Springer
New York
Berlin
Heidelberg
Hong Kong
London
Milan
Paris
Tokyo

Bruce A. Reed Cláudia L. Sales
Editors

Recent Advances in Algorithms and Combinatorics

With 52 Illustrations

Springer

Bruce A. Reed
Equipe Combinatoire
CNRS, Paris, France
and
McGill University
Montreal Canada

Cláudia L. Sales
Universidade Federal do Ceara
Departamento de Computacao—LIA
Campus do Pici—Bloco 910
CEP 60455-760 Fortaleza-CE Brasil
linhares@lia.ufc.br

Editors-in-Chief
Rédacteurs-en-chef
Jonathan Borwein
Peter Borwein
Centre for Experimental and Constructive Mathematics
Department of Mathematics and Statistics
Simon Fraser University
Burnaby, British Columbia V5A 1S6
Canada
cbs-editors@cms.math.ca

Mathematics Subject Classification (2000): 05-06

Library of Congress Cataloging-in-Publication Data
Recent advances in algorithms and combinatorics / Bruce Reed, Cláudia L. Sales.
p. cm. — (CMS books in mathematics ; 11)
Includes bibliographical references.
ISBN 0-387-95434-1 (alk. paper)
1. Combinatorial analysis. I. Reed, Bruce A. II. L. Sales, Cláudia. III. Series.
QA164 .R395 2002
511′6—dc21 2002017379

ISBN 0-387-95434-1 Printed on acid-free paper.

Printed in the United States of America.

9 8 7 6 5 4 3 2 1 SPIN 10864587

Typesetting: Pages created by the authors using a Springer TEX macro package.

www.springer-ny.com

Springer-Verlag New York Berlin Heidelberg
A member of BertelsmannSpringer Science+Business Media GmbH

Preface

Combinatorics is one of the fastest growing fields of mathematics. In large measure this is because many practical problems can be modeled and then efficiently solved using combinatorial theory. This real world motivation for studying algorithmic combinatorics has led not only to the development of many software packages but also to some beautiful mathematics which has no direct application to applied problems. In this volume we highlight some exciting recent developments in algorithmic combinatorics.

Most practical applications of algorithmic combinatorics would be impossible without the use of the computer. As computers become ever more powerful, more and more applications become possible. Computational biology is one example of a relatively new field in which algorithmic combinatorics plays a key role. The chapter by Sagot and Wakabayashi in this volume discusses how combinatorial tools can be used to search for patterns in DNA and protein sequences.

The information technology revolution has not only allowed for the resolution of practical problems using combinatorial techniques, it has also been the source of many new combinatorial problems. One example is radio channel assignment. In this problem we have a number of transmitters each of which must handle a number of calls. Each call must be assigned a frequency in such a way that interference is avoided (thus calls handled by the same transmitter are assigned different frequencies as are calls handled by transmitters which are *near* each other). The explosive growth in the use of the frequency spectrum due to, e.g., mobile telephone networks, has made it a very valuable resource. Indeed spectrum licenses were sold for billions of dollars in recent actions. So, efficiently assigning radio channels is of great importance. In his chapter in this volume, McDiarmid describes how to model radio channel assignment as a graph colouring problem and surveys the results that have been obtained using this approach.

Using graph colouring models to aid in studying how to direct the flow of information through transmission channels is not new. Shannon defined the zero-error capacity of a noisy (memoryless) channel as the maximum number of bits per symbol which could be sent through the channel whilst avoiding the introduction of errors. In 1961, Berge noted that determining the Shannon capacity of a channel could be modeled as a graph theory problem. In this context, he defined the class of perfect graphs, and noted that for certain channels, the Shannon capacity was simply the chromatic number of the associated perfect graph.

Berge's work motivated considerable research into efficient algorithms for colouring perfect graphs. This problem was finally resolved by Grötschel, Lovász, and Schrijver in 1981 using the (then) recently developed ellipsoid method. They modelled the problem as a semi-definite program(SDP) and then showed how the ellipsoid method could be used to solve this specific SDP. They later showed that in fact the ellipsoid method could be used to solve (actually approximately solve to arbitrary precision) a wide variety of SDP. It turned out that many combinatorial problems can be solved, at least approximately, by solving a related SDP. The most well-known example is the Goemans-Williamson algorithm to approximate Max-Cut. We are fortunate to have a chapter by Lovász in the volume which presents the basic theory of semi-definite programming and surveys its role in combinatorial optimization.

The ellipsoid method is a heavy piece of artillery, and researchers still hope to develop a combinatorial algorithm for colouring perfect graphs, which does not require its use. In his chapter, Maffray surveys some of the approaches with which this problem has been attacked. Many of the techniques for graph colouring he discusses are of interest in their own right and have applications to other graph colouring problems.

Although, the SDP artillery developed by Grötschel, Lovász, and Schrijver is incredibly powerful and beautiful, solving a graph theory problem using this artillery generally yields little insight as to how the optimal solution is determined by the graph's structure. Algorithms developed using decomposition theory, in contrast, often provide such information. Typically when using this paradigm, we decompose the graph into pieces which are easy to deal with, in such a way that it is easy to paste the solutions on the pieces together to obtain a global solution.

The first chapter in this volume is a beautifully written overview of one very important theory of this type. The theory was developed by Lovász to characterize the matching lattice (the matching lattice of a graph is the set of vectors indexed by its edges generated by the incidence vectors of perfect matchings). It was further refined by the authors of this chapter Carvalho, Lucchesi, and Murty.

Another very important theory of this type, that of tree width and tree decompositions, was developed by Robertson and Seymour as part of their seminal work characterizing graphs without a given fixed graph as a minor. In his chapter, Reed discusses the algorithmic aspects of tree decompositions, and mentions some applications to the theory of such diverse fields as databases, code optimization, and bioinformatics.

The third decomposition theorem discussed in this book is Szemerédi's regularity lemma. Roughly speaking, this result tells us that any large graph can be decomposed into edge disjoint random-looking bipartite graphs.

Thus the pieces in this decomposition are easy to deal with because they have many of the properties of random graphs. The basics of this theory is presented in the chapter of Kohayakawa and Rödl, who also survey some of its algorithmic applications. There are many equivalent definitons of what it means to be random looking, or formally *quasi-random*. In their chapter, Kohayakawa and Rödl present a new definition and show that this allows for more efficient algorithms to test this property. This important new result leads to similar efficiency gains in many of the algorithms developed using this theory.

Probability plays a different role in Steger's chapter on approximation algorithm. Recently, a link has been developed between the length of time needed to solve a problem using a deterministic algorithm and the number of bits needed to solve it using a random algorithm (with a given time complexity). This link has allowed researchers to show that many NP-complete optimization problems cannot be approximated unless $P = NP$. Steger's chapter provides an overview of this and other developments in this important field.

One use of graphs as models is to capture the intersection properties of various structures. In this context, the vertices correspond to the structures and two are joined by an edge if they intersect. For example, we can crudely model radio channel assignment in this way, To do so, we think of the vertices as discs around the transmitters representing the area which their broadcast signal covers, and join two vertices by an edge if these discs intersect. Then transmitters with an edge between them must use different frequencies.

Szwarcfiter's chapter considers a self-referential use of graphs of this kind. Here, the vertices of a graph G correspond to the cliques of some other graph H. We join two vertices of G by an edge if the corresponding cliques of H intersect in a vertex. We say that G is the clique graph of H. Szwarcfiter discusses various results on the class of clique graphs.

We have tried to point out some of the intersections between the topics treated in the various chapters of this work. The reader will stumble upon many more as he makes his way through it. More importantly, he will discover that each chapter can be appreciated and enjoyed in its own right.

Bruce A. Reed
Cláudia L. Sales

July 2002

This page intentionally left blank

Contents

This page intentionally left blank

List of Contributors

Marcelo H. de Carvalho
Departamento de Computação e Estatística, Universidade Federal do Mato Grosso do Sul, Campo Grande, MS, Brazil
Yoshi Kohayakawa
Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, SP, Brazil
László Lovász
Microsoft Research, Edmond, WA, USA
Cláudio L. Lucchesi
Instituto de Computação, Universidade de Campinas, Campinas, SP, Brazil
Frédéric Maffray
C.N.R.S., Laboratoire Leibniz - IMAG, Grenoble, France
Colin McDiarmid
Department of Statistics, University of Oxford, Oxford, UK
U. S. R. Murty
University of Waterloo, Waterloo, Canada
Bruce A. Reed
School of Computer Science, McGill University, Montreal, Canada
Vojtěch Rödl
Department of Mathematics and Computer Science, Emory University, Atlanta, GA, USA
Marie-France Sagot
Inria Rhône-Alpes, Laboratoire de Biométrie et Biologie Évolutive, Université Claude Bernard, Lyon, France
Angelika Steger
Institut für Informatik, Technische Universität München, München, Germany
Jayme L. Szwarcfiter
Instituto de Matemática, NCE and COPPE, Universidade Federal do Rio de Janeiro, Rio de Janeiro, RJ, Brazil
Yoshiko Wakabayashi
Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, SP, Brazil

1

The Matching Lattice

M.H. de Carvalho[1]
C.L. Lucchesi[2]
U.S.R. Murty[3]

1.1 Perfect Matchings

A set M of edges of a graph G is a *matching* of G if each vertex of G is incident with at most one edge of M and a *perfect matching* of G if each vertex of G is incident with precisely one edge of M.

The fundamental problem of characterizing graphs that admit a perfect matching was settled first for bipartite graphs by Hall in 1935, and more generally, for all graphs by Tutte in 1947. The two well-known theorems which provide these characterizations are given below:

Theorem 1.1.1 (Hall, 1935 [10]) *A graph G, with bipartition $\{A, B\}$, has a perfect matching if and only if $|A| = |B|$ and, $|I(G - S)| \leq |S|$, for each subset S of B, where $I(G - S)$ denotes the set of isolated vertices of $G - S$.*

Hall's Theorem is usually stated differently, but this version, in terms of isolated vertices, has a strong similarity with the statement of Tutte's Theorem, stated below.

Theorem 1.1.2 (Tutte, 1947 [24]) *A graph G has a perfect matching if and only if $|\mathcal{O}(G - S)| \leq |S|$, for each subset S of $V(G)$, where $\mathcal{O}(G - S)$ denotes the set of odd components of $G - S$.*

We denote by $\mathcal{M}(G)$ the set of all perfect matchings of G, or simply by $\mathcal{M}$, if G is understood. Tutte's Theorem gives a characterization of graphs for which $\mathcal{M}$ is nonempty. There are a number of interesting problems in

[1]Supported by CNPq, Brazil. Member of PRONEX 107/97 (MCT/FINEP)
[2]Supported by CNPq, and FAPESP, Brazil. Member of PRONEX 107/97 (MCT/FINEP)
[3]Partially supported by CNPq and FAPESP during his visit to UNICAMP, Brazil.

graph theory which are concerned with properties of graphs which have a perfect matching. A natural setting for the study of these problems is the theory of matching covered graphs.

1.2 Matching Covered Graphs

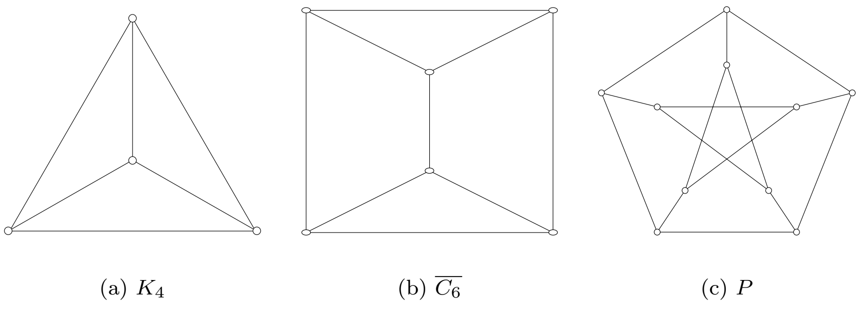
An edge of a graph G is *admissible* if it lies in some perfect matching of G. A *matching covered* graph is a connected graph each edge of which is admissible. Using Hall's and Tutte's Theorems, it is easy to derive the following characterizations of matching covered graphs.

Theorem 1.2.1 *Let G be a bipartite graph with bipartition $\{A, B\}$ that has a perfect matching. Graph G is matching covered if and only if for each nontrivial partition (A', A'') of A and each partition (B', B'') of B such that $|A'| = |B'|$, at least one edge of G joins some vertex of A' to some vertex of B''.*

Theorem 1.2.2 *A connected graph G is matching covered if and only if for each subset S of $V(G)$ the inequality $|\mathcal{O}(G - S)| \leq |S|$ holds, with equality only if set S is independent.*

(A set of vertices S of a graph G is *independent* (or *stable*) if the subgraph $G[S]$ of G spanned by S is free of edges.)

It can be shown that every matching covered graph is 2-connected. Using Theorem 1.2.2, it is easy to show that every 2-edge-connected cubic graph is matching covered. There are three cubic graphs which play particularly important roles in this theory. They are the complete graph K_4, the triangular prism $\overline{C_6}$, and the Petersen graph P (see Figure 1.1).



(a) K_4 (b) $\overline{C_6}$ (c) P

Figure 1.1. Three important cubic graphs.

For a history of the matching covered graphs, see Lovász and Plummer [13]. The most important source for this work is Lovász [12]. Murty [14] is also a very useful reference.

1.3 The Matching Lattice

Let G be a matching covered graph. For each set F of edges of G, we denote by χ^F the *incidence vector* of F in $\mathbb{R}^E$, that is, the vector w of 0's and 1's such that a coordinate $w(e)$ corresponding to edge e of G is equal to 1 if and only if edge e lies in set F. For any integer k, we denote by $\underline{\mathbf{k}}$ the vector of $\mathbb{R}^E$ whose coordinates are all equal to k. For every set F of edges of G and any vector w in $\mathbb{R}^E$, $w(F)$ denotes the scalar product of w and χ^F, that is, $w(F) = \sum_{e \in F} w(e)$.

The linear space generated by the incidence vectors of perfect matchings in G is the *matching space* of G and is denoted by $Lin(G)$:

$$Lin(G) := \{w \in \mathbb{R}^E : w = \sum_{M \in \mathcal{M}} \alpha_M \chi^M, \alpha_M \in \mathbb{R}\}.$$

Likewise, the lattice generated by the incidence vectors of perfect matchings in G is the *matching lattice* of G and is denoted $Lat(G)$:

$$Lat(G) := \{w \in \mathbb{Z}^E : w = \sum_{M \in \mathcal{M}} \alpha_M \chi^M, \alpha_M \in \mathbb{Z}\}.$$

We may restrict the set of coefficients used in the linear combinations to that of the set of nonnegative rationals $\mathbb{Q}_{\geq 0}$ or integers $\mathbb{Z}_{\geq 0}$, thereby obtaining the *rational cone* or *integer cone* spanned by the incidence vectors of matchings in $\mathcal{M}(G)$, denoted $Rat.Con(G)$ or $Int.Con(G)$, respectively:

$$\begin{aligned} Rat.Con(G) &:= \{w \in \mathbb{Q}^E_{\geq 0} : w = \sum_{M \in \mathcal{M}} \alpha_M \chi^M, \alpha_M \in \mathbb{Q}_{\geq 0}\}, \\ Int.Con(G) &:= \{w \in \mathbb{Z}^E_{\geq 0} : w = \sum_{M \in \mathcal{M}} \alpha_M \chi^M, \alpha_M \in \mathbb{Z}_{\geq 0}\}. \end{aligned}$$

Tait showed that the Four-Colour Conjecture is equivalent to the following assertion:

Conjecture 1.3.1 ([23]) *Every 2-connected cubic planar graph is 3-edge colourable.*

If G is a k-regular graph, then G is k-edge colourable if and only if $\underline{\mathbf{1}}$ can be expressed as a sum of incidence vectors of k perfect matchings of G. In other words, G is k-edge colourable if and only if $\underline{\mathbf{1}}$ lies in $Int.Con(G)$. Tutte made the following generalization of the Four-Colour Conjecture:

Conjecture 1.3.2 ([25]) *Every 2-connected cubic graph free of Petersen minors is 3-edge colourable.*

A *minor* of a graph G is any graph that may be obtained from a subgraph of G by edge contractions. Clearly, $Int.Con \subseteq Lat$. This observation led Seymour to study the matching lattice of certain cubic graphs:

Theorem 1.3.3 ([21]) *For every 2-edge-connected cubic graph G, if G does not contain the Petersen graph as a minor, then $\underline{\mathbf{1}} \in Lat(G)$.*

We note that Theorem 1.3.3 may be regarded as a proof of a relaxation of Conjecture 1.3.2.

Robertson, Sanders, Seymour and Thomas gave a new proof of the Four-Colour Theorem in [15]. Extending the techniques they developed in that proof, they gave a proof of Conjecture 1.3.2 [17, 19, 18, 16]. Thus, vector $\underline{\mathbf{1}}$ lies in the integer cone of any cubic 2-edge-connected planar graph.

Seymour also proved the following assertion:

Theorem 1.3.4 ([21]) *For every 2-edge-connected cubic graph G, vector $\underline{\mathbf{2}}$ lies in $Lat(G)$.*

Theorem 1.3.4 may be regarded as a proof of a relaxation of a conjecture due to Fulkerson and Berge [8]:

Conjecture 1.3.5 *For every 2-edge-connected cubic graph G, vector $\underline{\mathbf{2}}$ lies in $Int.Con(G)$.*

Lovász, [12], generalized Theorems 1.3.3 and 1.3.4 of Seymour by establishing a complete characterization of the matching lattice of any matching covered graph. More specifically, for any matching covered graph G, and any w in $\mathbb{Z}^E$, Lovász determined the necessary and sufficient conditions for w to be in $Lat(G)$.

We begin with an obvious necessary condition. For any subset S of V, $C = \nabla_G(S)$ (or simply $C = \nabla(S)$) denotes the (edge-) cut of G with S and $\overline{S} = V - S$ as its *shores*; in other words, $\nabla(S)$ is the set of all edges of G which have precisely one end in S. Clearly, for any perfect matching M and any vertex v, $\chi^M(\nabla(v)) = 1$. Therefore, if $w = \sum_{M \in \mathcal{M}} \alpha_M \chi^M$, then, for any vertex v, $w(\nabla(v)) = \sum_{M \in \mathcal{M}} \alpha_M$. A vector w in $\mathbb{R}^E$ is *regular* over a set $\mathcal{C}$ of cuts of G if $w(C) = w(D)$ for any two cuts C and D in $\mathcal{C}$. Vector w is *regular* if it is regular over the set of all stars $\{\nabla(v) : v \in V\}$. In view of the above observation, we have:

Lemma 1.3.6 *For every matching covered graph G, if a vector w lies in $Lat(G)$ then w is regular.*

For matching covered bipartite graphs, the regularity of a vector is also sufficient:

Lemma 1.3.7 *For every bipartite matching covered graph G, a vector in $\mathbb{Z}^E$ lies in $Lat(G)$ if and only if it is regular.*

However, in general, the regularity condition is not sufficient for an integral vector w to belong to $Lat(G)$. For the Petersen graph, $\underline{\mathbf{1}}$ satisfies the regularity condition, but it is not in the matching lattice[4].

In the study of the matching lattice, one makes use of two types of decompositions of matching covered graphs. These decompositions are tight cut decompositions and ear decompositions. We will consider both in the next sections.

1.4 Tight Cut Decompositions

The first type of decomposition of matching covered graphs, known as the *tight cut decomposition*, was introduced by Lovász in [12]. In this section, we describe this procedure and explain its relevance to the study of the matching lattice.

1.4.1 Tight Cuts

Let G be a matching covered graph. A cut C of G is *tight* in G if every perfect matching of G has precisely one edge in C.

Let S be a shore of a cut C of a matching covered graph G. Then, the graph obtained from G by contracting $\overline{S}$ to a single vertex $\overline{s}$ is denoted by $G\{S;\overline{s}\}$ and the graph obtained from G by contracting S to a single vertex s is denoted by $G\{\overline{S};s\}$. We shall refer to these two graphs $G\{S;\overline{s}\}$ and $G\{\overline{S};s\}$ as the *C-contractions* of G. If the names of the new vertices in the C-contractions are irrelevant, we shall simply denote them by $G\{S\}$ and $G\{\overline{S}\}$. Observe that this notation is similar to the notation $G[S]$ used for the subgraph of G induced by S; $G\{S;\overline{s}\}$ is the subgraph induced by S, together with a new vertex $\overline{s}$ such that each edge in $\nabla_G(S)$ joins its end in S to the vertex $\overline{s}$.

Lemma 1.4.1 ([12]) *Let G be a matching covered graph, and let $C = \nabla(X)$ be a tight cut of G. Then the two C-contractions $G_1 = G\{X\}$ and $G_2 = G\{\overline{X}\}$ obtained by contracting the two shores $\overline{X}$ and X, respectively, are also matching covered.*

[4]This follows from the fact that, for any pentagon C of the Petersen graph, $\underline{\mathbf{1}}(C) = 5 \equiv 1$ $(mod\ 2)$, whereas $\chi^M(C) \equiv 0$ $(mod\ 2)$, for any perfect matching M.

For the Petersen graph, a vector w in Z^E is in the matching lattice if and only if it is regular and, for any pentagon Q, $w(E(Q))$ is even.

1.4.2 Barrier Cuts and 2-Separation Cuts

There are two types of tight cuts, namely barrier cuts and 2-separation cuts, which are of special interest in this theory. To define the cuts of the first type, we need the notion of a barrier in a matching covered graph.

Let G be a matching covered graph. By Tutte's Theorem, $|\mathcal{O}(G-S)| \leq |S|$, for each subset S of V. A nonempty subset S of V is called a *barrier* of G if $|\mathcal{O}(G-S)| = |S|$. If v is any vertex of G, then $\{v\}$ is a barrier of G. Such a barrier is *trivial.*

Barrier Cut: If B is a nontrivial barrier and H is any nontrivial odd component of $G-B$, then $\nabla(V(H))$ is a tight cut. Such a cut is called a *barrier cut* (see Figure 1.2(a)).

The second type of tight cut is defined below.

2-Separation cut: Let $\{u,v\}$ be a 2-separation that is not a barrier. Let H be a component of $G-\{u,v\}$. Since $\{u,v\}$ is not a barrier, graph H is even. Let $X := V(H) \cup \{u\}$, $Y := V(H) \cup \{v\}$. Then $\nabla(X)$ and $\nabla(Y)$ are both tight cuts. Such cuts are referred to as *2-separation cuts* (see Figure 1.2(b)).

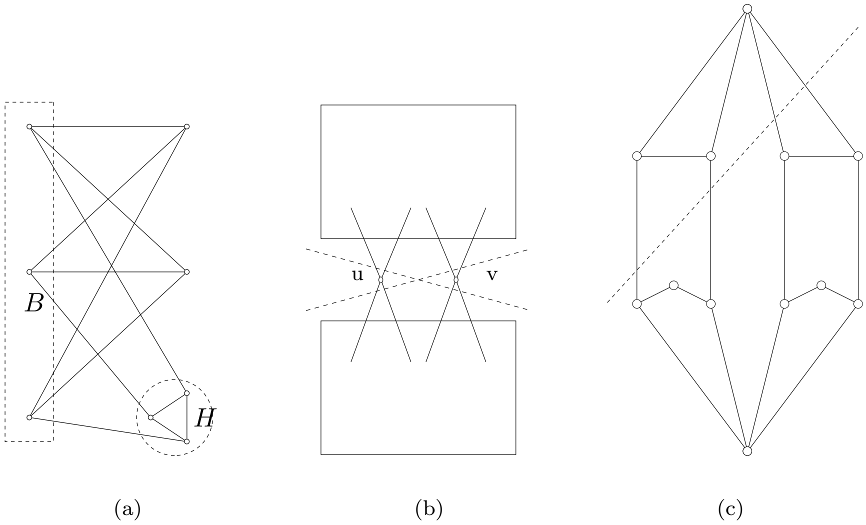


Figure 1.2. (a) A barrier cut (b) a 2-separation cut and (c) a tight cut that is neither a barrier nor a 2-separation cut.

A matching covered graph may have a tight cut that is neither a barrier cut nor a 2-separation cut (see Figure 1.2(c)). However, Edmonds, Lovász and Pulleyblank proved the following fundamental result:

Theorem 1.4.2 ([7]) *If a matching covered graph G has a nontrivial tight cut, then it either has a nontrivial barrier cut or it has a 2-separation cut.*

Until recently, the only known proof of Theorem 1.4.2 was based on LP-duality. Recently Szigeti [22] obtained a simple proof which does not use LP-duality.

1.4.3 Bricks and Braces

A matching covered graph G in which all barriers are trivial is *bicritical.* (A graph G is bicritical if and only if $G - \{u, v\}$ has a perfect matching for any two distinct vertices u and v of G.) A *brick* is a 3-connected bicritical graph. By definition, a brick cannot have barrier cuts or 2-separation cuts. Hence, by Theorem 1.4.2, we have:

Lemma 1.4.3 *A brick has no nontrivial tight cuts.*

Let G be a bipartite matching covered graph G with bipartition (A, B), Let (A_1, A_2) be a partition of A into non-empty sets A_1 and A_2, and (B_1, B_2) be a partition of B into non-empty sets B_1 and B_2. Suppose that $|A_1| = |B_1| + 1$, $|B_2| = |A_2| + 1$, and that there are no edges of G linking B_1 with A_2. Then, the cut $\nabla((A_1 \cup B_1))$ is a nontrivial tight cut of G (See Figure 1.3). In fact, every tight cut in a bipartite matching covered graph must be of this form (see Lovász, [12]). A bipartite matching covered graph which does not have any nontrivial tight cuts is called a *brace.* The above

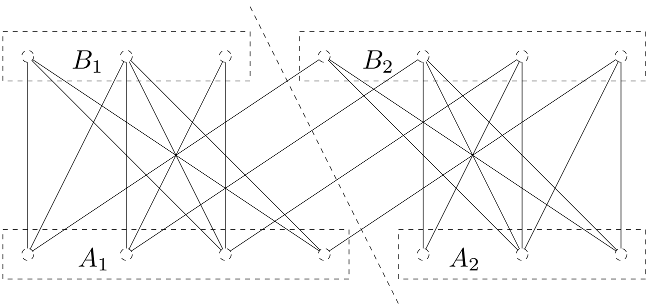


Figure 1.3. A tight cut in a bipartite graph.

mentioned facts concerning matching covered graphs without nontrivial tight cuts are summarized by the following statement.

Theorem 1.4.4 (See [12]) *A matching covered graph has no nontrivial tight cuts if and only if it is either a brick or a brace.*

1.4.4 Tight Cut Decompositions

Let G be a matching covered graph, and let $C = \nabla(S)$ be a nontrivial tight cut of G. Then, as already noted, the two C-contractions G_1 and G_2 of G are also matching covered. If either G_1 or G_2, say G_1 has a nontrivial tight cut D, then we can take D-contractions of G_1, in the same manner as above, and obtain smaller matching covered graphs than G_1. Thus, given any matching covered graph G, by repeatedly applying cut-contractions, we can obtain a list of graphs which do not have nontrivial tight cuts (i.e. bricks and braces. See Figure 1.4).

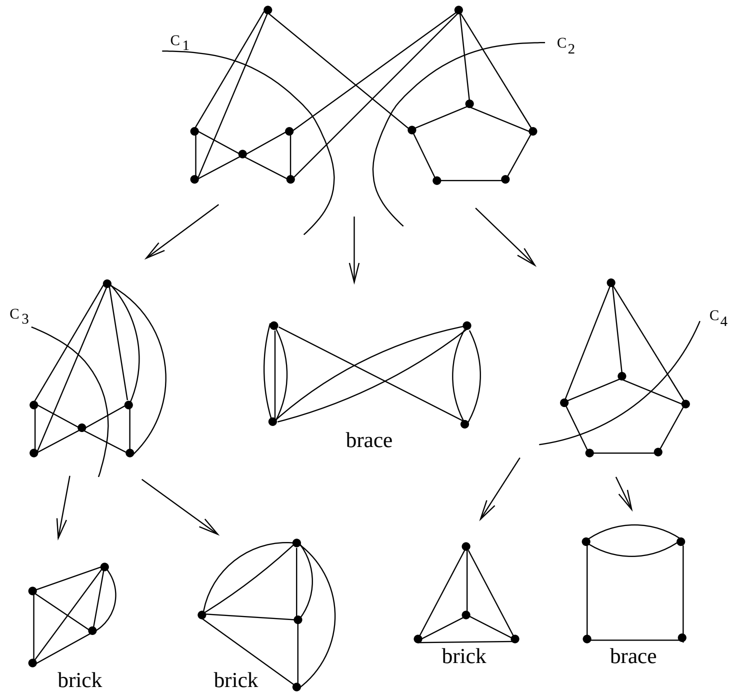


Figure 1.4. Tight cut decomposition.

Lovász gave a very elegant proof of the following remarkable result:

Theorem 1.4.5 (See [12]) *The results of any two applications of the tight cut decomposition procedure on a matching covered graph G are the same list of bricks and braces, except possibly for the multiplicities of edges.*

In particular, the numbers of bricks and braces resulting from a tight cut decomposition of a matching covered graph G is independent of the tight cut decomposition. We shall let $b(G)$ denote the number of bricks of G. The number of bricks of G whose underlying simple graphs are Petersen graphs is also an invariant of G; we shall denote it by $p(G)$. The numbers $b(G)$, and (b+p)$(G) = b(G) + p(G)$ play important roles in the theory of matching covered graphs.

Note that $b(G) = 0$ if and only if G is bipartite, and $b(G) = 1$ if and only if for every tight cut C of G one of the C-contractions of G is bipartite and the other C-contraction has exactly one brick. We shall refer to a matching covered graph G with $b(G) = 1$ as a *near-brick*. Many useful properties that bricks satisfy are quite often satisfied more generally by near-bricks. Furthermore, for proving theorems concerning bricks, it is often convenient to consider the wider class of near-bricks.

1.4.5 Tight Cuts and the Matching Lattice

The matching lattice of a matching covered graph may be expressed in terms of the matching lattices of tight cut contractions of the graph:

Theorem 1.4.6 *Let G be a matching covered graph, let $C := \nabla(S)$ be a tight cut of G, and let G_1 and G_2 be the two C-contractions of G. Let w be a vector in $\mathbb{Z}^E$, and let w_1 and w_2 be the restrictions of w to $E(G_1)$ and $E(G_2)$, respectively. Then w is in Lat(G) if and only if w_1 and w_2 are in Lat(G_1) and Lat(G_2), respectively.*

Thus, the matching lattice of a matching covered graph G may be expressed in terms of the matching lattices of the bricks and braces of G. As already noted, the characterization of the matching lattices of bipartite matching covered graphs is simple. Thus, braces, which are bipartite, pose no problem. Lovász proved the following interesting theorem concerning bricks. (Note that a brick G with $p(G) = 0$ is a brick whose underlying simple graph is not isomorphic to the Petersen graph.)

Theorem 1.4.7 ([12]) *Let G be a brick with $p(G) = 0$, and let w be a vector in $\mathbb{Z}^E$. Then, w is in the matching lattice of G if and only if w is regular.*

Let G be a near-brick. Let C be any nontrivial tight cut of G. Let w be any vector in $\mathbb{Z}^E$ that is regular. One of the C-contractions of G, say G', is bipartite, the other a near-brick, say G'', such that $p(G'') = p(G) = 0$. Then, a counting argument in G' may be used to show that $w(C) = w(\nabla(v))$, for any vertex v of G'. Thus, the restrictions of w to $E(G')$ and $E(G'')$ are both regular. The next assertion then follows by induction:

Lemma 1.4.8 *For any near-brick G, a vector in $\mathbb{R}^E$ is regular if and only if it is regular over the set of tight cuts of G.*

The next statement then follows from Theorem 1.4.7:

Theorem 1.4.9 *Let G be a near-brick with $p(G) = 0$, and let w be a vector in $\mathbb{Z}^E$. Then, w is in the matching lattice of G if and only if w is regular.*

1.5 Separating Cut Decompositions

We have seen in Theorem 1.4.1, that for any tight cut C of a matching covered graph G, both C-contractions of G are matching covered. The converse, however, is not true. For example, if we consider a cut C in $\overline{C_6}$ spanned by the vertices of a triangle, then both C-contractions of $\overline{C_6}$ are equal to K_4, a brick, whence matching covered. However, there exists in $\overline{C_6}$ a perfect matching that contains all three edges of C.

We now consider the class of cuts C of a matching covered graph G such that both C-contractions of G are matching covered. These cuts play a crucial role in many proofs in this theory; typically, we find a suitable cut such that the two cut-contractions with respect to that cut are matching covered, and prove the desired theorem by applying induction to the resulting smaller graphs.

1.5.1 Separating Cuts

Let G be a matching covered graph. Cut $C := \nabla(S)$ is a *separating cut* of G if both C-contractions of G are matching covered. In particular, every tight cut of G is separating. The following lemma provides a useful characterization of separating cuts in a matching covered graph.

Lemma 1.5.1 ([4]) *A cut C of a matching covered graph G is separating if and only if for each edge e of G, there exists a perfect matching that contains e and just one edge in C.*

1.5.2 Solid Matching Covered Graphs

A matching covered graph G is *solid* if each separating cut of G is tight. We now describe some classes of solid matching covered graphs. Bipartite matching covered graphs are solid. Bipartite matching covered graphs are a particular case of a more general class of solid matching covered graphs, which will be described now. A graph is *odd-intercyclic* if any two of its odd circuits have at least one vertex in common. It can be shown that if a matching covered graph G has a separating cut C that is not tight, each shore of C spans a nonbipartite graph. Thus, every nonsolid matching covered graph has at least two disjoint odd circuits. We then have the following consequence:

Lemma 1.5.2 *Every odd-intercyclic matching covered graph is solid.*

There are various classes of graphs which are obviously odd-intercyclic. For example, every bipartite graph is odd-intercyclic. Similarly, any graph which has a vertex whose deletion results in a bipartite graph is odd-intercyclic. We describe now an important class of odd-intercyclic graphs. A *wheel* is a simple graph obtained from a circuit by adding a new vertex and joining that vertex to each vertex of the circuit; the circuit is called the *rim*, the new vertex the *hub* and each edge joining the hub to the rim a *spoke*. The *order* of the wheel is the number of vertices of its rim; a wheel of order n is denoted W_n. A wheel is *even* or *odd*, according to the parity of n. Note that the hub of a wheel is uniquely identified, except for W_3, which is K_4, the complete graph on four vertices; in this case, we may say that any of its vertices is a hub. It is easy to see that every wheel is odd-intercyclic and that every odd wheel is a brick.

The complete graph K_5 is an odd-intercyclic graph which does not belong to any of the above described families of graphs. Gerards et al [9] discovered an interesting class of odd-intercyclic graphs that are embeddable in the projective plane. Suppose that H is a 2-connected bipartite plane graph. Let $u_1, u_2, \cdots, u_n$, and $v_1, v_2, \cdots, v_n$ be vertices of H which appear in the cyclic order $(u_1, u_2, \cdots, u_n, v_1, v_2, \cdots, v_n)$ on the outer face of H. Now obtain G from H by joining, for $1 \leq i \leq n$, u_i and v_i by an edge e_i. Using the fact that H is planar and bipartite, and the fact that, for $1 \leq i < j \leq n$, the ends u_i and v_i of e_i are 'separated' by the ends u_j and v_j of e_j, it is easy to verify that G is odd-intercyclic. For convenience, we shall denote by $\mathcal{G}$ the class of graphs G that can be constructed in this manner.

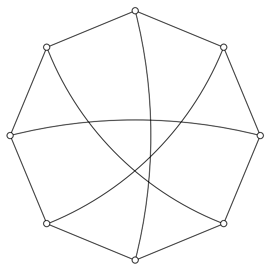
Other examples of odd-intercyclic graphs may be obtained by suitably gluing together graphs in the families described above. (For example, suppose that G is a graph in the family $\mathcal{G}$, such that the associated bipartite plane graph H has a vertex v of degree three. Then the graph G' obtained by splicing G with $K_{3,3}$ at v is also odd-intercyclic.) A complete characterization of odd-intercyclic graphs has been given by Gerards et al [9].

Let $(v_0, v_1, \cdots, v_{n-1})$ and $(v_n, v_{n+1}, \cdots, v_{2n-1})$ be two disjoint paths, for some positive integer n. The graph obtained from these paths by adding the edges $v_i v_{i+n}$, for each i such that $0 \leq i < n$, is a *ladder*. If we also add edges $v_{n-1}v_n$ and $v_{2n-1}v_0$, we get a *Möbius ladder* M_n of order n. Figure 1.5 (a) depicts the Möbius ladder of order four. The following result is easy to prove:

Lemma 1.5.3 *For each positive integer n, the Möbius ladder M_n, of order n, is odd-intercyclic (belonging to the family $\mathcal{G}$). Moreover, if n is even then M_n is a (solid) brick.*

We have seen that the property of being odd-intercyclic is sufficient for a matching covered graph to be solid. However, the condition is not necessary.

Figure 1.5 (b) shows an example, due to Murty, of a solid brick that has two disjoint odd circuits.


(a)

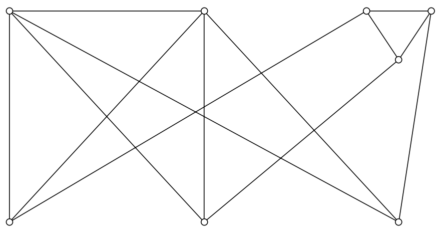

(b)

Figure 1.5. (a) Möbius ladder M_4 (b) A solid brick that is not odd-intercyclic

1.5.3 Separating Cut Decompositions

Let G be a matching covered graph, and let $C = \nabla(S)$ be a nontrivial separating cut of G. By definition of separating cut, the two C-contractions G_1 and G_2 of G are also matching covered. If either G_1 or G_2, say G_1, has a nontrivial separating cut D, then we can take D-contractions of G_1, in the same manner as above, and obtain matching covered graphs smaller than G_1. Thus, given any matching covered graph G, by repeatedly applying cut-contractions, we can obtain a list of solid bricks and braces. Note that this is a generalization of the tight cut decomposition.

It turns out that for each separating cut C of a matching covered graph G, both shores of C span connected graphs. Thus, both C-contractions of G are minors of G. Consequently, the bricks and braces obtained by an application of a separating cut decomposition procedure to a matching covered graph G are solid minors of G. A graph H is a *separating cut minor* of a matching covered graph G if H may be obtained from G by a (possibly partial) application of a separating cut decomposition. More formally, the collection of separating cut minors of a matching covered graph G is defined recursively as follows:

(i) Graph G is a separating cut minor of itself.

(ii) For each nontrivial separating cut C of G, each separating cut minor of either C-contraction of G is a separating cut minor of G.

(iii) The graphs obtained by the application of the two rules above are the only separating cut minors of G.

Although separating cuts are, in some sense, a generalization of tight cuts, some nice properties of tight cuts are lost by this generalization. For example, an analog of Theorem 1.4.5 is not valid for separating cuts in general. That is, separating cut decompositions are not unique. Carvalho obtained a graph that has two distinct separating cut decompositions. The graph G depicted in Figure 1.6 is a variation of Carvalho's original example: the two D-contractions of G are isomorphic, up to multiple edges, to the solid brick depicted in Figure 1.5(b), whereas the two C-contractions of G are isomorphic nonsolid bricks, each of which has a separating cut decomposition with three bricks of four vertices. Therefore, it is possible to obtain two essentially distinct separating cut decompositions of G, one with two isomorphic graphs, the other with six graphs, each of which a brick of four vertices. Using this example, it is possible to build a family G_n of graphs ($n \geq 0$) such that for each n, G_n has two distinct separating cut decompositions such that the number of bricks obtained by the two decompositions differ by at least $4 \cdot 2^n$. (For example, by splicing two copies of the graph in Figure 1.6, we can obtain a graph which has one separating cut decomposition resulting in four solid bricks, and another decomposition resulting in ten solid bricks.)

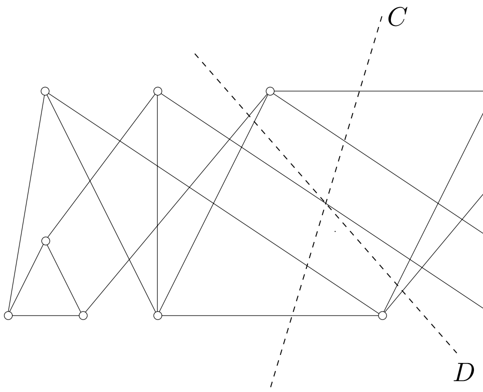


Figure 1.6. A brick with distinct separating cut decompositions.

1.5.4 The Characteristic of a Separating Cut

Let G be a matching covered graph. For each separating cut C of G, the *characteristic* $\lambda(C)$ *of* C is defined as follows:

$$\lambda(C) := \begin{cases} \min\{|M \cap C| > 1 : M \in \mathcal{M}(G)\}, & \text{if } C \text{ is not tight} \\ \infty, & \text{otherwise.} \end{cases}$$

The *characteristic* $\lambda(G)$ *of* a matching covered graph G is defined as follows:

$$\lambda(G) := min\{\lambda(C) : \ C \text{ is separating}\}.$$

It is easy to see that $\lambda(\overline{C}_6) = 3$ and that the characteristic of the Petersen graph is five. We remark that a matching covered graph G is solid if and only if its characteristic is infinite. The following result, proved by Carvalho in his Ph. D. thesis [2] (see also [4]), establishes the quintessential property of the characteristic of a brick.

Theorem 1.5.4 *The characteristic of every brick G lies in $\{3, 5, \infty\}$. Moreover, if $\lambda(G) = 5$ then the underlying simple graph of G is the Petersen graph.*

The next two results, proved by Christiane N. Campos [1], a doctoral student who is writing her dissertation under the supervision of the second author, extend Theorem 1.5.4 to every separating cut of a matching covered graph:

Theorem 1.5.5 *The characteristic of every separating cut C of every brick G lies in $\{3, 5, \infty\}$. Moreover, if $\lambda(C) = 5$ then the underlying simple graph of G is the Petersen graph.*

Theorem 1.5.6 *The characteristic of every separating cut C of every matching covered graph G lies in $\{3, 5, \infty\}$. Moreover, if $\lambda(C) = 5$ then graph G has a separating cut minor P such that cut C is nontrivial and separating in P, and the underlying simple graph of P is the Petersen graph.*

Figure 1.7 shows an example of a separating cut of characteristic five in a matching covered graph G with $b(G) = p(G) = 2$, due to Campos [1].

1.6 Removable Edges

An edge e of a matching covered graph G is *removable* if $G - e$ is matching covered. Lovász [12] proved the following important theorem. (He used it in his inductive proof of Theorem 1.4.7.)

Theorem 1.6.1 *Every brick different from K_4 and $\overline{C}_6$ has a removable edge.*[5]

[5]It is in fact true that every brick different from K_4 and $\overline{C}_6$ has $(\Delta - 2)$ removable edges, where Δ denotes the maximum degree in the brick. For a simple proof of this result, see [3].

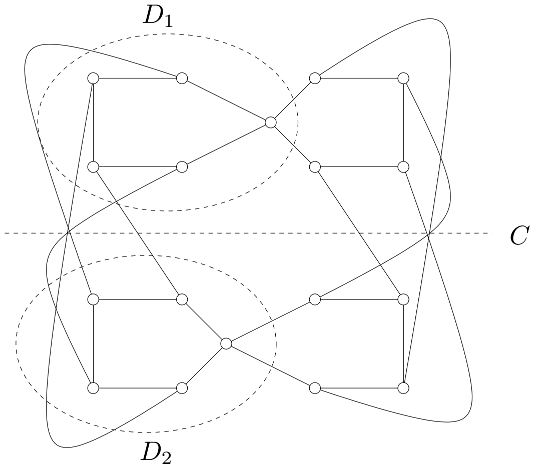


Figure 1.7. A cut C of characteristic five and two cuts, D_1 and D_2, used in a partial separating cut decomposition of G that yields the Petersen brick having C as a nontrivial separating cut.

For a removable edge e in a brick G, $b(G-e)$ and $p(G-e)$ can be arbitrarily large. In general, for every matching covered graph G, both invariants, b and $(b+p)$, are monotonic:

Lemma 1.6.2 *For every matching covered graph G and every removable edge e of G, $b(G-e) \geq b(G)$ and $(b+p)(G-e) \geq (b+p)(G)$.*

It is worth mentioning that p is not monotonic: for example, in the Petersen graph, which is a brick, every edge e is removable, yet $p(G-e) = 0 < p(G)$.

A removable edge e of a matching covered graph G is *b-invariant* if $b(G-e) = b(G)$, and is *$(b+p)$-invariant* if $(b+p)(G-e) = (b+p)(G)$. An edge e of a matching covered graph G is *b-removable* if it is removable and b-invariant, and is *$(b+p)$-removable* if it is removable and $(b+p)$-invariant. In 1987, Lovász made the following conjecture [12]:

Conjecture 1.6.3 *Every brick different from K_4, $\overline{C}_6$, and the Petersen graph has a b-removable edge.*

This conjecture was generalized by Carvalho and Lucchesi, in 1993. The first author proved the validity of this more general conjecture in his Ph. D. dissertation, written under the supervision of the two other authors [2, 4, 5]:

Theorem 1.6.4 *Every brick distinct from K_4 and $\overline{C}_6$ has a $(b+p)$-removable edge.*

Thus, a simple brick distinct from K_4, $\overline{C}_6$ and the Petersen graph not only has a b-removable edge, but it has one, e, such that the underlying simple brick of near-brick $G-e$ is not isomorphic to the Petersen graph. Figure 1.8 shows an example of an edge that is both b-removable and $(b+p)$-removable and another that is b-removable but not $(b+p)$-removable.

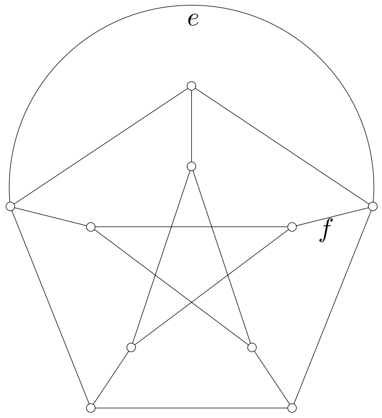


Figure 1.8. An edge e that is b-removable but not $(b+p)$-removable, and an edge f that is both b-removable and $(b+p)$-removable.

The proof of Theorem 1.6.4 proceeds along the following general lines. We first show that if G is a solid brick, then any removable edge of G is $(b+p)$-removable. Thus, it is sufficient to prove the theorem for bricks which have nontrivial separating cuts. We show that every such brick in fact has a separating cut such that the two cut-contractions with respect to that cut are near-bricks. By induction, the bricks of these near-bricks have $(b+p)$-removable edges. From this we deduce that the given brick has a $(b+p)$-removable edge. A proof of this result can be found in [4] and [5]

1.7 Ear Decompositions

Let G' be a subgraph of a graph G. Then, a path P of odd length in $G - E(G')$ is a *single ear* of G' if (i) both ends of P are in $V(G')$, and (ii) P is internally disjoint from G'. A *double ear* of a subgraph G' of a graph G is a pair of vertex-disjoint single ears of G'. An *ear* of G' is either a single ear or a double ear of G'. An *ear decomposition* of a matching covered graph G is a sequence

$$G_1 \subset G_2 \subset \ldots \subset G_r = G$$

of matching covered subgraphs of G where (i) $G_1 = K_2$, and (ii) for $1 \leq i \leq r-1$, G_{i+1} is the union of G_i and an ear (single or double) of G_i. The following fundamental theorem was established by Lovász and Plummer.

Theorem 1.7.1 (The two-ear Theorem [13], [3]) *Every matching covered graph has an ear decomposition.*

A bipartite matching covered graph has an ear decomposition which uses only single ears, but an ear decomposition of a nonbipartite matching covered graph must have at least one double ear. There are matching covered graphs which have no ear decompositions with just one double ear. For example, in any ear decomposition of the Petersen graph, one has at least two double ears. In fact, there is essentially only one ear decomposition of the Petersen graph, with $r = 5$ and two double ears. (See [13], page 178.)

A given matching covered graph may have different ear decompositions with different numbers of double ears. For example, consider the graph $P + e$ obtained from the Petersen graph P by adding an edge e joining two nonadjacent vertices of P (Figure 1.8). By extending the ear decomposition of P to that of $P + e$, using e as the last ear, we can obtain an ear decomposition of G with two double ears. However, it is possible to find an ear-decomposition of $P + e$ using just one double ear, in which the last (single) ear is edge f (Figure 1.8).

If G is any matching covered graph, we denote by $d_*(G)$ the minimum possible number of double ears an ear decomposition of G may have, and refer to an ear decomposition of G with exactly $d_*(G)$ double ears as an *optimal ear decomposition* of G. For any bipartite matching covered graph $d_* = 0$. For any nonbipartite matching covered graph, in particular for any brick, $d_*(G) \geq 1$, and as already noted, $d_* = 2$ for the Petersen graph.

The relevance of optimal ear decompositions to finding bases of matching lattices is described in the next section. We conclude this section with the following simple identity relating the number n of vertices, the number m of edges of a matching covered graph G, and the length r of an ear decomposition $(G_1 = K_2 \subset G_2 \subset \ldots \subset G_r = G)$ of G and d, the number of double ears in the decomposition.

Theorem 1.7.2 *The numbers m, n, r, and d satisfy the following identity.*

$$r = m - n + 2 - d.$$

1.7.1 Perfect Matchings and Ear Decompositions

A matching covered subgraph H of a matching covered graph G is a *nice* subgraph of G if $G - V(H)$ has a perfect matching. It is easy to see that all the subgraphs G_i in an ear decomposition of a matching covered graph G are nice. Using this property, it is possible to associate with any given ear decomposition of length r, a set $M_1, M_2, \cdots, M_r$ of r perfect matchings of G such that $\chi^{M_1}, \chi^{M_2}, \cdots, \chi^{M_r}$ are linearly independent.

Theorem 1.7.3 *Let G be a matching covered graph. Let*

$$\mathcal{D} = (G_1 = K_2 \subset G_2 \subset \ldots \subset G_r = G)$$

be an ear decomposition of G. Then there exists a list

$$\mathcal{M}(\mathcal{D}) = (M_1, M_2, \cdots, M_r)$$

of perfect matchings of G such that, for $1 \leq i \leq r$,

(i) $M_i \cap E(G_j)$ is a perfect matching of G_j for $i \leq j \leq r$, and

(ii) there is an edge e_i in $M_i \cap E(G_i)$ such that $e_i \notin M_k$, for $1 \leq k < i$.

We shall refer to the list $\mathcal{M}(\mathcal{D}) = (M_1, M_2, \cdots, M_r)$ of perfect matchings obtained as in the above theorem as a list of perfect matchings *associated* with the given ear decomposition. Since, for each i, $1 \leq i \leq r$, there is an edge e_i which is in M_i but not in any one of the matchings $M_1, M_2, ...M_{i-1}$, clearly, the incidence vectors of $M_1, M_2, ...M_r$ are linearly independent. This observation and Lemma 1.7.2 lead to the following simple, but very useful corollary.

Theorem 1.7.4 *If a matching covered graph G has an ear decomposition with d double ears, then there exist $m - n + 2 - d$ perfect matchings of G whose incidence vectors are linearly independent, where m and n are the numbers of edges and vertices of G, respectively.*

Thus, one means of obtaining a 'large' independent set of perfect matchings of a matching covered graph G would be to obtain an ear decomposition of G with as few double ears as possible. However, it is not always the case that the set of perfect matchings associated with an optimal ear decomposition of a matching covered graph G yields a basis for the matching lattice of G. For example, an optimal ear decomposition of the Petersen graph has length five, and so yields an independent set of five perfect matchings. However, the Petersen graph has six perfect matchings, and they are all linearly independent.

1.7.2 A Lower Bound for $d_*(G)$

Using the next Lemma and the observations made in the previous section, one is then able to determine a lower bound for the number of double ears of an ear decomposition of a matching covered graph.

Lemma 1.7.5 *Let G be a matching covered graph and $C := \nabla(S)$ a tight cut in G. Let G' and G'' be the two C-contractions of G. Then $d_*(G) \geq d_*(G') + d_*(G'')$.*

Theorem 1.7.6 *For any matching covered graph G, $d_*(G) \geq b(G)+p(G)$.*

1.7.3 Removable Ears

In all the previous approaches, existence of ear decompositions of a matching covered graphs were established by showing how a nice matching covered proper subgraph G_i of a matching covered graph G could be extended to a nice matching covered subgraph G_{i+1} by the addition of a single or double ear. Another approach is to build an ear decomposition of a matching covered graph in the reverse order. To state the theorem precisely, we need to define the notion of a removable ear in a matching covered graph.

Let G be a matching covered graph, and let P be a path in G. Then, P is said to be a *removable single ear* in G if (i) P is a path of odd length whose internal vertices have all degree two in the graph, and (ii) the graph obtained from G by deleting all the edges and internal vertices of P is matching covered. (A removable ear of length one is *a removable edge.*) A *removable double ear* in G is a pair (P_1, P_2) of disjoint paths, each of which of odd length, such that (i) each internal vertex of each of P_1 and P_2 has degree two in the graph, (ii) the graph obtained from G by deleting all the edges and internal vertices of each of P_1 and P_2 is matching covered, and (iii) neither P_1 nor P_2 is a removable single ear. (A removable double ear each path of which has length one is a *removable doubleton.*) A *removable ear* in G is either a single or a double ear which is removable.

In trying to establish the existence of ear decompositions with special properties, it is convenient to find the subgraphs in the ear decomposition in the reverse order starting with $G_r = G$. Thus, after obtaining a subgraph G_i in the sequence which is different from K_2, we find a suitable removable ear (single or double) and obtain G_{i-1} from G_i by removing that ear from G_i. For example, to show that every matching covered graph G has an ear decomposition, it suffices to show that every matching covered graph different from K_2 has a removable ear. Similarly, if we are trying to find an ear decomposition of a matching covered graph G with, say, at most d double ears, it is necessary to show that G has a removable ear Q such that the graph $G - Q$ has an ear decomposition with at most $d - t$ double ears, where $t = 0$, or 1, depending on whether or not Q is a single or a double ear. This is the motivation for the notion of a $(b+p)$-removable ear given below.

A removable ear Q of a matching covered graph G is $(b+p)$-*removable* if

$$(b+p)(G-Q) = \begin{cases} (b+p)(G) & \text{if } Q \text{ is a single ear, and} \\ (b+p)(G) - 1 & \text{if } Q \text{ is a double ear.} \end{cases}$$

In particular, a removable edge e of a matching covered graph G is $(b+p)$-removable, if $(b+p)(G-e) = (b+p)(G)$.

1.7.4 Canonical Ear Decompositions

In any ear decomposition, by definition, G_1 is K_2, and G_2 is an even circuit. It is not difficult to check that if G_3 is nonbipartite, then it must in fact be an odd subdivision of K_4, and if G_3 is bipartite and G_4 is nonbipartite, then G_4 must be an odd subdivision of $\overline{C}_6$. We shall refer an ear decomposition

$$G_1 \subset G_2 \subset \ldots \subset G_r = G$$

of a nonbipartite matching covered graph G as a *canonical ear decomposition* if either its third member G_3 or its fourth member G_4 is nonbipartite. The following fundamental theorem was proved by Lovász in 1983.

Theorem 1.7.7 ([11]) *Every nonbipartite matching covered graph G has a canonical ear decomposition.*

The following generalization of the above theorem is given in [3].

Theorem 1.7.8 *Every nonbipartite matching covered graph G has an optimal ear decomposition which is canonical.*

A lemma which was used in proving the above Theorem, and which we shall need in the next section is the following.

Lemma 1.7.9 *Let G be a matching covered graph. Let $C = \nabla(S)$ be a nontrivial tight cut of G. Suppose that $G_1 = G\{S, \overline{s}\}$ has a canonical ear decomposition with d double ears and that $G_2 = G\{\overline{S}, s\}$ is bipartite. Then, G also has a canonical ear decomposition with d double ears.*

A proof of the above Lemma can be found in [3].

1.8 Optimal Ear Decomposition

In this section we shall present the result that every matching covered graph admits an ear decomposition that uses exactly $(b + p)$ double ears. In fact, to prove this for bricks, it is more convenient to prove the following slightly more general result.

Theorem 1.8.1 *Let G be a near-brick. Then there is a canonical ear decomposition of G that uses precisely $(b + p)$ double ears.*

The proof of this Theorem follows immediately from Lemma 1.7.9 and Theorem 1.6.4.

1.8.1 Optimal Ear Decompositions of Matching Covered Graphs

Let G be a matching covered graph. If G is a brick or a brace, we have seen how to find optimal ear decompositions of G. Suppose that G is neither a brick nor a brace. Then it has nontrivial tight cuts. Unfortunately, there is no obvious way of obtaining an ear decomposition (much less an optimal ear decomposition) of G from arbitrary (optimal) ear decompositions of G_1 and G_2, where G_1 and G_2 are the two C-contractions of G with respect to a tight cut C of G. For example, if the last ear in an ear decomposition of G_1 is Q, and $E(Q) \cap C$ is not a removable edge in G_2, then Q will not be a removable ear of G. For this reason, it is convenient to have some flexibility in selecting the optimal ear decompositions of G_1 and G_2 in order for us to be able to combine them to obtain an optimal ear decomposition of G itself. The following theorem is motivated by the above consideration. It may be viewed as a strengthening of Theorem 1.6.4.

Theorem 1.8.2 *Every brick has two $(b+p)$-removable ears.*

Even with the aid of the above Theorem, if C is an arbitrary tight cut of G, it is not clear how to combine ear decompositions of G_1 and G_2. (In the notation described earlier, if C is an arbitrary tight cut of G, there is no obvious reason why $E(Q) \cap C$ should be removable in G_2.) However, it is possible to show that, if G is a matching covered graph which is neither a brick nor is bipartite, then one can choose a tight cut C of G which is either a 2-separation cut or a suitable barrier cut so that the above mentioned difficulty does not arise. This makes it possible to prove the following theorem [6].

Theorem 1.8.3 (Optimal Ear Decomposition Theorem) *The minimum number, $d_*(G)$, of double ears an ear decomposition of a matching covered graph G may have is equal to $b(G) + p(G)$.*

1.9 A Characterization of the Matching Lattice

We have noted that the difficult part of obtaining a characterization of the matching lattice consists of proving Theorem 1.4.9. We shall see how Theorem 1.8.1 can be used to prove Theorem 1.4.9. We first deal with matching covered graphs which are odd subdivisions of K_4 or $\overline{C}_6$ (by direct verification).

Theorem 1.9.1 *Let G be an odd subdivision of either K_4 or $\overline{C}_6$, and let w be any regular vector in $\mathbb{Z}^E$. Then, w is in the matching lattice of G.*

Theorem 1.9.2 *Let G be a matching covered graph with $b(G) = 1$ and $p(G) = 0$, let w be a regular vector in $\mathbb{Z}^E$. Then w is in the matching lattice of G. Moreover, if $\mathcal{D} = (G_1, G_2, \cdots, G_{r-1}, G_r = G)$ is an optimal canonical ear decomposition of G, and $\mathcal{M}(\mathcal{D}) = (M_1, M_2, \cdots, M_{r-1}, M_r)$ is a set of perfect matchings of G associated with $\mathcal{D}$, then w is an integer linear combination of incidence vectors of perfect matchings in $\mathcal{M}(\mathcal{D})$.*

Using similar (but more straightforward) arguments, it is easy to prove the following theorem.

Theorem 1.9.3 *Let G be a bipartite matching covered graph. Let w be a regular vector in $\mathbb{Z}^E$. Then w is in the matching lattice of G. Moreover, if $\mathcal{D} = (G_1, G_2, \cdots, G_{r-1}, G_r = G)$ is any ear decomposition of G, and $\mathcal{M}(\mathcal{D}) = (M_1, M_2, \cdots, M_{r-1}, M_r)$ is a set of perfect matchings of G associated with $\mathcal{D}$, then w is an integer linear combination of incidence vectors of perfect matchings in $\mathcal{M}(\mathcal{D})$.*

1.9.1 A Basis for the Matching Lattice

It is well-known that if $\mathcal{L}$ is any lattice generated by a set of integral vectors, then $\mathcal{L}$ has a basis consisting of integral vectors [20, Corollary 4.1b, page 47]. However, unlike the case with linear spaces, a generating set of a lattice need not contain a basis of the lattice. Murty[14] raised the question whether it is always possible to find a basis for the matching lattice of a matching covered graph G consisting solely of incidence vectors of perfect matchings of G. Henceforth, by a *basis* of a lattice $\mathcal{L}$, we mean a linearly independent set $\{a_1, \cdots, a_k\}$ of vectors in $\mathcal{L}$ such that every element a in $\mathcal{L}$ may be expressed as

$$a = \lambda_1 a_1 + \cdots + \lambda_k a_k,$$

where the coefficients $\lambda_1, \cdots, \lambda_k$ are all integers.

Theorem 1.9.4 *The matching lattice of a matching covered graph G is of dimension $m - n + 2 - b$ and has a basis consisting of incidence vectors of perfect matchings.*

1.9.2 A Characterization of the Matching Lattice

A collection of cuts is *laminar* if no two of its cuts cross. Two cuts $\nabla(X)$ and $\nabla(Y)$ *cross* if each of $X \cap Y$, $X \cap \overline{Y}$, $\overline{X} \cap Y$ and $\overline{X} \cap \overline{Y}$ is nonnull. Using splitting along nontrivial tight cuts of G and ad hoc observations in the Petersen graph, it is then possible to prove the following result, originally proved by Lovász:

Theorem 1.9.5 *Let G be a matching covered graph, $\mathcal{C}$ a maximal laminar collection of nontrivial tight cuts of G. A vector w in $\mathbb{Z}^E$ lies in $Lat(G)$ if and only if it satisfies the following properties, for each brick and brace H obtained by a tight cut decomposition of G using the cuts in $\mathcal{C}$:*

(i) *the restriction of w to H is regular, and*

(ii) *if the underlying simple graph of H is the Petersen graph then, for any pentagon C of H, $w(C)$ is even.*

As a Corollary of the above Theorem, one then derives the following generalization of Theorem 1.3.4.

Theorem 1.9.6 *Let G a matching covered graph, $\mathcal{C}$ a maximal laminar collection of tight cuts of G, w a vector in $\mathbb{Z}^E$. Vector $2w$ lies in $Lat(G)$ if and only if w is regular over the set of cuts in $\mathcal{C}$.*

1.10 Unsolved Problems

We conclude this survey with a list of three of the most attractive problems concerning matching covered graphs.

Problem 1. Determining the characteristic of a matching covered graph: The notion of the characteristic of a matching covered graph has played a central role in our work. But as yet we do not know if there exists a polynomial algorithm for determining the value of this parameter. The characteristic of a matching covered graph is the minimum of the characteristics of its bricks and braces, and all braces have characteristic ∞. Thus, in view of Theorem 1.5.4, the problem boils down to determining whether a given brick is solid. We do not even know if this decision problem is in $\mathcal{NP}$.

Problem 2. Finding a good lower bound for the number of perfect matchings in a matching covered graph: Clearly the number of perfect matchings in a matching covered graph is at least the dimension of its matching lattice. There are a number of graphs for which these two numbers coincide; we refer to such graphs as *extremal graphs*. For example, every odd wheel is extremal. The dimension of the matching lattice of a cubic brick on $2n$ vertices is $n+1$. Using Theorem 1.6.4, we have been able to determine all extremal cubic matching covered graphs (they all have fewer than eighteen vertices).

Lovász and Plummer have conjectured that there exist constants $c_1 > 0$ and $c_2 > 1$, such that every 2-connected cubic graph on $2n$ vertices has at least $c_1 c_2^n$ perfect matchings. Our result mentioned above says that, for $n \geq 9$, the number of perfect matchings in a cubic brick on $2n$ vertices is at least $n+2$. Insignificant though it is, this is the best lower bound we know for the number of perfect matchings in cubic bricks.

Problem 3. Characterizing the integer cone of a matching covered graph: As noted in section 3, a 2-connected cubic graph G is 3-edge-colourable if and only if $\underline{\mathbf{1}}$ is in $Int.Con(G)$. Since the problem of determining the edge-chromatic number of a cubic graph is $\mathcal{NP}$-complete, one cannot expect to be able to find a good characterization of the integer cone of a matching covered graph. Nevertheless, there may be special classes of graphs for which this is feasible. For example, it is easy to show that if G is a bipartite matching covered graph, then a non-negative integer vector w is in $Int.Con(G)$ if and only if it is regular. Generalizing this result, the second author has recently shown that if G is any solid matching covered graph, then a non-negative integer vector w is in $Int.Con(G)$ if and only if it is in $Lat(G)$.

Clearly, for any matching covered graph G,

$$Int.Con(G) \subseteq Rat.Con(G) \cap \mathbb{Z}^{E}_{\geq 0}.$$

As a generalization of Tutte's conjecture, it has been suggested by Seymour [21] that equality holds for every matching covered graph that does not contain the Petersen graph as a minor. (We learnt of this conjecture, in this form, from Lovász, through a private communication from Vempala). By the four-colour theorem, this statement is true for cubic planar graphs. It is not known whether it is true for all planar matching covered graphs.

References

[1] C. N. Campos and C. L. Lucchesi.On the characteristic of a separating cut in a matching covered graph.Technical Report 22, Institute of Computing, University of Campinas, Brazil, 2000.

[2] M. H. de Carvalho.*Decomposição Ótima em Orelhas para Grafos Matching Covered.*PhD thesis, Institute of Computing–University of Campinas, Brazil, 1997.In Portuguese.

[3] M. H. de Carvalho, C. L. Lucchesi, and U. S. R. Murty.Ear decompositions of matching covered graphs.*Combinatorica*, 19:151–174, 1999.

[4] M. H. de Carvalho, C. L. Lucchesi, and U. S. R. Murty.On a conjecture of Lovász concerning bricks. I. The characteristic of a matching covered graph.Submitted for publication, 1999.

[5] M. H. de Carvalho, C. L. Lucchesi, and U. S. R. Murty.On a conjecture of Lovász concerning bricks. II. Bricks of finite characteristic.Submitted for publication, 1999.

[6] M. H. de Carvalho, C. L. Lucchesi, and U. S. R. Murty.Optimal ear decompositions of matching covered graphs.Submitted for publication, 1999.

[7] J. Edmonds, L. Lovász, and W. R. PulleyblanK.Brick decomposition and the matching rank of graphs.*Combinatorica*, 2:247–274, 1982.

[8] D. R. Fulkerson.Blocking and antiblocking pairs of polyhedra.*Math. Programming*, 1:168–194, 1971.

[9] A. M. H. Gerards, L. Lovász, K. Truemper, A. Schrijver, P. Seymour, and S. Shih.Regular matroids from graphs.Under preparation.

[10] P. Hall.On representatives of subsets.*J. London Math. Soc.*, 10:26–30, 1935.

[11] L. Lovász.Ear decompositions of matching covered graphs.*Combinatorica*, 3:105–117, 1983.

[12] L. Lovász.Matching structure and the matching lattice.*J. Combin. Theory (B)*, 43:187–222, 1987.

[13] L. Lovász and M. D. Plummer.*Matching Theory*.Number 29 in Annals of Discrete Mathematics. Elsevier Science, 1986.

[14] U. S. R. Murty.The matching lattice and related topics.Technical report, University of Waterloo, 1994.Preliminary Report.

[15] N. Robertson, D. Sanders, P. D. Seymour, and R. Thomas.The four-colour theorem.*J. Combin. Theory (B)*, pages 2–44, 1997.

[16] N. Robertson, P. D. Seymour, and R. Thomas.Excluded minors in cubic graphs.Manuscript.

[17] N. Robertson, P. D. Seymour, and R. Thomas.Tutte's edge-coloring conjecture.*J. Combin. Theory (B)*, pages 166–183, 1997.

[18] D. Sanders, P. D. Seymour, and R. Thomas.Edge three-coloring cubic doublecross graphs.Manuscript.

[19] D. Sanders and R. Thomas.Edge three-coloring cubic apex graphs.Manuscript.

[20] A. Schrijver.*Theory of Linear and Integer Programming*.Wiley, 1986.

[21] P. D. Seymour.On multicolourings of cubic graphs and conjectures of Fulkerson and Tutte.*Proc. London Math. Soc. Series 3*, 38:423–460, 1979.

[22] Z. Szigeti.Perfect matchings versus odd cuts.submitted for publication, November 1998.

[23] P. G. Tait.Note on a theorem in geometry of position.*Trans. Roy. Soc. of Edinburgh*, pages 657–660, 1880.

[24] W. T. Tutte.The factorizations of linear graphs.*J. London Math. Soc.*, 22:107–111, 1947.

[25] W. T. Tutte.On the algebraic theory of graph colorings.*J. Combin. Theory*, 1:15–50, 1966.

2

Discrete Mathematics and Radio Channel Assignment

C. McDiarmid

2.1 Introduction

The following generalization of graph colouring arises naturally in the study of channel assignment for cellular radiocommunications networks.

Given a graph G and a length $l(uv)$ for each edge uv of G, determine the least positive integer t (the 'span') such that the nodes of G can be assigned channels (or colours) from $1, ..., t$ so that for every edge uv, the channels assigned to u and v differ by at least $l(uv)$. The nodes correspond to transmitter sites, and the lengths $l(uv)$ specify minimum channel separations to avoid interference. This 'constraint matrix' model provides the central focus of the chapter.

The plan of the chapter is as follows. We start by giving a brief introduction to the constraint matrix model which we have just met. Then we present a variety of general results about this model, for example giving bounds on the span of channels required which are natural extensions of well known results about graph colouring. We also discuss briefly the related T-colouring model. This general discussion is followed by a short section on the difficulty of finding the span, where we discuss bipartite and nearly bipartite graphs and graphs of bounded tree-width.

There follow three substantial sections which focus on three aspects of the constraint matrix model. First we consider the natural special case where the transmitter sites are located in the plane, and the required minimum separation between channels assigned to two sites depends on the distance between them. We are led to consider unit disk graphs, and more generally to consider frequency-distance models. Next, we introduce demands into the picture. When all required channel separations are 0 or 1, and there are large numbers of channels demanded at the sites, we find that we are led into the world of imperfect graphs. After that we consider two sorts of random models for channel assignment, one set in the plane, and one a natural generalisation of the usual random graphs.

In each of these sections, at some stage we let some parameter tend to infinity in order to allow analysis and reveal structure: the parameters correspond to the minimum channel re-use distance, the maximum demand, and the number of sites. Also in each of these sections, we focus on the ratio of chromatic number to clique number or generalisations of this idea.

Finally, we close by giving a fuller story concerning the modelling of the radio channel assignment problem, and set the constraint matrix model (and the T-colouring model) in a more general framework.

There has been a flood of work recently on applying heuristic methods such as simulated annealing and tabu search to attack channel assignment problems. See for example [42] for a particularly successful approach, and see [65] for a recent review and for further references. We discuss here the mathematical ideas that inform and guide such approaches but we do not discuss the methods themselves.

2.2 The constraint matrix model

Let $V = \{v_1, \ldots, v_n\}$ be a set of n transmitter sites. We are given a graph $G = (V, E)$ on the sites, the *interference* or *constraint* graph, together with a non-negative integer length $l(e)$ for each edge e. An assignment $\phi : V \to \{1, \ldots, t\}$ is *feasible* if $|\phi(u) - \phi(v)| \geq l(uv)$ for each edge uv. Here we use uv to denote the undirected edge between u and v. The idea is that if sites are close together then they must use widely separated channels.

The *span* of the problem, $\text{span}(G, l)$, is the least t such there is a feasible assignment. (Some authors call $t - 1$ the span.) We want to determine or approximate the span, and find corresponding assignments. Note that if $\mathbf{1}$ denotes the appropriate all 1's function, then $\text{span}(G, \mathbf{1})$ equals the chromatic number $\chi(G)$. Also, with any positive edge lengths the least *number* of colours required is just $\chi(G)$, but it is the span that is of interest to us.

Sometimes more than one channel may be required at a site, and then it is natural to phrase the problem in terms of a 'constraint matrix'. In the *constraint matrix with demands* model, we are given a graph on V with edge-lengths $l(e)$ as before, and a co-site constraint value $c(v) \geq 1$ for each node v. Equivalently, we are given an $n \times n$ symmetric matrix A (the constraint matrix) of non-negative integers with off-diagonal entries the edge lengths $l(uv)$ (or 0 for non-edges) and diagonal entries $c(v) \geq 1$.

We are also given a demand vector $\mathbf{x}$, which is an n-vector of non-negative integers x_v, which specifies how many channels are needed at each site v. A *feasible assignment* ϕ is a family $(\phi(v) : v \in V)$ where for each $v \in V$ the set $\phi(v)$ contains x_v positive integers such that the following condition holds: for each distinct $u, v \in V$ and each $i \in \phi(u)$ and $j \in \phi(v)$ we have $|i - j| \geq l(uv)$, and for each $v \in V$ and each distinct $i, j \in \phi(v)$ we have

$|i-j| \geq c(v)$. The diagonal entries $c(v)$ typically are the largest. We may denote the span by $\text{span}(A, \mathbf{x})$.

Examples

1. If G is a triangle with each edge of length 3, then the span is 7. More generally, if G is the complete graph K_n and each edge length is k then the span is $k(n-1)+1$ – see Proposition 2.3.1.
2. Is G is the 4-cycle C_4 with each edge length 3, then the span is 4. More generally if G is any bipartite graph then the span is 1+ the maximum edge length – see Proposition 2.4.1.
3. Let G consist of a triangle with each edge of length 1, together with a pendant edge of length 2 attached to each of the nodes. Then the span is 4.
4. Let G be the 5-cycle C_5, let each edge length be 1 and each co-site constraint value be 2, and let each node have demand 2. Then the span is 5.

2.3 General results for the constraint matrix model

In this section we give various results, some introductory, about the span in the constraint matrix model. We restrict our attention here to the case of unit demands.

2.3.1 All equal edge lengths

When the edge lengths are all the same, we are almost back to colouring. The following result was perhaps first shown in [71]. Let $\mathbf{1}$ denote the appropriate all 1's function.

Proposition 2.3.1 *If each edge length is k then*

$$\text{span}(G, k\mathbf{1}) = k(\chi(G)-1)+1.$$

Proof. Observe that the span is at most the right hand side, since we could always first colour G with $\chi(G)$ colours and then assign a channel to each colour, using channels $1, k+1, \ldots, k(\chi(G)-1)+1$.

Now let us show that the span is at least the right hand side. Let t be the span, and consider a feasible assignment ϕ using channels $0, 1, \ldots, t-1$ which uses as few as possible channels which are not multiples of k. Then in fact ϕ must use only multiples of k, for otherwise the least channel not a multiple of k could be pushed down to the nearest multiple of k, giving a contradiction. But now if we let $c(v) = \phi(v)/k$ we obtain a (proper) colouring of G, and so $\chi(G) \leq (t-1)/k + 1$, which yields the desired inequality. □

2.3.2 *Lower bounds for the span*

It follows from Proposition 2.3.1 that if G is the complete graph K_n and all edge lengths are at least k then

$$\operatorname{span}(G,l) \geq k(n-1)+1. \tag{2.1}$$

This result may be extended as follows, see [44, 81]. Since we allow the length of an edge to be 0, we could always assume that the graph G is complete, though usually this is not helpful.

Proposition 2.3.2 *If G is complete, then*

$$\operatorname{span}(G,l) \geq \operatorname{hp}(G,l),$$

where $\operatorname{hp}(G,l)$ *is the minimum length of a hamiltonian path.*

Proof. Given a feasible assignment ϕ, list the nodes as $v_1,\ldots,v_n$ so that $\phi(v_1) \leq \phi(v_2) \cdots \leq \phi(v_n)$. This gives a hamiltonian path in G, and

$$\phi(v_n) - \phi(v_1) = \sum_{i=1}^{n-1} \phi(v_{i+1}) - \phi(v_i) \geq \sum_{i=1}^{n-1} l(v_i v_{i+1}),$$

which is the length of the path. □

This last result has the drawback that it is NP-hard to calculate $\operatorname{hp}(G,l)$, but there are good lower bounds which may be efficiently calculated, for example the minimum length of a spanning tree. Observe that Proposition 2.3.2 is tight if the edge-lengths satisfy the triangle inequality, but we should not expect this to hold for minimum channel separations.

Since we can apply the last two bounds on the span to any complete subgraph of a graph, we may think of them as extending the lower bound that $\chi(G) \geq \omega(G)$. Now let us consider another lower bound on $\chi(G)$. The *stability number* (or independence number) $\alpha(G)$ is the maximum size of a stable set in G. We have

$$\chi(G) \geq |V|/\alpha(G). \tag{2.2}$$

The inequality (2.2) can be extended as follows. For each node v let α_v denote the maximum size of a stable set containing v. Then

$$\chi(G) \geq \sum_v 1/\alpha_v. \tag{2.3}$$

For, given any proper k-colouring of G, with colour sets $S_1, \ldots, S_k$, we have $\alpha_v \geq |S_i|$ if $v \in S_i$, and so

$$\sum_v 1/\alpha_v = \sum_{i=1}^{k} \sum_{v \in S_i} 1/\alpha_v \leq \sum_{i=1}^{k} \sum_{v \in S_i} 1/|S_i| = k.$$

There are lower bounds for the span extending these ideas. Let m be a positive integer, and let us keep m fixed throughout. Consider an instance G, l of the constraint matrix problem. Call a subset U of nodes *m-assignable* if the corresponding subproblem has span at most m. Let α^m denote the maximum size of an m-assignable set. Similarly, for each node v let α_v^m denote the maximum size of an m-assignable set containing v. Then

$$\operatorname{span}(G, l) \geq m|V|/\alpha^m \ - (m-1), \tag{2.4}$$

and indeed ([81])

$$\operatorname{span}(G, l) \geq m \sum_v 1/\alpha_v^m - (m-1). \tag{2.5}$$

It is perhaps most natural to prove these results (2.4) and (2.5) using weak LP duality (exercise!), but that approach does not seem easily to give the following slight extension of (2.5).

Let the index i always run through $1, \ldots, m$. For each node v and each i, let α_{vi}^m denote the maximum size of an m-assignable set U containing v, such there is a feasible assignment $\phi : U \to \{1, \ldots, m\}$ with $\phi(v) = i$. For example, if G is the path with three nodes u, v, w (v in the middle) and both edges of length 2, then

$$\alpha_v^3 = \alpha_{v1}^3 = \alpha_{v3}^3 = 3 \ \text{ and } \ \alpha_{v2}^3 = 1.$$

Proposition 2.3.3

$$\operatorname{span}(G, l) \geq \sum_v \sum_i 1/\alpha_{vi}^m - (m-1). \tag{2.6}$$

Observe that $\alpha_{vi}^m \leq \alpha_v^m$, and so the bound (2.6) is always at least as good as (2.5).

Proof. Let $t = \operatorname{span}(G, l)$, and fix a feasible assignment $\phi : V \to \{1, \ldots, t\}$. For each set I of integers let $\hat{I}$ denote $\phi^{-1}(I)$. For each v and i let I_{vi} denote the set $\{\phi(v) - i + 1, \ldots \phi(v) + m - i\}$ of m consecutive integers, and let $\beta_{vi} = |\hat{I_{vi}}|$. Then $1 \leq \beta_{vi} \leq \alpha_{vi}^m$. Let $\mathcal{I}$ denote the collection of sets $I = \{j, \ldots, j + m - 1\}$ of m consecutive integers such that $\hat{I} \neq \emptyset$. Then $|\mathcal{I}| \leq t + m - 1$. Hence

$$\begin{aligned}
\sum_v \sum_i 1/\alpha_{vi}^m \ &\leq \ \sum_v \sum_i 1/\beta_{vi} \\
&= \ \sum_v \sum_i \sum_{I \in \mathcal{I}} \mathbf{1}_{(I = I_{vi})} (1/|\hat{I}|)
\end{aligned}$$

$$= \sum_{I\in\mathcal{I}} (1/|\hat{I}|) \sum_{v\in\hat{I}} \sum_i \mathbf{1}_{(I=I_{vi})}.$$

But for each $v \in \hat{I}$ we have $\sum_i \mathbf{1}_{(I=I_{vi})} = 1$, and so the last quantity above equals

$$\sum_{I\in\mathcal{I}} (1/|\hat{I}|) \sum_{v\in\hat{I}} 1 = \sum_{I\in\mathcal{I}} 1 = |\mathcal{I}| \leq t + m - 1.$$

□

2.3.3 Span and orientations

The Gallai-Roy Theorem (see for example [88]) relates the chromatic number $\chi(G)$ to the maximum length of a path (with no repeated nodes allowed) in an orientation of G. The theorem states that if D is an orientation of G with maximum path length $\varrho(D)$, then

$$\chi(G) \leq 1 + \varrho(D);$$

and further, equality holds for some acyclic orientation D. This theorem extends directly to the weighted graph case, that is to constraint matrix problems - see [4] which discusses the acyclic case and related algorithms.

Proposition 2.3.1 *Given (G, l) and an orientation D of G, let $\varrho(D, l)$ denote the maximum length of a path. Then*

$$\text{span}(G, l) \leq 1 + \varrho(D, l);$$

and further, equality holds for some acyclic orientation D.

Observe that if G is complete, then an acyclic orientation D yields a hamiltonian path, and so $\varrho(D, l) \geq \text{hp}(G, l)$: thus the 'equality part' of Proposition 2.3.1 extends the lower bound given in Proposition 2.3.2.

Proof. List the arcs of D in non-increasing order of length. Form a maximal acyclic subdigraph D' of D, by running through the list of arcs, and including an arc whenever it does not create a cycle. For each node v let $\phi(v)$ be the maximum length of a path in D' ending at v.

Observe that if there is a path Q in D' from u to v of length d then $\phi(v) \geq \phi(u) + d$; for since D' is acyclic, if we start with a maximum length path P in D' ending at u we can continue along the path Q without repeating a node.

Consider an arc uv of D. If it is in D', then the above observation gives $\phi(v) \geq \phi(u) + l(uv)$. If uv is not in D', then there is a path Q in D' from v to u which consists of arcs each of length at least $l(uv)$, and which thus has length at least $l(uv)$: hence the observation gives $\phi(u) \geq \phi(v) + l(uv)$. This shows that ϕ is a feasible assignment, taking values in $\{0, 1, \ldots, \varrho(D, l)\}$. Hence $\text{span}(G, l) \leq 1 + \varrho(D, l)$, as required.

For the last part, let ϕ be an optimal assignment. If nodes u and v are adjacent in G, orient the edge from u to v if $\phi(u) < \phi(v)$. Call the resulting acyclic orientation D. Consider any path $v_1, v_2, \ldots, v_k$ in D. Since ϕ increases along the path, we may argue as in the proof of Proposition 2.3.2 to see that

$$\sum_{i=1}^{k-1} l(v_i v_{i+1}) \leq \sum_{i=1}^{k-1} (\phi(v_{i+1}) - \phi(v_i)) = \phi(v_k) - \phi(v_1),$$

and so the path has length at most $\mathrm{span}(G,l) - 1$. Hence $1 + \varrho(D,l) \leq \mathrm{span}(G,l)$, which completes the proof. □

2.3.4 Sequential assignment methods

Suppose that we want to colour the nodes of a graph with colours $1, 2, \ldots$, and we have a given ordering on the nodes. Let us consider two variants of the greedy colouring algorithm. In the 'one-pass' method, we run through the nodes in order and always assign the smallest available colour. In the 'many-passes' method, we run through the nodes assigning colour 1 whenever possible, then repeat with colour 2 and so on. Both methods yield exactly the same colouring, and show that

$$\chi(G) \leq \Delta(G) + 1, \tag{2.7}$$

since at most $\Delta(G)$ colours are ever denied to a node.

Now consider a constraint matrix problem (G,l). Define the *weighted degree* of a node v by $\deg_l(v) = \sum\{l(uv) : uv \in E\}$, and define the *maximum weighted degree* by $\Delta_l(G) = \max_v \deg_l(v)$. The above greedy methods generalise immediately [61].

Example

Let G be the 4-cycle C_4, with nodes a, b, c, d and edge lengths $l(ab) = 1$ and $l(bc) = l(cd) = l(ad) = 2$. Note that $\Delta_l = 4$. The one-pass method assigns channels 1,2,4,6 to the nodes a, b, c, d respectively, with span 6. The many-passes method assigns channel 1 to nodes a and c, channel 2 to none of the nodes, and channel 3 to nodes b and d, with span 3.

In fact the many passes method always uses a span of at most $\Delta_l + 1$, and so we may extend (2.7) as follows.

Proposition 2.3.2

$$\mathrm{span}(G,l) \leq \Delta_l(G) + 1.$$

Proof. In order to show that the many passes method needs a span of at most the above size, suppose that it is about to assign channel c to node v. Let A be the set of neighbours u of v to which it has already assigned a channel $\phi(u)$. For each channel $j \in \{1, \ldots, c-1\}$ there must be a node $u \in A$ with $\phi(u) \leq j$ and $\phi(u) + l(uv) \geq j + 1$. Hence the intervals $\{\phi(u), \ldots, \phi(u) + l(uv) - 1\}$ for $u \in A$ cover $\{1, \ldots, c-1\}$. Thus

$$c - 1 \leq \sum_{u \in A} l(uv) \leq \deg_l(v) \leq \Delta_l(G),$$

and this completes the proof. □

There is a straightforward extension of (2.7), involving the 'degeneracy' of a graph – see for example [88]. Given an ordering $\sigma = (v_1, \ldots, v_n)$ of the nodes, let $g(\sigma)$ be the maximum over $1 < j \leq n$ of the degree of node j in the subgraph induced by nodes $1, \ldots, j$. We call the minimum value of $g(\sigma)$ over all such orderings σ the *degeneracy* of G, and denote it by $\delta^*(G)$. We can compute $\delta^*(G)$ as follows. Find a node v of minimum degree, delete it and put it at the end of the order, and repeat. This shows that $\delta^*(G)$ equals the maximum over all induced subgraphs of the minimum degree, and that we can compute it and find a corresponding order in $O(n^2)$ steps.

If we colour the nodes of G in an order yielding the minimum above, then at each stage at most $\delta^*(G)$ colours are denied to a node. Hence

$$\chi(G) \leq \delta^*(G) + 1, \tag{2.8}$$

and further we can find a corresponding colouring quickly. (The quantity $\delta^*(G) + 1$ is sometimes called the *colouring number* of G.)

Does this result extend to span(G, l)? The answer is 'not well', since the colouring method which yields the inequality (2.8) above does just what we avoided earlier, namely it considers the nodes in order and colours one after another. Consider the example where G consists of a triangle with one edge of length 2 and two of length 1 adjacent to a node v, and one pendant edge of length 2 attached to this node v: the span is 4, but in each induced subgraph there is a node with weighted degree at most 2. However, the inequality (2.8) does extend if we replace the degree of each node v not by its weighted degree $\deg_l(v)$ but by the sum of the values $2l(uv) - 1$ over all the nodes $u \neq v$ with $l(uv) \geq 1$. For, observe that if we have a feasible assignment for the graph without v and we wish to extend it to v, then the above sum bounds the number of channels denied to v – see Proposition 6 of [81].

How well do related upper bounds or other results on $\chi(G)$ extend to the constraint matrix case? In particular, when can we save the +1 in (2.7) as in Brooks' Theorem? Is there any analogue of the Hajnal-Szemerédi theorem that a graph G has a $\Delta + 1$ colouring in which the colour sets differ in size by at most 1? What about Wilf's result that $\chi(G) \leq \lambda(G) + 1$, where $\lambda(G)$ is the maximum eigenvalue of the adjacency matrix (this result follows from (2.8)). Is there any analogue of the Hajós construction? For all these, see for example [88].

2.3.5 An IP model

The following integer programme (IP) gives a simple reformulation of the constraint matrix model, though other formulations may be better suited to computations for particular types of problem, see also [65].

Choose an upper limit $f_{\max}$, and let $F = \{1, \ldots, f_{\max}\}$ be the set of available channels. We let u and v run through the node set V, and let i and j run through F. We introduce a binary variable y_{ui} for each transmitter u and channel i: setting $y_{ui} = 1$ will correspond to assigning channel i as one of the channels at transmitter u. Then $\text{span}(A, \mathbf{x})$ is given by the following integer programme.

$\min z$ subject to

$$\begin{array}{lcll} z & \geq & j\ y_{vj} & \forall v, j \\ \sum_j y_{vj} & = & x_v & \forall v \\ y_{ui} + y_{vj} & \leq & 1 & \forall ui \neq vj \text{ with } |i-j| < a_{uv} \\ y_{vj} & \in & \{0,1\} & \forall v, j \end{array}$$

When we write the shorthand

$$\forall ui \neq vj \text{ with } |i-j| < a_{uv}$$

above, we mean

$$\forall u, v \in V \text{ and } i, j \in F \text{ such that } (u,i) \neq (v,j) \text{ and } |i-j| < a_{uv}.$$

To see that this IP formulation is correct, consider an optimal assignment $\phi : V \to F$. Run through the transmitters $v \in V$ and the channels $j \in F$, and set $y_{vj} = 1$ if $j \in \phi(v)$ and $y_{vj} = 0$ otherwise; and set z to be the maximum channel used. It is easy to see that this gives a feasible solution to the IP, with $z = \text{span}(A, \mathbf{x})$. Conversely, given a feasible solution to the IP with value t, we may obtain in a similar way a feasible assignment $\phi : V \to \{1, \ldots, t\}$.

2.3.6 Counting feasible assignments

Given a graph G, for each positive integer t let $f(t)$ be the number of (proper) t-colourings of G. Thus for example if G consists of two adjacent nodes then $f(t) = t(t-1)$. It is well known and easy to see that there is a unique polynomial $p(t)$ defined for all real t which agrees with f on the positive integers: this is the *chromatic polynomial* of G.

Does this result extend to the constraint matrix problem? Let G be a graph with n nodes, and with edge lengths as usual. For each positive integer t let $f(t)$ be the number of feasible assignments from V to $\{1, \ldots, t\}$.

For example let G consist of two adjacent nodes u and v with $l(uv) = 3$. Then it is easy to check that $f(t)$ agrees with the polynomial $p(t) = (t-2)(t-3)$ for each $t \geq 2$, but $f(1) = 0$ which does not agree with $p(t)$. Thus there is no 'feasible assignment counting polynomial'. However, there is nearly one.

Theorem 2.3.4 *There is a monic polynomial $p(x)$ of degree n such that $f(t) = p(t)$ for all sufficiently large integers t. Indeed, if the maximum edge length is k then this is true for all $t > (n-1)k$.*

This result was shown independently in [87] by methods based on counting hyperplane arrangements, and in [56] by elementary methods. See also these papers for extensions of the above result.

2.3.7 Cyclic channel distances

Since the available channels are evenly spaced in the spectrum, we have taken them to be the consecutive integers $1, 2, \ldots, t$ or $0, 1, \ldots, t-1$ for some t. Sometimes it is convenient to 'wrap the channels around a circle', and work with 'cyclic channel distance' – see for example [39]. For $i, j \in \{0, 1, \ldots, t-1\}$ let

$$d_t(i,j) = \min\{|i-j|, t - |i-j|\}.$$

We say that an assignment $\phi : V \to \{0, 1, \ldots, t-1\}$ is *t-cyclically-feasible* if the usual constraints are satisfied when we use the cyclic channel distance d_t as above. (Thus we are imposing more constraints than before.) The least t for which there is such an assignment is the *cyclic span* of the problem. Observe that the cyclic span is at least the span and at most the span $+(k-1)$, where k denotes the maximum constraint value.

There are two reasons to work with cyclic channel distances. Firstly, as noted in [17], if an assignment ϕ is t-cyclically feasible for unit demands, then we can satisfy demand $x+1$ at each node in a very straightforward manner, by assigning channels $\phi(v), \phi(v)+t, \ldots, \phi(v)+xt$ to each node v. Secondly, cyclic channel distances are sometimes mathematically more tractable, as there are no 'end effects'. For example, suppose that G is bipartite (with at least one edge), each edge length is 1 and each co-site constraint value is 2. If we have a (non-zero) demand vector with maximum entry $x_{\max}$ then it is easy to see that the cyclic span is $2x_{\max}$. For clearly this is a lower bound, and we can assign even channels from $\{0, 2, \ldots, 2x_{\max}-2\}$ to the nodes in one part and odd channels from $\{1, 3, \ldots, 2x_{\max}-1\}$ to the nodes in the other part. In the usual linear case, we need to think more about this problem – see section 2.6.3.

Cyclic channel distances are related to the *circular chromatic number* (originally called the star chromatic number) of a graph G. This may be defined as the infimum of the values t/k such there is a t-cyclically-feasible assignment for G with each edge length k, see for example [45] and the references therein. We shall not discuss cyclic channel distances further here, but see for example [11, 79, 39, 55].

2.3.8 Graph distance between sites

Given a graph G and positive integer k, let $G^{(k)}$ denote the graph with the same vertices as G, and with distinct vertices u and v adjacent whenever their distance in G is at most k. [The graph distance between u and v is the least number of edges in a path joining them.] Thus $G^{(1)}$ is just G.

Consider the triangular lattice T in the plane, with minimum distance 1, as described in section 2.5.2 below. If we join two points T when their Euclidean distance is 1, we obtain the infinite 6-regular graph G_T. Similarly from the square lattice S we obtain the 4-regular graph G_S. The following result from [39, 63] concerns the chromatic number χ and the clique number ω of the graphs $G_T^{(k)}$ and $G_S^{(k)}$.

Theorem 2.3.5 *For each positive integer k, the graph G_T of the triangular lattice satisfies*

$$\chi(G_T^{(k)}) = \omega(G_T^{(k)}) = \lceil \frac{3}{4}(k+1)^2 \rceil,$$

the graph G_S of the square lattice satisfies

$$\chi(G_S^{(k)}) = \omega(G_S^{(k)}) = \lceil \frac{1}{2}(k+1)^2 \rceil.$$

There has been much related work concerning graph distance. For example, an $L(2,1)$-*labelling* or *radio colouring* of a graph G is an assignment such that channels assigned to adjacent nodes differ by at least 2 and channels assigned to nodes at distance 2 are distinct. Thus it is a feasible assignment for the graph $G^{(2)}$, where each edge from G has length 2, and each 'new' edge has length 1.

We shall be very interested in the Euclidean distance between points (sites) in the plane, but we shall not discuss graph distance further here, see [10, 20, 31, 39, 38, 35, 73, 75, 79].

2.3.9 The T-colouring model

We start by introducing the rather general T_e-sets model. We are interested in two specialisations of this model. One is the now familiar constraint matrix problem, which has proved fruitful in terms of providing a model both useful to engineers and tractable for mathematicians. The other specialisation involves T-colourings of graphs. This topic took its motivation from radio channel assignment, and set off from there to generate some attractive mathematics. We shall discuss this topic very briefly.

The *T_e-sets model* is specified by a constraint graph $G = (V, E)$ together with a set T_e for each edge e of G, where always $0 \in T_e$. The sets T_e contain the 'forbidden differences'. An assignment ϕ is *feasible* if for each distinct $u, v \in V$ we have $|\phi(u) - \phi(v)| \notin T_{uv}$. As before we are interested in the span.

When each set T_e is of the form $\{0, 1, \ldots, l(e)\}$ we are back to the constraint matrix model. In the *T-colouring model*, we are given a single set T, that is each set $T_e = T$. The idea is not to insist that the forbidden differences are as in the constraint matrix model, and thus to allow for interference caused by phenomena such as intermodulation products (see section 2.8), but to specify only one such forbidden set T, in the interests of mathematical tractability rather than practical use. To the mathematician this is of course a natural problem to extract from the general T_e-sets model, and there is some practical interest in this case, see for example [18].

Let us denote the span by $\text{span}_T(G)$. Observe that

$$\text{span}_T(G) \leq \text{span}_T(K_{\chi(G)}),$$

since we could always first colour G with $\chi(G)$ colours and then assign a channel to each colour. Observe also that

$$\text{span}_T(K_n) \leq |T|(n-1)+1.$$

For, if we assign channels to the nodes one after another, when we come to assign a channel to the ith node at most $|T|(i-1)$ channels are forbidden. [Indeed, we may extend the inequality in Proposition 2.3.2 if we replace $l(uv)$ in the definition of the weighted degree by $|T_{uv}|$.] From the last two inequalities we have [84]

$$\text{span}_T(G) \leq \text{span}_T(K_{\chi(G)}) \leq |T|(\chi(G)-1)+1$$

for any set T. In the special case when $T = \{0, 1, \ldots, k-1\}$ for some positive integer k, Proposition 2.3.1 shows that the last inequalities hold at equality throughout. A central focus in the theory of T-colourings is to investigate for which sets T (always containing 0) is it true that

$$\text{span}_T(G) = \text{span}_T(K_{\chi(G)})$$

for every graph G; that is, that $\text{span}_T(G)$ is determined by $\chi(G)$. We have just seen that this is true when $T = \{0, 1, \ldots, k-1\}$: for many further examples see [73, 65] and the references therein.

2.4 How hard is channel assignment?

We noted earlier that the special case when all lengths are 1 is essentially the graph colouring problem. Since graph colouring is NP-hard – see for example [19] – we cannot expect an easy ride. Indeed, it is hard even to approximate the chromatic number $\chi(G)$: if $P \neq NP$ then no polynomial time algorithm can guarantee to colour an n-node graph with at most $n^{\frac{1}{7}-\varepsilon}\chi(G)$ colours for any fixed $\varepsilon > 0$, see for example [3]. In fact the general problem seems to be harder than graph colouring – see below and see section 2.6.2.

2.4.1 Bipartite graphs and odd cycles

Bipartite graphs are easy. For any graph G clearly $\mathrm{span}(G, l) \geq L$, where $L = \max\{l(xy) + 1 | xy \in E(G)\}$.

Proposition 2.4.1 *If G is bipartite, then* $\mathrm{span}(G, l) = L$.

Proof. If we set $\phi(x) = 1$ for x in one part of the bipartition and $\phi(x) = L$ for x in the other part, then we obtain a feasible assignment with span L. □

After bipartite graphs the next thing to consider is odd cycles. Here again it is easy to determine the span.

Proposition 2.4.2 *If G is an odd cycle then* $\mathrm{span}(G, l) = \max(L, M)$, *where* $M = \min\{l(uv) + l(vw) + 1 | uv, vw \in E(G)\}$.

Proof. Since G is an odd cycle, in any feasible assignment ϕ there exist edges uv and vw of G such that $\phi(u) \leq \phi(v) \leq \phi(w)$, and then $|\phi(w) - \phi(u)| \geq l(uv) + l(vw)$. Thus the span of G is at least M, and so it is at least $\max(L, M)$.

On the other hand, let us choose two edges uv and vw in G with $l(uv) + l(vw) = M - 1$. Form an even cycle G' by deleting v and adding the edge uw. Consider the length function l' on $E(G')$ which satisfies $l'(uw) = l(uv) + l(vw)$ and agrees with l elsewhere. Since G' is bipartite we see that an optimal feasible assignment c for G' has span $\max(M, L)$. Furthermore, since u and w are at distance at least $l(uv) + l(uw)$, we can choose $\phi(v)$ between $\phi(u)$ and $\phi(w)$ to obtain a feasible assignment for G with the same span. The result follows. □

Let us call a graph 1-*nearly bipartite* if by deleting at most one node we may obtain a bipartite graph. It is of course easy to tell if this is the case, by simply deleting each node in turn. It is also easy to determine the chromatic number $\chi(G)$ of a 1-nearly bipartite graph G, as it is at most 3. However, it is NP-hard to determine $\mathrm{span}(G, l)$, even if we restrict the edge lengths to be 1 or 2, see [64]. Further the span must then be at most 5, and it is NP-complete to tell if it is at most 4. Thus, we cannot hope to obtain a polynomial time approximation algorithm with performance ratio better than $\frac{5}{4}$, even for such restricted constraint matrix problems.

We discuss bipartite graphs further in section 2.6.3, where there are demands and a co-site constraint.

2.4.2 Bounded tree-width graphs

The 'tree-width' of a connected graph measures how far the graph is from being a tree – see for example Chapter 4 in this book. On trees, many problems can be solved quickly (in polynomial time) by simple dynamic programming, and often a similar approach works for graphs of bounded tree-width. For example it is easy to determine the chromatic number of such graphs.

It may be natural for us to consider constraint matrix problems where at most a fixed number b of different lengths allowed (for example there may be a fixed number of frequency-distance constraints – see 2.5). For such problems, if we consider graphs of bounded tree-width, the standard dynamic programming approach will determine the span in polynomial time. The key point is that there will be at most n^b possible values for the span, where n is the number of nodes: for, if the edge lengths are $l_1, \ldots, l_b$ then by Proposition 2.3.1 the span equals $1 + \sum_{i=1}^{b} a_i l_i$ for some integers $0 \leq a_i \leq n - 1$.

However, such a dynamic programming approach does not work if we do not restrict the lengths. Indeed, the problem of determining the span for graphs of tree-width at most 3 with arbitrary lengths is NP-hard [64].

2.5 Channel assignment in the plane

It is natural to specialise the constraint matrix model to the case where the transmitter sites are located in the plane, and the minimum channel separation for a pair of sites depends on the distance between them. We are led to consider unit disk graphs, and more generally to consider frequency-distance models. A theme throughout is the comparison of chromatic number to clique number and its generalisations.

2.5.1 Disk graphs

Let us consider only co-channel interference, which corresponds to each minimum channel separation being 0 or 1. Suppose that we are given a threshold distance d or d_0, such that interference will be acceptable as long as no channel is re-used at sites less than distance d apart. Given a set V of points in the plane and given $d > 0$, let $G(V, d)$ denote the graph with node set V in which distinct nodes u and v are adjacent whenever the Euclidean distance $d(u, v)$ between them is less than d. Equivalently, we may centre an open disk of diameter d at each point v, and then two nodes are adjacent when their disks meet. Such a graph is called a *unit disk* (or proximity) graph.

Our basic version of the channel assignment problem involves colouring such unit disk graphs. We are naturally also interested in the clique number $\omega(G)$ for such graphs G. The following result from [12], see also [51], shows that the clique and chromatic numbers are not too far apart.

Proposition 2.5.1 *For a unit disk graph G,*

$$\chi(G) \leq 3\omega(G) - 2.$$

Proof. In a realisation of G with diameter 1, consider the 'bottom left' point v. All its neighbours lie within an angle of less than 180 degrees at v. Thus we can cover all the neighbours with three sectors, each with radius less than 1 and angle less than 60 degrees. But the points in each sector together with v form a complete graph, and so the degree of v is at most $3(\omega(G) - 1)$. Hence the degeneracy of G is at most $3\omega(G) - 3$, and the result follows from (2.8). □

It would be nice to improve this result: perhaps the factor 3 could be replaced by 3/2? It is shown in [8] that it is NP-hard to recognise unit disk graphs. Many problems are hard for unit disk graphs, even given a realisation in the plane, see [12]: for example finding $\chi(G)$ or $\alpha(G)$. However, a polynomial time algorithm has recently been given [70] to find $\omega(G)$ without being given a realisation in the plane, see also [9]. It builds on an earlier method [12] which needed a realisation in the plane.

The idea of the method to find $\omega(G)$ is as follows. Firstly, in polynomial time we can find $\omega(H)$ if the graph H is *co-bipartite*, that is, if the complementary graph $\overline{H}$ is bipartite. For, a set K of nodes forms a maximum clique in $H = (V, E)$ if and only if $V \setminus K$ is a minimum cover (of edges by nodes) in $\overline{H}$; and we can find a minimum cover in a bipartite graph when we find a maximum matching.

Now let us call an ordering $e_1, \ldots, e_m$ of the edges of a graph G *good* if for each $i = 1, \ldots, m$ the following condition holds: the set N_i of common neighbours of the two end nodes of e_i in the subgraph with edges $e_{i+1}, \ldots, e_m$ is such that the subgraph $G[N_i]$ it induces in G is co-bipartite. [In [70] such an ordering is called a 'cobipartite neighbourhood edge elimination ordering'.] Now $\omega(G) = \max_i \omega(G[N_i]) + 2$ – to see this, consider a maximum clique K and the first edge e_i in K in the ordering. Hence, given a good edge ordering we can determine $\omega(G)$ in polynomial time.

Every unit disk graph has a good edge ordering: given a realisation in the plane we may simply order the edges by non-decreasing length. For consider two nodes u and v with distance $d(u, v) = d < 1$. Let W be the set of nodes in the 'lozenge' L of points in the plane within distance at most d of both u and v. The line uv cuts L into two halves: if x and y are nodes in the same half then $d(x, y) \leq d$ and so x and y are adjacent.

Finally, for any graph with a good edge ordering, a greedy method finds such an ordering quickly. For if we have a partial list $e_1, \ldots, e_{k-1}$ so far, we may take e_k as any edge which satisfies the condition above. There must be such an edge - consider a good ordering and the first edge in this ordering not amongst $e_1, \ldots, e_{k-1}$.

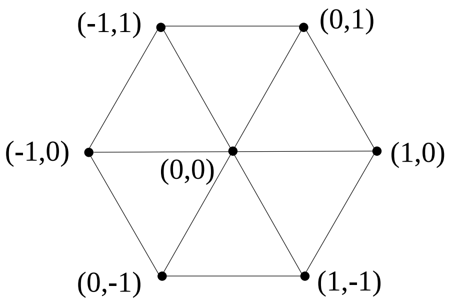


Figure 2.1. The neighbours of $(0,0)$

If the transmitters can have different powers, we are led to consider disk graphs, which are defined as for unit disk graphs except that the diameters may be different.

Proposition 2.5.2 *For a disk graph G,*

$$\chi(G) \leq 6\omega(G) - 5.$$

Proof. Consider a node v with disk of smallest diameter, and proceed as in the proof of Proposition 2.5.1 to show that the degeneracy is at most $6(\omega(G) - 1)$. □

As with the result for unit disk graphs, it would be nice to improve this result. It does not seem to be known whether for disk graphs there is a polynomial time algorithm to find $\omega(G)$, even given a realisation in the plane. In polynomial time we can approximate to within any fixed factor the stability number α and the fractional chromatic number χ_f (defined in Section 2.6) – see [53, 40, 16]. For related work see [51, 50, 29].

2.5.2 The triangular lattice

The triangular lattice T crops up naturally in radio channel assignment. It is sensible to aim to spread the transmitters out to form roughly a part of a triangular lattice, with hexagonal cells, since that will give the best 'coverage', that is, for a given number of transmitters in a given area this pattern minimises the maximum distance to a transmitter.

The triangular lattice graph may be described as follows. The vertices are all integer linear combinations $x\mathbf{p} + y\mathbf{q}$ of the two vectors $\mathbf{p} = (1, 0)$ and $\mathbf{q} = (\frac{1}{2}, \frac{\sqrt{3}}{2})$: thus we may identify the vertices with the pairs (x, y) of integers. Two vertices are adjacent when the Euclidean distance between them is 1. Thus each vertex (x, y) has the six neighbours : $(x \pm 1, y), (x, y \pm 1), (x + 1, y - 1), (x - 1, y + 1)$, see Figure 2.1. We always assume that the lattice T has this natural embedding in the plane with minimum distance 1. The cells are hexagons centered on the points, with diameter $2/\sqrt{3}$.

For any $d > 0$, we let $\hat{d}$ be the minimum Euclidean distance between two points in T subject to that distance being at least d. Then $d \leq \hat{d} \leq \lceil d \rceil$, and we can compute the $\hat{d}^2$ quickly, in $O(d)$ arithmetic operations.

Theorem 2.5.1 *The triangular lattice T satisfies*

$$\chi(G(T, d)) = \hat{d}^2$$

for any $d > 0$.

This result [63] appears to have been known to engineers at least since 1979 – see [49, 17] – and see also Theorem 3 in [5]. In section 2.6.1 we shall consider the triangular lattice again, focussing on the effect of demands.

2.5.3 Large distances in the plane

In order to gain insight without getting lost in details, one might consider the case when d is large. It turns out (cf. [62]) that it is possible to make quite precise statements in the limit as $d \to \infty$. These results are phrased in terms of the upper density of the set of sites, which is roughly the maximum number of sites per unit area over large areas. Both the chromatic number and the clique number tend to be large when the upper density is large.

So how do we define 'upper density'? Let V be any countable set of points in the plane. For $x > 0$ let $f(x)$ be the supremum of the ratio $|V \cap S|/x^2$ over all open $(x \times x)$ squares S with sides aligned with the axes. The *upper density* of V is $\sigma^+(V) = \inf_{x>0}: f(x)$. In fact $f(x) \to \sigma^+(V)$ as $x \to \infty$; and the definition could equally well be phrased in terms of disks say rather than squares. The square lattice and the triangular lattice (with minimum distance 1) have upper density 1 and $2/\sqrt{3}$ respectively.

Theorem 2.5.2 *Let V be a countable non-empty set of points in the plane, with upper density $:\sigma^+(V) = \sigma$. For any $d > 0$, denote the clique number $\omega(G(V, d))$ by ω_d, and use χ_d, Δ_d and δ_d^* similarly for the chromatic number, maximum degree and degeneracy. Then $\omega_d/d^2 \geq \sigma\pi/4$ and $\chi_d/d^2 \geq \sigma\sqrt{3}/2$ for any $d > 0$; and, as $:d \to \infty$, $\Delta_d/d^2 \to \sigma\pi:$, $\delta_d^*/d^2 \to \sigma\pi/2:$, $\omega_d/d^2 \to \sigma\pi/4:$ and $\chi_d/d^2 \to \sigma\sqrt{3}/2$.*

It follows for example that for any countable set V of points in the plane with a finite positive upper density, the ratio of the chromatic number of $G(V, d)$ to its clique number tends to $:2\sqrt{3}/\pi \sim 1.103:$ as $d \to \infty$. It was suggested in [18] that such a result should hold for the triangular lattice.

A key step in proving the result on χ_d in Theorem 2.5.2 is provided by Theorem 2.5.1. The idea is to scale the triangular lattice T so the density is slightly greater than σ, and then transfer a good colouring of T over to V.

2.5.4 The frequency-distance model

Unit disk graphs are interesting, but for channel assignment problems we may want to consider more than just co-channel interference, and more general trade-offs between geographical distance and channel separation. Suppose that we are given a non-zero vector $\mathbf{d} = (d_0, d_1, \ldots, d_{k-1})$ of $k \geq 1$ distances, where $d_0 \geq d_1 \geq \cdots \geq d_{k-1} \geq 0$. We call such a vector a *distance k-vector*. An assignment $f : V \to \{1, 2, \ldots, t\}$ is called $\mathbf{d}$-*feasible* if it satisfies the *frequency-distance constraints*

$$d(u, v) < d_i \ \Rightarrow \ |f(u) - f(v)| > i$$

for each pair of distinct points u, v in V and for each $i = 0, 1, \ldots, k-1$. This yields a constraint matrix problem where $l(uv) = i+1$ if $d_{i+1} \leq d(u, v) < d_i$ (set $d_k = 0$). As usual, $\text{span}(V, \mathbf{d})$ denotes the least integer t for which there is such an assignment. This *frequency-distance model* is a popular standard model for channel assignment, see for example [34], with k typically equal to 2 or 3 or 4.

When $k = 1$, so that there is just one distance d_0 given, we are back to colouring proximity graphs as discussed above. For an example with $k = 2$, suppose that $\mathbf{d} = (\sqrt{2}, 1)$ and the set V of sites is the set Z^2 of integer points (i, j). Then we may obtain a $\mathbf{d}$-feasible assignment $f : V \to \{1, 2\}$ from the natural 2-colouring of the sites: indeed we may set $f((i, j)) = 1$ if $i + j$ is odd, and $= 2$ if $i + j$ is even. Clearly $:\text{span}(\mathbf{d}; V) = 2$ here. The values $d_0, d_1, \ldots$ are set with the intention that any $\mathbf{d}$-feasible assignment will lead to acceptable levels of interference. As discussed above, the d_0-constraint limits co-channel interference, and similarly the d_1-constraint limits the contribution to the interference from first adjacent channels.

2.5.5 Large distances and frequency-distance constraints

When the distances $d_0, d_1, \ldots$ are small, small changes in them (or in the set V) can lead to large proportional changes in the span. In order to gain insight into the problem without getting lost in details, much as before in Section 2.5.3, we consider the case when the distances are large. Suppose then that $\mathbf{d} = d\mathbf{x}$ where $\mathbf{x} = (x_0, x_1, \ldots, x_{k-1})$ is a fixed distance k-vector and $d \to \infty$. Are there results for this case corresponding to Theorem 2.5.2 above on unit disk graphs? It turns out [57] that indeed $\text{span}(V, d\mathbf{x})/d^2$ tends to a limit as $d \to \infty$, and some partial results are known about the limit. The limit is specified as the product of the upper density of V and the 'inverse channel density' $\chi(\mathbf{x})$ of the distance vector $\mathbf{x}$.

Let $\mathbf{x}$ be a distance k-vector, that is $\mathbf{x} = (x_0, x_1, \ldots, x_{k-1})$, where $x_0 \geq x_1 \geq \cdots \geq x_{k-1} \geq 0$ and $x_0 > 0$. For each $i = 1, 2, \ldots$, the *i-channel density* $\alpha_i(\mathbf{x})$ is the supremum of the upper density $\sigma^+(V)$ over all sets V of points in the plane for which there is an $\mathbf{x}$-feasible assignment using channels $1, \ldots, i$. The 1-channel density $\alpha_1(1)$ is thus the maximum density of a packing of pairwise disjoint unit-diameter circles in the plane; and so $\alpha_1(1) = 2/\sqrt{3}$ and corresponds to taking V as the triangular lattice with unit edge lengths. This is the classical result of Thue on packing circles in the plane – see for example [74, 67, 78]. [We write $\alpha(1)$ instead of $\alpha((1))$ and so on.]

We shall be interested in particular in the 2-channel density $\alpha_2(1, x_1)$. This quantity is the solution of the following red-blue-purple circle packing problem. We wish to pack in the plane a pairwise disjoint family of red unit-diameter circles and a pairwise disjoint family of blue unit-diameter circles, where a red and a blue circle may overlap, forming a purple patch, but their centres must be at least distance x_1 apart. What is the maximum density of such a packing? [Equivalently we may think of packing unit-diameter balls in $\mathbf{R}^3$, where the balls must be in two layers, one with centres on the plane $z = 0$ and one with centres on the plane $z = (1 - x_1^2)^{\frac{1}{2}}$.]

The *channel density* $\alpha(\mathbf{x})$ is the infimum over all positive integers i of $\alpha_i(\mathbf{x})/i$. It is not hard to see that $\alpha(1) = \alpha_1(1)$ and so $\alpha(1) = 2/\sqrt{3}$; and that always $: 0 < \alpha(\mathbf{x}) < \infty$. Further, define the *inverse channel density* $\chi(\mathbf{x})$ to be $1/\alpha(\mathbf{x})$.

Theorem 2.5.3 *For any set V of points in the plane, and any distance k-vector* $\mathbf{x}$

$$\mathrm{span}(V, d\mathbf{x})/d^2 \to \sigma^+(V) \colon \chi(\mathbf{x}) \quad \textit{as } d \to \infty.$$

Thus in particular, for any set V of points in the plane with upper density 1, such as the set of points of the unit square lattice, the ratio $\mathrm{span}(V, d\mathbf{x})/d^2$ tends to the inverse channel density $: \chi(\mathbf{x}) :$ as $: d \to \infty$.

We wish to develop an understanding of the quantity $\mathrm{span}(V, \mathbf{d})$, and in particular of how it compares with certain natural lower bounds. One of these lower bounds on the span comes from considering the 'distance-s cliques'. A family of points forms a *distance-s clique* if each pair of points in the set is at distance less than s. If there is a distance-d_j clique (sometimes called a level-$(j+1)$ clique) with t elements then by (2.1) in section 2.3.2,

$$\mathrm{span}(V, \mathbf{d}) \geq 1 + (t-1)(j+1) = (j+1)t - j.$$

Let us call the maximum value of these bounds over all $j = 0, \ldots, l-1$ and all distance-d_j cliques the *clique bound* for the problem and denote it by $\mathrm{cliquebound}(V, \mathbf{d})$.

The quantity $\mathrm{cliquebound}(V, \mathbf{d})$ just introduced may be defined by

$$\mathrm{cliquebound}(V, \mathbf{d}) = \max_j \left((j+1)\omega(G(V, d_j)) - j\right),$$

where the maximum is over $j = 0, \ldots, k-1$. Let us consider also

$$\mathrm{colourbound}(V, \mathbf{d}) = \max_j \left((j+1)\chi(G(V, d_j)) - j\right).$$

Clearly colourbound$(V, \mathbf{d}) \geq$ cliquebound$(\mathcal{V}, \mathbf{d})$ since $\chi(G) \geq \omega(G)$ for every graph G, and it follows for example from Proposition 2.3.1 that span$(V, \mathbf{d}) \geq$ colourbound$(V, \mathbf{d})$. Theorem 2.5.2 yields easily that for any set V of points in the plane and any distance k-vector $\mathbf{x}$, as $: d \to \infty$

$$\text{colourbound}(V, d\mathbf{x})/d^2 \to \sigma^+(V)\colon \max_j \{(j+1)x_j^2\}\sqrt{3}/2 \qquad (2.9)$$

and

$$\text{cliquebound}(V, d\mathbf{x})/d^2 \to \sigma^+(V)\colon \max_j \{(j+1)x_j^2\}\pi/4. \qquad (2.10)$$

It follows using Theorem 2.5.3 that for any set V of points with finite non-zero upper density, and any distance k-vector $\mathbf{x}$, as $: d \to \infty$,

$$\text{span}(V, d\mathbf{x})/\text{colourbound}(V, d\mathbf{x}) \to (2/\sqrt{3})\ \chi(\mathbf{x})/\max_j \{(j+1)x_j^2)\} \quad (2.11)$$

and

$$\text{colourbound}(V, d\mathbf{x})/\text{cliquebound}(V, d\mathbf{x}) \to 2\sqrt{3}/\pi. \qquad (2.12)$$

Perhaps there is most interest in the case $k = 2$, when we have just two distances $d_0 \geq d_1$. The current knowledge on the value of $\chi(1, x)$ is summarised in the following theorem – see also Figure 2.2.

Theorem 2.5.4 *There are exact results, that* $: \chi(1, x) = \sqrt{3}/2$: *for* $: 0 \leq x \leq 1/\sqrt{3}$, $: \chi(1, 1/\sqrt{2}) = 1$: *and* $: \chi(1, 1) = \sqrt{3}$. *There are lower bounds, that* $: \chi(1, x)$: *is at least*

$$\begin{array}{ll} \sqrt{3}/2 & \textit{for } 1/\sqrt{3} < x \leq 3^{\frac{1}{4}}/2 \simeq 0.658 \\ 2x^2 & \textit{for } 3^{\frac{1}{4}}/2 \leq x < 1/\sqrt{2} \simeq 0.707 \\ 1 & \textit{for } 1/\sqrt{2} < x \leq 3^{-\frac{1}{4}} \simeq 0.760 \\ \sqrt{3}x^2 & \textit{for } 3^{-\frac{1}{4}} \leq x < 1. \end{array}$$

Finally, there are upper bounds, that $: \chi(1, x)$: *is at most*

$$\begin{array}{ll} \frac{3\sqrt{3}}{2}x^2 & \textit{for } 1/\sqrt{3} < x \leq 4/\sqrt{43} \simeq 0.610 \\ 2x\sqrt{1-x^2} & \textit{for } 4/\sqrt{43} \leq x < 1/\sqrt{2} \simeq 0.707 \\ 2\sqrt{x^2 - \frac{1}{4}} & \textit{for } 1/\sqrt{2} < x < 1. \end{array}$$

Suppose for example that $k = 2$ and $\mathbf{x} = (1, 1/\sqrt{2})$. Then we have $\chi(\mathbf{x}) = 1$, and $\max\{x_0^2, 2x_1^2\} = 1$, and so the limit in (2.11) above is $2/\sqrt{3}$; and hence

$$\text{span}(V, d\mathbf{x})/\text{cliquebound}(V, d\mathbf{x}) \to 4/\pi$$

as $d \to \infty$. If $\mathbf{x} = (1, x_1)$ where $0 < x_1 \leq 1/\sqrt{3}$ then we have $\chi(\mathbf{x}) = \sqrt{3}/2$ and $\max\{x_0^2, 2x_1^2\} = 1$, and so the limit in (2.11) above is 1.

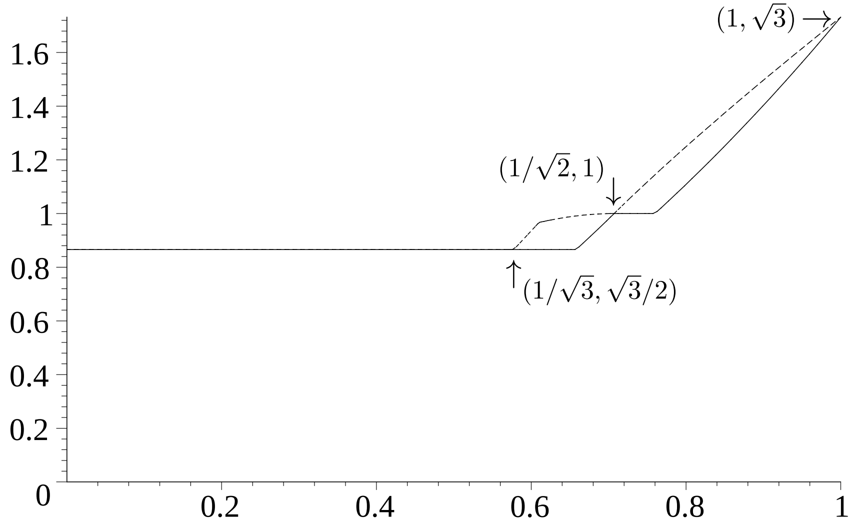


Figure 2.2. Upper and lower bounds on $\chi(1,x)$

2.6 Channel assignment with demands

2.6.1 The triangular lattice with demands

Given a finite induced subgraph of the triangular lattice graph together with demands, how well can we assign channels? Here we are assuming that each edge length $l(uv)$ is 1 and each co-site constraint value $c(v)$ is 1. We shall present two theorems from [63], one a hardness result and one a result on algorithmic approximation. First, however, we make some preliminary comments.

Given a graph G together with a demand vector $\mathbf{x}$, there is a natural associated 'replicated' graph $G_{\mathbf{x}}$, obtained by replacing each node v by a complete graph on $:x_v$ nodes. An assignment of channels for the pair $(G,\mathbf{x})$ corresponds to a colouring of the graph $G_{\mathbf{x}}$.

A graph G is *perfect* if for each induced subgraph H of G, the chromatic number $\chi(H)$ equals the maximum number $\omega(H)$ of vertices in a complete subgraph of H. If a graph G is perfect then so is the replicated graph $G_{\mathbf{x}}$ for any demand vector $\mathbf{x}$, and further an optimal weighted colouring can be found in polynomial time by using the ellipsoid method – see [32]. If G is bipartite, for example if it is a finite subgraph of the square or hexagonal lattice, then things are even easier. Of course, finite subgraphs of the triangular lattice graph need not be perfect – see the remarks at the end of this section. Recall from section 2.5.2 that we may represent the points of the triangular lattice by pairs of integers.

Theorem 2.6.1 *It is NP-complete to determine, on input a set F of pairs of integers determining an induced subgraph G of the triangular lattice graph together with a demand vector $\mathbf{x}$, if the graph $G_{\mathbf{x}}$ is 3-colourable.*

This theorem extends the result mentioned earlier that it is NP-hard to determine the chromatic number of a unit disk graph. Given an input as above, it is of course easy to find the maximum size $\omega(G_{\mathbf{x}})$ of a complete subgraph of $G_{\mathbf{x}}$ in polynomial time, since each clique in G has at most three nodes.

Theorem 2.6.2 *There is a polynomial time combinatorial algorithm which, on input a set F of pairs of integers determining an induced subgraph G of the triangular lattice graph together with a demand vector $\mathbf{x}$, finds a feasible assignment which uses at most $\frac{4\hat{\omega}+1}{3}$ colours, where $\hat{\omega} = \omega(G_{\mathbf{x}})$.*

For related results see [46, 66, 77]. The algorithm is quite simple and practical. It has a distributed phase, in which it constructs colour sets similar to the colour sets of the 3-colouring of the triangular lattice graph, and then a tidy-up phase which corresponds to colouring a forest. By using the algorithm, we can find quickly a weighted colouring for an induced subgraph of the triangular lattice such that the number of colours used is no more than about 4/3 times the corresponding clique number of G_w, and hence is no more than about 4/3 times the optimal number. Further by Theorem 2.6.1 we cannot guarantee to improve on the ratio 4/3, assuming that $P \neq NP$.

However, perhaps we are being pessimistic. In typical radio channel assignment problems, the maximum number of channels demanded at a transmitter may be quite large. For example, the 'Philadelphia problem' described in [18] involves a 21 vertex subgraph of the triangular lattice with demands ranging from 8 to 77 (though it also has constraints on the colours of vertices at distances up to 3, and so it is not a simple weighted colouring problem). Perhaps we can improve on the ratio 4/3 if there are large demands?

We note that the 9-cycle C_9 is an induced subgraph of the triangular lattice graph. Further, for any positive integer k, if we start with a C_9 and replicate each node k times, we obtain a graph with clique number $2k$ and chromatic number $\lceil \frac{9k}{4} \rceil$. Is this ratio $\frac{9}{8}$ of chromatic number to clique number asymptotically worst (greatest) possible? This question has sparked off much further work on the 'imperfection ratio' of graphs [24, 25, 58, 27], but it is still not resolved. It has been shown [36] that for any triangle-free subgraph of the triangular lattice graph together with demands, the ratio is at most $\frac{7}{6}$. See also [37].

Throughout the above, we assumed that each co-site constraint value was 1. There is a very similar result if each co-site constraint value is 2, ([77], see also [27]. If each co-site constraint value is 3, things are rather different: for since $\chi(G) \leq 3$ the span is at most $3x_{\max}$ (see Proposition 2.3.1), and of course it is a least $3x_{\max} - 2$.

2.6.2 The dumbell problem

Co-site constraints can cause difficulties. A (2×2) constraint matrix $\begin{pmatrix} a & b \\ b & c \end{pmatrix}$ with a demand vector (m, n) yields a deceptively simple-looking problem. Note that this corresponds to a constraint graph consisting of just a single edge with length b, co-site constraint values a and c (not necessarily the same), and demands. Dominic Welsh christened this the *dumbell problem*, and asked if there is a formula for the span in terms of a, b, c, m, n, or if we can at least compute the span in time bounded by a polynomial in the input size, even for a fixed constraint matrix. Some partial results are given in [68].

2.6.3 Bipartite graphs with co-site constraint value 2

Suppose that the constraint graph is bipartite, and each edge length is 1. If each co-site constraint value (that is, each diagonal entry in the constraint matrix) is 1, then the problem is easy, as we noted earlier. Let us consider the case when each co-site constraint value is 2. (We discussed this problem with cyclic channel distances in section 2.3.7.)

The span is at most $2x_{\max}$, since we could use the odd channels $1, 3, \ldots, 2x_{\max} - 1$ on one part and the even channels $2, 4, \ldots, 2x_{\max}$ on the other. Further, clearly the span is at least $2x_{\max} - 1$. Can we tell which value is correct?

Call a path in G *critical* if it has an odd number t of edges, the end nodes have demand $x_{\max}$ and any internal nodes have demand $< x_{\max}$. Suppose that the span is $2x_{\max} - 1$, with a feasible assignment $\phi : V(G) \to \{1, \ldots, 2x_{\max} - 1\}$, and consider such a path. The end nodes must get precisely all the odd channels, so each of the $x_{\max} - 1$ even channels can be used at most $(t-1)/2$ times, and each of the $x_{\max}$ odd channels can be used at most $(t+1)/2$ times. Thus the total number of appearances of channels on the path is at most

$$(x_{\max} - 1)(t-1)/2 + x_{\max}(t+1)/2 = t x_{\max} - (t-1)/2.$$

The *critical path condition* is that for each critical path, the sum of the demands on the nodes is at most $t x_{\max} - (t-1)/2$.

Theorem 2.6.3 *Let G be a bipartite graph, let each edge length be 1, and let each co-site constraint value be 2. Let $\mathbf{x}$ be a demand vector. Then the span is either $2x_{\max} - 1$ or $2x_{\max}$, and it is the lower value if and only if the critical path condition holds. Further, we can determine the span in polynomial time.*

It is easier to handle the case when each co-site constraint value is 3.

Theorem 2.6.4 *Let G be a bipartite graph, let each edge length be 1, and let each co-site constraint value be 3. Let $\mathbf{x}$ be a demand vector. Then the span is either $3x_{\max} - 2$ or $3x_{\max} - 1$, and it is the lower value if and only if no two nodes with maximum demand are adjacent.*

For the above results and related results, see [22, 23].

2.6.4 Large demands

Since in applications demands are unpredictable, there is some advantage (in terms of the ratio of numbers of channels needed to mean demand) in having cells large enough so that the mean demand is not too small. This effect is called 'trunking gain'. (It need not concern us that, with a time division multiple access scheme like GSM, there may be up to 8 users to a channel.) Certainly, some standard test problems have large demands, as mentioned in section 2.6.1.

Let us assume that each edge length is 1 and each co-site constraint is 1, as in section 2.6.1, and focus on the demands. We pick up the discussion from the end of that section.

Consider a (fixed) graph G. For each positive integer k, let

$$r_k(G) = \max\{\frac{\chi(G_{\mathbf{x}})}{\omega(G_{\mathbf{x}})} : x_{\max} = k\},$$

where the maximum is over all non-negative integral demand vectors $\mathbf{x}$ with maximum entry $x_{\max}$ equal to k. Observe that $: r_k(G) \geq 1:$ always.

Theorem 2.6.2 above shows that if G is an induced subgraph of the triangular lattice T, then $: r_k(G) \leq \frac{4k+1}{3k}$ for all positive integers k. But we are interested in the values of $r_k(G)$ for *large* k rather than the maximum value over *all* k, as this corresponds to large demands.

Recall from section 2.6.1 that if a graph G is perfect then so is each graph obtained from G by replication.Thus G is perfect if and only if $r_k(G) = 1$ for each positive integer k. Since any bipartite graph is perfect, a natural starting point for further investigation is to consider odd cycles. If n is an odd integer at least 5, then $: |r_k(C_n) - \frac{n}{n-1}| < \frac{1}{k}$, and so $r_k(C_n) \to \frac{n}{n-1}$ as $k \to \infty$. In fact $r_k(G)$ always tends to a limit as $k \to \infty$, namely the imperfection ratio of G, which we now proceed to define.

First we need to recall the definition of the fractional chromatic number of a graph G. Introduce a variable y_S for each stable set S in G. The *fractional chromatic number* $\chi_f(G)$ is the value of the linear program: $: \min \sum_S y_S$ subject to $: \sum_{v \in S} y_S \geq 1$ for each node v: and $: y_S \geq 0$ for each stable set S. Alternatively, $\chi_f(G)$ is the least value of the ratio a/b such that with a colours we may colour each node of G exactly b times, see for example [76]. It is easily seen that

$$\omega(G) \leq \chi_f(G) \leq \chi(G).$$

Now we define the *imperfection ratio*, imp(G), by setting

$$\text{imp}(G) = \max_{\mathbf{x}}\{\frac{\chi_f(G_{\mathbf{x}})}{\omega(G_{\mathbf{x}})}\}, \tag{2.13}$$

where the maximum is over all non-zero integral weight vectors $\mathbf{x}$. (The ratios on the right hand side above do indeed attain a maximum value.) Observe that imp$(G) \geq 1$.

Theorem 2.6.5 *For any graph G, $r_k(G) \to \text{imp}(G)$: as* $: k \to \infty$.

For example, we see immediately from the above result on odd cycles that imp$(C_n) = \frac{n}{n-1}$ for the odd cycle C_n with $n \geq 5$ nodes. The following theorem records two basic properties of the imperfection ratio. It is most naturally proved in the context of equivalent polyhedral definitions of the imperfection ratio, see [24].

Theorem 2.6.6 *For any graph G,* imp$(G) = 1$ *if and only if G is perfect; and* imp$(G) = \text{imp}(\bar{G})$,: *where $\bar{G}$ denotes the complement of G.*

Further, imp$(G) \leq 4/3$ for any finite induced subgraph G of the triangular lattice T, by Theorem 2.6.2; and the question we asked above about asymptotic ratios may now be rephrased in terms of imp(G), as follows.

Conjecture 2.6.7 *If G is a finite induced subgraph G of the triangular lattice T, then* imp$(G) \leq 9/8$.

It turns out that the imperfection ratio is related to the 'entropy' of G and its complement, see [58, 80]. Let us mention a few further results on the imperfection ratio: for these and many more see [22, 24, 25, 58].

- Suppose that G is a line-graph. If G has no odd holes then G is perfect, so imp$(G) = 1$. If G has an odd hole, and the shortest length of one is g, then imp$(G) = g/(g-1)$.
- For any planar triangle-free graph G, imp$(G) \leq 3/2$, and the constant $3/2$ is best possible.
- For a unit disk graph G, imp$(G) \leq 1 + 2/\sqrt{3} < 2.2$. The cycle power graph C_{3k-1}^{k-1} is a unit disk graph (see [50]), which shows that this bound cannot be reduced below $3/2$. Perhaps this is the right value; that is, do we have imp$(G) \leq 3/2$ for any unit disk graph?
- For an n-node graph, the integer weights x_v required to achieve imp(G) in the definition (2.13) can grow exponentially with n, though they can always be bounded by $n^{\frac{n}{2}}$.
- For the random graph $G_{n,\frac{1}{2}}$, the imperfection ratio is close to $n(2\log_2 n)^{-2}$ with high probability.

2.7 Random channel assignment problems

2.7.1 Random models in the plane

It is not easy to model satisfactorily the distribution of sites in the plane. Sometimes it is assumed that they form part of a regular lattice, but we consider quite a different case here. We assume here that the sites are generated by picking n points $\mathbf{X}_1, \mathbf{X}_2, \ldots$ independently according to some fixed probability distribution on the plane, and we let $n \to \infty$. The following results are taken from [59]. The proofs lean heavily on the deterministic work described in section 2.5.3.

Random unit disk graphs

It is of interest to investigate how large the ratio $\chi(G)/\omega(G)$ is 'usually' for unit disk graphs. We saw earlier that always $\chi(G)/\omega(G) \leq 3$ for a unit disk graph G. To give some meaning here to the word 'usually', we need either much empirical data or many simulations – see for example [89], or a suitable random model. We adopt the latter approach here.

Let the random variable $\mathbf{X}$ be distributed in the plane with some distribution ν. Let $\mathbf{X}_1, \mathbf{X}_2, \ldots,$ be independent random variables, each distributed like $\mathbf{X}$. Let $\mathbf{X}^{(n)}$ denote the family consisting of the first n random points $\mathbf{X}_1, \ldots, \mathbf{X}_n$. Let $d = d(n) > 0$ for $n = 1, 2, \ldots$, and let G_n denote the random (embedded) unit disk graph $G(\mathbf{X}^{(n)}, d(n))$.

Previous work on random unit disk graphs, see [69], shows the importance of the ratio $d^2n/\ln n$. For example, suppose that the underlying distribution is the uniform distribution on the unit square, so that for large n the average degree of a node is close to $\pi d^2 n$. Then as $n \to \infty$, the probability that G_n is connected tends to 0 if $d^2n/\ln n \to 0$ and tends to 1 if $d^2n/\ln n \to \infty$. It does not seem clear for the application to channel assignment problems how we should wish the average degree to behave, though slow growth of some sort seems reasonable.

An important quantity will be the *maximum density* $\nu_{\max}$ of the distribution. This may be defined in many equivalent ways, for example as

$$\nu_{\max} = \sup_B \nu(B)/\text{area}(B), \tag{2.14}$$

where the supremum is over all open balls B, $\nu(B) = P(\mathbf{X} \in B)$, and of course $\text{area}(B) = \pi r^2$ if B has radius r. Typically this is just the maximum value of the density function. We shall be interested in the case when this quantity is finite.

As before we focus on the ratio of chromatic number to clique number. We consider the 'sparse' case, when $d = d(n)$ is such that the average degree grows more slowly than $\ln n$; and the 'dense' case, when the average degree grows faster than $\ln n$.

Theorem 2.7.1 *Let the distribution ν have finite maximum density. Let $d = d(n)$ satisfy $d(n) \to 0$ as $n \to \infty$.*

(a) (Sparse case) As $n \to \infty$, if $d^2n/\ln n \to 0$ but $d^2n \geq n^{-o(1)}$ then

$$\chi(G_n)/\omega(G_n) \to 1 \text{ in probability.}$$

(b) (Dense case) As $n \to \infty$, if $d^2n/\ln n \to \infty$ then

$$\chi(G_n)/\omega(G_n) \to 2\sqrt{3}/\pi \sim 1.103 \text{ a.s.}$$

In part (b) above we use 'a.s.' or 'almost surely' in the standard sense in probability theory, that is we are asserting that

$$P\left(\chi(G_n)/\omega(G_n) \to 2\sqrt{3}/\pi \text{ as } n \to \infty\right) = 1.$$

Let us amplify Theorem 2.7.1, and split the sparse and dense cases into separate results.

We consider the upper bounds $\Delta(G) + 1$ and $\delta^*(G) + 1$ on $\chi(G)$, and the lower bound $\omega(G)$. It is of interest also to consider a natural lower bound on $\omega(G)$. We define the *disk containment number* of $G(\mathcal{V}, d)$ to be the maximum over all open disks of diameter d of the number of points of $\mathcal{V}$ in the disk. Let us denote this quantity by $\omega^-(G(\mathcal{V}, d))$. Of course we always have

$$\omega(G(\mathcal{V}, d)) \geq \omega^-(G(\mathcal{V}, d)).$$

It is straightforward to compute this quantity in $O(n^3)$ steps. We shall see that with high probability ω and ω^- are very close, which may help somewhat to explain why it has been found to be easy to calculate ω. In practice for problems arising in radio channel assignment (which do not necessarily give rise to a unit disk graph) usually it turns out to be easy to determine ω, and colouring methods that start from large cliques or near cliques have proved to be very successful [42].

Theorem 2.7.2 *(On sparse random disk graphs)*

Let $d = d(n)$ satisfy $d^2n = o(\ln n)$ and $d = n^{-\frac{1}{2}+o(1)}$. Let

$$k = k(n) = \frac{\ln n}{\ln(\frac{\ln n}{d^2 n})}.$$

Then $k \to \infty$ as $n \to \infty$:, and in probability $:\Delta(G_n)/k \to 1$: and $:\omega^-(G_n)/k \to 1,$: and so $:\chi(G_n)/\omega^-(G_n) \to 1$.

Theorem 2.7.3 *(On dense random disk graphs)*

Let $d = d(n)$ satisfy $d^2n/\ln n \to \infty$ as $n \to \infty$. Let

$$k = k(n) = \nu_{\max}(\pi/4)d^2n.$$

Then as $n \to \infty$, almost surely $:\omega^-(G_n)/k \to 1$, $:\omega(G_n)/k \to 1$, $:\chi(G_n)/k \to 2\sqrt{3}/\pi$, $:\delta^*(G_n)/k \to 2$, and $:\Delta(G_n)/k \to 4$.

This last result may be proved using many of the same ideas as for Theorem 2.5.2.

There is an unfortunate gap between the sparse and dense cases above. It would be interesting to learn about the behaviour of $\chi(G_n)/\omega(G_n)$ when $d^2n/\ln n \to \beta$ where $0 < \beta < \infty$. See [69] for the behaviour of $\omega(G_n)$ in this case, and for further related results.

Random frequency-distance problems

Let $\mathbf{c} = (c_0, c_1, \ldots, c_{l-1})$ be a fixed distance l-vector and let $d = d(n) \to 0$ as $n \to \infty$. We shall use d to scale the vector $\mathbf{c}$ appropriately, and focus on the problem generated by the family $\mathbf{X}^{(n)}$ consisting of the first n random points $\mathbf{X}_1, \ldots, \mathbf{X}_n$ together with the distance vector $d\mathbf{c}$. Denote the corresponding span by $\mathrm{span}(\mathbf{X}^{(n)}, d\mathbf{c})$, and similarly for the colour and clique bounds. Then

$$\mathrm{span}(\mathbf{X}^{(n)}, d\mathbf{c}) \geq \mathrm{colourbound}(\mathbf{X}^{(n)}, d\mathbf{c}) \geq \mathrm{cliquebound}(\mathbf{X}^{(n)}, d\mathbf{c}).$$

How good are these bounds usually?

Theorem 2.7.4 *Suppose that the distribution ν has finite maximum density. Let $\mathbf{c} = (c_0, c_1, \ldots, c_{l-1})$ be a fixed distance l-vector. Let $d = d(n)$ satisfy $d(n) \to 0$ as $n \to \infty$.*

(a) (Sparse case) As $n \to \infty$, if $d^2n/\ln n \to 0$ but $d^2n \geq n^{-o(1)}$, then in probability

$$\mathrm{span}(\mathbf{X}^{(n)}, d\mathbf{c})/\mathrm{cliquebound}(\mathbf{X}^{(n)}, d\mathbf{c}) \to 1.$$

(b) (Dense case) As $n \to \infty$, if $d^2n/\ln n \to \infty$ then a.s.

$$\mathrm{span}(\mathbf{X}^{(n)}, d\mathbf{c})/\mathrm{colourbound}(\mathbf{X}^{(n)}, d\mathbf{c}) \to (2/\sqrt{3})\chi(\mathbf{c})/\max_j\{(j+1)c_j^2\} \tag{2.15}$$

where the maximum is over $j = 0, \ldots, l-1$, and

$$\mathrm{colourbound}(\mathbf{X}^{(n)}, d\mathbf{c})/\mathrm{cliquebound}(\mathbf{X}^{(n)}, d\mathbf{c}) \to 2\sqrt{3}/\pi. \tag{2.16}$$

The limits in (2.15) and (2.16) above should be familiar from the earlier results (2.11) and (2.12). By Theorems 2.7.2 and 2.7.3, it would make no difference in the above result if we replaced ω by ω^- in the definition of the clique bound. More detailed results extending Theorem 2.7.4 are given in [59].

2.7.2 Random $\{0,1,2\}$-valued constraints

In this section, we follow [60] and consider random constraint matrix problems, where each constraint value is 0,1 or 2. We stick to the case of unit demands.

Let us start with the complete graph on the set of n nodes $V = \{v_1, \ldots, v_n\}$, together with a length $l(e) \in \{0, 1, 2\}$ for each edge e. Let E_i denote the set of edges of length i. Then E_2 contains the 'long' edges, E_1 the 'short' edges, and E_0 the 'missing' edges. Let G be the graph obtained by omitting the 'missing edges', that is $G = G(V, E_1 \cup E_2)$.

We have seen already that the span is at least $\chi(G)$ and is at most $2\chi(G) - 1$. Trivally, if E_2 is empty then we are back to ordinary node colouring and the span is $\chi(G)$. Also, if E_1 is empty then the span is $2\chi(G) - 1$ by Proposition 2.3.1.

Let us introduce the random model. Given $0 \le p \le 1$ and a positive integer n, the standard random graph $G_{n,p}$ has nodes $v_1, \ldots, v_n$ and the $\binom{n}{2}$ possible edges appear independently each with probability p.

Now let p_0, p_1 and p_2 be non-negative and sum to 1, and let $\mathbf{p} = (p_0, p_1, p_2)$. We call $\mathbf{p}$ a *probability vector*. The *random network* $G_{n,\mathbf{p}}$ has nodes $v_1, \ldots, v_n$ and the $\binom{n}{2}$ edges e have independent *lengths* X_e, where $P(X_e = i) = p_i$. An edge of length 0 corresponds to a missing edge, so the graph associated with the network has distribution G_{n,p_1+p_2}.

It is well known [6] that

$$\chi(G_{n,p}) \sim 2\ln(1/(1-p)) \colon (n/\ln n). \tag{2.17}$$

(We take p as fixed, $0 < p < 1$.) This notation means that the ratio of left hand side to right hand side tends to 1 in probability. (Much more precise results are known, see [6, 54].) At the Workshop on Radio Channel Assignment in Brunel University in July 2000, Jan van den Heuvel asked for similar results for the asymptotic behaviour of span$(G_{n,\mathbf{p}})$.

Fix a probability vector $\mathbf{p} = (p_0, p_1, p_2)$. It turns out that there is an abrupt change of behaviour (a 'phase transition') in span$(G_{n,\mathbf{p}})$ around the curve $p_1 = p_2(1 - p_2)$. If $p_1 \le p_2(1 - p_2)$ we are in the 'few short edges' regime, where we may as well treat short edges as long and leave about half the channel sets empty. In contrast, if $p_1 \ge p_2(1 - p_2)$ we are in the 'few long edges' regime, and it turns out that it is best to choose the channel sets nearly uniform in size.

Theorem 2.7.5 *Fix a probability vector* $\mathbf{p} = (p_0, p_1, p_2)$, *where* $p_0, p_1, p_2 > 0$ *and* $p_0 + p_1 + p_2 = 1$. *If* $p_1 \le p_2(1 - p_2)$ *then*

$$\text{span}(G_{n,\mathbf{p}}) \sim 2\chi(G_{n,p_1+p_2}) \sim \ln(1/p_0) \colon (n/\ln n);$$

and if $p_1 \ge p_2(1 - p_2)$ *then*

$$\text{span}(G_{n,\mathbf{p}}) \sim ((1/2)\ln(1/p_0) + \ln(1/(1-p_2))) : (n/\ln n).$$

On the 'critical curve' $p_1 = p_2(1-p_2)$ we have $\ln(1/p_0) = 2\ln(1/(1-p_2))$, so the two expressions for span$(G_{n,\mathbf{p}})$ are indeed equal there. It follows from the theorem together with (2.17) that when $p_1 \le p_2(1 - p_2)$ we have

$$\text{span}(G_{n,\mathbf{p}}) \sim \chi(G_{n,p_1+p_2}) + 2\chi(G_{n,p_2}),$$

but it is not clear what to make of this.

2.8 Modelling radio channel assignment

In this section we discuss the background to radio channel assignment problems. We shall end up with the (T_e)-sets model, which is where we started in section 2.8.

We may think of the radio channel assignment problem as the final stage in the design of a cellular radio communication system. The general idea of such a system is that many low-powered transmitters (base stations) each serve the customers in their local cell, and thus the same radio channel can be used simultaneously in many different cells, as long as these cells are sufficiently well separated. Since the radio spectrum is a finite resource which is heavily in demand, we want to assign the channels to the transmitters carefully in order to take maximum advantage of this re-use possibility.

Suppose then that transmitters are located at various sites in a geographical region, perhaps a city, with power levels set. Engineers often aim to spread the transmitters out to form roughly a part of a triangular lattice, since it gives the best 'coverage', that is, it minimises the maximum distance to a transmitter. Sometimes the transmitters may be spread out very differently, for example along a major road. We shall suppose that the channel bandwidth has been fixed, so that we may without loss of generality take the available channels to be the integers $\{1, \ldots, t\}$ for some t. The service region is divided into cells around each transmitter. We may think of the cell around transmitter v as consisting of the potential receiver sites which are closer to v than to any other transmitter, at least in the case when each transmitter has the same power. When such transmitters are spread out like part of the triangular lattice, the cells are hexagonal.

For each cell, there is an estimate of the (peak period) expected demand. Using these demand estimates, the requirement that calls be blocked say at most 2% of the time, and a simple queuing model, an appropriate number x_v of channels is chosen for each transmitter v. Note that we are considering a static model : there is interest also in dynamic models, where the demand levels change over time, and the focus is on the method for re-assigning channels. We shall not pursue this topic here, but see for example ..??

We have now described the input to the channel assignment problem from the early stages of the design of the cellular communication system. The aim in the final stage is to find an assignment of x_v channels to each transmitter v, such that the corresponding interference is acceptable, and the span of channels used is minimised. (Alternatively, we might wish to minimise the interference subject to a given span of channels being available.)

So, when will interference be acceptable? (For a general treatment partially ducking this question see [26].) Typically a 'protection ratio' θ

is set, depending on engineering considerations involving the selectivity of the equipment used and the width of the channel. We say that the interference arising from some channel assignment is acceptable if the signal-to-interference ratio (SIR) is at least the 'protection ratio' θ at each potential receiver site, or at all but a small proportion of test sites. In order to estimate signal-to-interference ratios we need a model for the propagation of radio waves, or many empirical measurements.

A typical simplified propagation model assumes that the signal power received at distance r from the transmitter is proportional to $r^{-\alpha}$ for an appropriate constant α, where $3 \leq \alpha \leq 4$ for a typical urban environment. (In free space $\alpha = 2$.) Here the 'power received' refers to a receiver tuned to the same channel c as the transmitter. For a receiver tuned to channel $c \pm i$ the received power drops off rapidly with i. In the model proposed in [28] (and used for example in [89]), the received power drops off by a factor of about $(2i)^5$. Thus for a receiver tuned to one of the adjacent channels $c \pm 1$, the power received is reduced by a factor 32, and for a receiver tuned to a more distant channel the power received is negligible.

It is assumed that the power received does not depend on the frequency used (which is realistic since typically the range of frequencies involved is small). For omnidirectional transmitters, it is sometimes assumed for simplicity that this power depends only on the distance from the transmitter. (We shall not consider directional versions here, but see for example [83]). More detailed propagation models consider also 'fading effects' due to shadowing (perhaps from intervening buildings or rain) or to multiple path interference effects, though these can be allowed for by adding a safety margin to the protection ratio θ. Typically external effects such as thermal noise are ignored.

Now consider a transmitter v and a potential receiver R in the cell around v, where R is tuned to channel c, one of the channels at v. On the basis of a propagation model or empirical measurements, we can estimate the signal power received at R from transmitter v. We can also estimate the unwanted power received at R from each of the other transmitters using channel c: these combine to form the 'co-channel' interference. Similarly, we can estimate the unwanted power received at R from each transmitter using the adjacent channels $c \pm 1$, and the resulting 'first-adjacent channel' interference, and so on for more distant channels with decreasing importance.

The question remains of how to combine the interfering unwanted signal powers to yield the total interference. The usual way to do this is simply to take the maximum value, and again allow a margin for error in the protection ratio θ. We follow this method here, which leads to models with binary constraints, that is, involving only pairs of channels. If we do not make the simplifying 'dominant interferer' assumption, we are led to hypergraph colouring models (see for example [41]) or models where we compute the interference 'globally' from the entire assignment, see for example [89].

A natural aim is to find a feasible assignment that achieves this minimum span or is close to it. Alternatively, there might be a fixed range of channels available, and the aim is to find a feasible assignment using only channels within this range which then perhaps minimises the maximum interference. Another possibility is that we cannot find a feasible assignment within the given span, and we have to settle for some violated constraints. We shall restrict our attention to the first aim.

Another simplifying assumption that seems reasonable from the physics of interference is that only the difference between two channels matters. Typically the smaller the difference the greater the interference, but this is not always the case as there may for example be 'intermodulation products', in particular at transmitters on the same site.

Consider a pair of transmitters u and v, and suppose that they transmit on channels differing by c. If there is a potential receiver in the cell around u such that the ratio of the received power from u to that from v is less than the protection ratio θ, then we make c a 'forbidden difference' for u and v; and similarly with u and v interchanged. We have now got to the stage where the most general problem we wish to consider specifies for each pair uv of transmitters a set T_{uv} of forbidden differences $|i-j|$ for channels $i \in f(u)$ and $j \in f(v)$. Clearly we may assume that always $T_{uv} = T_{vu}$. Thus an assignment f is feasible if for each distinct $u, v \in V$ and each $i \in f(u)$ and $j \in f(v)$ we have $|i-j| \notin T_{uv}$, and for each $v \in V$ and each distinct $i, j \in f(v)$ we have $|i-j| \notin T_{vv}$. We might add lists $L(v)$ of available channels, and insist that $\phi(v) \subseteq L(v)$.

The corresponding interference graph G has a node for each transmitter v, and distinct nodes u and v are adjacent if T_{uv} is non-empty. It is often convenient to think of the problem as being specified by the graph G together with a set T_e for each edge e of G, where always $0 \in T_e$. This is the T_e-sets model we met in Section 2.3.9, with its two special cases the constraint matrix model and the T-colouring model.

In order to keep the discussion reasonably brief some simplifications have naturally been made. For example, typically communication involves not one but two radio channels; a 'down-link' for signals from the base station transmitter to the moble station, and an 'up-link' for the signals back, perhaps at a fixed offset. Some useful references for further reading include [28, 39, 41, 49, 79, 86].

Acknowledgements I would like to acknowledge helpful comments from Stefanie Gerke and Bruce Reed.

References

[1] S.M. Allen and N. Dunkin, Frequency assignment problems: representations and solutions, Technical report, University of Glamorgan, 1997.

[2] S.M. Allen, D.H. Smith and S. Hurley, Lower bounding techniques for frequency assignment, *Discrete Mathematics* **197/198** (1999) 41 – 52.

[3] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela and M. Protasi, *Complexity and Approximation*, Springer, 1999.

[4] F. Barasi and J. van den Heuvel, Graph labelling, acyclic orientations, and greedy algorithms, manuscript, 2001.

[5] M. Bernstein, N.J.A. Sloan and P.E. Wright, On sublattices of the hexagonal lattice, *Discrete Mathematics* **170** (1997) 29 – 39.

[6] B. Bollobás, The chromatic number of random graphs, *Combinatorica* **8** (1988) 49 – 55.

[7] B. Bollobas, *Modern Graph Theory*, Graduate Texts in Mathematics 184, Springer, 1998.

[8] H. Breu and D.G. Kirkpatrick, Unit disk graph recognition is NP-hard, *Comput. Geom.* **9** (1998) 3 – 24.

[9] S. Ceroi, Clique number of intersection graphs of convex bodies, manuscript, 2001.

[10] G.J. Chang and D. Kuo, The $L(2,1)$-labeling problem on graphs, *SIAM J. Discrete Math.* **9** (1996) 309 – 316.

[11] G. Chang, L. Huang and X. Zhu, Circular chromatic numbers and fractional chromatic numbers of distance graphs, *Discrete Math.* **19** (1997) 223 – 230.

[12] B.N. Clark, C.J. Colbourn and D.S. Johnson, Unit disk graphs, *Discrete Mathematics* **86** (1990) 165 – 177.

[13] I. Csiszár, J. Körner, L. Lovász, K. Marton and G. Simonyi, Entropy splitting for antiblocking corners and perfect graphs, *Combinatorica* **10** (1990) 27 – 40.

[14] N. Dunkin, S.M. Allen, D.H. Smith and S. Hurley, Frequency assignment problems: benchmarks and lower bounds, Technical report UG-M-98-1, University of Glamorgan, 1998.

[15] P. Erdős, Some remarks on chromatic graphs, *Colloq. Math.* **16** (1967) 253 – 256.

[16] T. Erlebach, K. Jansen and E. Seidel, Polynomial-time approximation schemes for geometric graphs, Proceedings of the Twelth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2001), 2001, pp. 671-679.

[17] A. Gamst, Homogeneous distribution of frequencies in a regular hexagonal cell system, *IEEE Transactions on Vehicular Technology* **VT-31** (1982) 132 – 144.

[18] A. Gamst, Some lower bounds for a class of frequency assignment problems, *IEEE Transactions on Vehicular Technology* **VT-35** (1986) 8 – 14.

[19] M.R. Garey and D.S. Johnson, *Computers and Intractability*, Freeman, 1979.

[20] J.F. Georges and D.W. Mauro, Generalized vertex labelings with a condition at distance two, *Congressus Numerantium* **109** (1995) 141 – 159.

[21] J.F. Georges, D.W. Mauro and M.I. Stein, Labelling products of complete graphs with a condition at distance two, *SIAM J. Discrete Mathematics* **14** (2001) 28 – 35.

[22] S. Gerke, Channel assignment problems, DPhil thesis, University of Oxford, 2000.

[23] S. Gerke, Colouring weighted bipartite graphs with a co-site constraint, *Discrete Mathematics* **224** (2000) 125 – 138.

[24] S. Gerke and C. McDiarmid, Graph imperfection, *J. Comb. Th. B*, to appear.

[25] S. Gerke and C. McDiarmid, Graph imperfection II, *J. Comb. Th. B*, to appear.

[26] S. Gerke and C. McDiarmid, Channel assignment with large demands, manuscript, 2000.

[27] S. Gerke and C. McDiarmid, On the k-imperfection ratio of graphs, manuscript, 2001.

[28] R.A.H. Gower and R.A. Leese, The sensitivity of channel assignment to constraint specification, in *Proceedings of EMC97 Symposium*, Zurich, 131 – 136, 1997.

[29] A. Gräf, M. Stumpf and G. Weissenfels, On coloring unit disk graphs, *Algorithmica* **20** (1998) 277 – 293.

[30] J.R. Griggs and D. Der-Fen Liu, The channel assignment problem for mutually adjacent sites, *J. Combinatorial Theory A* **68** (1994) 169 – 183.

[31] J.R. Griggs and R.K. Yeh, Labelling graphs with a condition at distance two, *SIAM J. Discrete Math.* **5** (1995) 586 – 595.

[32] M. Grötschel, L. Lovász and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*, Springer-Verlag, 1988.

[33] A. Gyárfás, Problems from the world surrounding perfect graphs, *Zastosowania Matematyki Applicationes Mathematicae* XIX (1987) 413 – 441.

[34] W.K. Hale, Frequency assignment, *Proceedings of the IEEE* **68** (1980) 1497 – 1514.

[35] F. Harary and M. Plantholt, Graphs whose radio coloring number equals the number of nodes, in *Graph Colouring and Applications*, P. Hansen and O. Marcotte editors, CRM Proceedings and Lecture Notes 23, American Mathematical Society, 1999.

[36] F. Havet, Channel assignment and multicolouring of the induced subgraphs of the triangular lattice, *Discrete Mathematics* **233** (2001) 219 – 231.

[37] F. Havet and J. Zerovnik, Finding a five bicolouring of a triangle-free subgraph of the triangular lattice. *Discrete Mathematics*, to appear.

[38] J. van den Heuvel and S. McGuinness, Colouring the square of a planar graph, manuscript.

[39] J. van den Heuvel, R.A. Leese and M.A. Shepherd, Graph labelling and radio channel assignment, *J. Graph Theory* **29** (1998) 263 – 283.

[40] H.B. Hunt, M.V. Marathe, V. Radhakrishnan, S.S. Ravi, D.J. Rosenkrantz and R.E. Stearns, NC-approximation schemes for NP- and PSPACE-hard problems for geometric graphs, *J. Algorithms* **26** (1998) 238 – 274.

[41] S. Hurley and R. Leese (editors), *Models and Methods for Radio Channel Assignment*, Oxford University Press, to appear.

[42] S. Hurley, D.H. Smith and S.U. Thiel, FASoft: a system for discrete channel frequency assignment, *Radio Science* **32** 1921 – 1939, 1997.

[43] S. Janson, T. Łuczak and A. Ruciński, *Random Graphs*, Wiley, 2000.

[44] J. Janssen and K. Kilakos, Polyhedral analysis of channel assignment problems (part 1: tours), Research Report CDAM-96-17, LSE, August 1996.

[45] T.R. Jensen and B. Toft, *Graph Colouring Problems*, Wiley, 1995.

[46] S. Jordan and E.J. Schwabe, Worst-case performance of cellular channel assignment policies, *ACM Journal of Wireless Networks* **2** (1996) 265 – 275.

[47] R.A. Leese, A unified approach to the assignment of radio channels on a regular hexagonal grid, *IEEE Trans. Vehicular Technology* **46** (1997) 968 – 980.

[48] L. Lovász, Normal hypergraphs and the perfect graph conjecture, *Discrete Mathematics* **2** (1972) 253 – 267.

[49] V.H. MacDonald, The cellular concept, *The Bell System Technical Journal* **58** (1979) 15 – 41.

[50] E. Malesińska, S. Piskorz and G. Weißenfels, On the chromatic number of disk graphs, *Networks* **32** (1998) 13 – 22.

[51] M.V. Marathe, H. Breu, H.B. Hunt, S.S Ravi and D.J. Rosencrantz, Simple heuristics for unit disk graphs, *Networks* **25** (1995) 59 – 68.

[52] M.V. Marathe, V. Radhakrishnan, H.B. Hunt and S.S Ravi, Hierarchically specified unit disk graphs, *Theoret. Comput. Sci* **174** (1997) 23 – 65.

[53] T. Matsui, Approximation algorithms for maximum independent set problems and fractional coloring problems on unit disk graphs, *Discrete and Computational Geometry* (Tokyo, 1998) 194 – 200, Lecture Notes in Computer Science 1763, Springer, Berlin, 2000.

[54] C. McDiarmid, On the chromatic number of random graphs, *Random Structures and Algorithms* **1** No.4 (1990) 435–442.

[55] C. McDiarmid, A doubly cyclic channel assignment problem, *Discrete Applied Mathematics* **80** (1997) 263 – 268.

[56] C. McDiarmid, Counting and constraint matrices, manuscript, 1998.

[57] C. McDiarmid, Frequency-distance constraints with large distances, *Discrete Applied Mathematics* **223** (2000) 227 – 251.

[58] C. McDiarmid, Channel assignment and graph imperfection, to appear as a chapter in *Perfect Graphs*, J. Ramirez and B. Reed, editors, Wiley, 2001.

[59] C. McDiarmid, Random channel assignment in the plane, manuscript 2001.

[60] C. McDiarmid, On the span of a random channel assignment problem, manuscript 2001.

[61] C. McDiarmid, On the span in channel assignment problems: bounds, computing and counting, manuscript, 2001.

[62] C. McDiarmid and B.A. Reed, Colouring proximity graphs in the plane, *Discrete Mathematics* **199** (1999) 123 – 137.

[63] C. McDiarmid and B.A. Reed, Channel assignment and weighted colouring, *Networks* **36** (2000) 114 – 117.

[64] C. McDiarmid and B.A. Reed, Channel assignment and bounded tree-width graphs, manuscript, 2000.

[65] R.A. Murphy, P.M. Pardalos and M.G.C Resende, Frequency assignment problems, chapter in *Handbook of Combinatorial Optimization* Vol 4 (1999) (D.-Z. Dhu and P.M. Pardalos, editors).

[66] L. Narayanan and S. Shende, Static frequency assignment in cellular networks, *Proc. 4th Colloquium on Structural Information and Communications Complexity*, July 1997.

[67] J. Pach and P.K. Agarwal, *Combinatorial Geometry*, Wiley, 1995.

[68] A. Paster, MSc thesis, University of Oxford, 2000.

[69] M.D. Penrose, *Random Geometric Graphs*, forthcoming book.

[70] V. Raghavan and J. Spinrad, Robust algorithms for restricted domains, Proceedings of the Twelth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2001), 2001.

[71] A. Raychaudhuri, Intersection assignments, T-coloring, and powers of graphs, PhD thesis, Department of Mathematics, Rutgers University, NJ, 1985.

[72] A. Raychaudhuri, Further results on T-coloring and frequency assignment problems, *SIAM J. Discrete Mathematics* **7** (1994) 605 – 613.

[73] F.S. Roberts, T-colourings of graphs: recent results and open problems, *Discrete Math.* **93** (1991) 229 – 245.

[74] C.A. Rogers, *Packing and Covering*, Cambridge University Press, 1964.

[75] D. Sakai, Labelling chordal graphs: distance two condition, *SIAM J. Discrete Math.* **7** (1994) 133 – 140.

[76] E.R. Scheinerman and D.H. Ullman, *Fractional Graph Theory*, Wiley, 1997.

[77] N. Schnabel, S. Ubéda and J. Žerovnik, A note on upper bounds for the span of the frequency planning in celular networks, manuscript, 1999.

[78] R. Schneider, *Convex bodies: The Brunn - Minkowski Theory,* Encyclopedia of Mathematics and its Applications, vol. 44, Cambridge University Press, 1993.

[79] M. Shepherd, Radio Channel Assignment, DPhil thesis, University of Oxford, 1999.

[80] G. Simonyi, Imperfection ratio and graph entropy, manuscript, 1999.

[81] D.H. Smith and S. Hurley, Bounds for the frequency assignment problem, *Discrete Mathematics* **167/168** (1997) 571 – 582.

[82] D.H. Smith, S.M. Allen and S. Hurley, Lower bounds for channel assignment, in S. Hurley and R. Leese (editors), *Models and Methods for Radio Channel Assignment*, Oxford University Press, to appear.

[83] C.-E. Sundberg, Alternative cell configurations for digital mobile radio systems, *The Bell System Technical Journal* **62** (1983) 2037 – 2065.

[84] B.A. Tesman, Applications of forbidden distance graphs to T-colorings, *Congressus Numerantium* **74** (1990) 15 – 24.

[85] W.J. Watkins, S. Hurley and D.H. Smith, Evaluation of models for area coverage, Report to UK Radiocommunications Agency, Department of Computer Science, Cardiff University, December 1998.

[86] W. Webb, *The Complete Wireless Communications Professional*, Artech House Publishers, 1999.

[87] D.J.A. Welsh and G. Whittle, Arrangements, channel assignment, and associated polynomials, *Adv. in Appl. Math* **23** (1999) 375 – 406.

[88] D.B. West, *Introduction to Graph Theory*, Prentice Hall, 1996.

[89] R.M. Whitaker, S. Hurley and D.H. Smith, Frequency assignment heuristics for area coverage problems, Report to UK Radiocommunications Agency, Department of Computer Science, Cardiff University, September 2000.

[90] X. Zhu, Circular chromatic number: a survey, *Discrete Math.* **229** (2001) 371 – 410.

3

On the coloration of perfect graphs

F. Maffray

3.1 Introduction

We consider only finite graphs, without loops. Given an undirected graph $G = (V, E)$, a *k-coloring* of the vertices of G is a mapping $c : V \to \{1, 2, \ldots, k\}$ for which every edge xy of G has $c(x) \neq c(y)$. If $c(v) = i$ we say that v has color i. Those sets $c^{-1}(i)$ $(i = 1, \ldots, k)$ that are not empty are called the *color classes* of the coloring c. Each color class is clearly a stable set (i.e., a subset of vertices with no edge between any two of them), hence we will frequently view a coloring as a partition into stable sets. The graph G is called *k-colorable* if it admits a k-coloring, and the *chromatic number* of G, denoted by $\chi(G)$, is the smallest integer k such that G is k-colorable. We refer to [9, 16, 29] for general results on graph theory.

A classical result [59], and indeed one of the earliest, from complexity theory states that determining if a graph is k-colorable is an NP-complete problem for every fixed integer $k \geq 3$. (For $k = 2$ the problem is simply to determine if G is bipartite, which is an easy exercise.) For $k = 3$ the problem remains NP-complete even for triangle-free graphs with maximum degree four [71] (see also [32]). Approximation approaches to the chromatic number also seem to be doomed; indeed, it is known [38] that, unless P=NP, there is no polynomial-time algorithm that can find a $2\chi(G)$-coloring for every graph G (see [69] for even stronger results). These complexity results show that determining the chromatic number of a graph, or even, more modestly, trying to color it with not too many colors, is a very hard problem. Thus one feels that, for a class $\mathcal{C}$ of graphs to be such that there exists an algorithm finding the chromatic number of every graph in $\mathcal{C}$ in polynomial time, it must be that the graphs in $\mathcal{C}$ have a “strong structure”. There are not very many such classes (apart from trivial ones). One of the few outstanding classes of graphs for which the problem of the chromatic number becomes tractable is the class of *perfect graphs*. This article attempts to present a survey on methods for coloring the vertices of a perfect graph. It is not possible to review all the results, problems and questions arising in the domain of graph coloring in a short survey paper. We prefer to refer the interested reader the excellent book by Jensen and Toft [57]. For the specific subject of perfect graphs, see [10, 67, 83].

3.2 Basic definitions and notation

Let $G = (V, E)$ be an undirected graph. The *neighborhood* $N_G(v)$ of a vertex v is the set of vertices in G that are adjacent to v in G; when there is no ambiguity we will write $N(v)$. A chordless path (resp. cycle, clique) on k vertices is denoted by P_k (resp. C_k, K_k). A complete bipartite graph with bipartition classes of size p and q is denoted by $K_{p,q}$. A *hole* is a chordless cycle on at least four vertices. A hole is *odd* if it has an odd number of vertices. An *antihole* is the complement of a hole. The maximum size of a clique contained in a graph G is denoted by $\omega(G)$. The maximum size of a stable set of G is denoted by $\alpha(G)$. For two vertices u, v of G, $G + uv$ is the graph obtained by adding the edge uv, and G/uv is the graph obtained by identifying u, v into one vertex, adjacent to all vertices of $N_G(u) \cup N_G(v)$ (this operation is called contraction).

A graph G is *perfect* if the equality $\omega(H) = \chi(H)$ holds for every induced subgraph H of G. A *minimal imperfect* graph is not perfect but every proper subgraph is. Berge [7, 8, 9] conjectured:

Strong Perfect Graph Conjecture [8]: *Every minimal imperfect graph is an odd hole or the complement of an odd hole.*

A graph with no odd hole and no odd antihole is frequently called a *Berge graph.* A weaker conjecture of Berge was proved by Lovász:

Perfect Graph Theorem [64, 65] (See also [39]): *A graph is perfect if and only if its complement is perfect.*

Grötschel, Lovász and Schrijver [44] have shown that the problem of computing $\omega(G)$ and $\chi(G)$ can be solved in polynomial time for all perfect graphs. Their algorithm computes a certain value $\theta(G)$ which, if the graph is perfect, is equal to $\chi(\overline{G})$. Moreover, it is possible to implement this algorithm in such a way that a $\chi(G)$-coloring is produced if the graph is perfect. However, the algorithm is based on the ellipsoid method applied to convex bodies that are not necessarily polyhedra, and it is not easy to summarize. Thus it may be frustrating and opaque to the purely graph-theory minded. The results presented here, on the other hand, are always based on rather simple graph-theoretic concepts.

3.3 Bounds on the chromatic number

Lower bounds

An easy bound is $\chi(G) \geq n/\alpha(G)$; this follows from the fact that a $\chi(G)$-coloring is a partition into $\chi(G)$ stable sets. This inequality can be arbitrarily bad: taking a graph H with chromatic number k, and adding p isolated vertices, we obtain a graph G with $n = |V(H)| + p$ vertices and with $\alpha(G) = \alpha(H) + p$. For p arbitrarily large the ratio $n/\alpha(G)$ tends to 1 while $\chi(G) = k$.

Another lower bound for $\chi(G)$ is $\omega(G)$, since the vertices of any clique in G must receive different colors. This bound too can be arbitrarily bad. To illustrate this, Mycielski [82] proposed for every integer $k \geq 2$ a graph G_k which has chromatic number k and contains no triangle, i.e., $\omega(G_k) = 2$. Such graphs can be defined recursively as follows. Set $G_2 = K_2$. For $k \geq 3$, the graph G_k is obtained by taking a copy H of G_{k-1}, for every vertex $v \in H$ adding a vertex v^* with an edge from v^* to each vertex of $N_H(v)$, and adding a vertex z with an edge from z to each $*$-vertex. It is easy to check that G_k contains no triangle. To see that $\chi(G_k) = k$, suppose on the contrary that G_k is $k-1$-colorable; assume that z receives color $k-1$; for each vertex $v \in H$ of color $k-1$, assign to v the color of v^* instead of $k-1$; it is easy to check that this yields a $k-2$-coloring of H, a contradiction.

Erdős [33] (see also [87]) proved that for all integers k, g there exists a graph G with $\chi(G) = k$ and such that G contains no cycle of length strictly less than g. This result is non-constructive and is a famous example of the power of the Probabilistic Method developed by Erdős. Later, Lovász [63] gave a constructive proof of the existence of such graphs.

Upper bounds

Let Δ denote the maximum degree in a graph G. An easy observation is that every graph G has $\chi(G) \leq \Delta + 1$. Indeed, take any vertex $v \in G$, and consider a $\Delta+1$-coloring of $G - v$; such a coloring may be assumed to exist by induction on the size of V. Since v has at most Δ neighbours, it is possible to find among the colors $1, \ldots, \Delta + 1$ one that is not used on any neighbour of v, and which can be assigned to v.

A famous result of Brooks [17] states that a graph satisfies $\chi(G) = \Delta+1$ if and only if either $\Delta = 2$ and G has a connected component that is an odd cycle, or G has a connected component that is a $K_{\Delta+1}$. Thus every graph different from these two exceptional cases has $\chi(G) \leq \Delta$. See also Lovász [66] for an alternate proof of Brooks's theorem. Again it is easy to find graphs for which the difference between $\chi(G)$ and Δ can be arbitrarily bad, e.g., $G = K_{1,q}$. Reed [85] proved that for large enough Δ, a graph G has $\chi(G) \geq \Delta$ if and only if G contains a K_Δ. Strengthenings of these results were obtained by Molloy and Reed [81]: in particular, for given Δ, deciding if a graph ¡ith maximum degree Δ is k-colorable is in the class P if $k \geq \Delta - e\sqrt{\Delta}$.

Exact value

It is possible to compute the chromatic number of a graph exactly, with a method that may take much time but is combinatorially simple. Let $G = (V, E)$ be a graph. If G is a complete graph then $\chi(G) = |V|$. If G is not a complete graph, consider two non-adjacent vertices u, v. Any coloring c of G has either $c(u) \neq c(v)$, and in this case c is a coloring of $G + uv$, or $c(u) = c(v)$, and in this case c is a coloring of G/uv. The converse is also true. It follows that $\chi(G) = \min\{c(G + uv), c(G/uv)\}$. We can then repeat this for the graphs $G + uv$ and G/uv. Thus we obtain a recursive procedure which finds the exact value of $\chi(G)$ and indeed finds a $\chi(G)$-coloring of G. However, the tree representing this recursive procedure may have exponentially many nodes.

3.4 Edge coloring

Let us mention briefly the problem of coloring the edges of a graph (rather than the vertices) in such a way that no two adjacent edges receive the same color. Clearly, every edge-coloring of a simple graph G with maximum degree Δ uses at least Δ colors. Vizing [92] proved that every simple graph G with maximum degree Δ can be edge-colored with at most $\Delta + 1$; more generally, every multigraph with maximum degree Δ and maximum edge multiplicity λ can be edge-colored with at most $\Delta + \lambda$ colors. Holyer [53] proved that it is an NP-complete problem to decide if a given simple graph can be edge-colored with Δ colors. This problem seems to remain difficult even for fairly small classes of graphs, see e.g., [26, 27].

The problem of coloring the edges of a graph G is equivalent to coloring the vertices of its *line-graph*, which is the graph $L(G)$ whose vertices are the edges of G and whose edges are the pairs of adjacent edges in G. Clearly every vertex-coloring of $L(G)$ is an edge-coloring of G and vice-versa. As proved by Beineke [6], line-graphs of simple graphs are exactly the graphs that do not contain as an induced subgraph any of a list of nine forbidden subgraphs; similarly, line-graphs of multigraphs are characterized by a list of seven forbidden subgraphs. Thus the edge-coloring problem is just the vertex-coloring problem for a certain subclass of graphs. Kierstead and Schmerl [60] proved that by excluding only the *claw* $K_{1,3}$ and the graph $K_5 - e$ (which are among the nine forbidden subgraphs above) one obtains a class of graphs G with $\chi(G) \leq \omega(G) + 1$.

3.5 Sequential Algorithms

One of the simplest kind of heuristics that can be imagined to color the vertices of a graph $G = (V, E)$ is the following.

> *Sequential Algorithm:*
> Input: An ordering $v_1 < \cdots < v_n$ of the vertices of a graph G.
> Output: a coloring c of the vertices of G.
> First step: give color 1 to vertex v_1.
> Main Step: For $i := 2$ to n, if some already used color does not appear on any neighbour v_j of v_i with $j < i$, then assign to v_i any color from $C_i \setminus U_i$; else assign to v_i the color $1 + \max C_i$.

As defined here, the Sequential Algorithm is not deterministic, because we may have a choice for the color to be assigned at each step. To make it deterministic, we can add a rule for the choice of an available color (when there is a choice). Usually one chooses the smallest available integer. This gives the so-called GREEDY COLORING ALGORITHM. We note that the greedy coloring algorithm is easy to implement so as to work in time $O(|V| + |E|)$.

The quality of the solution produced by the greedy coloring algorithm certainly depends on the ordering of the vertices given as input. Of course, a lucky choice for the ordering could produce an optimal coloring. Indeed suppose that $S_1, \ldots, S_q$ are stable sets that partition $V(G)$ with $q = \chi(G)$ and consider the ordering obtained by putting first the vertices of S_1, then those of S_2 and so on. If a kind person gives us this ordering as input, it is easy to check that the greedy coloring along this ordering will produce a coloring with q colors. But in general no one has a way to guess a good ordering in polynomial time (unless P=NP) since the coloring problem is NP-complete. Moreover, an ordering that is good for G might be bad for some subgraphs of G.

This idea can be refined a little by choosing for the last vertex of the ordering a vertex of small degree (hence increasing our chances that we would not need a new color for v) [93]. Thus we may order the vertices as $v_1 < \cdots < v_n$ so that $d(v_1) \geq \cdots \geq d(v_n)$ and apply the greedy algorithm on this ordering. This implies the following bound:

$$\chi(G) \leq \max \min\{i, d(v_i) + 1\}.$$

See [58, 89] for comments on the performance of this method.

Perhaps a better ordering consists in choosing for v_n a vertex of minimum degree in G, then for v_{n-1} a vertex of minimum degree *among the vertices of* $G - v_n$, etc (see [75, 76]). Thus, if $v_1 < \cdots < v_n$ is an ordering determined by this procedure, and d'_i denotes the degree of v_i in the subgraph induced by $v_1, \ldots, v_i$, we have:

$$\chi(G) \leq \max \min\{i, d'_i + 1\}.$$

Consequently, if $\delta(G)$ is the minimum degree in G, we obtain:

$$\chi(G) \leq \max\{\delta(H) + 1 \mid H \text{ is an induced subgraph of } G\}.$$

Let us denote by $\beta(G)$ the value on the right-hand side of the preceding inequality. So we have $\chi(G) \leq \beta(G)$ for every graph. We note that, if v is a vertex of minimum degree in G, i.e., $d(v) = \delta(G)$, then $\beta(G) = \max\{d(v) + 1, \beta(G - v)\}$. This suggests a simple way of computing the value of $\beta(G)$ for every graph.

Markossian, Gapsparian and Reed [74] call a graph G *β-perfect* if every induced subgraph H of G satisfies $\chi(H) = \beta(H)$. They note that odd cycles are β-perfect, while even cycles are not. Thus β-perfect graphs in general are different from perfect graphs. However, they seem to show some similarities. It was proved in [74] that every graph G such that G and $\overline{G}$ contain no even hole C_{2k} $(k \geq 2)$ is β-perfect, and that every graph in which every even cycle has at least two chords is β-perfect. This result was recently generalized in [28].

Perfectly orderable graphs

Chvátal [19] proposed to call *perfectly orderable* any graph G that admits an ordering $<$ such that the greedy algorithm produces an optimal coloring for every induced subgraph H of G (with the induced ordering on H). Such an ordering is called *perfect*. One of the interesting features here is that there is a simple characterization of perfect orderings.

Theorem 3.5.1 ([19]) *A linear ordering $<$ of a graph G is perfect if and only if there is no induced P_4 $abcd$ with edges ab, bc, cd and with $a < b$ and $d < c$.*

A P_4 $abcd$ with edges ab, bc, cd and with $a < b$ and $d < c$ is called an *obstruction*. This theorem implies that the class of perfectly orderable graphs is in the class NP, which was not obvious from the definition: indeed, we only need to verify that a given ordering is such that the graph has no obstruction. On the other hand, finding such an ordering may be hard; in fact, the recognition of perfectly orderable graphs is an NP-complete problem [80].

Another interesting feature is revealed in the next theorem.

Theorem 3.5.2 ([19]) *Perfectly orderable graphs are perfect.*

Examples of perfectly orderable graphs: Perfectly orderable graphs generalize two famous classes of perfect graphs: the *triangulated* graphs, and the *comparability* graphs.

A triangulated graph (or *chordal graph*) is a graph that has no hole as an induced subgraph. Triangulated graphs have been extensively studied since the last 1950s (see [42]), and we can only briefly summarize some basic results about their structure. A *simplicial vertex* is a vertex whose neighbourhood induces a complete subgraph.

Theorem 3.5.3 (see [9, 42]) *Let G be a triangulated graph that is not a clique. Then G admits at least two non-adjacent simplicial vertices.*

This theorem yields a method to find a perfect ordering for a triangulated graph G. Let v_n be a simplicial vertex of G, then v_{n-1} be a simplicial vertex of $G - v_n$, etc. Let us call the ordering $v_1 < \cdots < v_n$ a *simplicial elimination ordering* for G. It is easy to check that a simplicial elimination ordering has no obstruction. Thus, and by Theorem 3.5.1, any simplicial elimination ordering is perfect. In fact, using the greedy algorithm on a simplicial elimination ordering is still the most efficient way to color optimally the vertices of a triangulated graph.

A comparability graph is a graph that admits a transitive orientation of its edges, that is, an acyclic orientation such that whenever $\vec{uv}$ and $\vec{vw}$ are two arcs then u and w are adjacent and $\vec{uw}$ is an arc of the orientation. Comparability graphs too have been the object of much study, starting with the seminal work of Ghouila-Houri [40], Gilmore and Hoffman [41] and Gallai [37, 73]. Note that even if a transitive orientation of a comparability graph G is not given a priori, it is nonetheless possible to find one in linear time [77]. Given such an orientation, consider a topological ordering $<$ that is compatible with the orientation (i.e., if $\vec{uv}$ is an arc then $u < v$); since the orientation is acyclic it is always possible to find such an ordering. Then it is a routine matter to check that any such ordering has no obstruction. Thus, and by Theorem 3.5.1, such an ordering is perfect. Again, using the greedy algorithm on such an ordering is still the most efficient way to color optimally the vertices of a comparability graph.

Apart from triangulated graphs and comparability graphs, there are actually many classes of perfect graphs that were proved to be perfectly orderable. It is not possible to mention all of them here, as they are numerous and many have lengthy technical definitions. We refer the interested reader to [52] for the most recent recent and complete survey, and to [51] for algorithms on perfectly orderable graphs with vertex weights.

3.6 Sequential coloring with bichromatic exchange

After looking at the greedy (or 'sequential') coloring algorithm, one may have the idea, at the i-th step, of trying to improve the current coloring on the vertices $v_1, \ldots, v_{i-1}$ before making a choice of a color for the vertex v_i. Hopefully the improvement will reduce the chance of having to use a new color for v_i.

One way to modify a coloring c of a graph H is as follows. Consider two color classes S, T of c. The subgraph $H[S \cup T]$ is bipartite, and it may have several connected components. If this subgraph has several connected components, then, by swapping ('exchanging') the colors along one component, we obtain a coloring which is not the same as the original one and might be "better" for later purposes. This is called *bichromatic exchange.* Given a graph G, a vertex v of G and a coloring c of $G - v$, let us say that two color classes S, T of c are *indifferent* for v if no connected component of $G_{S,T}$ contains a neighbor of v in S and a neighbor of v in T. This is equivalent to saying that the subgraph induced by $S \cup T \cup \{v\}$ is bipartite. Now we can define an improved form of sequential algorithm, involving bichromatic exchanges, called Sequential with Bichromatic Exchange (SBX).

```
SBX Algorithm:
Input: An ordering $v_1 < \cdots < v_n$ of the vertices of a graph $G$.
Output: A coloring of the vertices of $G$.
First assign color 1 to $v_1$.
Main Step: For $i := 2$ to $n$, if some already used color does not appear on any neighbour $v_j$ of $v_i$ with $j < i$, then give this color to $v_i$; else, if there are two colors $S$ and $T$ that are indifferent for $v_i$, then perform the $S,T$-bichromatic exchanges on the components of $G_{S,T}$ that contain the neighbours of $v_i$ colored by $S$, and give the color $S$ to $v_i$; else give a new color to $v_i$.
```

Historically, this idea was exploited early in the proof of Brooks's theorem [17] that every connected graph G different from an odd cycle or a complete graph has $\chi(G) \leq \Delta$; indeed, this proof is constructive and uses bichromatic exchanges to find a Δ-coloring of such a graph G.

We can note that the proof of Vizing's theorem [92] that every simple graph G can be edge-colored with at most $\Delta + 1$ colors also uses bichromatic exchanges on the colored edges; looking at the line-graph of G, one sees that these are exactly bichromatic exchanges on the vertices of $L(G)$. Hence we have another application of bichromatic exchanges.

This idea was also exploited by Tucker [90] for coloring the vertices of diamond-free perfect graphs. The *diamond* is the graph $K_4 - e$, with four vertices and five edges (see Figure 3.1). Tucker [90] proved that in a diamond-free graph G with no odd hole, if the edges that do not lie in a triangle are removed (thus obtaining a graph G' with no odd hole), there exists in G' a vertex v whose neighbourhood induces at most two cliques with no edges between them. Tucker [90] then proved that, given any $\omega(G' - v)$ coloring c of $G' - v$, either $\omega(G') = \omega(G' - v) + 1$ (and we use a new color for v), or c has two color classes that are indifferent for v, thus $\omega(G') = \omega(G' - v)$ and G' can be $\omega(G')$-colored (as in the main step of Algorithm SBX). Finally, adding back the edges that were removed, the

$\omega(G')$ coloring of G' can be turned into an $\omega(G)$-coloring of G using again only bichromatic exchanges.

Another example of coloring the vertices of a graph sequentially and using bichromatic exchanges is the algorithm of Hsu and Nemhauser [56] for coloring any claw-free perfect graph G with $\omega(G)$ colours. In that algorithm, given a vertex v and an optimal coloring c of $G-v$, not just one but several bichromatic exchanges may be necessary to "improve" the coloring c.

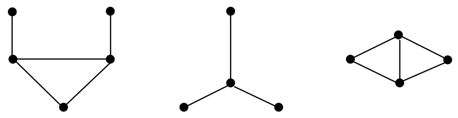


Figure 3.1. bull, claw, diamond

Just as with the greedy algorithm, every graph admits an ordering such that any execution of algorithm SBX on this ordering provides an optimal coloring. Call an ordering of the vertices of a graph *SBX-perfect* if for every induced ordered subgraph the algorithm SBX produces an optimal coloring. Likewise call a graph *SBX-perfect* if it admits an SBX-perfect ordering. Unlike Chvátal's perfectly orderable graphs, the SBX-perfect graphs are not all perfect: for example every odd hole is in this class (every ordering of the vertices of an odd hole is SBX-perfect). It is not known whether the recognition problem for SBX-perfect graphs is in the class P, or even in NP. The definition of SBX-perfect graphs was stated in [72] (extending results from [70]), where the following result was established. Call a vertex v of a graph G *excellent* if it does not lie in an odd hole of G and its neighborhood contains no induced P_4, $3K_2$ or P_3+K_2. An ordering $v_1 < v_2 < \cdots < v_n$ of the vertices of a graph G is then called excellent if, for every $i = 1, 2, \ldots, n$, vertex v_i is excellent in the subgraph of G induced by $v_1, v_2, \ldots, v_i$. It was proved in [72] that Algorithm SBX applied along an excellent ordering of the vertices of a graphs produces a coloring with $\omega(G)$ colors. Consequently (noting that an excellent ordering of a graph remains excellent in each induced ordered subgraph) the graphs that admit an excellent ordering are perfect. It can be verified that the class of graphs having an excellent ordering contains the bipartite graphs, the triangulated graphs (any simplicial vertex is excellent), and the graphs that contain no diamond and no odd hole. There are perfect graphs such that neither they nor their complements have excellent vertices, for example the self-complementary graph obtained from a P_4 by substituting each vertex with a P_4. This graph is actually perfectly orderable, hence the class of perfectly orderable graphs and the complementary class are different and incomparable with the class of excellent graphs.

3.7 Sequential coloring with trichromatic exchange

The following sequential method for coloring some perfect graphs was proposed in [1, 3], based on an idea of Hacène Aït Haddadène. We recall for now that Tucker [91] found a combinatorial algorithm for coloring every K_4-free perfect graph G with $\omega(G)$ colors, in time $O(|V|^3)$. We will describe this algorithm in the next section.

Let v be a vertex of a perfect graph G. Assume that $G - v$ admits a coloring c using $\omega(G) \geq 3$ colors, with color classes $S_1, \ldots, S_{\omega(G)}$. Suppose that there exist three distinct colors i, j, k such that the subgraph H induced by $S_i \cup S_j \cup S_k \cup \{v\}$ contains no K_4. Thus this subgraph is a K_4-free perfect graph, and so we can apply Tucker's algorithm to H in order to color it with three colors. Hence, keeping the other $\omega(G) - 3$ colors, we get an $\omega(G)$-coloring of G. If a vertex v has the property that for every coloring c of $G - v$ there are three color classes whose union with v form a K_4-free subgraph, one can call v a *Tucker vertex.* It may then be interesting to find classes of graphs that always have a Tucker vertex. This question was investigated in [1, 2, 3, 22, 43] for several classes of graphs; as their definitions are quite technical, we refer the interested reader directly to these articles.

3.8 Coloring by contraction

As observed at the beginning, if u, v are two non-adjacent vertices in a graph G, then any coloring of G/uv is a coloring of G, simply by assigining to u and v the color of the contracted vertex uv in G/uv, and leaving the same color for all other vertices. This procedure may be called "decontracting" the coloring; clearly it is a simple and (timewise) efficient procedure. However, the choice of the contracted pair is crucial: for example, if G is a path P_4 $uabv$, then $\chi(G) = 2$ while $\chi(G/uv) = 3$; in this case, it was not a good idea to contract the pair u, v. In a general graph G, it will not be possible (unless P=NP) to guess a pair of non-adjacent vertices u, v such that $\chi(G) = \chi(G/uv)$, for otherwise we would be able to find a $\chi(G)$-coloring of G after at most $n - 1$ contractions. Note that guessing such a pair u, v is equivalent to assuming that there is an optimal coloring in which u, v have the same color. As it turns out, the domain of perfect graphs offers more hope in this direction than general graphs. This section will illustrate this point. The main definition here is the following.

Definition 3.8.1 (Even pair [79]) *Two non-adjacent vertices x, y in a graph G form an even pair if every induced path between them has an even number of edges.*

Before exploring the notion of even pairs, let us mention one of the most astute applications of the method of coloring by contraction for perfect graphs. Indeed Tucker [91] gave a polynomial-time algorithm that colors every K_4-free perfect graph G with 3 colors, as follows. If G is diamond-free, use the other algorithm due to Tucker [90] and mentioned in Section 3.6. If G contains a diamond, then observe the two vertices of degree two in the diamond must necessarily have the same color in every 3-coloring of G; so it is natural to contract them. Repeating this for every diamond, we end up with a diamond-free graph G'. Unfortunately, the graph G' might not be perfect, as it could contain some odd hole. (We will see below that the contraction of even pairs ensure that no odd hole is created, but the vertices that are contracted by Tucker are not necessarily even pairs.) However, Tucker found in this specific case a way around the problem caused by the appearance of odd holes, by decomposing the graph, see [91].

General graph-coloring heuristics based on the contraction of non-adjacent vertices have been proposed by Dutton and Brigham [31] and by Hertz [48]. Dutton and Brigham choose at each step a pair x, y that has the largest number of common neighbours among all non-adjacent pairs in G. Hertz fixes a vertex x and, as long as x has non-neighbours, picks a non-neighbour y of x that has the most common neighbours with x. The vertices x, y are then contracted and the procedure is iterated until a clique K is obtained. One can get a $|K|$-coloring of G by proceeding backwards along the contraction sequence. See [31, 48] for comments on the performance of these heuristics.

Now we return to the study of even pairs. The following results motivate our interest in even pairs and the contraction operation.

Lemma 3.8.1 *Let G be a graph that contains an even pair $\{x, y\}$. Then,*

1. *$\omega(G/xy) = \omega(G)$;*
2. *$\chi(G) = \chi(G/xy)$.*
3. *If G is perfect then G/xy is perfect.*
4. *If G contains no odd hole then G/xy contains no odd hole;*
5. *If G contains no antihole then G/xy contains no antihole different from $\overline{C}_6$.*
6. *If G contains no odd antihole then G/xy contains no odd antihole;*
7. *If G is a Berge graph then G/xy is a Berge graph.*

More precisely, it can be prove that, whenever x, y is an even pair in a graph G, if Q is a largest clique of G/xy, then either Q does not contain vertex xy and is a largest clique of G, or Q contains xy and then one of $Q \cup \{x\}, Q \cup \{y\}$ is a largest clique of G. In parallel, if c is a $\chi(G)$ coloring of G, then, using bichromatic interchange and the fact that x, y is an even pair, one can find a $\chi(G)$-coloring c' of G in which x, y have the same color (hence c' is a coloring of G/xy). A precise proof of these results can be found in [36, 55].

Lemma 3.8.2 (The Even Pair Lemma [14, 79]) *No minimal imperfect graph contains an even pair.*

Meyniel [79] defined the following two classes of graphs. A graph G is a *strict quasi-parity* (SQP) graph if every induced subgraph of G either has an even pair or is a clique. A graph G is a *quasi-parity* (QP) graph if every induced subgraph of G on at least two vertices either has an even pair or its complement has an even pair. The Even Pair Lemma implies that every strict quasi-parity graph is perfect; the Even Pair Lemma and Lovász's Perfect Graph Theorem implies that every quasi-parity graph is perfect. These two classes are very interesting, but they are somewhat out of the scope of this article. We refer the interested reader to [34, 35] for more information on strict quasi-parity graphs and quasi-parity graphs.

Theorem 3.8.1 ([15]) *It is co-NP-complete to decide whether a graph admits an even pair.*

Despite this theorem, since contracting an even pair of vertices in a perfect graph G is a good operation from the point of view of coloring, we may want to iterate it as long as possible. Hopefully the graph G' we obtain at the end of such a sequence of even-pair contractions will be easy to color (for some other reason), and thus, going backward along the sequence and decontracting the coloring we will obtain an optimal coloring of G. The easiest graphs to color are complete graphs, and this leads to the following definitions.

Definition 3.8.2 ([13]) *A graph G is even-contractile if there exists a sequence G_0, G_1, ..., G_k of graphs such that $G_0 = G$, G_k is a clique, and, for $i \leq k-1$, G_{i+1} is obtained from G_i by contracting an even pair of G_i. A graph G is perfectly contractile if every induced subgraph of G is even contractile.*

Facts 1 and 2 imply that *every perfectly contractile graph is perfect.* Several classical families of perfect graphs are perfectly contractile, especially weakly triangulated graphs, Meyniel graphs and perfectly orderable graphs.

As an illustration, let $G_0, \ldots, G_k$ be the sequence of graphs as in Figure 3.2. For $i \leq k-1$, G_{i+1} is obtained from G_i by contracting the even pair $\{x_i, y_i\}$. Since G_k is a clique, we have an optimal coloring and a largest clique of G_k with no difficulty. We can then work backwards and decontract the coloring to find a largest clique and an optimal coloring of G_0. We note that the size of the largest clique in G_0 is equal to its chromatic number.

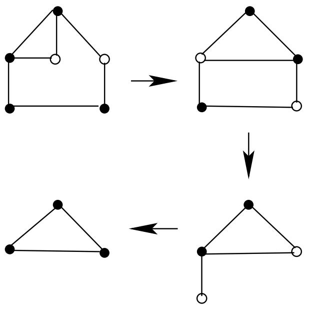


Figure 3.2. A sequence of even-pair contractions

3.8.1 Perfectly orderable graphs

It was proved by Hertz and de Werra [49] that every non-complete perfectly orderable graph has an even pair whose contraction yields a new perfectly orderable graph. Thus perfectly orderable graphs are perfectly contractile. However, the proof of [49] uses a given perfect ordering in order to find such an even pair. But it is NP-complete to determine if a graph is perfectly orderable [80]. Thus, if an oracle tells us that a graph G is perfectly orderable but does not show a perfect ordering of G, we can probably not find such an ordering. It is still possible to determine which pairs of vertices form even pairs in polynomial time in a perfectly orderable graph [4], but after contracting an arbitrary even pair in G, we do not know and can probably not check whether the resulting graph is perfectly orderable. There may be a sophisticated way of finding quickly an even pair whose contraction yields another perfectly orderable graph, but this problem is open and seems hard.

3.8.2 Weakly triangulated graphs

A graph is *weakly triangulated* if it contains no hole or antihole of length at least five. These graphs generalize triangulated graphs and their complements. Hayward [45] proved that all weakly triangulated graphs are perfect. Two vertices x, y in a graph G form a *2-pair* if every chordless path between them has length two. Hayward, Hoàng and Maffray [46] proved that every weakly triangulated graph which is not a clique contains a 2-pair. Moreover, the contraction of any 2-pair in a weakly triangulated graph yields a weakly triangulated graph. Thus, every weakly triangulated graph is perfectly contractile.

A nice feature of 2-pairs is that they are not hard to find; more precisely, it is easy to check that two non-adjacent vertices x, y in a graph G form a 2-pair if and only if x and y lie in different connected components of the subgraph $G - N(x) \cap N(y)$, a condition which can be tested in time $O(m+n)$ for a given pair. Thus, we can find a 2-pair in a graph (if it contains any) in $O(n^2m)$ time. Consequently, for every weakly triangulated graph G we can find, in $O(n^3m)$ time, a sequence of 2-pair contractions that reduces G to a clique, and we can use such a sequence to find an optimal coloring and largest clique in G. See [46] for the details and for $O(n^3m)$ algorithms that solve the maximum clique and minimum coloring problems on weakly triangulated graphs, and also solve the *weighted* versions of these optimization problems in $O(n^4m)$ time. Later, Arikati and Pandu Rangan [5] have developed an $O(nm)$ algorithm to find a 2-pair (if any) in a graph, thus entailing a speedup in the optimization algorithms. Spinrad and Sritharan [88] further improved on the complexity of the weighted version of the four optimization problems.

3.8.3 Meyniel graphs

Henry Meyniel studied the graphs in which every odd cycle of length five or more has at least two chords, and he proved in [78] that they graphs are perfect. It has then become usual to call them *Meyniel graphs*. Later, he showed that every such graph is either a clique or contains an even pair [79].

Using even-pair contractions to color Meyniel graphs could be a problem, as there are non-complete Meyniel graphs which do not contain any even pair whose contraction yields a Meyniel graph: consider the graph obtained by substituting every vertex in a C_6 by a pair of adjacent vertices (this example is due to Sarkossian, and Hougardy; see also [12]). Hertz [47] proposed to call a graph G *quasi-Meyniel* if it contains no odd hole and there exists a vertex t of G such that every edge which is the only chord of some odd cycle is incident with t (any such vertex is called a *tip* of G). Note that every Meyniel graph is quasi-Meyniel. Hertz [47] proved that every non-complete quasi-Meyniel graph contains an even pair whose contraction yields a quasi-Meyniel graph. More precisely, Hertz proved that if t is a tip of a quasi-Meyniel graph G and t is not adjacent to every other vertex, and if u is a non-neighbour of t with the most common neighbours with t, then t, u form an even pair in G, G/tu is quasi-Meyniel, and the contracted vertex tu is a tip of G/tu.

It follows that every non-complete quasi-Meyniel graph has an even pair whose contraction yields a quasi-Meyniel graph, and that all quasi-Meyniel graphs are perfectly contractile. We obtain an $O(nm)$ algorithm for the maximum clique and minimum coloring problems on Meyniel graphs. This is the fastest way to color Meyniel graphs.

There are several other classes of graphs that might be perfectly contractile, in particular *strongly perfect graphs* [11], *alternately orientable graphs* [50], and graphs in the class Bip* [20]. See [34, 35] for more details.

3.8.4 Structure of perfectly contractile graphs

It is easy to see that antiholes and odd holes are not perfectly contractile, as they have no even pair. A *refinement of* $\overline{C}_6$ is any graph that can be obtained from a $\overline{C}_6$ by subdividing the edges that do not lie in a triangle. An *odd refinement of* $\overline{C}_6$ is a refinement of $\overline{C}_6$ in which each of the three subdivided paths has odd length. Everett and Reed (see [34, 35]) have conjectured that *a graph is perfectly contractile if and only if it contains no antiholes, no odd holes and no odd refinement of* $\overline{C}_6$. This conjecture on the charaterization of perfectly contractile graphs is still open. It has been verified for a few classes, in particular planar graphs [62], bull-free graphs [24], and claw-free graphs [61].

3.9 Other methods

The preceeding sections gave examples of classes of (usually perfect) graphs that can be optimally colored using either the sequential method, or the method by contraction, or a combination of the two (e.g., for K_4-free graphs). Here we show some other methods.

Coloring the vertices of a co-comparability graph G is equivalent to partitioning the vertices of its complement $\overline{G}$ into cliques. Given a transitive orientation of $\overline{G}$, every clique is a directed paths in this transitive orientation. Build a graph B as follows: replace each vertex x of $\overline{G}$ by two vertices, a "source" x^+ and a "sink" x^-, so that x is replaced by x^+ in all incoming arcs to x and by x^- in all outgoing arcs from x. Clearly B is bipartite, and it is not had to check that a partition of $\overline{G}$ into directed paths corresponds to a matching in B. Thus our initial problem of finding an optimal coloring for G is reduced to finding a maximum matching in a bipartite graph B, a well-solved problem [68]. Note that this method also applies in the special when G is the complement of a bipartite graph; in this case we simply have $B = \overline{G}$. See also [18] and [51] for the weighted version of these problems.

Recently, Roussel and Rusu [86] gave an $O(n^2)$ algorithm which colors Meyniel graphs without using even pairs. This novel method is based on a Lexicographic Breadth-First Search of the graph. With every vertex a label is associated (the label is a word carrying information on the neighbours of the vertex and on the colors). At each step, the next vertex to color as well as the color to assign to this vertex is determined by an examination of the labels. Thus this method could be considered as a sequential method with a sophisticated rule of choice, except that the ordering is not given all at once but is determined along the way.

De Figueiredo and Maffray [23] give a polynomial-time algorithm to color bull-free perfect graphs. The algorithm works by decomposing the graph into pieces, along a decomposition tree that has a linear number of nodes. The leaves of the tree are graphs that are either comparability graphs or co-comparability graphs or weakly triangulated graphs, and hence can be colored with the methods above. It is then shown how to combine coloring of the children of a node of the tree into a coloring of the node. The determination of this decomposition tree uses many results from [21, 24].

References

[1] H. Aït Haddadène, S. Gravier. On weakly diamond-free Berge graphs. *Disc. Math.* 159 (1996), 237–240.

[2] H. Aït Haddadène, S. Gravier, F. Maffray. An Algorithm for coloring some perfect graphs. *Disc. Math.* 183 (1998), 1–16.

[3] H. Aït Haddadène, F. Maffray. Coloring degenerate perfect graphs. *Disc. Math.* 163 (1997), 211–215.

[4] S.R. Arikati, U.N. Peled. A polynomial algorithm for the parity path problem on perfectly orientable graphs. *Disc. App. Math.* 65 (1996), 5–20.

[5] S.R. Arikati, C. Pandu Rangan. An efficient algorithm for finding a two-pair, and its applications. *Disc. App. Math.* 31 (1991), 71–74.

[6] L.W. Beineke.Characterizations of derived graphs.*J. Comb. Th.* 9 (1970), 129–135.

[7] C. Berge, Les problèmes de coloration en théorie des graphes. *Publ. Inst. Stat. Univ. Paris* 9 (1960), 123–160.

[8] C. Berge.Färbung von Graphen, deren sämtliche bzw. deren ungerade Kreise starr sind (Zusammenfassung).*Wiss. Z. Martin Luther Univ. Math.-Natur. Reihe*, 10 (1961), 114–115.

[9] C. Berge.*Graphs.*North-Holland, Amsterdam/New York, 1985.

[10] C. Berge, V. Chvátal (editors). *Topics on Perfect Graphs.* Ann. Disc. Math. 21 (1984), North Holland, Amsterdam.

[11] C. Berge, P. Duchet. Strongly perfect graphs. In *Topics on Perfect Graphs*, C. Berge and V. Chvátal, editors, *Ann. Disc. Math.* 21 (1984), 57–62, North Holland, Amsterdam.

[12] M.E. Bertschi. *La colorabilité unique dans les graphes parfaits*, PhD thesis, Math. Institute, University of Lausanne, Switzerland, 1988.

[13] M. E. Bertschi, Perfectly contractile graphs. *J. Comb. Th. B* 50 (1990), 222–230.

[14] M. E. Bertschi, B.A. Reed. A note on even pairs. *Disc. Math.* 65 (1987), 317–318.

[15] D. Bienstock, On the complexity of testing for odd holes and odd induced paths. *Disc. Math.* 90 (1991), 85–92.

[16] B. Bollobás. *Modern Graph Theory.* Grad. Texts in Math. 184, Springer, 1998.

[17] R.L. Brooks.On colouring the nodes of a network.*Proc. Cambridge Phil. Soc.* 37(1941), 194–197.

[18] K. Cameron.Antichain sequences.*Order*, 2 (1985), 249–255.

[19] V. Chvátal, Perfectly ordered graphs, In *Topics on Perfect Graphs*, C. Berge and V. Chvátal, editors, *Ann. Disc. Math.* 21 (1984), 63–68, North Holland, Amsterdam.

[20] V. Chvátal, Star cutsets. *J. Comb. Th. B* 39 (1985), 189–199.

[21] V. Chvátal, N. Sbihi. Bull-free Berge graphs are perfect. *Graphs and Combin.* 3 (1987), 127–139.

[22] C.M.H. de Figueiredo, S. Gravier, C. Linhares Sales. On Tucker's proof of the Strong Perfect Graph Conjecture for $K_4 - e$-free graphs. To appear in *Disc. Math.*

[23] C.M.H. de Figueiredo, F. Maffray. Optimizing bull-free perfect graphs. Manuscript, Universidade Federal do Rio de Janeiro, Brazil, 1998. To appear in *Graphs and Combinatorics.*

[24] C.M.H. de Figueiredo, F. Maffray, O. Porto. On the structure of bull-free perfect graphs. *Graphs and Combin.* 13 (1997), 31–55.

[25] C.M.H. de Figueiredo, F. Maffray, O. Porto. On the structure of bull-free perfect graphs, 2: the weakly triangulated case. RUTCOR Research Report 45-94, Rutgers University, 1994. To appear in *Graphs and Combinatorics.*

[26] C.M.H. de Figueiredo, J. Meidanis, C. Mello.On edge-colouring indifference graphs.*Theor. Comp. Sci.* 181 (1997), 91–106.

[27] C.M.H. de Figueiredo, J. Meidanis, C. Mello.Local conditions for edge-coloring.*J. Comb. Math. and Comb. Comp.* 32 (2000), 79–91.

[28] C.M.H. de Figueiredo, K. Vušković.A class of beta-perfect graphs.*Disc. Math.* 216 (2000), 169-193.

[29] R. Diestel. *Graph Theory.* Grad. Texts in Math. 173, Springer, 1998.

[30] G.A. Dirac.On rigid circuit graphs.*Abh. Math. Sem. Univ. Hamburg*, 25 (1961), 71–76.

[31] R.D. Dutton, R.C. Brigham. A new graph coloring algorithm. *Computer Journal* 24 (1981), 85-86.

[32] Th. Emden-Weinert, S. Hougardy, B. Kreuter. Uniquely colourable graphs and the hardness of colouring graphs of large girth. *Comb., Prob. & Comp.* 7 (1998), 375–386.

[33] P. Erdős. Graph theory and probability. *Canad. J. Math.* 11 (1959), 34–38.

[34] H. Everett, C.M.H. de Figueiredo, C. Linhares-Sales, F. Maffray, O. Porto, B.A. Reed. Path parity and perfection. *Disc. Math.* 165/166 (1997), 223–242.

[35] H. Everett, C.M.H. de Figueiredo, C. Linhares-Sales, F. Maffray, O. Porto, B.A. Reed.Even pairs. To appear in *Perfect Graphs*, J. L. RamírezAlfonsín and B.A. Reed, ed., John Wiley and Sons, 2001.

[36] J. Fonlupt, J.P. Uhry. Transformations which preserve perfectness and *h*-perfectness of graphs. *Ann. Disc. Math.* 16 (1982), 83–85.

[37] T. Gallai.Transitiv orientierbare Graphen.*Acta Math. Acad. Sci. Hungar.* 18 (1967), 25–66.

[38] M. Garey, D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman, San Francisco (1979).

[39] G.S. Gasparian. Minimal imperfect graphs: a simple approach. Combinatorica 16 (1996), 209–212.

[40] A. Ghouila-Houri. Caractérisation des graphes non orientés dont on peut orienter les arêtes de manière à obtenir le graphe d'une relation d'ordre. *C.R. Acad. Sci. Paris* 254 (1962), 1370–1371.

[41] P.C. Gilmore, A.J. Hoffman. A characterization of comparability graphs and of interval graphs. *Canadian J. Math.* 16 (1964), 539–548.

[42] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York (1980).

[43] S. Gravier. On Tucker vertices of graphs. *Disc. Math.* 203 (1999), 121–131.

[44] M. Grötschel, L. Lovász, A. Schrijver. Polynomial algorithms for perfect graphs. In *Topics on Perfect Graphs*, C. Berge and V. Chvátal, editors, *Ann. Disc. Math.* 21 (1984), 325–356, North Holland, Amsterdam.

[45] R. Hayward, Weakly triangulated graphs. *J.Comb. Th. B* 39 (1985), 200–208.

[46] R. Hayward, C .T. Hoàng, F. Maffray. Optimizing weakly triangulated graphs. *Graphs and Combin.*, 5 (1989), 339-349. Erratum in vol. 6 (1990), 33–35.

[47] A. Hertz, A fast algorithm for coloring Meyniel graphs. *J. Comb. Th. B* 50 (1990), 231–240.

[48] A. Hertz, COSINE, a new graph coloring algorithm. *Operations Research Letters* 10 (1991), 411–415.

[49] A. Hertz, D. de Werra. Perfectly orderable graphs are quasi-parity graphs: a short proof. *Disc. Math.* 68 (1988), 111–113.

[50] C.T. Hoàng. Alternating orientation and alternating coloration of perfect graphs. *J. Comb. Th. B* 42 (1987), 264–273.

[51] C.T. Hoàng. Algorithms for minimum weighted coloring of perfectly ordered, comparability, triangulated and clique-separable graphs. *Disc. Appl. Math.* 55 (1994), 133–143.

[52] C.T. Hoàng.Perfectly orderable graphs.To appear in *Perfect Graphs*, J.L. Ramírez-Alfonsín and B.A. Reed, ed., John Wiley and Sons, 2001.

[53] I. Holyer.The NP-completeness of edge-coloring.*SIAM J. Computing* 10 (1981), 718–720.

[54] S. Hougardy. *Perfekte Graphen.* PhD thesis, Institut für Ökonometrie und Operations Research, Rheinische Friedrich Wilhelms Universität, Bonn, Germany, 1991.

[55] W. L. Hsu. Decomposition of perfect graphs. *J. Comb. Th. B* 43 (1987), 70–94.

[56] W.L. Hsu, G.L. Nemhauser. Algorithms for maximum weighted cliques, minimum weighted clique covers, and minimum colourings of claw-free perfect graphs. In *Topics on perfect graphs*, C. Berge, and V. Chvátal ed., *Ann. Disc. Maths* 21, North-Holland, Amsterdam, 1984.

[57] T.R. Jensen, B. Toft. *Graph Coloring Problems.* Wiley-Interscience Series in Disc. Math. and Optimization, 1995.

[58] D.S. Johnson. Worts case behavior of graph coloring algorithms. *Proc. 5th Southeastern Conf. on Comb., Graph Th. & Comput., Utilitas Mathematica* (Winnipeg, 1979), 513–527.

[59] R.M. Karp.Reducibility among combinatorial problems.In R.E. Miller and J.W. Thatcher, editors, *Complexity of computer computations*, pages 85–104. Plenum Press, New York, 1972.

[60] H. Kierstead, J.H. Schmerl. The chromatic number of graphs which induce neither $K_{1,3}$ nor $K_5 - e$. *Disc. Math.* 58 (1986) 253–262.

[61] C. Linhares Sales, F. Maffray. Even pairs in claw-free perfect graphs. *J. Comb. Th. B* 74 (1998), 169–191.

[62] C. Linhares Sales, F. Maffray, B.A. Reed. On planar perfectly contractile graphs. *Graphs and Combin.* 13 (1997), 167–187.

[63] L. Lovász. On chromatic number of graphs and set-systems. *Acta Math. Hung.* 19 (1968), 59–67.

[64] L. Lovász. Normal hypergraphs and the perfect graph conjecture. *Disc. Math.* 2 (1972), 253–267.

[65] L. Lovász.A characterization of perfect graphs.*J. Comb. Th. B*, 13 (1972), 95–98.

[66] L. Lovász.Three short proofs in Graph Theory.*J. Comb. Th. B* 19 (1975), 269–271.

[67] L. Lovász.Perfect Graphs.In *Selected Topics in Graph Theory 2*, L.W. Beineke and R.J. Wilson ed., Academic Press, 1983, 55–87.

[68] L. Lovász, M.D. Plummer.*Matching Theory.Annals of Disc. Maths* 29, North-Holland, 1986.

[69] C. Lund, M. Yannakakis. On the hardness of approximating minimization problems. *J. Assoc. Comp. Mach.* 41 (1994), 960–981.

[70] F. Maffray, O. Porto, M. Preissmann.A generalization of simplicial elimination orderings.*J. Graph Th.*, 23 (1996), 203–208.

[71] F. Maffray, M. Preissmann.On the NP-completeness of the k-colorability problem for triangle-free graphs.*Disc. Math.* 162 (1996), 313–317.

[72] F. Maffray, M. Preissmann.Sequential colorings and perfect graphs.*Disc. Appl. Math.* 94 (1999), 287–296.

[73] F. Maffray, M. Preissmann.A translation of Tibor Gallai's article 'Transitiv orientierbare Graphen'.To appear in *Perfect Graphs*, J.L. Ramírez-Alfonsín and B.A. Reed, ed., John Wiley and Sons, 2001.

[74] S.E. Markossian, G.S. Gasparian, B.A. Reed.β-perfect graphs.*J. Comb. Th. B* 67 (1996), 1–11.

[75] D.W. Matula. A min-max theorem with application to graph coloring. *SIAM Rev.* 10 (1968), 481–482.

[76] D.W. Matula, L.L. Beck. Smallest last ordering and clustering and graph coloring algorithms. *J. Assoc. Comp. Mach.* 30 (1983), 417–427.

[77] R.M. McConnell, J.P. Spinrad.Linear-time modular decomposition and efficient transitive orientation of undirected graphs.Proc. 7th Annual ACM-SIAM Symp. Disc. Algorithms. SIAM, Philadelphia, 1997.

[78] H. Meyniel. The graphs whose odd cycles have at least two chords. In *Topics on Perfect Graphs*, C. Berge and V. Chvátal, editors, *Ann. Disc. Math.* 21 (1984), 115–120, North-Holland, Amsterdam.

[79] H. Meyniel. A new property of critical imperfect graphs and some consequences. *European J. Comb.* 8 (1987), 313–316.

[80] M. Middendorf, F. Pfeiffer. On the complexity of recognizing perfectly orderable graphs. *Disc. Math.* 80 (1990), 327–333.

[81] M. Molloy, B.A. Reed. Colouring graphs whose chromatic number is near their maximum degree. *Lecture Notes in Comp. Sci.*, vol. 1380 (Proc. LATIN'98 Conf.), 216–225, 1998.

[82] J. Mycielski.Sur le coloriage des graphes.*Colloq. Math.* 3 (1955), 161–162.

[83] J.L. Ramírez-Alfonsín, B.A. Reed (editors). *Perfect Graphs.* John Wiley and Sons, 2001.

[84] B.A. Reed. Problem session on parity problems (Public communication). *DIMACS Workshop on Perfect Graphs*, Princeton University, New Jersey, 1993.

[85] B.A. Reed.A strengthening of Brooks's theorem.*J. Comb. Th. B* 76 (1999), 136–149.

[86] F. Roussel, I. Rusu. An $O(n^2)$ algorithm to color Meyniel graphs. Manuscript, LIFO, University of Orléans, France, 1998.

[87] J. Spencer. *Ten Lectures on the Probabilistic Method.* CMBS-NSF Region. Conf. Ser. in Appl. Math., SIAM, Philadelphia, 1994.

[88] J. Spinrad, R. Sritharan. Algorithms for weakly triangulated graphs. *Disc. Appl. Math.* 59 (1995), 181–191.

[89] M.M. Syslo. Sequential coloring versus Welsh-Powell bound. *Disc. Math.* 74 (1989), 241–243.

[90] A. Tucker. Coloring perfect $(K_4 - e)$-free graphs. *J. Comb. Th. B* 42 (1987), 313–318.

[91] A. Tucker. A reduction procedure for colouring perfect K_4-free graphs. *J. Comb. Th. B* 43 (1987), 151-172.

[92] V.G. Vizing. On an estimate of the chromatic class of a p-graph (in Russian). *Diskret. Analiz.* 3 (1964), 23–30.

[93] D.J.A. Welsh, M.B. Powell. An upper bound on the chromatic number of a graph and its applications to timetabling problems. *Computer J.* 10 (1967), 85–87.

4

Algorithmic Aspects of Tree Width

B.A. Reed

4.1 Dividing and Conquering

Divide and conquer is a technique which is effective when preparing military campaigns, planning political strategy, and manipulating your parents. So, it is not too surprising that it also has an important role to play in algorithmic graph theory.

In this article, we present the rudiments of the theory of tree width and brambles, which can be viewed as the theory of the restriction of a "divide and conquer" approach to graph theory. From a different point of view, this theory is a new way of looking at connectivity in graphs. In discussing it, we assume the reader is familiar with the classical notions of connectivity.

To begin we present three examples of the use of the divide and conquer approach for constructing algorithms to solve optimization problems on graphs.

4.1.1 Colouring, Clique Cutsets, and Chordal Graphs

We note that a clique has a unique colouring, up to relabelling the colours, as each colour class must have size one. So, given a clique cutset C in a graph G, and a c-colouring of the graph induced by $C \cup U$ for each component U of $G - C$, it is straightforward to obtain a c-colouring of G. We simply permute the colour class names on each colouring so that the colourings agree on C. This fact, along with a polynomial-time algorithm for finding a clique cutset in a graph which has one (see [47]), yields efficient polynomial-time divide and conquer algorithms for colouring many classes of graphs.

One such class is the chordal graphs. A graph is *chordal* if it contains no induced cycle of length four or more. It is not difficult to show that every chordal graph which is not a clique has a clique cutset (see [22]) . So, given a chordal graph we can either (i) optimally colour it because it is a clique and hence has a unique colouring, or (ii) find a clique cutset within it, and reduce our coluring problem to a number of smaller colouring problems. It turns out that if we repeat this reduction process until the subproblems we are left with are all cliques then we will consider only a polynomial number of subproblems.

Hence, this divide and conquer approach is polynomial. There are faster (indeed linear-time) algorithms for colouring chordal graphs, which take fuller advantage of their special structure. The advantage of the divide and conquer approach is that it is much more widely applicable. Indeed given any base class B of graphs for which we can find optimal colourings in polynomial time, our divide and conquer technique yields a poynomial time algorithm for colouring any graph in the class:

$$B^* = \{G| \text{ every induced subgraph } H \text{ of } G \text{ is in } B \text{ or has a clique cutset}\}.$$

See e.g. [21] where B is the class of complete multi-partite graphs.

4.1.2 Embedding Planar Graphs

Obviously every planar graph has infinitely many drawings in the plane as we are free to wiggle its edges. Such a transformation, however, does not change the basic structure of the drawing. To capture this structure, we define an embedding of a planar graph G as a set of closed walks of G, which are the set of boundaries of the regions for some planar drawing of G. Note that if G is biconnected then these closed walks are cycles. It turns out that every three connected planar graph has a unique embedding (see [28]).

This fact makes it relatively easy to find an embedding of a three connected planar graph G. We begin by finding a subgraph H of G which also has a unique embedding, We then perform a sequence of iterations, adding more of G to the embedding at each step whilst maintaining the property that the embedded subgraph has a unique embedding. An appropriate implementation of this approach yields a polynomial time algorithm for embedding three connected planar graphs.

By applying divide and conquer, we can extend this algorithm, so that it finds embeddings of all planar graphs. If a planar graph is disconnected then we can construct a planar embedding of G by combining embeddings of its components in a straightforward way. Furthermore, if x is a cutvertex of G and $G - x$ has components $U_1, ..., U_k$, then we can paste together embeddings of the graphs $x + U_1, ..., x + U_k$ at x to obtain an embedding of G. In the same way, if $\{x, y\}$ is an edge such that $G - x - y$ is disconnected with components $U_1, ...U_k$ then we can paste together embeddings of the $U_i + x + y$ along the edge xy to obtain an embedding of G.

If $\{x, y\}$ is a cutset of G which is not an edge then we need to be a wee bit more clever. We can assume that neither x nor y is a cutvertex as otherwise we can decompose the problem as discussed above. Thus, for every component U_i there is a path from x to y in $U_i + x + y$ and hence also in $G - U_i$ (consider $j \neq i$). The existence of this second path implies that since G is planar so is $H_i = U_i + x + y + xy$. Now, we can paste embeddings of the H_i together along xy to obtain an embedding of $G + xy$. By deleting xy, we obtain an embedding of G.

An appropriate implementation of this approach yields a polynomial time algorithm for embedding planar graphs.

4.1.3 Dynamic Programming in Trees

Many problems can be solved using divide and conquer in trees, to illustrate we consider the maximum weight stable set problem. I.e, given a tree T, and an integer weight $w(t)$ for each node t of T, determine

$$max\{\sum_{t\in S} w(t)|\ S \text{ is an independent set in } T\}.$$

In order to solve this problem, we actually need to solve a generalization: Rooted Maximum Weight Stable Set (RMWSS). Given a tree T, a root r which is a node of T, and an integer weight function w on the nodes of T, determine both:

$$W_1(T,r,w) = max\{\sum_{t\in S} w(t)|\ \text{S is an independent set in } \ T \text{ containing } r\}.$$

and

$$W_2(T,r,w) = max\{\sum_{t\in S} w(t)|\ S \text{ is an independent set in } T-r\},$$

Now, if r is the only node of T then $W_1(T,r,w) = w(r)$ and $W_2(T,r,w) = 0$. Otherwise, r has neighbours $s_1, ..., s_k$ for some $k \geq 1$ and $T-r$ has corresponding components $T_1, .., T_k$ with $s_i \in T_i$. If $k = 1$, then we note that $W_1(T,r,w) = w(r) + W_2(T_1, s_1, w)$ and $W_2(T,r,w) = max(W_1(T_1,s_1,w), W_2(T_1,s_1,w))$. So we can restrict our attention to the smaller instance (T_1, s_1, w) of RMWSS. If $k \geq 2$ then we apply divide and conquer. We let T_i' be the tree obtained from T_i by adding the edge rs_i. We solve the smaller (RMWSS) instance (T_i', r, w) for each i. Having done so, we simply note that

$$W_1(T,r,w) = \sum_{i=1}^{k} W_1(T_i', r, w) - (k-1)w(r)$$

and

$$W_2(T,r,w) = \sum_{i=1}^{k} W_2(T_i', r, w).$$

By traversing the tree using a postorder traversal, we can actually implement this divide and conquer algorithm to solve RMWSS on trees in linear time.

Many other optimization problems can be solved on trees in linear time using the same divide and conquer approach.

4.1.4 Pasting Solutions Together

In all three of the above examples, we found a cutset C for G and we decomposed our problem into similar subproblems on the subgraphs of G induced by $\{U + C|\ U \text{ is a component of } G - C\}$. We then pasted together the subproblem solutions along the cutset C to obtain a solution to the original problem. In doing so, we had to ensure that the partial solutions "agreed" on C.

In our chordal graph colouring algorithm, doing so was straightforward. For, there is essentially only one way of colouring a clique and so the subproblem solutions always agreed on C.

When embedding planar graphs, we had to pay a bit more attention to this issue. When solving the subproblem on $U + C$ for a cutset C consisting of two non-adjacent vertices x and y, we insisted that the embedding of $U + C$ be obtained from an embedding of $U + C + xy$ by deleting xy. This allowed us to paste together solutions to the subproblems along the edge xy. So, in this case we ensured that we could paste the embeddings together by restricting the set of permissible solutions to each subproblem.

To solve MWSS on trees, we applied a different approach. For $k \geq 2$, we decomposed our subproblem into subproblems on $T'_1, ..., T'_k$. To determine if we could paste together a set or solutions to the subproblems, we needed to know for each i, whether or not r was in the solution for T'_i. However, we did not restrict our attention to stable sets containing r or not containing r. Rather, we computed both the maximum weight of a stable set in T'_i containing r and the maximum weight of a stable set in T'_i not containing r.

This latter approach is the one that interests us. When decomposing the graph using a cutset C, for each possible restriction Π of a solution of the subproblem on $C+U$ to C, we will record the optimal extension of Π to a solution on $C+U$. To obtain an efficient algorithm, we need to ensure that the number of possiblities for such a Π is small (i.e. polynomial in the size on the input). To do so, we will actually bound the size of C, which typically allows us to bound the number of possible Π. For example, if C has size w then there are at most c^w possible c-colourings of C. Note that we also bound the size of the subgraphs which we do not decompose as we also need to compute all possible solutions for these subgraphs.

4.1.5 Knowing Your Roots

To apply the approach above, we actually solve rooted versions of the optimization problems we consider. Thus for a given set of roots R, we need to compute a table containing the best extension of each possible partial solution on the roots to a complete solution. When we decompose our problem along a cutset C, the set of roots R_U of the subproblem defined on $C + U$ for some component U of $G - C$ must include the vertices of C, because we need to see how to paste our solutions together. R_U must also include $R \cap U$, as we are now solving the rooted version of the problem so need to record how the solutions to the subproblems behave with respect to R. We will actually set $R_U = C + R \cap C$.

We can build a *rooted cutset decomposition tree* for an instance (G, R) to record the rooted subproblems we consider. The root of the tree is labelled with the inital rooted problem (G, R). If we decompose (G,R) along a cutset C such that $G - C$ has components $U_1, ..., U_k$ then the children of the root will be labelled with the instances $(C \cup U_i, R_{U_i})$. The subtree of our decomposition tree rooted at the node labelled $(C \cup U_i, R_{U_i})$ will be a decomposition tree for the corresponding subproblem. For a node t of the tree, we let $(H_t, R - t)$ be the label of t.

These then are essentially the tree decompositions which we want to study. However, we actually want to decompose graphs rather than rooted graphs so we will consider a slightly different definition.

4.1.6 The Rest of The Paper

We present a formal definition of a tree decomposition of a graph in the next section. These tree decompositions point out how to split a graph up into pieces using cutsets corresponding to the nodes of the tree. We will show that a wide variety of optimization problems can be solved efficiently, using divide and conquer, on graphs which have tree decompositions where the size of these cutsets is bounded. More interestingly, we see that this theory can be used to solve many optimization problems on arbitrary graphs (both in theory and practice), in part by exploiting structural properties of the graphs which do not have such tree decompositions. Along the way, we introduce a dual of tree width.

4.2 Tree decompositions and tree width

A *tree decomposition* of a graph G consists of a tree T and for each node t of T, a subset W_t of $V(G)$ such that:

(i) For each vertex v of G, the set $S_v = \{t \mid t \text{ is a node of } T,\ v \text{ is in } W_t\}$ induces a non-empty subtree of T, and

(ii) For each edge of G with endpoints x and y, S_x intersects S_y.

We let $\mathcal{W} = (W_t \mid t \text{ is a node of } T)$, and speak of the tree decomposition $[T, \mathcal{W}]$. We use $S_v([T, \mathcal{W}])$ instead of S_v if this precision is necessary.

As depicted in Figures 4.1 and 4.2, given a tree decomposition $[T, \mathcal{W}]$ of G, we can choose for each node t in T, a subgraph X_t of G with node set W_t such that each edge of G is in precisely one of these subgraphs. To do so, we place each edge e with endpoints x and y in X_t for some arbitrary element t of $S_x \cap S_y$. Originally a tree decomposition was defined in terms of the tree T and the specification of such a set of subgraphs: $\mathcal{X} = (X_t \mid t \text{ is a node of } T)$ (cf. [32]). Our definition focuses on $W_t = V(X_t)$. We justify this abuse of notation by remarking that we are really always considering a partition of the edge set and hence a tree decomposition of the second type. However, we simply don't care which element of $S_x \cap S_y$ contains a particular edge with endpoints x and y. So, we use W_t instead of $V(X_t)$ to ease our exposition.

Obviously, every graph has a tree decomposition using a tree with one node. We are particularly interested in tree decompositions in which the W_t are small and in graphs which have such tree decompositions. The *width* of a tree decomposition $[T, \mathcal{W}]$ is $\max\{|W_t| - 1,\ t \text{ a node of } T\}$. The *tree width* of G, denoted $\mathrm{TW}(G)$, is the minimum of the widths of its tree decompositions.

We remark that the -1 in the definition is present, to ensure that the following holds:

Fact 4.2.1 *Any tree which contains an edge has tree width* 1.

Proof *If xy is an edge of G then $\{x, y\} \subseteq W_t$ for some T and so G has tree width at most* 1.

On the other hand, if G is a tree we obtain a tree decomposition of width 1 as follows. We arbitrarily root G at some node r. We set $T = G$ and for each vertex

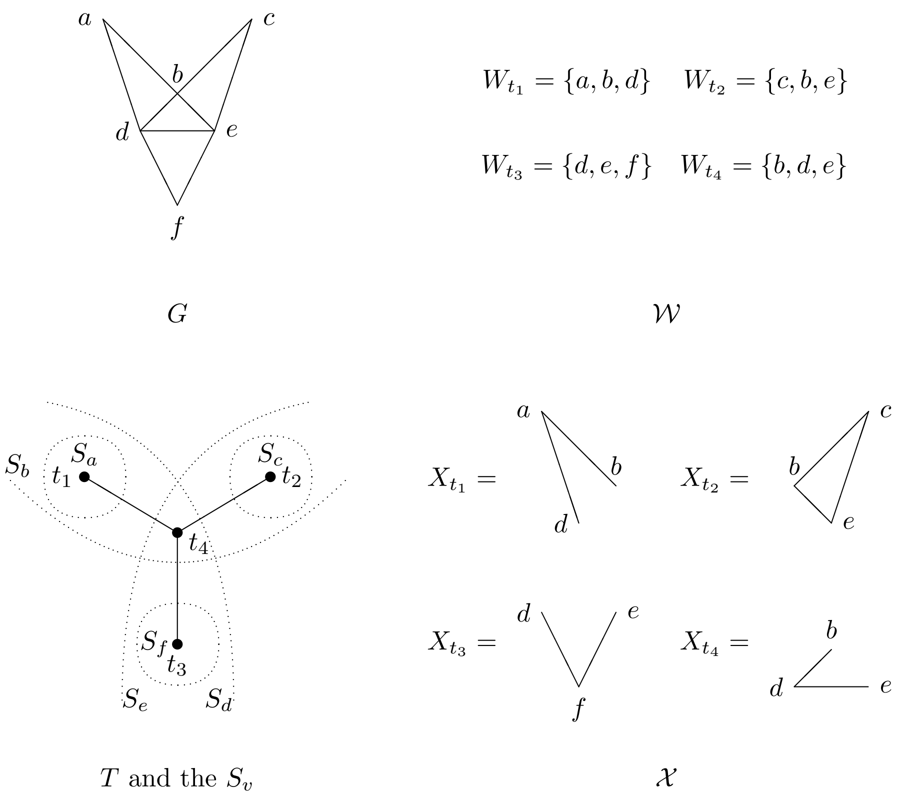

$W_{t_1} = \{a, b, d\}$ $W_{t_2} = \{c, b, e\}$
$W_{t_3} = \{d, e, f\}$ $W_{t_4} = \{b, d, e\}$
G
$\mathcal{W}$
S_a S_b S_c S_d S_e S_f
t_1 t_2 t_3 t_4
$X_{t_1} =$ $X_{t_2} =$ $X_{t_3} =$ $X_{t_4} =$
T and the S_v
$\mathcal{X}$

Figure 4.1. An example of a tree decomposition

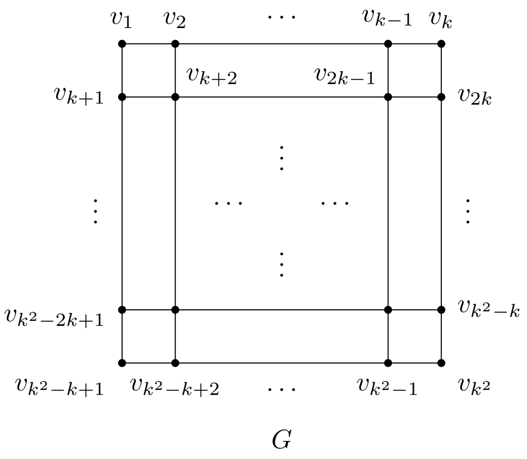
v let S_v consists of v and its children. Thus, $W_r = r$ and for $t \neq r$, W_t consists of t and its parent. Clearly this is a tree decomposition of width one for G.

We show now that a tree decomposition of G points out a way of decomposing G using cutsets corresponding to its nodes.

Definition 4.2.1 *Let $[T, \mathcal{W}]$ be a tree decomposition of a graph G. For any subtree S of T, by V_S we mean $\bigcup\{W_t \mid t$ is a node of $S\}$.*

Lemma 4.2.2 *Let $[T, \mathcal{W}]$ be a tree decomposition of a graph G. Let rs be an arc of T and let R and S be the components of $T - rs$ containing r and s respectively. Then, $(V_R - W_s, V_S - W_r)$ is a partition of $V - (W_r \cap W_s)$ and furthermore, there is no edge of G between $V_R - W_s$ and $V_S - W_r$.*

Proof *First, we note that for each vertex v of G, exactly one of the following holds: $S_v \subseteq R$ and hence $v \in V_R - W_s$, $S_v \subseteq S$ and hence $v \in V_S - W_r$, or S_v contains the arc rs and hence $v \in W_s \cap W_r$. Thus, $(V_R - W_s, V_S - W_r)$ is a partition of $V - (W_r \cap W_s)$. Now, if u is in $V_R - W_s$ and v is in $V_S - W_r$ then $S_u \subseteq R$ while $S_v \subseteq S$. Thus, $S_u \cap S_v = \emptyset$ and so $uv \notin E(G)$.* ■



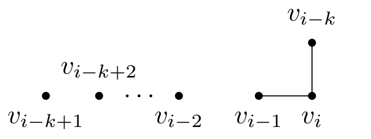
$$W_{t_i} = \{v_{i-k}, v_{i-k+1}, \ldots, v_i\}$$
for $k < i \leq k^2$

$$W_{t_i} = \{v_1, v_2, \ldots, v_i\}$$
for $1 \leq i \leq k$

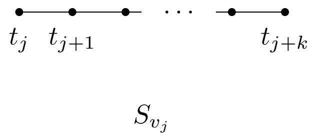
$$W_{t_i} = \{v_{i-k}, \ldots, v_{k^2}\}$$
for $k^2 + 1 \leq i \leq k^2 + k$

$\mathcal{W}$

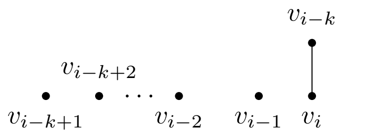
$X_i =$



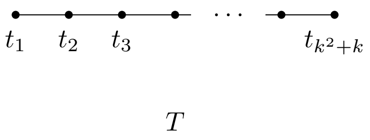
for $k + 2 \leq i \leq k^2$ with $k \not| (i-1)$



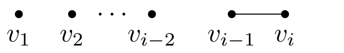
$X_i =$



for $i = k + 1,\ 2k + 1,\ \ldots,\ k^2 - k + 1$

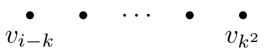


$X_i =$



for $1 \leq i \leq k$

$X_i =$



for $i > k^2$

$\mathcal{X}$

Figure 4.2. Another example of a tree decomposition

Corollary 4.2.3 *Let $[T, \mathcal{W}]$ be a tree decomposition of a graph G and let t be a node of T with l neighbours $s_1, \ldots, s_l$. Let S_i be the component of $T - t$ containing s_i. Then $(V_{S_1} - W_t, \ldots, V_{S_l} - W_t)$ is a partition of $V - W_t$. Furthermore, for $1 \le i < j \le l$, there is no edge between $V_{S_i} - W_t$ and $V_{S_j} - W_t$.*

4.2.1 Rooted Tree Decompositions

In order to use tree decompositions to solve optimization problems, we need to root them.

Definition 4.2.2 *A rooted tree decomposition of a graph G consists of a tree decomposition $[T, \mathcal{W}]$ of G and a rooted tree obtained by rooting T at some node r.*

For brevity's sake, we often use $[(T, r), \mathcal{W}]$ to denote this tree decomposition. It has the same width as $[T, \mathcal{W}]$.

Definition 4.2.3 *Let s be a node of a rooted tree (T, r). We define T_s to be the rooted tree with root s consisting of s and all its descendants.*

Definition 4.2.4 *We use G_t to denote the subgraph of G induced by V_{T_t}.*

The following consequence of Corollary 4.2.3 will allow us to solve optimization problems on graphs of bounded tree width using our dynamic programming approach.

Fact 4.2.4 *For each node s in T, there are no edges between $G_s - W_s$ and $G - G_s$. Furthermore, for any two children t and t' of s, there are no edges between $G_t - W_s$ and $G_{t'} - W_s$.*

The following two results point out the relationship between tree decompositions and rooted cutset decomposition trees.

Theorem 4.2.5 *If G has a rooted cutset decomposition tree such that:*

(a) for every node t in the tree $|R_t| \le w$, and

(b) for every leaf t of the tree $|V(H_t)| \le w$,

then it has tree width at most $2w - 1$.

Theorem 4.2.6 *If G has tree width at most w then for any set R of at most $2w + 1$ vertices of G, (G, R) has a rooted cutset decomposition tree such that:*

(a) for every node t in the tree $|R_t| \le 2w + 1$, and

(b) for every leaf t of the tree $|V(H_t)| \le 2w + 1$.

Proof of 4.2.5: *Let T be the tree in a rooted cutset decomposition tree for G which satisfies (a) and (b). For each node t of T let (H_t, R_t) be the label of t. For each leaf t of T, set $W_t = V(H_t)$. For each internal node t of T, letting C_t*

be the cutset on which we decomposed H_t, set $W_t = R_t \cup C_t$. A routine top down recursive analysis verifies that this is a tree decomposition and it is easy to verify that it has width at most $2w - 1$. We leave the details to the reader.

We will present an algorithmic proof of Theorem 4.2.6 in Section 4.3

4.2.2 Two Alternative Definition

We motivated the definition of tree width by linking it to rooted cutset decomposition trees, which are defined in a top-down manner starting at the root. As we now show, we can also define tree width by considering the leaves.

Definition 4.2.5 *A k-tree is defined recursively as follows:*

(i) The empty graph is a k-tree, and

(ii) every graph which is obtained from a k-tree G by adding a vertex v and the edges between v and a clique of size at most k in G, is also a k-tree.

Definition 4.2.6 *A graph is a partial k-tree if it is a subgraph of a k-tree.*

The class of partial k-trees was defined by Arnborg, Corneil, and Proskurowski [4]. As we now show, the class of partial k-trees coincides with the class of graphs of tree width k. To do so, we need:

Definition 4.2.7 *A subtree intersection representation for G is a tree decomposition such that $S_u \cap S_v$ is non-empty if and only if uv is an edge. Its width is its width as a tree decomposition.*

Lemma 4.2.7 (The Helly Property for Trees) *If $\mathcal{F}$ is a family of subtrees of T every two of which intersect then $\bigcap_{S \in \mathcal{F}} S$ is non-empty.*

We leave the proof of the Helly property for trees as an exercise.

Observation 4.2.8 *A graph has a subgraph intersection representation of width at most k if and only if it is a k-tree.*

Proof *We prove that every k-tree has a subgraph intersection representation of width at most k by induction. Clearly, if G has at most one node then the result holds. So consider a k-tree G such that $|V(G)| \geq 2$. By the definition of a k-tree, there is a node v of G such that $G - v$ is a k-tree and the neighbours of v induce a clique with at most k nodes. By induction $G - v$ has a subtree intersection representation $[T, \mathcal{W}]$ of width at most k. For any two neighbours of v, since uv is an edge, S_u intersects S_v. So, by the Helly property for trees, there is a node t of T such that W_t contains $N(v)$. We construct a new tree by adding a vertex s and an edge st to T. We construct a subgraph intersection representation of G by setting $W_t = v + N(v)$. (we leave it to the reader to verify that this does indeed yield a subtree intersection representation). Since $|W(t)| \leq k + 1$ this subtree intersection representation has width at most k, as desired.*

Now, to prove that any graph which has a subtree intersection representation of width at most k is a k-tree, we proceed by induction on the number of nodes in the tree decomposition and subject to this by induction on the number of vertices in the graph. If the tree decomposition has only one node then the graph is a clique of size at most $k+1$ so the desired result holds. So, consider a graph G which has a subtree intersection representation $[T, \mathcal{W}]$. of width at most k. We can assume T has at least two nodes and choose a leaf t of T with a unique neighbour s in T.

If $W_t \subseteq W_s$ then any S_u and S_v which intersect in t also intersect in s. Thus, deleting t and W_t yields a new subtree intersection representation of G of width at most k. So, we are done by the induction hypothesis.

If W_t is not contained in W_s then let v be a vertex of $W_t - W_s$. Note that this implies that $S_v = t$. Thus, the neighbour set of v is precisely $W_t - v$ which is a clique by the definition of subtree intersection representation, and has at most k vertices because of our bound on the width of $[T, \mathcal{W}]$. Furthermore, replacing W_t by $W_t - v$ yields a subtree intersection representation of $G - v$ of width at most k, so by our inductive hypothesis, $G - v$ is a k-tree. Combining these two facts, we obtain that G is a k-tree.

Corollary 4.2.9 *A graph is a partial k-tree if and only if it has tree width at most k.*

Proof *If G is a partial k-tree then it is a subgraph of some k-tree H with $V(H) = V(G)$. Now, H has a subgraph representation of width at most k which is a tree decomposition of G.*

Conversely, if G has a tree decomposition of width at most k then the graph H on $V(G)$ obtained by setting $uv \in E(H)$ precisely if $S_u \cap S_v \neq \emptyset$ is a k-tree, by Observation 4.2.8. Hence, G is a partial k-tree.

As a corollary of Observation 4.2.8, we see that a graph has a subtree intersection representation if and only if each of its subgraphs contains a vertex whose neighbourhood induces a clique. Gallai[21] proved that a graph satisfies this property precisely if it is chordal. Using this result and the recursive definition of k-tree it is easy to prove that a graph is a k-tree if and only if it is chordal and has no clique with more than $k+1$ vertices. Thus, we have:

Theorem 4.2.10 *The tree width of G is one more than the minimum over all chordal graphs H such that $G \subseteq H$ of the size of the largest clique in H.*

4.3 Finding Bounded Width Decompositions

In this section, we present a straightforward polynomial time algorithm which given a graph of tree width w and a subset R of at most $2w+1$ of its vertices, constructs a rooted cutset tree decomposition for it satisfying (a) and (b) of Theorem 4.2.6, thereby proving that theorem. Furthermore, since the proof of Theorem 4.2.5 can be made algorithmic, our algorithm can also be used to build a tree decomposition of G with width at most $4w+1$ in polynomial time, if w

is fixed (this problem is NP-complete if k is part of the input; see [26]). We then go on to discuss faster algorithms and a related duality theorem.

Our algorithm for finding rooted cutset decomposition trees is recursive and top down. Given a rooted graph (G, R) with $|R| \leq 2w+1$ and $|V(G)| > 2w+1$, we will find a cutset C such that for each component U of $G-C$, $R_U = C \cup (R \cap C)$ satisfies $|R_U| \leq 2w+1$. A judicous choice of our cutsets will allow us to continue this process until we have decomposed our problem into a family of subproblems, each of which has at most $2w+1$ vertices.

Actually the following result, shows that we do not need to take any care in choosing our cutset.

Lemma 4.3.1 *If G has tree width at most w, then for every $R \subseteq V(G)$, $\exists C \subseteq V$ with $|C| \leq w+1$ such that for each component U of $G-C, |R \cap U| \leq \frac{1}{2}R$.*

Proof *Consider a tree decomposition $[T, \mathcal{W}]$ of G of width at most w. Suppose the desired C does not exist. Then, for each node t of T, since $|W_t| \leq w+1$, there is a component U_t of $G - W_t$ containing more than half the vertices of R. By Corollary 4.2.3, there is a unique arc st of T such that for every $v \in U_t$, S_v is contained in the component of $T - st$ containing s. We define a function f by setting $f(t) = s$. Since T has n nodes and $n-1$ arcs, there exists an arc t_1, t_2 of T such that $f(t_1) = t_2$ and $f(t_2) = t_1$. Now, there must be a vertex v in $R \cap U_{t_1} \cap U_{t_2}$. But this contradicts either our choice of $f(t_1)$ or $f(t_2)$.*

This lemma tells us that in each iteration of our top down algorithm the desired cutset C exists. We can find such a cutset in polynomial time via complete enumeration. We need only determine for each of the $O(|V(G)|^{w+1})$ subsets X of $V(G)$ with $|V(G)| \leq w+1$ whether we can set $X = C$. To test if we can set $X = C$ we just compute the connected components of $G - X$ which can be done in linear time. So, we can carry out an iteration in polynomial time.

A straightforward recursive argument which we omits shows that (provided we insist that each C is non-empty) the recursive cutset decomposition tree which we construct has at most $|V(G) - R|$ nodes. This linear bound on the number of iterations we perform ensures that the total running time reuqired by the algorithm is also polynomial. It will also be useful later when we come to use these tree decompositions.

As we discuss in Section 4.3.2, this algorithm can actually be implemented in $O(|V(G)|^2)$ time and a different approach yields a linear time algorithm.

4.3.1 An Approximate Dual to Tree Width

We note that our algorithm actually computes a tree decomposition of width $4w+1$ for any graph such that for every $R \subseteq V(G)$, a set C as described in the statement of Lemma 4.3.1 exists. This fact motivates the following:

Definition 4.3.1 *A set S is k-linked if for every set X of fewer than k vertices, there is a component U of $G - X$ containing more than half the vertices of S. the linkedness of G, denoted $li(G)$, is the maximum k for which G has a k-linked set.*

Lemma 4.3.2 *$li(G) - 1 \leq TW(G) \leq 4li(G) + 1$.*

Proof *Lemma 4.3.1 is simply a restatement of $li(G) - 1 \leq TW(G)$. The algorithm just described is a constructive proof that $TW(G) \leq 4li(G) + 1$.*

Thus the maximum k such that G has a k-linked set is approximately the minimum width of a tree decomposition of G. For an exact duality result for tree width see [44].

4.3.2 Faster Algorithms

We now describe a variant of our algorithm for constructing tree decompositions which can be implemented in $O(|V(G)|^2)$ time. In this variant, rather than choosing a cutset C such that $|R \cap U| \leq \frac{1}{2}|R|$ for every component U of $G - C$, we settle for insisting that $|R \cap U| \leq \frac{2}{3}|R|$ for each such component. We also raise our bound on the size of R_t and $max\{|V(H_t)| : t \textit{ is a leaf}\}$ to $3w + 1$. Having made these two changes, our recursive procedure yields the desired rooted cutset decomposition tree of G and corresponding tree decomposition of G of width at most $6w + 1$.

This version of the algorithm can also be implemented in polynomial time, by computing, in each iteration, the components of $G - X$ for each $X \subseteq V(G)$ with $|X| \leq w + 1$. The following fact allows us to speed the algorithm up.

Observation 4.3.3 *For $S \subseteq V(H)$, there is no component U of H containing more than $\frac{2|S|}{3}$ vertices of S if and only if we can partition $V(H)$ into A and B such that $|A \cap S| \leq \frac{2|S|}{3}$, $|B \cap S| \leq \frac{2|S|}{3}$, and there are no edges between A and B.*

Proof *Left to the reader.*

To test for a fixed w and subset R of V with $R \leq 3w + 1$, whether or not V has a partition into A, B, and C such that $|C| \leq w + 1$, $|A \cap R| \leq 2w$, and $|B \cap R| \leq 2w$, we proceed as follows.

For each of the fewer than 3^{3w+1} partitions of R into A_R, B_R, C_R with $|A_R| \leq 2w$, $|B_R| \leq 2w$, and $|C_R| \leq w + 1$ we test whether we can choose the desired A,B, and C so that $A \cap R = A_R, B \cap R = B_R$, and $C \cap R = C_R$. I.e. we test if there is a set C' of at most $w + 1 - |C_R|$ vertices such that there are no paths between $A_R - C'$ and $B_R - C'$ in $G - C_R - C'$. By Menger's Theorem, if such a set of vertices does not exist then there are $w + 2 - |C_R|$ vertex disjoint paths between A_R and C_R. Furthermore, classical techniques (see e.g. [19]) allow us to either find the desired C' or such a set of paths in $O(w|E(G)|)$ time.

Using this procedure, we can implement each iteration of our recursive tree decomposition finding algorithm in $O(3^{3w+1}w|E(G)|)$ time and hence implement the algorithm using $O(|V(G)||E(G)|)$ time in total, for a fixed w. Now, it is immediate from the definition that a a partial k-tree has at most $k|V(G)|$ edges, so we need only apply our algorithm to graphs with $O(|V(G)|)$ edges and hence can implement it in $O(|V(G)|^2)$ time.

The two earliest polynomial time algorithms for computing tree width can be found in [4] and [32]. In [38], Robertson and Seymour presented the algorithm discussed here. Reed [30], by introducing some technical complications, speeded up the algorithm. His variant runs in $O(nlog\ n)$ time.

Arnborg et al. in [5], gave a linear time algorithm for determining if the tree width of a graph is some fixed w. However, their algorithm requires much more than linear space, (reading in unwritten memory is permitted,) and requires $O(|V(G)|^2)$ time if we actually want to find the tree decomposition. Bodlaender in [13] (see also [14]), developed a linear time algorithm for determining if the tree width of a graph is at most w and constructing a decomposition of width w if possible.

4.4 Using Bounded Width Decompositions

Given a bounded width rooted tree decomposition of a graph, many hard (i.e. NP-complete) problems can be solved efficiently. To illustrate, we reconsider the maximum weight stable set problem.

So assume we are given a graph G, a positive integer weight $w(x)$ for each vertex x of G, and a rooted tree decomposition $[(T,r),\mathcal{W}]$ of G of width at most k with at most n nodes. For a subset X of V we let $w(X)=\sum_{x\in X} w(x)$.

We will traverse the tree using a post-order traversal and compute, for each $t\in T$ and stable set $S\subseteq W_t$:

$$f(S,t) = max\ (w(X)|X\ stable,\ X\subseteq V(G_T),\ X\cap W_t = S).$$

(Recall that G_t is the subgraph of G induced by those vertices v such that S_v intersects the rooted subtree underneath t.) The the solution to our optimization problem is simply $max\ \{f(S,r)|S\subseteq W_r,\ S\ stable\}$.

If t is a leaf then $f(S,t)=w(S)$. If t is not a leaf then we let $s_1,...s_l$ be the children of t, and $S_i = S\cap W_{s_i}$. Corllary 4.2.3 implies that if for each i we let X_i be a maximum stable set in G_{s_i} with $X_i\cap W_{s_i}\cap W_t = S_i$ then $\cup_{i=1}^{l} X_i$ is also a stable set. Thus

$$f(S,t) = w(S) + \sum_{i=1}^{l} max\ \{f(Y,s_i)|Y\ stable, Y\subseteq W_{s_i}, Y\cap W_t = S_i\} - w(S_i).$$

When computing the values of f corresponding to a node t, we need consider at most 2^k possibilities for S and treating each possibilities requires $l2^k$ time where l is the number of children of t. Thus, our algorithm runs in $O(2^{2k}|V(T)|)$ time which is linear in $|V(G)|$ if k is fixed since $|V(T)|\le|V(G)|$.

With a little bit of extra bookkeeping we can also find a maximum weight stable set of G in linear time.

Similar dynamic programming algorithms allow us to solve many NP-complete optimization problems in linear time on graphs of bounded tree width. Examples include Clique, Hamilton Cycle, Chromatic Number, Domination Number, and H-minor containment. In fact, it has been shown [16] that any problem which can be formulated as a certain kind of logical formula can be solved on graphs of bounded tree width in linear time.

We note that graph isomorphism is an example of a problem which can be solved in polynomial but not (yet) linear time in graphs of bounded tree width[11]. In the same vein, although the chromatic index of a bounded tree width graph can be computed in linear time [12], the fastest algorithm known to construct an optimal edge colouring of such graphs runs in polynomial but not linear time.

All the algorithms mentioned above, although polynomial in n, are exponential in k and thus unlikely to be practical for graphs of tree width exceeding ten.

The problems which can be solved using these techniques come in three different flavours: those in which the input has low tree width because of the nature of the problem, those in which we can somehow restrict our attention to a subproblem on a subgraph of low tree width, and those for which we can attack the problem using different techniques if the input graph has unbounded tree width.

One example of the first type of problem is the phylogeny problem which arises in computational biology [10]. Here we are trying to construct a family tree so it is natural that our input is tree-like. Another example is register allocation (choosing which variables to load into a computer's registers while running a program). This can be modelled as a colouring problem on the interference graph of the program's control flow graph. Although such graphs can have high tree width, if the program is structured then their tree width is at most seven (cf. [46]). This allows for efficient resolution of the register allocation problem.

One example of the second type of problem is the Travelling Salesman Problem. Often, when applying branch and bound, an optimal solution is approximated by the best of 3 or 4 candidate heuristic solutions. Alternatively, one can paste these solutions together and find the optimum solution contained in the union of their edge sets. This is possible because the cycles tend not to differ too much and hence their union may well have bounded tree width. This approach was used by Applegate, Bixby, Chvatal and Cook [8] in their prize winning program for solving large TSPs.

We discuss a number of problems of the third type in the last section of the article. First however we need to develop a new characterization of graphs of high tree width, which we do in the next section.

4.5 Walls

We now give a forbidden subgraph characterization of graphs of bounded tree width. The subgraphs we need to forbid are the walls.

An elementary wall of height 8 is depicted in Figure 4.3. An *elementary wall of height h* is similar. It is a piece of the hexagonal lattice consisting of h levels each containing h bricks. More precisely, an elementary wall of height h contains $h+1$ vertex disjoint paths, $R_1, \ldots, R_{h+1}$ which we call *rows*, and $h+1$ vertex disjoint paths, $C_1, \ldots, C_{h+1}$ which we call *columns*. The reader should be able to complete the definition by considering Figure 4.3, in which R_1 is the top row. (For fussy formalists: the first and last row, i.e. R_1 and R_{h+1}, both contain $2h+1$ vertices. All the other rows contain $2h+2$ vertices. All the columns contain $2h$ vertices. Column i joins the $(2i-1)$st vertex of R_1 with the $(2i-1)$st vertex of R_{h+1}; it contains the $(2i-1)$st and $2i$th vertex of every other row, as well as the edge between them. For $j \leq h$ and odd, each C_i contains an edge between the $(2i-1)$st vertex of R_j and the $(2i-1)$st vertex of R_{j+1}. For $j \leq h$ and even, each C_i contains an edge between the $2i$th vertex of R_j and the $2i$th vertex of R_{j+1}. These are all the edges of the wall.)

A *wall of height h* is obtained from the elementary wall by replacing the edge set by a corresponding set of internally vertex disjoint paths whose interiors

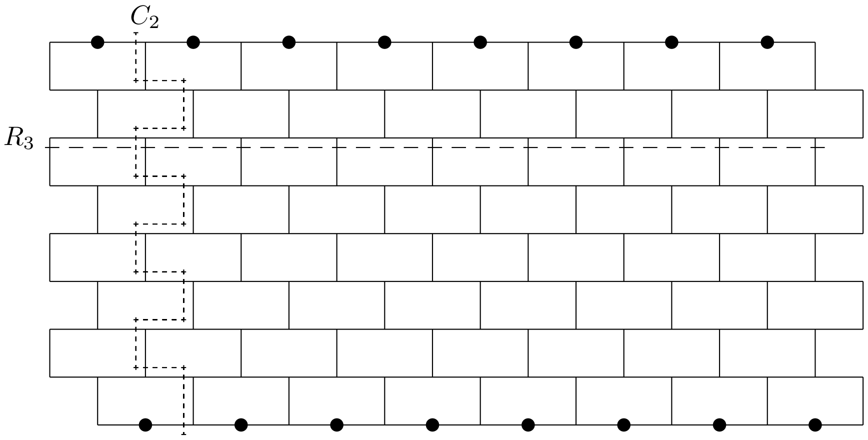


Figure 4.3. An elementary wall of height 8

are vertex disjoint from the original elementary wall, see Figure 4.4. The rows, columns, corners, and perimeter of the wall correspond to the same objects in the original elementary wall. The *nails* of the wall are the vertices of degree three within it as well as its corners.

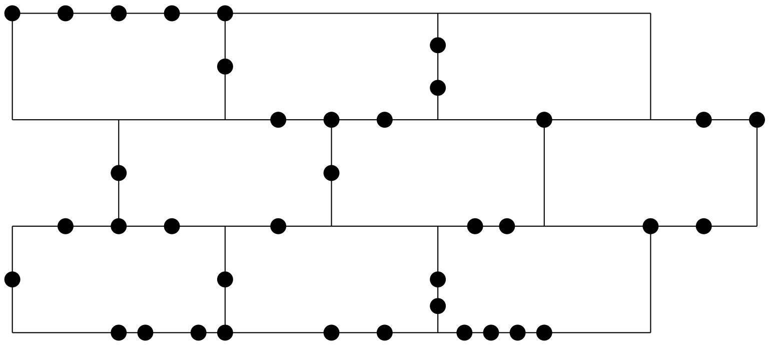


Figure 4.4. A wall of height 3

We note that the nails of any wall of height h can be shown to be $\lfloor h/2 \rfloor$-linked (since any set of at most $h/2$ vertices misses half the rows and the nails in these rows will clearly all be in the same component). This proves that $li(G)$ exceeds half the height of the largest wall in G. Since $TW(G) \geq li(G)$, this yields an approximation to the easy direction of the following result:

Theorem 4.5.1 ([42](see also [34])) *Let h be the maximum of the heights of the walls in G. Then $h + 1 \leq TW(G) \leq 25^{34h^5}$.*

The proof of this result is long and complicated so we will not present it. Instead, in the next section, we prove that if G is planar then $TW(G) \leq 96h+1$.

4.5.1 Excluding Walls in Planar Graphs

In this section, we prove:

Theorem 4.5.2 ([2]) *Let G be a planar graph and let $h \geq 2$ be the maximum of the heights of the walls in G, then $TW(G) \leq 96h+1$.*

We will need the following:

Definition 4.5.1 *A set S is strongly k-linked if for every set X of fewer than k vertices there is a component of $G-X$ containing more than two-thirds of the vertices of S.*

Observation 4.5.3 *Ever $2k$-linked set contains a strongly k-linked set.*

Proof *Let S be a $2k$-linked set in a graph G. If S is strongly k-linked we are done. Otherwise, there is a set X of fewer than k vertices of G such that every component of $G-X$ contains at most two-thirds the vertices of U. Since S is $2k$-linked, there is a (unique) component U^* of $G-X$ s.t $S^* = U^* \cap S$ satisfies; $|S^*| > \frac{|S|}{2}$.*

If S^* is strongly k-linked, we are done. Otherwise, there is a set X^* of fewer than k vertices such that every component U' of $G-X^*$ satisfies: $|U' \cap S^*| \leq \frac{2|S^*|}{3} \leq \frac{4|S|}{9}$.

Now, if U' is a component of $G-X-X^*$, then U' is contained in some component of $G-X$. If U' is not contained in U^*, then $|U' \cap S| \leq |S-S^*| \leq \frac{|S|}{2}$. If U' is contained in U^*, then $U' \cap S = U' \cap S^*$, and so by our choice of X^*, $|U' \cap S| \leq \frac{4|S|}{9}$. So, we see that every component U' of $G-X-X^*$ satisfies $|U' \cap S| \leq \frac{|S|}{2}$, contradicting the fact that S is $2k$-linked.

Proof of Theorem 4.5.2 *For some $h \geq 2$, let G be a planar graph which has tree width at least $96h+1$. By Lemma 4.3.2 and Observation 4.5.3, G contains a strongly $12k$ linked set, S. Fix a drawing D of G in the plane $\mathcal{R}^2$.*

We say that an arc or curve in the plane is G-normal if it intersects G only at vertices. For any simple closed curve C in the plane, we use $int(C)$ to denote the non-infinite component of $\mathcal{R}^2 - C$.

We choose a simple closed curve C in the plane satisfying:

(i) C is G-normal,

(ii) $|C \cap V(G)| \leq 8h$,

(iii) $|int(C) \cap S| > \frac{2|S|}{3}$, and

(iv) subject to (i) -(iii), $|int(C) \cap V(G)|$ is as small as possible.

Such a choice is possible because if we let J be a simple closed curve in the infinite face of our drawing which "surrounds" G (i.e. s.t. $D \subset int(J)$), then $J \cap V(G) = \emptyset$, so J satisfies (i)-(iii).

Now, if $C \cap V(G) < 8h$, then it is easy to shift C slightly so as to obtain a new G-normal curve C' which touches some vertex v of $int(C) \cap V(G)$ and satisfies $C' \cap V(G) = C \cap V(G) + v$, $int(C') \cap V(G) = int(C) \cap V(G) - v$. Our choice of C implies that $|int(C') \cap S| \leq \frac{2|S|}{3}$. Now, every component U of $G - (V(G) \cap C) - v$ is contained either in $int(C')$ or in $G - int(C)$ and in either case satisfies $|U \cap S| \leq \frac{|S|}{3}$. But this contradicts the fact that S is strongly $8h$-linked.

This contradiction implies $|V(G) \cap C| = 8h$. We enumerate the vertices on C as $v_1, ..., v_{8h}$ in the order they appear on when traversing C in the clockwise direction from an arbitrarily chosen starting point. We let G_C be the subgraph of G drawn in $C \cup int(C)$.

Fact 4.5.4 *There exist $2h + 2$ vertex disjoint paths $Q_1, ..., Q_{2h+2}$ between $v_1, ..., v_{2h+2}$ and $v_{4h+1}, ..., v_{6h+2}$ in G_C.*

Proof *Otherwise, by a planar version of Menger's Theorem, there is a G-normal arc A with its endpoints on C and otherwise contained in $int(C)$, such that letting A_1 and A_2 be the components of $C - A$, we have: $|V(G) \cap (A - C)| \leq |V(G) \cap A_1|$ and $|V(G) \cap (A - C)| \leq |V(G) \cap A_2|$.*

Now, letting $C_1 = A_1 \cup A$ and $C_2 = A_2 \cup A$, we see that both C_1 and C_2 are G-normal arcs containing at most $8h$ vertices of G. So, by our choice of C, we have $|int(C_1) \cap S| \leq \frac{2|S|}{3}$ and $|int(C_2) \cap S| \leq \frac{2|S|}{3}$. But now by considering $X = V(G) \cap (C \cup A)$, we see that S is not strongly $12h$-linked, a contradiction.

In the same vein, we obtain,

Fact 4.5.5 *There exist $h + 1$ vertex disjoint paths $R_1, \ldots, R_{h+1}$ between $v_{3h}, \ldots, v_{4h}$ and $v_{7h}, \ldots, v_{8h}$ in G_C.*

Now, it is easy to prove that there is a wall in G whose rows are subsets of $R_1, ...R_{h+1}$ and whose columns are subsets of $Q_1, ...Q_{2h+2}$.

As a corollary, we obtain that the vertex set of a planar graph is not $128\sqrt{n}+1$ linked, as a wall of height h contains more than h^2 vertices. A stronger result with the 128 replaced by a smaller constant was proven earlier by Lipton and Tarjan [25]. Using an argument similar to that given above, Alon Seymour and Thomas were abler to improve the constant in the Lipton-Tarjan result.

4.6 Some Applications

In this section, we discuss how to exploit the fact that graphs of high tree width contain high walls to solve optimization problems over arbitrary graphs.

A classical problem in graph theory is the following:

Disjoint Paths: Given a graph G and two sets of vertices S and T of G with $|S| = |T| = k$, determine whether there are k vertex disjoint paths from S to T, and if so find such a set of paths.

This problem can be solved in polynomial time, even if k is part of the input (see [23]). The algorithm used to solve this problem is one of the fundamental tools used in Operations Research, indeed we used it to speed up our algorithm for finding tree decompositions.

We consider a slightly different problem:

k-Rooted Routing: Given a graph G and two subsets $S = \{s_1, ..., s_k\}$ and $T = \{t_1, ..., t_k\}$ of vertices of G, determine if there are k vertex disjoints paths $P_1, .., P_k$ linking S and T so that P_i links s_i and t_i.

This problem is NP-complete if k is part of the input (see [26]). However, it is in P for any fixed k. In fact, Robertson and Seymour [38] developed an $\mathcal{O}(|V(G)|^3)$ time algorithm to solve k-Rooted Routing for a fixed k. Reed (cf. [31]) improved this to $\mathcal{O}(|V(G)|^2)$.)

We will now discuss Robertson and Seymour's algorithm. It is quite complicated, so we will focus on the role that tree decompositions play and give only a vague description of the remainder of the algorithm. We begin with a definition.

Definition 4.6.1 *Let (G, S, T) be an instance of k-Rooted Routing. A vertex v is irrelevant (with respect to (G, S, T)) if the desired paths exist in G if and only if they exist in $G - v$.*

Rooted Routing for fixed k is easy to solve in linear time on graphs of bounded tree width, using dynamic programming.

Robertson and Seymour [38] proved:

Theorem 4.6.1 *For every k there is an h_k such that if (G, S, T) is an instance of k-Rooted Routing and W is a wall of height h_k in G then there is an irrelevant vertex v in W. Furthermore, such a wall and corresponding irrelevant vertex can be found in polynomial time. (Robertson and Seymour's algorithm runs in $\mathcal{O}(|V(G)|^2)$ time, Reed improved this to $\mathcal{O}(|E(G)|)$).*

Now, obviously having found an irrelevant vertex v for (G, S, T) we can restrict our attention to $(G - v, S, T)$. Robertson and Seymour repeatedly apply Theorem 4.6.1 and delete the irrelevant vertex it returns until the graph they are considering contains no high wall. Theorem 4.5.1 implies that such a graph has tree width at most $25^{34h_k^5}$ and hence we can solve the k-Rooted Routing problem using dynamic programming.

We now briefly sketch the methods Robertson and Seymour use to prove Theorem 4.6.1. To begin we present two special cases for which it is easy to find the desired irrelevant vertex.

First, consider an instance (G, S, T) of of $k - RR$ such that G contains a clique C of size $2k + 1$.

If there is a set $\mathcal{P}$ of $2k$ vertex disjoint paths between $S \cup T$ and some subset C' of C then the desired paths $P_1, .., P_k$ exist, indeed we can choose each P_i to consist of the union of two elements of $\mathcal{P}$ and an edge of C (see Figure 4.5). So, we could simply stop having solved the problem. If, for some perverse reason, we actually want to find an irrelevant vertex then we note that if we choose the paths in P minimal then they are internally disjoint from C and by the above remarks, every vertex in $C - C'$ is irrelevant.

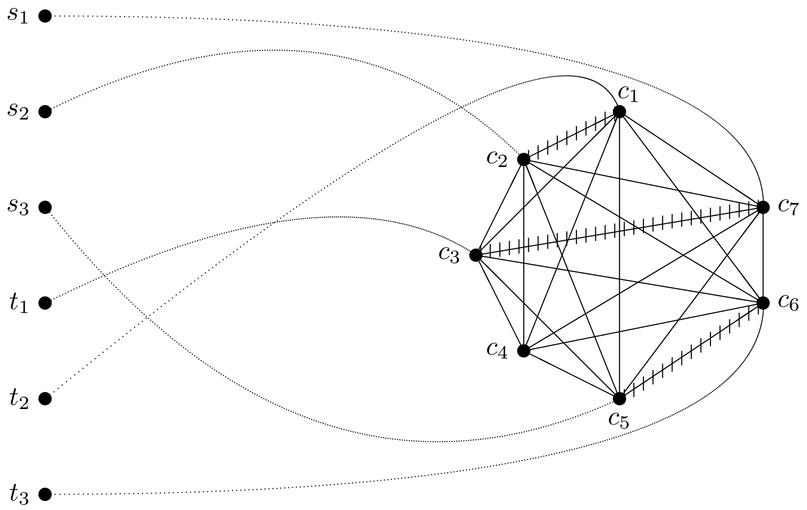


Figure 4.5. c_4 is irrelevant

Otherwise, by Menger's Theorem, there is a set X of less than $2k$ vertices such that there is no path between $S \cup T$ and C in $G - X$. That is, $S \cup T$ is disjoint from the component U of $G - X$ containing $C - X$. In fact, Menger's Theorem implies that if we choose such an X with $|X|$ minimum then there is a set $\mathcal{R}$ of $|X|$ vertex disjoint paths from X to some subset C' of C. By taking minimal paths, we can ensure these paths are disjoint from $C - C'$. Now, for any set $\mathcal{P}$ of k vertex disjoint paths linking S to T, the intersection of the paths in $\mathcal{P}$ with $X \cup U$ must be a set $\mathcal{P}'$ of paths with endpoints in X (since there are no edges from U to $G - X - U$). For any such set $\mathcal{P}'$ of paths, we can clearly obtain a set of paths with the same endpoints using the paths of $\mathcal{R}$ and an appropriate subset of the edges between vertices of C'. Thus, if a solution to our instance of $k - RR$ exists, there is a solution whose intersection with U uses only those vertices on some element of $\mathcal{R}$ and in particular uses none of the vertices of $C - C'$. Therefore the vertices in $C - C'$ are irrelevant.

So we obtain:

If (G, S, T) is an instance of $k - RR$, and C is a clique of size at least $2k + 1$ in G then in polynomial time, we can either find the desired paths or find an irrelevant vertex v.

It turns out that a similar statement holds if C is a sufficently large set of disjoint connected subgraphs every pair of which are joined by an edge, we call such a set a *clique minor*. In fact, Robertson and Seymour present a straightforward algorithm which uses Menger's Theorem to find an irrelevant vertex given a clique minor C consisting of $8k + 3$ such subgraphs (cf. [38]).

Next, consider an instance (G, S, T) of k-Rooted Routing such that G contains a wall W of height $2k$ and has a planar embedding such that the perimeter of

W forms the infinite face. Suppose that $S \cup T$ lies on the perimeter of W. Note that there are nested cycles $D_1, ..., D_k$ such that D_i lies in union of the rows R_i, R_{2k+2-i} and the columns C_i, C_{2k+2-i} and hence D_{i+1} lies inside D_i. If $k = 1$ then clearly the desired P_1 exists in D_1. If $k = 2$ then pushing the paths as close to the perimeter as possible, we can show that if the desired paths exist we can find them in subgraph consisting of D_2 and that part of G drawn outside D_2. In the same vein, for any k, if the desired paths exist then they exist in the subgraph of G on or outside D_k. Thus, every vertex in the intersection of C_{k+1} and R_{k+1} is irrelevant. So in this case, given the wall W, we can indeed find an irrelevant vertex of G quickly.

It turns out that given any instance (G, S, T) such that G contains a high wall, we will find ourselves in a situation similar to one of the two considered above. If the connections between W and $G - W$ are "highly non-planar" then G will contain a clique minor consisting of $8k + 3$ disjoint subgraphs and hence we are in the first situation discussed above. If the connections between W and $G - W$ are "sufficiently planar" then we can find a subwall W' of G such that the vertex in the middle of the subwall is irrelevant. The proof of this fact requires three hundred pages, and even a precise statement of the results is beyond the scope of this article.

The algorithm to solve k-Rooted Routing has a host of applications. In particular, Robertson and Seymour used it in developping a polynomial time algorithm to test membership in any class of graphs closed under the taking of minors.

References

[1] N. Alon, P. Seymour, and R. Thomas, A separator theorem for graphs with an excluded minor and its applications, *Proceedings of the 22nd Annual Association for Computing Machinery Symposium on Theory of Computing*, ACM Press, New York (1990) 293–299

[2] N. Alon, P. D. Seymour, and R. Thomas, Planar separators, *SIAM Journal on Discrete Mathematics* **7** (1994) 184–193

[3] D. Archdeacon and P. Huneke, A Kuratowski theorem for nonorientable surfaces, *Journal of Combinatorial Theory, Series B* **46** (1989) 173–231

[4] S. Arnborg, D. G. Corneil and A. Proskurowski, Complexity of finding embeddings in a k-tree, *SIAM Journal on Algebraic and Discrete Methods* **8** (1987) 277–284

[5] S. Arnborg, B. Courcelle, A. Proskurowski and D. Seese, An algebraic theory of graph reduction, *Journal of the Association for Computing Machinery* **40** (1993) 1134–1164

[6] S. Arnborg, J. Lagergren and D. Seese, Easy problems for tree-decomposable graphs, *Journal of Algorithms* **12** (1991) 308–340

[7] S. Arnborg and A. Proskurowski, Linear time algorithms for NP-hard problems restricted to partial k-trees, *Discrete Applied Mathematics* **23** (1989) (11–24)

[8] D. Applegate, B. Bixby, V. Chvátal and W. Cook, On the solution of travelling salesman problems, *Proceedings of the International Congress of Mathematicians* Vol. III (Berlin, 1998) 645–656

[9] M. Ball, T. Magnanti, C. Monma, and G. Nemhauser, Network Models, North Holland, Amsterdam, The Netherlands, 1995

[10] H. Bodlaender, M. Fellows, M. Hallett, T. Wareham and T. Warnow, The hardness of perfect phylogeny, feasible register assignment, and other problems on thin coloured graphs, *Theoretical Computer Science* **244** (2000) 167–188

[11] H. L. Bodlaender, Dynamic programming on graphs of bounded treewidth, *Proceedings of the 15th International Colloquium on Automata, Languages and Programming*, T. Lepistö and A. Salomaa (eds.), *Lecture Notes in Computer Science* **317** (1998) 105–118, Springer Verlag, Berlin

[12] H. L. Bodlaender, Polynomial algorithms for graph isomorphism and chromatic index on partial k-trees, *Journal of Algorithms* **11** (1990) 631–643

[13] H. L. Bodlaender, A linear time algorithm for finding tree decompositions of small treewidth, *SIAM Journal of Computing* **25** (1996) 1305-1317

[14] H. Bodlaender and T. Kloks, Efficient and Constructive algorithms for pathwidth and treewidth of graphs, *Journal of Algorithms* **21** (1996) 358-402

[15] F. R. K. Chung, Spectral Graph Theory, American Mathematical Society, Providence, Rhode Island, 1997

[16] B. Courcelle, The monadic second order logic of graphs. I. Recognizable sets of finite graphs, *Information and Computation* **85** (1990) 12–75

[17] M. R. Fellows and M. A. Langston, Nonconstructive advances in polynomial-time complexity, *Information Processing Letters* **26** (1987) 157–162

[18] M. R. Fellows and M. A. Langston, Nonconstructive tools for proving polynomial-time decidability, *Journal of the Association for Computing Machinery* **35** (1988) 727–739

[19] L. Ford and D. Fulkerson, Maximal flow through a network, *Canad. J. Math.* **8** (1956) 399-404

[20] L. Ford and D. Fulkerson, A simple algorithm for finding maximal network flows and an application to the Hitchcock Problem, *Canad. J. Math.* **9** (1957) 210-218

[21] F. Gavril, Algorithms on Clique Seperable Graphs, *Discrete Mathematics* **19** (1977), 159-165

[22] M. C. Golumbic, Algorithmic Graph Theory and Perfect Graphs, Academic Press, Toronto, Ontario, 1980

[23] T. C. Hu, Integer Programming and Network Flows, Addison-Wesley, Don Mills, Ontario, 1969

[24] R. M. Karp, On the complexity of combinatorial problems, *Networks* **5** (1975) 45–68

[25] R. J. Lipton and R. E. Tarjan, A separator theorem for planar graphs, *SIAM Journal on Applied Mathematics* **36** (1979) 177–189

[26] J. Lynch, The equivalence of theorem proving and the interconnection problem, *Association for Computing Machinery's Special Interest Group on Design Automation Newsletter* **5** (1976)

[27] K. Menger, Zur allgemeinen Kurventheorie, *Fundamenta Mathematicae* **10** (1927) 96–115

[28] B. Mohar and C. Thomassen, *Graphs on Surfaces*, John Hopkins Univeristy Press, Baltimore, 2001

[29] B. Reed, Tree width and tangles, a new measure of connectivity and some applications, *Surveys in Combinatorics*, R. Bailey (ed.), *LMS Lecture Note Series* **241** (1997) 87–162, Cambridge University Press, Cambridge, UK

[30] B. A. Reed, Finding approximate separators and computing tree width quickly, *Proceedings of the 24th Annual Association for Computing Machinery Symposium on Theory of Computing*, ACM Press, New York, 1992, 221–228

[31] B. Reed, Disjoint connected paths: faster algorithms and shorter proofs, manuscript.

[32] N. Robertson and P. D. Seymour, Graph Minors. II. Algorithmic aspects of tree-width, *Journal of Algorithms* **7** (1986) 309–322

[33] N. Robertson and P. D. Seymour, Graph Minors. IV. Tree-width and well-quasi-ordering, *Journal of Combinatorial Theory, Series B* **48** (1990) 227–254

[34] N. Robertson and P. D. Seymour, Graph Minors. V. Excluding a planar graph, *Journal of Combinatorial Theory, Series B* **41** (1986) 92–114

[35] N. Robertson and P. D. Seymour, Graph Minors. VII. Disjoint paths on a surface, *Journal of Combinatorial Theory, Series B* **45** (1988) 212–254

[36] N. Robertson and P. D. Seymour, Graph Minors. VIII. A Kuratowski theorem for general surfaces, *Journal of Combinatorial Theory, Series B* **48** (1990) 255–288

[37] N. Robertson and P. D. Seymour, Graph Minors. X. Obstructions to tree-decomposition, *Journal of Combinatorial Theory, Series B* **52** (1991) 153–190

[38] N. Robertson and P. D. Seymour, Graph Minors. XIII. The disjoint paths problem, *Journal of Combinatorial Theory, Series B* **63** (1995) 65–110

[39] N. Robertson and P. D. Seymour, Graph Minors. XVI. Excluding a non-planar graph, manuscript.

[40] N. Robertson and P. D. Seymour, Graph Minors. XX. Wagner's Conjecture, manuscript, 1988.

[41] N. Robertson, P. D. Seymour and R. Thomas, A survey of linkless embeddings, *Graph Structure Theory (Proceedings of the AMS-IMS-SIAM Joint Summer Research Conference on Graph Minors, Seattle, 1991)*, N. Robertson and P. Seymour (eds.), *Contemporary Mathematics* **147** (1993) 125–136, American Mathematical Society, Providence, Rhode Island

[42] N. Robertson, P. Seymour and R. Thomas, Quickly excluding a planar graph, *Journal of Combinatorial Theory, Series B* **62** (1994) 323–348

[43] P. Seymour, A bound on the excluded minors for a surface, manuscript.

[44] P. D. Seymour and R. Thomas, Graph searching and a min-max theorem for tree-width, *Journal of Combinatorial Theory, Series B* **58** (1993) 22–33

[45] R. Thomas, A Menger-like property of tree-width: the finite case, *Journal of Combinatorial Theory, Series B* **48** (1990) 67–76

[46] M. Thorup, All structured programs have small tree-width and good register allocation, *Information and Computation* **142** (1998) 159–181

[47] S. Whitesides, An Algorithm for Finding Clique Cutsets, *Information Processing Letters* **12** (1981) 31–32

5

A Survey on Clique Graphs

J.L. Szwarcfiter[1]

5.1 Introduction

Intersection graphs, in general, have been receiving attention in graph theory, for some time. For example, there are specific papers on this subject, dated some sixty years ago. On the other hand, two books, [14] and [56], appeared recently where intersection graphs play a central role. The book [30] also deals with various classes of intersection graphs.

Clique graphs form a class of intersection graphs. In this sense, the clique graph of a graph G exhibits the way in which the cliques of G are arranged. Clique graphs have been also studied in the context of graph operators [70]. A point of attraction in this study is the variety of different situations which arise as a result of taking the clique graph of a graph. Clique graphs were included in the books [14], [56] and [70]. Besides several papers have been written on the subject, since the sixties.

In this work, we survey some of the results on clique graphs. Comments related to the computational complexities of some clique graph problems have been included. In Section 2, we examine the effect on clique graphs, of some known binary graph operations. In the following section, we address the questions of characterization and recognition of the class. A study of clique graphs of classes of graphs is the subject of Section 4. On the other hand, Section 5 examines the inverse of the clique graph operation, for some classes of graphs. Iterated clique graphs form the subject of Section 6. In particular, topics of convergence and divergence of iterated clique graphs are described in Section 7. Diameters of iterated clique graphs are studied in Section 8. Finally, in the last section there is a brief description of some related topics and a list of open problems. An appendix contains definitions of graph classes mentioned in the text.

Consider undirected and simple graphs. Denote by $V(G)$ the vertex set and by $E(G)$ the edge set of a graph G. Represent by $N(v)$ and $N[v]$ the open and closed neighbourhood of a vertex $v \in V(G)$. If v and w are vertices of G satisfying $N[v] = N[w]$ then v, w are *twins*. If $N[v] \subseteq N[w]$ say that v is *dominated by w*.

[1]Partially supported by the Conselho Nacional de Desenvolvimento Científico e Tecnológico, CNPq, and Fundação de Amparo à Pesquisa do Estado do Rio de Janeiro, FAPERJ, Brazil

By $d(v, w)$ denote the *distance* between v and w in G, that is, the length of the shortest $v - w$ path. Let $G[v_1, \ldots, v_k]$ represent the subgraph induced in G by $\{v_1, \ldots, v_k\} \subseteq V(G)$. The symbol c_n denotes an induced cycle of length n. If v is a vertex of a subgraph H of G adjacent to every other vertex of H then v is *universal* in H. The *k-th power of* G, denoted by G^k, is the graph obtained from G by including the edge (v, w), whenever $d(v, w) \leq k$. Finally, the symbol $\overline{G}$ represents the *complement* of G.

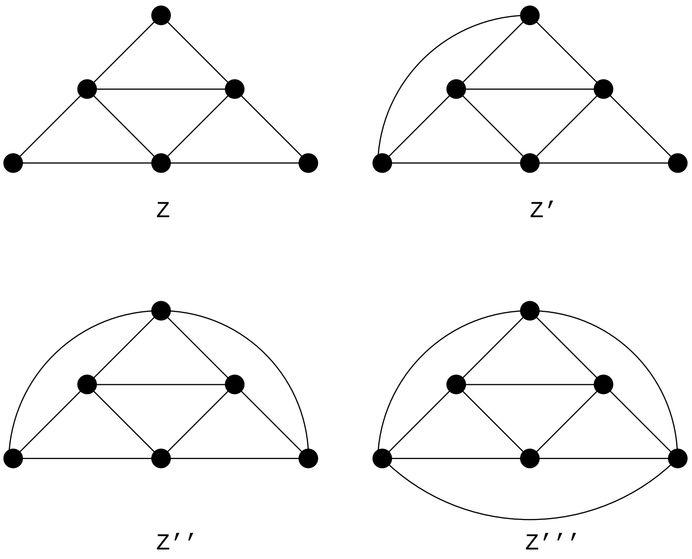


Figure 5.1. Extended Hajós Graphs

A *complete* of G is a subset of vertices pairwise adjacent. A *clique* is a maximal complete. An *independent set* is a subset of vertices pairwise non adjacent. The *clique graph* of G, denoted by $K(G)$, is the intersection graph of the cliques of G. In this case, G is a *clique inverse* graph of $K(G)$. The graph Z of Figure 1 is called *Hajós graph*, while the graphs Z, Z', Z'', Z''' are the *extended Hajós graphs.* For example, the graph Z is such that $K(Z) = K_4$. Hence K_4 is the clique graph of the Hajós graph, whereas the latter is a clique inverse graph of K_4. An induced star with four leaves is another clique inverse graph of K_4. On the other hand, not all graphs are clique graphs. Indeed, none of the extended Hajós graphs are clique graphs.

5.2 Operations on Clique Graphs

To start our study, it would be interesting to examine the effect of some binary operations in relation to clique graphs. Let G_1, G_2 be vertex disjoint graphs. In special, consider the following specific operations.

Union: $G_1 \cup G_2$ is the graph such that $V(G_1 \cup G_2) = V(G_1) \cup V(G_2)$ and $E(G_1 \cup G_2) = E(G_1) \cup E(G_2)$.

Join: $G_1 + G_2$ is defined as $V(G_1 + G_2) = V(G_1 \cup G_2)$ and $E(G_1 + G_2) = E(G_1 \cup G_2) \cup [V(G_1) \times V(G_2)]$, where $V(G_1) \times V(G_2)$ represents the set of unordered pairs (v_1, v_2), with $v_1 \in V(G_1)$ and $v_2 \in V(G_2)$.

Cartesian Product: $G_1 \times G_2$ is the graph where $V(G_1 \times G_2) = V(G_1) \times V(G_2)$ and for $v_1, w_1 \in V(G_1)$ and $v_2, w_2 \in V(G_2)$, (v_1, v_2) and (w_1, w_2) are adjacent vertices in $G_1 \times G_2$ precisely when $(v_1, w_1) \in E(G_1)$ and $(v_2, w_2) \in E(G_2)$.

Dot Product: $G_1 \circ G_2$ has vertex set $V(G_1 \circ G_2) = V(G_1 \times G_2)$ and edges as follows. For $v_1, w_1 \in V(G_1)$ and $v_2, w_2 \in V(G_2)$, (v_1, v_2) and (w_1, w_2) are adjacent vertices in $G_1 \circ G_2$ if (i) $v_1 = w_1$ and $(v_2, w_2) \in E(G_2)$, or (ii) $v_2 = w_2$ and $(v_1, w_1) \in E(G_1)$, or (iii) $(v_1, w_1) \in E(G_1)$ and $(v_2, w_2) \in E(G_2)$.

The following theorem relates the clique graphs of G_1, G_2, with that of a graph obtained by an operation of G_1, G_2.

Theorem 5.2.1 *[58]: Let G_1, G_2 be graphs with disjoint vertex sets. Then*

(1) $K(G_1 \cup G_2) = K(G_1) \cup K(G_2)$

(2) $\overline{K(G_1 + G_2)} = \overline{K(G_1)} \times \overline{K(G_2)}$

(3) $K(G_1 \circ G_2) = K(G_1) \circ K(G_2)$

Proof.Equality (1) is trivial. For (2), because each pair of vertices v_1, v_2, where $v_1 \in V(G_1)$ and $v_2 \in V(G_2)$, is adjacent in $G_1 + G_2$, it follows that each clique of $G_1 + G_2$ corresponds to a pair of cliques C_1 of G_1 and C_2 of G_2. Consequently, $V(K(G_1 + G_2)) = V(K(G_1)) \times V(K(G_2))$. Let C, C' be two cliques of $G_1 + G_2$, while C_1, C_2 and C'_1, C'_2 are their corresponding pairs of cliques in G_1 and G_2, respectively. Then $C \cap C' = \emptyset$ if and only if $C_1 \cap C'_1 = \emptyset$ and $C_2 \cap C'_2 = \emptyset$. Consequently, (2) holds. The proof of (3) is similar. □

5.3 A General Characterization

This section discusses the question of which graphs are clique graphs and the related recognition problem.

Let S be a family of subsets of some set. Say that S *satisfies the Helly property* when every subfamily of it, formed by pairwise intersecting subsets, contains a common element.

A graph is *clique-Helly* when its family of cliques satisfies the Helly property. The Hajós graph is the smallest graph which is not clique-Helly. This class of graphs has a central role in the study of clique graphs.

Given a graph H, the question is whether or not H is a clique graph. A sufficient condition has been formulated in [40], stating that H is a clique graph, whenever it is clique-Helly. On the other hand, examples of clique graphs which are not clique-Helly are given in Figure 3.

An *edge cover (by completes)* of a graph G is a collection $\mathcal{C}$ of completes of G, such that every edge of G has its both ends in some complete of $\mathcal{C}$.

By turning weaker the condition of [40], the following characterization has been formulated in [79]. In fact, it is the only general characterization for clique graphs so far known.

Theorem 5.3.1 *[79]: A graph is a clique graph if and only if it admits an edge cover satisfying the Helly property.*

Proof. Let $H = K(G)$, $V(G) = \{v_1, \dots, v_n\}$ and $C_1, \dots, C_\ell$ the cliques of G. For $v_i \in V(G)$, denote $L_i = \{C_j | v_i \in C_j\}$ and $L = \{L_1, \dots, L_n\}$. Clearly, each L_i is a complete of H, because every $C_j \in L_i$ contains v_i. In addition, L is an edge cover of H, because $(C_j, C_k) \in E(H)$ implies $v_i \in C_j \cap C_k$, for some v_i. Finally, let $L' \subseteq L$ be a subfamily of pairwise intersecting completes of L. Without loss of generality, let $L' = \{L_1, \dots, L_p\}$. Then there exists a vertex $C_{jk} \in V(H)$, such that $C_{jk} \in L_j \cap L_k$. Then $(v_j, v_k) \in E(G)$, for $1 \leq j \leq k \leq p$. The clique of G containing the complete $G[v_1, \dots, v_p]$ belongs to L_j, for $1 \leq j \leq p$. Consequently, L satisfies the Helly property. Conversely, let $L = \{L_1, \dots, L_n\}$ be an edge cover by completes, satisfying the Helly property. Construct a graph G, with vertex set $V(G) = V(H) \cup L$, as follows. For $C_i \in V(H)$ and $L_j \in L$, $(C_i, L_j) \in E(G)$ precisely when $C_i \in L_j$. For $L_i, L_j \in L$, $(L_i, L_j) \in E(G)$ when $i \neq j$ and $L_i \cap L_j \neq \emptyset$. The graph G contains no other edges. It follows that $H = K(G)$, completing the proof. □

A consequence of the above theorem is that a K_4-free graph is a clique graph if and only if it is clique-Helly.

So far, this characterization did not lead to a polynomial time algorithm for recognizing clique graphs. In fact, it is an open question to determine the complexity of the recognition problem [14, 71]. However, it is simple to conclude that the decision problem belongs to $\mathcal{NP}$. This is so because we need no more than $|O(E(H))|$ completes to cover the edges of a graph H and there is a polynomial time algorithm to verify if a given collection of subsets satisfies the Helly property [6, 79].

A graph is *clique-complete* when every pair of its cliques intersect. Clique-complete graphs were first considered in [57]. Clearly, a clique-complete graph is clique-Helly precisely when it contains a universal vertex. The following theorem describes a family of graphs, appearing as induced subgraphs in a clique-complete graph with no universal vertex. For $n \geq 3$, let Q_n be the graph whose vertices can be partitioned into two subsets V_1, V_2, both of size n, as follows: $V_1 = c_n$, $V_2 = K_n$, and each vertex of V_1 is adjacent in $\overline{Q_n}$ exactly to one vertex of V_2 and conversely. There are no other edges. Note that Q_3 is the Hajós graph. See Figure 2.

Theorem 5.3.2 *[55]: Every clique-complete graph with no universal vertex contains Q_{2n+1} as an induced subgraph, for some $n \geq 1$.*

On the other hand, in a clique graph, each induced subgraph which is isomorphic to an extended Hajós graph, must be contained in a (larger) special subgraph, as described by the following theorem.

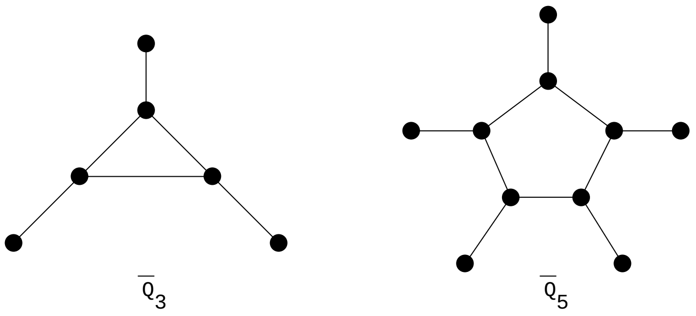


Figure 5.2. Minimal clique-complete graphs with no universal vertices

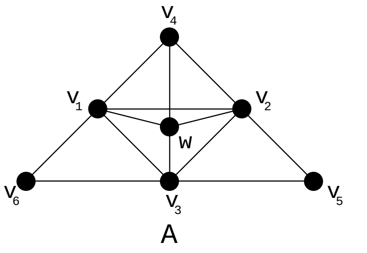
Theorem 5.3.3 *[36]: Let H be a clique graph. If H contains an induced subgraph H' isomorphic to an extended Hajós graph then H' must be contained in a (not necessarily induced) subgraph of H isomorphic to graphs A or B of Figure 3.*

Proof. Let H' be and induced subgraph of H, isomorphic to the Hajós graph. Let $V(H') = \{v_1, \ldots, v_6\}$, labelled as in Figure 3. By Theorem 2, H has a Helly edge cover $\mathcal{C}$, formed by completes. First, suppose $\{v_1, v_2, v_3\} \subseteq C_1$, for some complete $C_1 \in \mathcal{C}$. If there are completes $C_2, C_3, C_4 \in \mathcal{C}$, containing the subsets $\{v_1, v_2, v_4\}, \{v_1, v_3, v_6\}, \{v_2, v_3, v_5\}$, respectively, then H must contain a vertex adjacent to all vertices of H', meaning that A is a subgraph of H. Otherwise, without loss of generality, $\{v_1, v_4\}$ and $\{v_2, v_4\}$ are contained in distinct completes $C_2', C_2'' \in \mathcal{C}$. Because $C_1 \cap C_2' \cap C_2'' \neq \emptyset$ it follows that there exists a vertex w adjacent to all vertices of $C_1 \cup C_2' \cup C_2''$. Consequently H contains A. Otherwise, $\{v_1, v_2, v_3\} \not\subseteq C$, for any $C \in \mathcal{C}$, and consider the following alternatives. Suppose that $\{v_1, v_2, v_4\} \subseteq C_1$, for some $C_1 \in \mathcal{C}$. Then $\{v_1, v_3\}$ and $\{v_2, v_3\}$ must be covered by distinct completes C_2 and C_3. Since C_1, C_2, C_3 pairwise intersect, there exists $w \in C_1 \cap C_2 \cap C_3$. Then H contains A. The situations where $\{v_1, v_3, v_6\}$ or $\{v_2, v_3, v_5\}$ are contained in a complete of $\mathcal{C}$ are similar. The last alternative is the case where the edges in each of the subsets $\{v_1, v_2, v_3\}, \{v_1, v_2, v_4\}, \{v_1, v_3, v_6\}$ and $\{v_2, v_3, v_5\}$ belong to distinct completes of $\mathcal{C}$. Consequently, there exists a vertex w_i, $1 \leq i \leq 4$, adjacent to all the vertices of each of these subsets, respectively. If two $w_i's$ coincide then H contains A. Consider w_1, w_2, w_3, w_4 as distinct. Then w_1 must be adjacent to w_2, w_3, w_4, otherwise $\mathcal{C}$ does not satisfy Helly. Therefore H contains B. □

The above theorem might be useful in a recognition process for clique graphs.

5.4 Clique Graphs of Classes of Graphs

Given a class $\mathcal{A}$ of graphs, denote by $K(\mathcal{A})$ the class containing exactly the clique graphs of the graphs of $\mathcal{A}$. In this section, we discuss the problems of characterizing and recognizing the graphs of $K(\mathcal{A})$, for several classes $\mathcal{A}$. In general, we identify a class by capital letters. For example, $INTERVAL$ is the class of interval



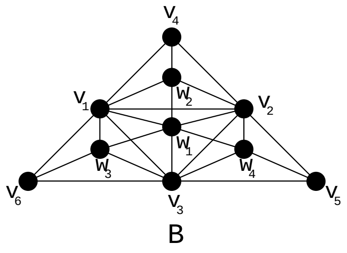


Figure 5.3. Graphs A and B

graphs. In the appendix there is a list of definitions of classes considered in this text.

A class of graphs $\mathcal{A}$ is *fixed* when $K(\mathcal{A}) = \mathcal{A}$. The result that clique-Helly graphs form a fixed class of graphs is fundamental in the study of clique graphs.

Theorem 5.4.1 *[28]: K(CLIQUE-HELLY) = CLIQUE-HELLY.*

Proof.Let G be a clique-Helly graph, $H = K(G)$, and $\mathcal{C}$ a family of pairwise intersecting cliques of H. Because G is clique-Helly, there is a vertex $v_i \in V(G)$ common to all cliques of $\mathcal{C}$, which form the clique $C_i \in \mathcal{C}$. The collection of such vertices v_i, for $C_i \in \mathcal{C}$, is a complete C of G. Therefore any maximal clique of G, containing C, is a vertex of H common to all $C_i \in \mathcal{C}$. Consequently, $K(G)$ is clique-Helly.

It remains to show that any clique-Helly graph H is the clique graph of some clique-Helly graph. From [40], it follows that $H = K(G)$, for some clique-Helly G. $\square$

As for the recognition problem, clique-Helly graphs can be recognized in polynomial time. With the purpose of describing such a method, let G be a graph and T a triangle of it. The *extended triangle* T' *of* G, *relative to* T, is the subgraph induced in G by the set formed by all the vertices adjacent to at least two of the vertices of T. The following is a characterization of clique-Helly graphs, which leads to a polynomial time recognition algorithm.

Theorem 5.4.2 *[25, 83]: A graph G is clique-Helly if and only if every extended triangle of it contains a universal vertex.*

Proof.Let T be a triangle of G. Suppose that the extended triangle T', relative to T, does not contain a universal vertex. Let $\mathcal{C}$ be the collection of cliques of G, containing at least one edge of T. It follows that $\mathcal{C}$ is a collection of pairwise intersecting cliques of G, with no common vertex. Consequently, G is not clique-Helly, a contradiction. Conversely, by hypothesis, every extended triangle T' of G contains a universal vertex. Suppose that G is not clique-Helly. Let $\mathcal{C}$ be a minimal family of pairwise intersecting cliques C_i of G, with no common vertex. By the minimality of $\mathcal{C}$, there exists a triangle T with vertices v_1, v_2, v_3 such that v_i is a common vertex of $\mathcal{C} \setminus C_i$, $1 \leq i \leq 3$. The extended triangle T' of G contains a universal vertex. This leads to conclude that $\mathcal{C}$ has a common vertex, a contradiction. Therefore, G is clique-Helly. $\square$

The following theorem shows that clique graphs of interval graphs also remain in the class, but do not cover all its domain.

Theorem 5.4.3 *[43]: K(INTERVAL) = PROPER INTERVAL*

Classes as those of interval graphs, such that $K(A) \subseteq A$ are called *closed.* Chordal graphs are not closed, because their clique graphs are not necessarily chordal. In fact they correspond to the following class.

Theorem 5.4.4 *[12, 32, 84]: K(CHORDAL) = DUALLY CHORDAL*

The class of dually chordal graphs can be recognized in polynomial time [84]. Moreover, a linear time algorithm has been described in [12]. Additionally, the complexities of some optimization problems, specialized to dually chordal graphs, have been determined in [11]. Another main source for solving optimization problems on dually chordal graphs is the paper [13]. A relation between squares of chordal graphs and dually chordal graphs is described in [86].

Clique graphs of several subclasses of chordal graphs have been characterized, so far. On the other hand, there are proeminent classes of graphs, as planar graphs and comparability graphs, whose clique graphs have not yet been characterized.

Table 5.1 summarizes some classes of graphs, together with their corresponding classes of clique graphs. As it can be observed from the table, most of the classes whose clique graphs have been characterized so far, can be classified into three types: fixed classes, closed classes and classes $\mathcal{A}$, such that $\mathcal{A}$ and $K(\mathcal{A})$ overlap, but $K(K(\mathcal{A})) \subseteq \mathcal{A}$.

Finally, we mention some more general results on classes of clique graphs. In [5], it has been shown that the underlying graphs of certain families of hypergraphs form a fixed class, provided these families consist of self-dual conformal hypergraphs, closed under the operations of reduction and addition of isolated edges. On the other hand, [33, 34] describe a characterization of certain intersection graphs, which can be applied as a technique for proving several results on clique graph classes.

5.5 Clique Inverse Classes

Let $\mathcal{A}$ be a class of (clique) graphs. Denote by $K^{-1}(\mathcal{A})$ the class of all clique inverse graphs of the graphs of $\mathcal{A}$. In the present section, we describe results concerning the characterization and recognition of $K^{-1}(\mathcal{A})$, for some classes $\mathcal{A}$. The reference [73] is devoted to this topic.

Clique inverse graphs of K_3-free graphs can be characterized by forbidden subgraphs, as follows.

Theorem 5.5.1 *[74]: A graph G is a clique inverse graph of a K_3-free graph if and only if it does not contain an induced subgraph isomorphic to any of the graphs of Figure 4.*

CLASS $\mathcal{A}$	$K(\mathcal{A})$	*REFS*
BLOCK	*BLOCK*	[42]
CLIQUE-HELLY	*CLIQUE-HELLY*	[28]
CHORDAL	*DUALLY CHORDAL*	[12, 32, 84]
CLOCKWORK	*CLOCKWORK*	[48]
DE	*DUALLY DE*	[35]
DIAMOND FREE	*DIAMOND FREE*	[21]
DISK-HELLY	*DISK-HELLY*	[5]
DISMANTABLE	*DISMANTABLE*	[5]
DUALLY CHORDAL	*CHORDAL* $\cap$ *CLIQUE-HELLY*	[12, 32]
DUALLY DE	*DE*	[35]
DUALLY DV	*DV*	[38, 72]
DUALLY RDV	*RDV*	[10, 72]
DV	*DUALLY DV*	[38, 72]
$\mathcal{H}_1$	$\mathcal{H}_1$	[23]
HELLY CIRCULAR ARC	*CIRCULAR CLIQUE*	[26]
HELLY HEREDITARY	*HELLY HEREDITARY*	[68]
INTERVAL	*PROPER INTERVAL*	[43]
MIN PROPER INTERVAL	*PROPER INTERVAL*	[37]
PROPER INTERVAL	*PROPER INTERVAL*	[43]
PTOLOMAIC	*PTOLOMAIC*	[5]
RDV	*DUALLY RDV*	[10, 72]
SPLIT	*STAR*	
STRONGLY CHORDAL	*STRONGLY CHORDAL*	[5, 12]
TREE	*BLOCK*	[42]
UV	*DUALLY CHORDAL*	[84]

Table 5.1. Clique Graph Classes

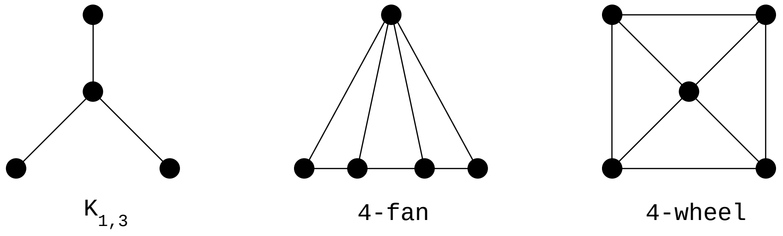


Figure 5.4. Forbidden subgraphs for clique-inverse graphs of K_3-free graphs

Proof.The graphs of Figure 4 have all three mutually intersecting cliques. Therefore if G contains any of those as an induced subgraph, $K(G)$ would contain a triangle. Conversely, by hypothesis, G does not contain any of these graphs as an induced subgraph. We show that $K(G)$ is triangle free.

Assume the contrary and let C_1, C_2, C_3 be three distinct pairwise intersecting cliques of G. For $I \subseteq \{1,2,3\}$, denote by V_I the subset of vertices of G lying exactly in every of the cliques C_i and in none of the cliques C_j, $i \in I$ and $j \in \{1,2,3\} \setminus I$. For simplicity, write V_{123}, instead of $V_{\{1,2,3\}}$, V_{12} instead of $V_{\{1,2\}}$ and so on. Consider the following situations.

Case 1 : $V_{123} \neq \emptyset$ and $V_{12} = V_{13} = V_{23} = \emptyset$
Let $u \in V_{123}$, $u_1 \in V_1$, $u_2 \in V_2$ and $u_3 \in V_3$, such that $(u_1, u_2), (u_1, u_3) \notin E(G)$. If $(u_2, u_3) \notin E(G)$ then $G[u, u_1, u_2, u_3] = K_{1,3}$. Otherwise, let $u_3' \in V_3$ satisfying $(u_2, u_3') \notin E(G)$. If $(u_1, u_3') \in E(G)$ then $G[u, u_1, u_2, u_3, u_3']$ is a 4-fan, otherwise $G[u, u_1, u_2, u_3'] = K_{1,3}$.

Case 2 : $V_{123}, V_{12} \neq \emptyset$ and $V_{13}, V_{23} = \emptyset$
Let $u \in V_{123}$, $u_{12} \in V_{12}$, $u_1 \in V_1$, $u_2 \in V_2$ and $u_3 \in V_3$, such that $(u_1, u_2), (u_{12}, u_3) \notin E(G)$. If $(u_1, u_3), (u_2, u_3) \notin E(G)$ it follows that $G[u, u_1, u_2, u_3] = K_{1,3}$. If $(u_1, u_3) \in E(G)$ and $(u_2, u_3) \notin E(G)$ (or $(u_2, u_3) \in E(G)$ and $(u_1, u_3) \notin E(G)$) then $G[u, u_1, u_{12}, u_2, u_3]$ is a 4-fan. If $(u_1, u_3), (u_2, u_3) \in E(G)$ it follows that the latter subgraph is a 4-wheel.

Case 3 : $V_{123}, V_{12}, V_{13} \neq \emptyset$ and $V_{23} = \emptyset$
Let $u \in V_{123}$, $u_{12} \in V_{12}$, $u_{13} \in V_{13}$, $u_2 \in V_2$, $u_3 \in V_3$, such that $(u_{13}, u_2), (u_{12}, u_3) \notin E(G)$. Then $G[u, u_{12}, u_{13}, u_2, u_3]$ is either a 4-wheel or a 4-fan, according to whether or not u_2, u_3 are adjacent, respectively.

Case 4 : $V_{123}, V_{12}, V_{13}, V_{23} \neq \emptyset$
Let $u \in V_{123}$, $u_{12} \in V_{12}$, $u_{13} \in V_{13}$, $u_{23} \in V_{23}$, $u_2 \in V_2$, $u_3 \in V_3$, such that $(u_{13}, u_2), (u_{12}, u_3) \notin E(G)$. Similarly as in Case 3, $G[u, u_{12}, u_{13}, u_2, u_3]$ is a 4-wheel or a 4-fan, according to whether u_2, u_3 are adjacent or not.

Case 5 : $V_{123} = \emptyset$
Then $V_{12}, V_{13}, V_{23} \neq \emptyset$. Choose $u_1 \in V_1$, $u_2 \in V_2$, $u_{12} \in V_{12}$, $u_{13} \in V_{13}$, $u_{23} \in V_{23}$, such that $(u_1, u_{23}), (u_{13}, u_2) \notin E(G)$. It follows that $G[u_1, u_{12}, u_{13}, u_2, u_{23}]$ is a 4-wheel or a 4-fan, depending on whether or not u_1, u_2 are adjacent, respectively.

All situations have been covered. In each of them a graph of Figure 4 has been obtained. This completes the proof. □

The list of forbidden subgraphs increases, when considering K_4-free graphs, instead of K_3-free.

Theorem 5.5.2 *[74]: A graph G is a clique inverse graph of a K_4-free graph if and only if it does not contain an induced subgraph isomorphic to any of the graphs of Figure 5.*

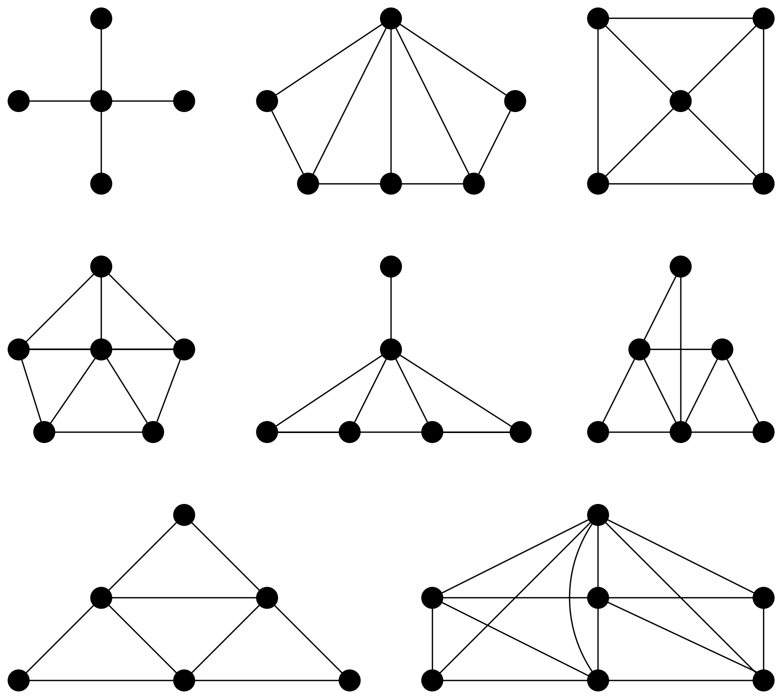


Figure 5.5. Forbidden subgraphs for clique-inverse graphs of K_4-free graphs

Clique inverse graphs of bipartite graphs can also be described by forbidden subgraphs.

Theorem 5.5.3 *[76]: A graph G is a clique inverse graph of a bipartite graph if and only if it does not contain as an induced subgraph any of the following: $K_{1,3}$, 4-fan, 4-wheel nor c_{2k+5}, $k \geq 0$.*

However, for some classes of graphs $\mathcal{A}$, recognizing graphs of $K^{-1}(\mathcal{A})$ is NP-hard. The following theorem shows that deciding whether a given graph is a clique inverse of a complete graph is Co-NP-complete.

Theorem 5.5.4 *[55]: Recognizing clique-complete graphs is Co-NP-complete.*

Proof. Transformation from the satisfiability problem. Let E be a boolean expression in conjunctive normal form, with clauses $L_i, 1 \leq i \leq p$, each L_i having q_i literals. Construct a graph G, as follows. There is one vertex v_i of G, for each clause L_i. In addition, one vertex w_{ij}, for each occurrance of a literal in L_i, $1 \leq i \leq p$ and $1 \leq j \leq q_i$. There are two additional vertices, u_1 and u_2. The edges of G are the following. For all $1 \leq i, k \leq p$, $i \neq k$ and $1 \leq j \leq q_i$, $(v_i, v_k), (v_i, w_{kj}) \in E(G)$. Denote by ℓ_{ij} the literal of

CLASS $\mathcal{A}$	$K^{-1}(\mathcal{A})$	*REFS*
BIPARTITE	$\mathcal{P}$	[76]
3-COLOURABLE	*NP-complete*	[75]
CHORDAL	*Co-NP-complete*	[75]
CHORDAL BIPARTITE	$\mathcal{P}$	[76]
CLIQUE-HELLY	*NP-hard*	[24]
CO-COMPARABILITY	*NP-hard*	[75]
CO-INTERVAL	*NP-hard*	[75]
COMPARABILITY	*NP-hard*	[75]
COMPLETE	*Co-NP-complete*	[55]
INTERVAL	*Co-NP-Complete*	[75]
$K_3 - FREE$	$\mathcal{P}$	[74]
$K_4 - FREE$	$\mathcal{P}$	[74]
SPLIT	*Co-NP-complete*	[75]
TREE	$\mathcal{P}$	[76]
TRIANGLE FREE	$\mathcal{P}$	[76]

Table 5.2. Complexity of Recognizing Clique Inverse Classes

L_i, corresponding to w_{ij}. The edges (w_{ij}, w_{kt}) exist precisely when $i \neq k$ and $\ell_{ij} \neq \overline{\ell_{kt}}$. Vertex u_1 is adjacent to all vertices of G, except u_2. The neighbours of u_2 are $v_1, \ldots, v_p$. The construction of G is completed. If E is satisfiable, let w_{ij_i} be the vertex of G corresponding to the literal of E, which satisfies clause L_i. In this case, $u_1, w_{1j_1}, \ldots, w_{pj_p}$ and $u_2, v_1, \ldots, v_p$ are disjoint cliques of G. Conversely, if G contains a pair of disjoint cliques then one of them, is $u_1, w_{1j_1}, \ldots, w_{pj_p}$ and the other is $u_2, v_1, \ldots, v_p$. Consequently, E is satisfiable if and only if G is not clique-complete. Finally, a certificate for G not to be clique-complete is a pair of disjoint cliques. Therefore recognizing clique-complete graphs is Co-NP-complete. □

Assuming that we know how to recognize graphs of a class $\mathcal{A}$, the task of verifying whether a given graph G belongs to $K^{-1}(\mathcal{A})$ becomes simple, if the number of cliques of G is bounded by a polynomial in $|V(G)|$. Sufficient conditions for a graph to have a polynomial number of cliques have been described in [4, 71, 75].

Table 5.2 illustrates some classes of graphs, whose complexity of recognizing the corresponding clique-inverse classes have been determined. That is, whether each recognition problem is NP-hard or belongs to the class $\mathcal{P}$.

Finally, the following concept provides additional information about minimal clique-inverse graphs. A graph G is *critical* when $K(G) \neq K(G - v)$, for any $v \in V(G)$. For a fixed graph H, the set of critical graphs G satisfying $H = K(G)$ is finite [29]. Observe that, for any clique graph H, there are infinite graphs G satisfying $H = K(G)$. However, it is simple to prove that recognizing critical graphs is NP-hard. The reduction is from the problem of recognizing clique-complete graphs, which is Co-NP-complete by Theorem 12.

5.6 Iterated Clique Graphs

Let G be a graph. Denote $K^0(G) = G$ and $K^i(G) = K(K^{i-1}(G))$, $i > 0$. Call $K^i(G)$ as the *i-th iterated clique graph of* G. In the present section, we examine questions related to this concept.

Clique-Helly graphs play a central role in the study of iterated clique graphs, once more. The following theorem is fundamental and its proof provides a simple and complete description of the second iterated clique graph of a clique Helly graph.

Theorem 5.6.1 *[28]: Let G be a clique-Helly graph. Then $K^2(G)$ is an induced subgraph of G.*

Proof. Let H be an induced subgraph of G, obtained by identifying each subset of pairwise twins of G and afterwards removing dominated vertices. We show that $H = K^2(G)$. Associate to each $v_i \in V(H)$ the family $\mathcal{C}_i$ of cliques of G, containing v_i. Since v_i is not dominated and G is clique-Helly, it follows that $\mathcal{C}_i$ corresponds to a clique of $K(G)$, hence to a vertex w_i of $K^2(G)$. In addition, since G is clique-Helly, every clique of $K(G)$ $\mathcal{C}_i$ contains a (unique) common vertex v_i. Consequently, there is a one-to-one correspondence between vertices of $V(H)$ and $V(K^2(G))$. Furthermore, $v_i, v_j \in V(H)$ are adjacent in H if and only if there is a clique of G common to $\mathcal{C}_i$ and $\mathcal{C}_j$, meaning that $w_i, w_j \in V(K^2(G))$ are adjacent. Consequently, $H = K^2(G)$. □

Induced subgraphs of clique-Helly graphs have been also considered in [54].

For a graph G, assume that $K^i(G) = G$, for some $i > 0$. The value of the smallest i satisfying the latter equation if the *period* of G, while G itself is called a *periodic graph.*

The above theorem implies that periodic clique-Helly graphs can be recognized, using the following simple assertion. Let G be a clique-Helly graph. Then G is periodic if and only if $N[v] \not\subseteq N[w]$, for all distinct vertices $v, w \in V(G)$. Moreover, when G is a periodic clique-Helly graph it follows that the period of G is 1 or 2.

In general, a graph G of period 1 is called a *self-clique* graph. The one-vertex graph and cycles of length greater than 3 are simple examples of these graphs. Figure 6 illustrates some other examples. Further examples of self-clique graphs have been described in [3]. On the other hand, Theorem 13 also implies that the period of a periodic triangle free graph is always two, except if it consists of a simple cycle. The latter has been extended in [20], where there is a characterization of self-clique graphs whose cliques have all sizes at most 2, except precisely for one clique. The problem of characterizing selff-clique graphs remains open, even if restricted to clique-Helly graphs. Self-clique graphs have been also considered in [29].

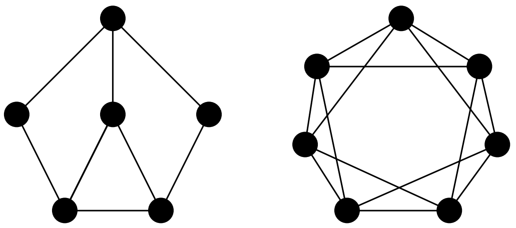


Figure 5.6. Self-clique graphs

The following is a sufficient condition for a graph to be self-clique.

Theorem 5.6.2 *[7]: Let G be a graph with minimum degree at least 2 and girth at least $6k + 1$, $k \geq 1$. Then G^{2k} is a self-clique graph.*

In a sense, the above theorem is best possible. This is so because if G has minimum degree 1 or girth $6k$ then G^{2k} is not necessarily self-clique. A graph G formed by a c_7, together with an additional vertex adjacent exactly to one vertex of the cycle is an example where the degree condition of the theorem fails, while the girth condition is satisfied for $k = 1$. However G^2 is not self-clique. On the other hand, c_6 is an example of a graph where the degree condition is satisfied for $k = 1$, the girth condition fails and ${c_6}^2$ is not a self-clique graph.

As for higher periods, there are examples of periodic (non clique-Helly graphs), for any desired period [28].

If a graph is not clique-Helly, one might wonder whether its iterated clique graph could become clique-Helly. For a graph G, define the *Helly defect* of G as the smallest value i, such that $K^i(G)$ is clique-Helly. In [5] it has been shown that the Helly defect of a chordal graph is at most 1. On the other hand, answering a question of [5], it has been proved in [8] that there are graphs with any desired Helly defect. An example is the family of graphs G_i, $i \geq 1$, whose first three members are depicted in Figure 7. The Helly defect of G_i is $i-1$. However, it is NP-hard to recognize whether the Helly defect of a given graph is equal to 1 [24].

5.7 Convergence and Divergence

Let G and H be graphs. Say that G is *convergent to H* when $K^i(G) = K^{i+1}(G) = H$, for some $i \geq 0$. When H is the one-vertex graph, call G,

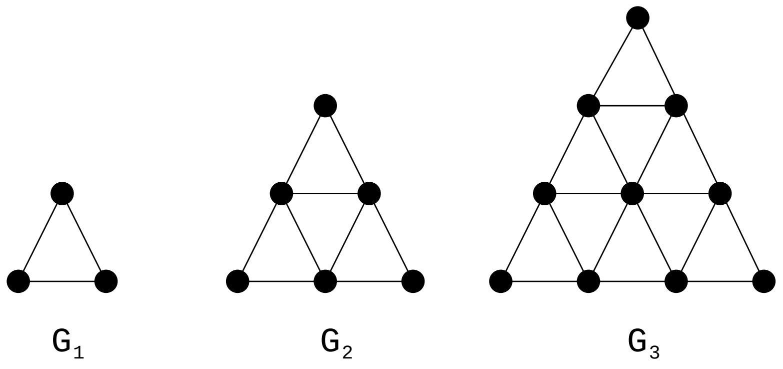


Figure 5.7. Graphs with increasing Helly defects

simply, *convergent.* On the other hand, when $\lim_{i\to\infty} |V(K^i(G))| = \infty$, call G a *divergent* graph. In this section, we examine convergence and divergence of graphs.

For the study of convergence, we remark that convergent clique-Helly graphs have been completely characterized. The following theorem implies a polynomial time algorithm for recognizing graphs of this class.

Theorem 5.7.1 *[5]: Let G be a clique-Helly graph. Then G is convergent if and only if G is dismantable.*

Proof.Because G is finite and by Theorems 5 and 13, G clique-Helly implies the existence of an integer $j \geq 0$ satisfying $K^i(G) = K^{i+2}(G)$, for all $i \geq j$. Suppose that G is convergent. Then $K^j(G)$ and $K^{j+1}(G)$ are the one-vertex graph. For any $0 \leq i < j$, if $K^i(G)$ is not dismantable then it contains a non-empty induced subgraph H formed by vertices which are not dominated, both in H and $K^i(G)$. By Theorem 13, H is preserved as an induced subgraph in $K^{i+2m}(G)$, for all $m \geq 0$. This contradicts $K^j(G)$ and $K^{j+1}(G)$ to be the one-vertex graph. Consequently, $K^i(G)$ is dismantable and so is G.

Conversely, by hypothesis G is a dismantable graph. Then $K^i(G)$ is dismantable, for all $i \geq 0$. Since G is clique-Helly, $K^j(G) = K^{j+2}(G)$, for some j. If G is not convergent, by Theorem 13, $K^j(G)$ has no dominated vertices, meaning that $K^j(G)$ is not dismantable, a contradiction. Therefore G must be convergent. □

Moreover, G is a disk-Helly graph if and only if it is dismantable and clique-Helly [5]. Dismantable graphs were considered in [63, 77], while disk-Helly graphs in [62, 78].

On the other hand, convergent graphs are not necessarily dismantable. Figure 8 illustrates such an example of a graph [67].

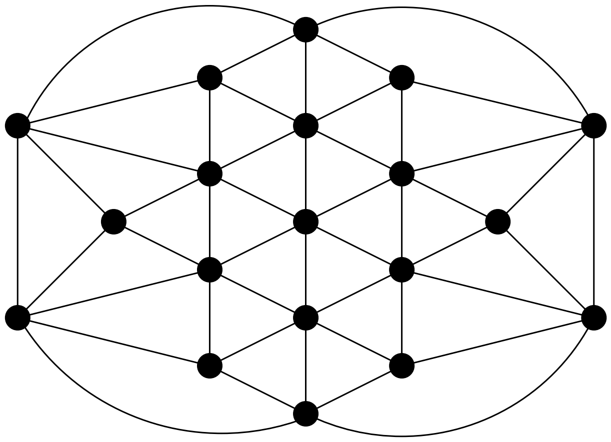


Figure 5.8. A convergent non dismantable graph

In general, much less is known about convergence, when non clique-Helly graphs are considered. If G is dismantable (and not clique-Helly) then it remains convergent [67]. The *index* of a convergent graph G is the smallest value of i, such that $K^i(G)$ equals the one-vertex graph. For example, the index of the graph of Figure 8 is equal to 7. Clique-complete graphs having at least two cliques correspond exactly to the graphs of index 2. By Theorem 12, it follows that recognizing convergent graphs of index 2 is Co-NP-complete. In fact, it is NP-hard to recognize convergent graphs of any given fixed index.

The study of convergence may have applications to other areas. For example, in [41] it has been shown that a finite order has the fixed point property whenever its comparability graph is convergent. However, there are finite orders with the fixed point property whose comparability graphs are divergent [53].

Next, we examine divergence. The class of divergent graphs has been investigated in [58, 59, 60]. For $n \geq 3$, denote by O_n the complement of a perfect matching on $2n$ vertices. Then O_3 is the extended Hajós graph Z'''. It follows that $K(O_n) = O_{2^{n-1}}$, meaning that O_n is divergent [28, 58].

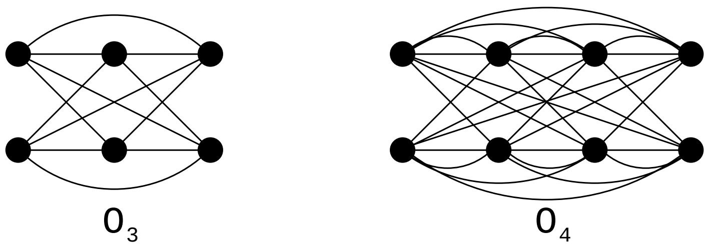


Figure 5.9. Graphs O_n

The following concepts are used for formulating a general sufficient condition for divergence. Let G_1, G_2 be graphs. A *homomorphism* is a function $\alpha : V(G_1) \rightarrow V(G_2)$, such that the image under α of adjacent vertices of G_1 either coincide or are adjacent in G_2. A homomorphism $\alpha : V(G_1) \rightarrow V(G_2)$ is a *retraction from G_1 to G_2*, when there exists a homomorphism $\beta : V(G_2) \rightarrow V(G_1)$, such that the composition $\alpha\beta$ is the identity fuction. In this case, G_2 is a *retract* of G_1. The concept of retraction has been introduced in [46] and has been later studied in many papers. The following theorem describes a relationship between retracts and clique graphs.

Theorem 5.7.2 *[59]: Let G_1, G_2 be graphs, such that G_2 is a retract of G_1. Then*

(i): *$K(G_2)$ is a retract of $K(G_1)$, and*

(ii): *If G_2 is divergent, so is G_1.*

Proof. For part (i), let $\alpha : V(G_1) \rightarrow V(G_2)$ and $\beta : V(G_2) \rightarrow V(G_1)$ be homomorphisms realizing G_2 as a retract of G_1. Let $C \subseteq V(G_2)$ be a complete of G_2 and $M_1(C)$ a clique of G_1, containing the complete $\beta(C)$. Similarly, define $M_2(C)$. Define functions $\alpha' : V(K(G_1)) \rightarrow V(K(G_2))$ and $\beta' : V(K(G_2)) \rightarrow V(K(G_1))$, as follows. For cliques C_1 and C_2 of G_1 and G_2, respectively, $\alpha'(C_1) = M_2(\alpha(C_1))$ and $\beta'(C_2) = M_1(\beta(C_2))$. It follows that α' and β' are homomorphisms satisfying $\alpha'\beta'(C_2) = C_2$, meaning that $K(G_2)$ is a retract of $K(G_1)$.

For part (ii), apply (i). Then $K^i(G_2)$ is a retract of $K^i(G_1)$. If G_2 is divergent, so is G_1. Otherwise, $V(K^i(G_1))$ is bounded meaning that $V(K^i(G_2))$ is bounded, an impossibility. □

It is possible to exhibit a retraction from the complete multipartite graph $K_{r_1,\ldots,r_m}$ to O_m, for $r_1, \ldots, r_m > 1$ and $m > 2$ [58]. Since O_m is divergent, the above theorem implies that $K_{r_1,\ldots,r_m}$ is divergent too. Similarly, the theorem also leads to conclude that $\overline{c_8}$ is divergent. In fact, $\overline{c_n}$ is divergent for $n \geq 8$. However, for a proof of the latter, the following additional concepts would be needed.

An *automorphism* α of a graph G is a bijective homomorphism of G into itself. An automorphism is *affine* when v and $\alpha(v)$ are adjacent, for all $v \in V(G)$. Similarly, α is *coaffine* when v and $\alpha(v)$ neither coincide nor are adjacent in G. The existence of coaffine automorphisms is preserved under the clique graph operation.

Theorem 5.7.3 *[59]: Let α be a coaffine automorphism for a graph G, and $\alpha_k : V(K(G)) \rightarrow V(K(G))$ a function satisfying $\alpha_k(C) = \alpha(C)$, where $\alpha(C)$ denotes the images under α of the vertices of C, for any clique C of G. Then α_k is a coaffine automorphism of $K(G)$*

Theorem 17 leads to the description of further families of divergent graphs, as $\overline{c_n}$, for $n \geq 8$. The above mentioned divergent graphs G are of exponential growth. In [60] it has been asked whether there are divergent graphs with polynomial growth. An affirmative answer has been given in [49], where there are descriptions of divergent graphs, whose growth is bounded by a polynomial of degree d, for any desired d. Furthermore, there are divergent graphs in which the number of vertices of its (finite) iterated clique graphs increase exactly by one, at each application of the clique graph operation [48].

The question to determine whether a graph converges to some graph or is divergent has been solved for the following family. Say that a graph is *locally* c_n when $N(v) = c_n$, for all $v \in V(G)$. Clearly, K_4 is the only connected locally c_3 graph and it is, of course, convergent. The extended Hajos graph Z''' is the only connected locally c_4 graph and it has been already mentioned that it is divergent. The icosahedron is the only connected locally c_5 graph. The question whether or not the icosahedron is divergent has been mentioned as open in [58, 59]. This has been answered in the affirmative in [65]. There is an infinite number of connected locally c_t graphs, for any $t \geq 6$. In [51] it has been proved that locally c_6 graphs are divergent. In contrast, locally c_t graphs are not divergent, for any $t > 6$ [52].

5.8 Diameters

In this section, we examine the behaviour of diameters of iterated clique graphs. The basic property is a close relationship between the diameters of a graph and that of its clique graph, as below described. Let v, v' be vertices of a graph G, while C, C' are cliques of G containing v and v', respectively. A shortest path $v - v'$ is called *diametral for* C, C', when its length equals $diam(G)$. In this case, C, C' are *diametral* cliques.

Theorem 5.8.1 *[43]: $diam(K(G)) - 1 \leq diam(G) \leq diam(K(G)) + 1$*

Proof. Let $t = diam(G)$ and $v_0, \ldots, v_t$ be a diametral path in G. Then each edge (v_{i-1}, v_i), $1 \leq i \leq t$, belongs to a clique C_i of G, disjoint of any C_j, except when C_i, C_j are consecutive in the sequence. A simple argument concludes that $C_1, \ldots, C_t$ is a shortest path between C_1 and C_t in $K(G)$, meaning that $diam(G) - 1 \leq diam(K(G))$.

For the righmost inequality, let $t = diam(K(G))$ and $C_0, \ldots, C_t$ be a diametral path of $K(G)$. Then G contains a shortest path $v_1, \ldots, v_t$, for $v_i \in C_{i-1} \cap C_i$. Hence $diam(K(G)) \leq diam(G) + 1$. $\square$

A similar relation holds for line graphs [47, 61]. The above theorem can be generalized, so that an equivalent result is valid for induced subgraphs of G [8].

A natural question is to classify all graphs G into *classes* 1,2 or 3, according to whether $diam(K(G)) - diam(G)$ equals to -1,0, or 1, respectively. With this purpose, we classify pairs of diametral cliques C, C', into the following types.

Type 1: There is a diametral path in G, containing both one edge of C and one edge of C'.

Type 2: There is a diametral path for C, C', containing exactly one edge of $C \cup C'$, while any further diametral path for C, C' contains at most one edge of $C \cup C'$.

Type 3: $d(v, v') = diam(G)$, for all $v \in C, v' \in C'$.

Theorem 5.8.2 *[3]: Let G be a connected graph. Then*

(1): *G is of class 1 if and only if all pairs of diametral cliques are of type 1.*

(2): *G is of class 2 if and only if G contains a pair of type 2 diametral cliques, but no type 3 pair.*

(3): *G is of class 3 if and only if G contains a pair of type 3 diametral cliques.*

In spite of the above characterizations, the problem of classifying graphs according to their classes does not seem simple, as implied by the following theorem.

Theorem 5.8.3 *: Recognizing whether a graph is of class 1,2 or 3 is NP-hard.*

Proof. Theorem 12 states that recognizing clique-complete graphs is Co-NP-complete. In the proof, a graph G is constructed, such that G is clique-complete if and only if a given boolean equation is not satisfiable. It follows that $diam(G) = 2$ and $diam(K(G)) = 1$ or 2, according to whether G is clique-complete or not. A certificate for G not to be in class 1 is a pair of diametral cliques of type 2 or 3. Consequently, to recognize class 1 graphs is Co-NP-complete, while it is NP-hard to recognize class 2 graphs.

For class 3 graphs, we describe a transformation from the satisfiability problem. Let E be a boolean expression in conjunctive normal form, having clauses $L_i, 1 \leq i \leq p, p \geq 3$, each L_i consisting of q_i literals. Construct a graph G, as follows. There is a pair of vertices v_i, w_i, for each clause L_i. In addition, G contains a vertex u_{ij}, for each occurrance of a literal in L_i, $1 \leq i \leq p$ and $1 \leq j \leq q_i$. The following are the edges of G. For $i \neq k$, $(v_i, v_k), (v_i, w_k) \in E(G)$. For $i \neq k$ and all $1 \leq j \leq q_k$, $(w_i, u_{kj}) \in E(G)$.

Denote by ℓ_{ij} the j-th literal of L_i. Then $(u_{ij}, u_{kt}) \in E(G)$ precisely when $i \neq k$ and $\ell_{ij} \neq \overline{\ell_{kt}}$. There are no other edges in G. The construction of the graph is completed.

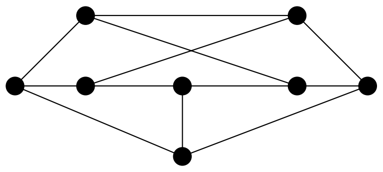
We show that G is a class 3 graph if and only if E is satisfiable. Suppose E is satisfiable and for each clause L_i, let $\ell_{i,j(i)}$ be the literal satisfying L_i. Then the vertices $u_{i,j(i)}, 1 \leq i \leq p$, form a complete C_1 of G. In fact, C_1 is a clique. On the other hand, the vertices $v_1, \ldots, v_p$ form a clique C_2 of G. It follows that C_1, C_2 constitute a pair of type 3 diametral cliques of G. By Theorem 19, G is a class 3 graph.

Conversely, by hypothesis, G contains a pair of type 3 diametral cliques C_1, C_2. It follows that $w_i \notin C_1 \cup C_2$, for $1 \leq i \leq p$, otherwise C_1 or C_2 is not maximal, or C_1, C_2 are not of type 3. Consequently, one of the cliques, say C_1 is formed solely by vertices u_{ij}, for each $1 \leq i \leq p$. The corresponding literals of E turn the boolean expression satisfiable, as required. Finally, a certificate for a class 3 graph is a pair of type 3 diametral cliques. Consequently, to recognize graphs of class 3 is NP-complete. □

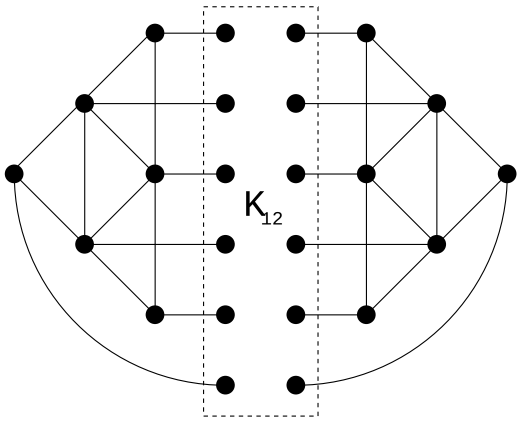
Further, we examine the diameters of iterated clique graphs. First, consider decreasing diameters. For a given integer $m > 0$, we look for a graph G, such that $diam(K^i(G)) < diam(K^{i-1}(G))$, $i = 1, \ldots, m$. Any convergent graph with equal index and diameter satisfies this condition. All chordal graphs fall into this category and therefore they are examples of diameter decreasing iterated clique graphs [5, 19]. On the other hand, self-clique graphs are trivial examples of diameter preserving iterated clique graphs.

Finally, consider iterated clique graphs with increasing diameters. The graphs G satisfying $diam(K(G)) = diam(G) + 1$ were characterized in Theorem 19 (iii). In [44], it has been asked whether there are graphs G, such that $diam(K^i(G)) = diam(G) + i$. In [44], itself, there is an example of a graph satisfying this condition for $i = 1$. Such a graph is that of Figure 10(a). In [3], there is an example for $i = 2$. Examples of graphs verifying the equality for $i = 2$, 3 and 4 were given in [64]. The graph of Figure 10(b) is that of $i = 2$. The question for arbitrary i has been answered positively in [9], where for each i, it has been described a family of graphs satisfying $diam(K^i(G)) = diam(G) + i$. A simpler family with this property has been reported in [66].

Another question mentioned in [45] concerns the existence or not of (divergent) graphs satisfying $\lim_{i\to\infty} diam(K^i(G)) = \infty$. This question has been answered in the affirmative in [50]. Furthermore, locally c_6 graphs also have this property [51].



(a)



(b)

Figure 5.10. Iterated clique graphs with increasing diameters

5.9 Remarks and Problems

We have summarized some of the results in the study of clique graphs, the intersection graphs of maximal cliques of a graph. There are other graph operators, closely related to clique graphs. The following can be mentioned, as examples. Intersection graphs of completes of sizes at most 2 of a graph (*middle graphs*) [1, 39, 81]; intersection graphs of all independent sets of G (*independence graphs*) [22, 82]; intersection graphs of all completes of G (*simplex graphs*) [80]; intersection graphs of all completes of size k and all cliques of size at most k ($\leq k -$ *clique graphs*) [69]; *edge-clique graphs* [2, 15, 16, 18]; among others. A comprehensive reference for graph operators is the book [70].

The following is a list of problems in clique graphs.

(1) Determine the complexity of recognizing clique graphs ([14, 71]).

(2) Let $\mathcal{G}$ be the set of all graphs and $K(\mathcal{G})$ be the set of the clique graphs of the graphs of $\mathcal{G}$. Denote $K^0(\mathcal{G}) = \mathcal{G}$ and $K^i(\mathcal{G}) = K(K^{i-1}(\mathcal{G}))$,

$i > 0$. Is it true that $K^i(\mathcal{G}) \neq K^{i-1}(\mathcal{G})$, for all i ? Clearly, the inequality holds for $i = 1$. And for $i = 2$ [36] ?

(3) Characterize intersection graphs of the chains of an order (clique graphs of comparability graphs).

(4) Characterize intersection graphs of antichains of an order (clique graphs of co-comparability graphs).

(5) Determine the complexity of recognizing divergent graphs.

(6) Characterize self-clique graphs.

(7) Determine if the period of a periodic clique-Helly graph is 1 or 2.

(8) Is there a graph G satisfying $\lim_{i\to\infty} diam(K^i(G)) = \infty$ and such that for every finite i, $diam(K^i(G)) = diam(G) + i$ [66] ?

(9) A graph is *clique irreducible* if each clique of it contains an edge which is not contained in any other clique. Characterize clique irreducible graphs [87].

(10) Two graphs G_1, G_2 are *clique-isomorphic* when $K(G_1)$ and $K(G_2)$ are isomorphic. Recognize if two given graphs are clique-isomorphic [29].

(11) A *clique-transversal* of a graph G is a subset of vertices intersecting every clique of G [27, 85]. A *clique-independent set* is a subset of pairwise disjoint cliques of G. Denote by $\tau_c(G)$ and $\alpha_c(G)$ the cardinalities of a minimum clique-transversal and maximum clique-independent set of G, respectively. Clearly, $\tau_c(G) \geq \alpha_c(G)$. For an arbitrary t, are there graphs G satisfying $t = \tau_c(G) - \alpha_c(G) = \theta(|V(G)|)$?

(12) A graph G is *clique-perfect* whenever $\tau_c(H) = \alpha_c(H)$, for every induced subgraph H of G, where τ_c and α_c are defined as above [17, 31]. Characterize clique perfect graphs.

(13) Prove or show a counter-example.
G is clique-perfect if and only G does not contain as induced subgraphs the following graphs: (i) c_{2k+1}, $k \geq 2$, (ii) $\overline{c_k}$, $k \geq 7$ and $k \neq 0$ *(mod 3)* and (iii) S_{2k+1}, $k \geq 1$, where S_t denotes a *t-sun*, that is, a graph consisting of a clique C and an independent set I, such that $|C| = |I| = t$, $C \cap I = \emptyset$, while the degree of a vertex of C is $t + 1$ and that of a vertex of I is 2, $t \geq 3$.

Acnowledgements: To Emerson M. Carmelo, Márcia R. Cerioli, Marisa Gutierrez and Cláudia L. Sales for their comments and suggestions.

References

[1] J. Akiyama, T.Hamada, and I. Yoshimura.On characterizations of the middle graph.*TRU Mathematics*, 11:35–39, 1975.

[2] M. O. Albertson and K. L. Collins.Duality and perfection for edges in cliques.*Journal of Combinatorial Theory B*, 36:298–309, 1984.

[3] R. Balakrishnan and P. Paulraja.Self-clique graphs and diameters of iterated clique graphs.*Utilitas Mathematica*, 29:263–268, 1986.

[4] E. Balas and C. S. Yu.On graphs with polynomially solvable maximum-weight clique problem.*Networks*, 19:247–253, 1989.

[5] H.-J. Bandelt and E. Prisner.Clique graphs and Helly graphs.*Journal of Combinatorial Theory B*, 51:34–45, 1991.

[6] C. Berge.*Hypergraphes.*Gauthier-Villars, Paris, 1987.

[7] A. Bondy, G. Durán, M. C. Lin, and J. L. Szwarcfiter.A sufficient condition for self-clique graphs (extended abstract).*Electronic Notes in Discrete Mathematics*, 2001.To appear.

[8] C. F. Bornstein and J. L. Szwarcfiter.On clique convergent graphs.*Graphs and Combinatorics*, 11:213–220, 1995.

[9] C. F. Bornstein and J. L. Szwarcfiter.Iterated clique graphs with increasing diameters.*Journal of Graph Theory*, 28:147–154, 1998.

[10] C. F. Bornstein and J. L. Szwarcfiter.A characterization of clique graphs of rooted path graphs.In Y. Alavi, D. R. Lick, and A. Schwenck, editors, *Proceedings of the 8th Quadriennial International Conference on Graph Theory, Algorithms, Combinatorics and Applications*, pages 117–122. Western Michigan University, New Issues Press, 1999.

[11] A. Brandstädt, V. D. Chepoi, and F. F. Dragan.Clique r-domination and clique r-packing problems on dually chordal graphs.*SIAM Journal on Discrete Mathematics*, 10:109–127, 1997.

[12] A. Brandstädt, V. D. Chepoi, F. F. Dragan, and V. I. Voloshin.Dually chordal graphs.*SIAM Journal on Discrete Mathematics*, 11:437–455, 1998.

[13] A. Brandstädt, V. D. Chepoi, and F. F.Dragan.The algorithmic use of hypertree structure and maximum neighbourhood orderings.*Discrete Applied Mathematics*, 82:43–77, 1998.

[14] A. Brandstädt, V. B. Le, and J. Spinrad.*Graph Classes: A Survey*, volume 3 of *SIAM Monographs on Discrete Mathematics and Applications.*SIAM, Philadelphia, 1999.

[15] M. R. Cerioli.*Grafos Clique de Arestas.*PhD thesis, Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brazil, 1999.

[16] M. R. Cerioli and J. L. Szwarcfiter.A characterization of edge clique graphs.*Ars Combinatoria*, 2001.To appear.

[17] G. J. Chang, M. Farber, and Z. Tuza.Algorithmic aspects of neighbourhood numbers.*SIAM Journal on Discrete Mathematics*, 6:24–29, 1991.

[18] G. Chartrand, S. F. Kapoor, T. A. McKee, and F. Saba.Edge-clique graphs.*Graphs and Combinatorics*, 7:253–264, 1991.

[19] B. L. Chen and K.-W. Lih.Diameters of iterated clique graphs of chordal graphs.*Journal of Graph Theory*, 14:391–396, 1990.

[20] G. L. Chia.On self-clique graphs with given clique sizes.*Discrete Mathematics*, 212:185–189, 2000.

[21] L. Chong-Keang and P. Yee-Hock.On graphs without multicliqual edges.*Journal of Graph Theory*, 5:443–451, 1981.

[22] E. J. Cockayne and S. T. Hedetnieme.Independence graphs.*Congressus Numerantium*, 10, 1974.

[23] C. L. Deng and C. K. Lim.A class of clique-closed graphs.*Discrete Mathematics*, 127:131–137, 1994.

[24] M. C. Dourado, F. Protti, and J. L. Szwarcfiter.The complexity of recognizing graphs with Helly defect one.In preparation.

[25] F. F. Dragan.*Centers of Graphs and the Helly Property*.PhD thesis, Moldava State University, Chisinău, Moldava, 1989.In russian.

[26] G. Durán and M. C. Lin.Clique graphs of Helly circular-arc graphs.*Ars Combinatoria*, 2001.To appear.

[27] P. Erdös, T. Gallai, and Z. Tuza.Covering the cliques of a graph with vertices.*Discrete Mathematics*, 108:279–289, 1992.

[28] F. Escalante.Über iterierte Clique-Graphen.*Abhandlungender Mathematischen Seminar der Universität Hamburg*, 39:59–68, 1973.

[29] F. Escalante and B. Toft.On clique-critical graphs.*Journal of Combinatorial Theory B*, 17:170–182, 1974.

[30] M. C. Golumbic.*Algorithmic Graph Theory and Perfect Graphs*.Academic Press, New York, 1980.

[31] V. Guruswami and C. P. Rangan.Algorithmic aspects of clique transversal and clique-independent sets.*Discrete Applied Mathematics*, 100:183–202, 2000.

[32] M. Gutierrez.Tree-clique graphs.In J. L. Szwarcfiter, editor, *Workshop Internacional de Combinatória*, pages 7–26, Rio de Janeiro, 1996. Universidade Federal do Rio de Janeiro.

[33] M. Gutierrez.Intersection graphs and clique application.*Graphs and Combinatorics*, 2001.To appear.

[34] M. Gutierrez and J. Meidanis.Algebraic theory for the clique operator.Manuscript.

[35] M. Gutierrez and J. Meidanis.Recognizing clique graphs of directed edge path graphs.Manuscript.

[36] M. Gutierrez and J. Meidanis.On the clique operator.*Lecture Notes in Computer Science*, 1380:261–272, 1998.Proceedings of the 3rd Latin American Conference on Theoretical Informatics.

[37] M. Gutierrez and L. Oubiña.Minimum proper interval graphs.*Discrete Mathematics*, 142:77–85, 1995.

[38] M. Gutierrez and R. Zucchello.Grafos ACI: Una generalización de los grafos de intervalos própios.Manuscript.

[39] T. Hamada and I. Yoshimura.Traversability and connectivity of the middle graph of a graph.*Discrete Mathematics*, 14:247–255, 1976.

[40] R. C. Hamelink.A partial characterization of clique graphs.*Journal of Combinatorial Theory*, 5:192–197, 1968.

[41] S. Hazan and V. Neumann-Lara.Fixed points of posets and clique graphs.*Order*, 13:219–225, 1996.

[42] S. T. Hedetniemi and P. J. Slater.Line graphs of triangleless graphs and iterated clique graphs.*Lecture Notes in Mathematics*, 303:139–147, 1972.

[43] B. Hedman.Clique graphs of time graphs.*Journal of Combinatorial Theory B*, 37:270–278, 1984.

[44] B. Hedman.Diameters of iterated clique graphs.*Hadronic Journal*, 9:273–276, 1986.

[45] B. Hedman.A polynomial algorithm for constructing the clique graph of a line graph.*Discrete Applied Mathematics*, 15:61–66, 1986.

[46] P. Hell.*Rétractions de Graphes*.PhD thesis, Université de Montreal, Montreal, Canada, 1972.

[47] M. Knor, L. Niepel, and L. Soltes.Centers in line graphs.*Math. Slovaca*, 43:11–20, 1993.

[48] F. Larrión and V. Neumann-Lara.On clique divergent graphs with linear growth.Manuscript.

[49] F. Larrión and V. Neumann-Lara.A family of clique divergent graphs with linear growth.*Graphs and Combinatorics*, 13:263–266, 1997.

[50] F. Larrión and V. Neumann-Lara.Clique divergent graphs with unbounded sequence of diameters.*Discrete Mathematics*, 197-198:491–501, 1999.

[51] F. Larrión and V. Neumann-Lara.Locally C_6 graphs are clique divergent.*Discrete Mathematics*, 2000.To appear.

[52] F. Larrión, V. Neumann-Lara, and M. A. Pizaña.Whitney triangulations, local girth and iterated clique graphs.Manuscript.

[53] F. Larrión, V. Neumann-Lara, and M. A. Pizaña.Clique divergent clockwork graphs and partial orders (extended abstract).*Electronic Notes in Discrete Mathematics*, 2001.To appear.

[54] C. K. Lim.A result on iterated clique graphs.*Journal of the Australian Mathematical Society A*, 32:289–294, 1982.

[55] C. L. Lucchesi, C. P. Mello, and J. L. Szwarcfiter.On clique-complete graphs.*Discrete Mathematics*, 183:247–254, 1998.

[56] T. A. McKee and F. R. McMorris.*Topics in Intersection Graph Theory*, volume 2 of *Monographs on Discrete Mathematics and Applications*.SIAM, Philadelphia, 1999.

[57] C. P. Mello.*Sobre Grafos Clique-Completos*.PhD thesis, Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brazil, 1992.

[58] V. Neumann-Lara.On clique-divergent graphs.In *Problèmes Combinatoires et Théorie des Graphes*, pages 313–315, Orsay, France, 1978. Colloques Internationaux C.N.R.S. 260.

[59] V. Neumann-Lara.Clique divergence in graphs.In *Algebraic Methods in Graph Theory*, volume 25, pages 563–569. Colloquia Mathematica Societatis János Bolyai, Szeged, Hungary, 1981.

[60] V. Neumann-Lara.Clique divergence in graphs - some variations.Technical report, Instituto de Matematicas, Universidad Nacional Autonoma de Mexico, 1991.

[61] L. Niepel, M. Knor, and L. Soltes.Distances in iterated line graphs.*Ars Combinatoria*, 43:193–202, 1996.

[62] R. Nowakowski and I. Rival.The smallest graph variety containing all paths.*Discrete Mathematics*, 43:223–234, 1983.

[63] R. Nowakowski and P. Winkler.Vertex-to-vertex porsuit of a graph.*Discrete Mathematics*, 43:235–239, 1983.

[64] C. Peyrat, D. F. Rall, and P. J. Slater.On iterated clique graphs with increasing diameters.*Journal of Graph Theory*, 10:167–171, 1986.

[65] M. A. Pizaña.The icosahedron is clique-divergent.Manuscript.

[66] M. A. Pizaña.Distances and diameters on iterated clique graphs (extended abstract).*Electronic Notes in Discrete Mathematics*, 2001.To appear.

[67] E. Prisner.Convergence of iterated clique graphs.*Discrete Mathematics*, 103:199–207, 1992.

[68] E. Prisner.Hereditary clique-Helly graphs.*Journal of Combinatorial Mathematics and Combinatorial Computing*, 14:216–220, 1993.

[69] E. Prisner.A common generalization of line graphs and clique graphs.*Journal of Graph Theory*, 18:301–313, 1994.

[70] E. Prisner.*Graph Dynamics*.Pitman Research Notes in Mathematics 338, Longman, 1995.

[71] E. Prisner.Graphs with few cliques.In Y. Alavi and A. Schwenk, editors, *Proceedings of the 7th Quadrennial International Conference on Graph Theory, Algorithms, Combinatorics ans Applications*, pages 945–956. Western Michigam University, John Wiley and Sons, Inc., 1995.

[72] E. Prisner and J. L. Szwarcfiter.Recognizing clique graphs of directed and rooted path graphs.*Discrete Applied Mathematics*, 94:321–328, 1999.

[73] F. Protti.*Classes de Grafos Clique Inversos*.PhD thesis, Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brazil, 1998.

[74] F. Protti and J. L. Szwarcfiter.Clique-inverse graphs of K_3-free and K_4-free graphs.*Journal of Graph Theory*, 35:257–272, 2000.

[75] F. Protti and J. L. Szwarcfiter.On clique graphs of linear size.*Congressus Numerantium*, 2000.To appear.

[76] F. Protti and J. L. Szwarcfiter.Clique-inverse graphs of bipartite graphs.*Journal of Combinatorial Mathematics and Combinatorial Computing*, 2001.To appear.

[77] A. Quilliot.*Homomorphismes, points fixes, retractions et jeux des poursuite dans les graphes, les ensembles ordonnés et les espaces metriques*.PhD thesis, Université de Paris, Paris, France, 1983.

[78] A. Quilliot.On the Helly property working as a compactness criterion on graphs.*Journal of Combinatorial Theory A*, 40:186–193, 1985.

[79] F. S. Roberts and J. H. Spencer.A characterization of clique graphs.*Journal of Combinatorial Theory B*, 10:102–108, 1971.

[80] E. Sampathkumar and H. B. Walikar.On the complete graph of a graph.*Abstract Graph Theory Newsletter*, 3, 1978.

[81] M. Skowronska and M. M. Syslo.An algorithm to recognize a middle graph.*Discrete Applied Mathematics*, 7:201–208, 1984.

[82] P. J. Slater.Irreducible point independence numbers and independence graphs.*Congressus Numerantium*, 10:647–660, 1974.

[83] J. L. Szwarcfiter.Recognizing clique-Helly graphs.*Ars Combinatoria*, 45:29–32, 1997.

[84] J. L. Szwarcfiter and C. F. Bornstein.Clique graphs of chordal and path graphs.*SIAM Journal on Discrete Mathematics*, 7:331–336, 1994.

[85] Z. Tuza.Covering all cliques of a graph.*Discrete Mathematics*, 86:117–126, 1990.

[86] W. D. Wallis and J. Wu.Squares, clique graphs and chordality.*Journal of Graph Theory*, 20:37–45, 1995.

[87] W. D. Wallis and G. H. Zhang.On maximal clique irreducible graphs.*Journal of Combinatorial Mathematics and Combinatorial Computing*, 8:187–193, 1990.

Appendix

Below are definitions of most of the classes mentioned in the text. An arbitrary graph in each class is denoted by G.

BIPARTITE: There is a partition of $V(G)$ into at most two independent sets.

BLOCK: G is the intersection graph of the blocks (maximal biconnected components) of a graph.

3-COLOURABLE: There is a partition of $V(G)$ into at most three independent sets.

CHORDAL: G is the intersection graph of subtrees of a tree.

CHORDAL BIPARTITE: G is bipartite and for every cycle c of G of length ≥ 6, there is an edge between two non adjacent vertices in c.

CIRCULAR CLIQUE: G admits a circular ordering α of its vertices and a Helly edge cover, formed by completes, each of them composed by vertices which are consecutive in α.

CLIQUE: G is the intersection graph of the cliques of a graph.

CLIQUE-COMPLETE: Every pair of cliques of G intersect.

CLIQUE-HELLY: The cliques of G satisfy the Helly property.

COMPARABILITY: The edges of G can be transitively oriented.

COMPLETE: All pairs of distinct vertices of G are adjacent.

CONVERGENT: $K^i(G)$ is the one-vertex graph, for some finite i.

DE: G is the edge intersection graph of paths of a directed tree, where two paths are considered as intersecting, when they share a common edge.

DIAMOND FREE: Every edge of G belongs to exactly one clique.

DISK HELLY: The disks of G satisfy the Helly property, where a disk is a subset of vertices lying at a distance $\leq i$, from some vertex of G.

DISMANTABLE: G is either the one-vertex graph or it has a dominated vertex v, such that $G - v$ is dismantable.

DUALLY CHORDAL: G admits a spanning tree T, such that each complete of G induces a (connected) subtree in T.

DUALLY DV: G admits a spanning directed tree T, such that each complete of G induces a (directed) path in T.

DUALLY RDV: G admits a spanning directed rooted tree T, such that each complete of G induces a (directed rooted) path in T.

DV: G is the intersection graph of paths of a directed tree.

EDGE-CLIQUE: The vertices of G correspond to the edges of some graph H, with two vertices adjacent in G whenever their corresponding edges in H belong to a same clique.

$\mathcal{H}_1$: G has a pair of cliques C_1, C_2, for each pair of vertices $v_1, v_2 \in V(G)$, satisfying C_1 contains v_1 and not v_2, while C_2 contains v_2 and not v_1.

HELLY CIRCULAR ARC: G is the intersection graph of arcs of a circle, satisfying the Helly property.

HELLY HEREDITARY: Every induced subgraph of G is clique-Helly.

INTERVAL: G is the intersection graph of intervals of a real line.

LINE: G is the intersection graph of the pairs of adjacent vertices of a graph.

MINIMAL PROPER INTERVAL: G is the intersection graph of proper intervals of a real line, whose number of distinct extreme points of the intervals is $2|\mathcal{C}(G)| - |\mathcal{C}(K(G))|$, where $\mathcal{C}(G)$ is the set of cliques of G.

PERIODIC: G satisfies $K^i(G) = G$, for some finite i.

PROPER INTERVAL: G is the intersection graph of proper intervals of a real line.

PTOLOMAIC: Every four vertices $u, v, w, t \in V(G)$ satisfy $d(u,v).d(w,t) \leq d(u,w).d(v,t) + d(u,t).d(v,w)$.

RDV: G is the intersection graph of paths of a directed rooted tree.

SELF-CLIQUE: $G = K(G)$.

SPLIT: G and $\overline{G}$ are chordal.

STAR: G contains a universal vertex.

STRONGLY CHORDAL: G is chordal and contains no induced t-suns, $t \geq 3$.

TREE: G is connected and acyclic.

UV: G is the intersection graph of paths of a tree.

6

Semidefinite Programs and Combinatorial Optimization

L. Lovász

6.1 Introduction

Linear programming has been one of the most fundamental and successful tools in optimization and discrete mathematics. Its applications include exact and approximation algorithms, as well as structural results and estimates. The key point is that linear programs are very efficiently solvable, and have a powerful duality theory.

A fundamental method in combinatorial optimization is to write a combinatorial optimization problem as a linear program with integer variables. There are usually many ways to do so; ideally, one tries to get the "tightest" description (in which the feasible set of the linear program is the convex hull of integer solutions); but this is often too complicated to determine, and we have to work with a "relaxed" description. We then forget the integrality constraints, thereby obtaining a *linear relaxation*, a linear program which can be solved efficiently; and then trying to restore the integrality of the variables by some kind of rounding (which is usually heuristic, and hence only gives approximately optimal integral solutions). In those particularly well-structured cases when we have the tightest description, the basic optimal solutions to the linear program are automatically integral, so it also solves the combinatorial optimization problem right away.

Linear programs are special cases of convex programs; *semidefinite programs* are more general but still convex programs, to which many of the useful properties of linear programs extend. Recently, semidefinite programming arose as a generalization of linear programming with substantial novel applications. Again, it can be used both in proofs and in the design of exact and approximation algorithms. It turns out that various combinatorial optimization problems have semidefinite (rather than linear) relaxations which are still efficiently computable, but approximate the optimum much better. This fact has lead to a real breakthrough in approximation algorithms.

In these notes we survey semidefinite optimization mainly as a relaxation of discrete optimization problems. We start with two examples, a proof and an approximation algorithm, where semidefinite optimization plays a important role. Still among the preliminaries, we survey some areas which play a role in semidefinite optimization: linear algebra (in particular, positive semidefinite matrices), linear programming (duality and algorithms), and polyhedral combinatorics (which we illustrate on the example of the stable set polytope).

After introducing semidefinite programs and discussing some of their basic properties, we show that semidefinite programs arise in a variety of ways: as certain geometric extremal problems, as relaxations (stronger than linear relaxations) of combinatorial optimization problems, in optimizing eigenvalue bounds in graph theory, as stability problems in engineering.

Next we show through examples from graph theory, number theory, and logic how semidefinite optimization can be used in proofs as well as in the design of approximation algorithms.

In Chapter 6.7 we try to put the combinatorial applications of semidefinite optimization in a broader perspective: they can be viewed as procedures to strengthen the descriptions of combinatorial optimization problems as integer linear programs. It turns out that such procedures can be formalized, and in some cases (like the stable set polytope, our favorite example) they lead to efficient ways of generating the tight linear descriptions for most cases when this description is known at all.

There are many unsolved problems in this area; indeed, progress has been quite slow (but steady) due to the difficulty of some of those. Several of these roadblocks are described in Chapter 6.8.

For more comprehensive studies of issues concerning semidefinite optimization, see [98].

6.1.1 Shannon capacity

Consider a noisy channel through which we are sending messages over a finite alphabet V. The noise may blur some letters so that certain pairs can be confounded. We want to select as many words of length k as possible so that no two can possibly be confounded. As we shall see, the number of words we can select grows as Θ^k for some $\Theta \geq 1$, which is called the *Shannon zero-error capacity* of the channel.

In terms of graphs, we can model the problem as follows. We consider V as the set of nodes of a graph, and connect two of them by an edge if they can be confounded. This way we obtain a graph $G = (V, E)$. We denote by $\alpha(G)$ the maximum number of independent points (the maximum size of a stable set) in the graph G. If $k = 1$, then the maximum number of non-confoundable messages is $\alpha(G)$.

To describe longer messages, we define the *strong product* $G \cdot H$ of two graphs $G = (V, E)$ and $H = (W, F)$ as the graph with $V(G \cdot H) = V \times W$, with $(i, u)(j, v) \in E(G \cdot H)$ iff $ij \in E$ and $uv \in F$, or $ij \in E$ and $u = v$, or $i = j$ and $uv \in F$. The product of k copies of G is denoted by G^k. Thus $\alpha(G^k)$ is the maximum number of words of length k, composed of elements of V, so that for every two words there is at least one i $(1 \le i \le k)$ such that the i-th letters are different and non-adjacent in G, i.e., non-confoundable.

The *Shannon capacity* of a graph G is the value $\Theta(G) = \lim_{k\to\infty} \alpha(G^k)^{1/k}$ (it is not hard to see that the limit exists). It is not known whether $\Theta(G)$ can be computed for all graphs by any algorithm (polynomial or not), although there are several special classes of graphs for which this is not hard. For example, if G is a 4-cycle with nodes (a, b, c, d), then for every $k \ge 1$, all words of length k consisting of a and c only can be used, and so $\alpha(C_4^k) \ge 2^k$. On the other hand, if we use a word, then all the 2^k words obtained from it by replacing a and b by each other, as well as c and d by each other, are excluded. Hence $\alpha(C_4^k) \le 4^k/2^k = 2^k$, and $\Theta(C_4) = 2$. More generally, we have $\alpha(G^k) \ge \alpha(G)^k$ for any graph G and so $\Theta((G) \ge \alpha(G)$. If we can also bound $\Theta(G)$ from above by $\alpha(G)$, then we have determined it exactly (this method works for all perfect graphs; cf section 6.2.3).

The smallest graph for which $\Theta(G)$ cannot be computed by such elementary means is the pentagon C_5. If we set $V(C_5) = \{0, 1, 2, 3, 4\}$ with $E(C_5) = \{01, 12, 23, 34, 40\}$, then C_5^2 contains the stable set $\{(0, 0), (1, 2), (2, 4), (3, 1), (4, 3)\}$. So $\alpha(C_5^{2k}) \ge \alpha(C_5^2)^k \ge 5^k$, and hence $\Theta(C_5) \ge \sqrt{5}$.

We show that equality holds here [64]. Consider an "umbrella" in $\mathbb{R}^3$ with the unit vector e_1 as its handle, and 5 ribs of unit length (Figure 6.1). Open it up to the point when non-consecutive ribs are orthogonal (i.e., form an angle of 90°). This way we get 5 unit vectors u_0, u_1, u_2, u_3, u_4, assigned to the nodes of C_5 so that each u_i forms the same angle with e_1 and any two non-adjacent nodes are labeled with orthogonal vectors. (Elementary trigonometry gives $e_1^\top u_i = 5^{-1/4}$).

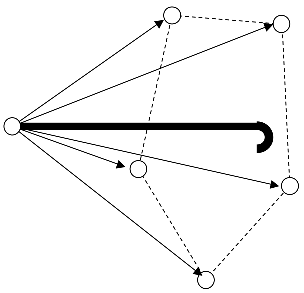


Figure 6.1. An orthogonal representation of C_5.

It turns out that we can obtain a similar labeling of the nodes of C_5^k by unit vectors $v_i \in \mathbb{R}^{3k}$, so that any two non-adjacent nodes are labeled with orthogonal vectors. Moreover, $e_1^\top v_i = 5^{-k/4}$ for every $i \in V(C_5^k)$. Such a labeling is obtained by taking tensor products. The *tensor product* of two vectors $(u_1, \dots, u_n) \in \mathbb{R}^n$ and $(v_1, \dots, v_m) \in \mathbb{R}^m$ is the vector

$$u \circ v = (u_1v_1, \dots, u_1v_m, u_2v_1, \dots, u_2v_m, \dots, u_nv_1, \dots, u_nv_m) \in \mathbb{R}^{nm}.$$

The tensor product of several vectors is defined similarly. The property one uses in verifying the properties claimed above is that if $u, x \in \mathbb{R}^n$ and $v, y \in \mathbb{R}^m$, then

$$(u \circ v)^\mathsf{T}(x \circ y) = (u^\mathsf{T}x)(v^\mathsf{T}y).$$

If S is any stable set in C_5^k, then $\{v_i : i \in S\}$ is a set of mutually orthogonal unit vectors, and hence

$$\sum_{i \in S}(e_1^\mathsf{T}v_i)^2 \le |e_1|^2 = 1$$

(if the v_i formed a basis then this inequality would be an equality).

On the other hand, each term on the left hand side is $5^{-1/4}$, hence the left hand side is $|S|5^{-k/2}$, and so $|S| \le 5^{k/2}$. Thus $\alpha(C_5^k) \le 5^{k/2}$ and $\Theta(C_5) = \sqrt{5}$.

This method extends to any graph $G = (V, E)$ in place of C_5: all we have to do is to assign unit vectors to the nodes so that non-adjacent nodes correspond to orthogonal vectors (such an assignment will be called an *orthogonal representation*). If the first coordinate of each of these vectors is s, then the Shannon capacity of the graph is at most $1/s^2$. The best bound that can be achieved by this method will be denoted by $\vartheta(G)$.

But how to construct an optimum (or even good) orthogonal representation? Somewhat surprisingly, the optimum representation can be computed in polynomial time using semidefinite optimization. Furthermore, it has many nice properties, most of which are derived using semidefinite duality and other fundamental properties of semidefinite programs (section 6.3.1), as we shall see in section 6.5.1.

6.1.2 Maximum cuts

A *cut* in a graph $G = (V, E)$ is the set of edges connecting a set $S \subseteq V$ to $V \setminus S$, where $\emptyset \subset S \subset V$. The *Max Cut Problem* is to find a cut with maximum cardinality. We denote by MC this maximum.

(More generally, we can be given a weighting $w : V \to \mathbb{R}_+$, and we could be looking for a cut with maximum total weight. Most other problems discussed below, like the stable set problem, have such weighted versions. To keep things simple, however, we usually restrict our discussions to the unweighted case.)

The Max Cut Problem is NP-hard; one natural approach is to find an approximately maximum cut. Formulated differently, Erdős in 1967 described the following simple heuristic: for an arbitrary ordering $(v_1, \dots, v_n)$ of the nodes, we color $v_1, v_2, \dots, v_n$ successively red or blue. For each i, v_i is colored blue iff the number of edges connecting v_i to blue nodes among $v_1, \dots, v_{i-1}$ is less than the number of edges connecting v_i to red nodes in this set. Then the cut formed by the edges between red and blue nodes contains at least half of all edges. In particular, we get a cut that is at least half as large as the maximum cut.

There is an even easier randomized algorithm to achieve this approximation, at least in expected value. Let us 2-color the nodes of G randomly, so that each node is colored red or blue independently, with probability 1/2. Then the probability that an edge belongs to the cut between red and blue is 1/2, and expected number of edges in this cut is $|E|/2$.

Both of these algorithms show that the maximum cut can be approximated from below in polynomial time with a multiplicative error of at most 1/2. Can we do better? The following strong negative result [10, 19, 45] shows that we cannot get arbitrarily close to the optimum:

Proposition 6.1.1 *It is NP-hard to find a cut with more than* $(16/17)\mathcal{MC} \approx .94\mathcal{MC}$ *edges.*

But we can do better than 1/2, as the following seminal result of Goemans and Williamson [37, 38] shows:

Theorem 6.1.2 *One can find in polynomial time a cut with at least* $.878\mathcal{MC}$ *edges.*

The algorithm of Goemans and Williamson makes use of the following geometric construction. We want to find an embedding $i \mapsto u_i$ $(i \in V)$ of the nodes of the graph in the unit sphere in $\mathbb{R}^d$ so that the following "energy" is minimized:

$$\mathcal{E} = -\sum_{ij\in E} \frac{1}{4}(u_i - u_j)^2 = -\sum_{ij\in E} \frac{1 - u_i^{\mathsf{T}} u_j}{2}.$$

(Note the negative sign: this means that the "force" between adjacent nodes is repulsive, and grows linearly with the distance.)

If we work in $\mathbb{R}^1$, then the problem is equivalent to MAX CUT: each node is represented by either 1 or -1, and the edges between differently labeled nodes contribute -1 to the energy, the other edges contribute 0. Hence the negative of the minimum energy $\mathcal{E}$ is an upper bound on the maximum size $\mathcal{MC}$ of a cut.

Unfortunately, the argument above also implies that for $d = 1$, the optimal embedding is NP-hard to find. While I am not aware of a proof of this, it is probably NP-hard for $d = 2$ and more generally, for any fixed d. The surprising fact is that for $d = n$, such an embedding can be found using semidefinite optimization (cf. section 6.4.1).

So $-\mathcal{E}$ is a polynomial time computable upper bound on the size of the maximum cut. How good is this bound? And how to construct an approximately optimum cut from this representation? Here is the simple but powerful trick: *take a random hyperplane H through the origin in* $\mathbb{R}^n$ (Figure 6.2). The partition of R^d given by H yields a cut in our graph. Since the construction pushes adjacent points apart, one expects that the random cut will intersect many edges.

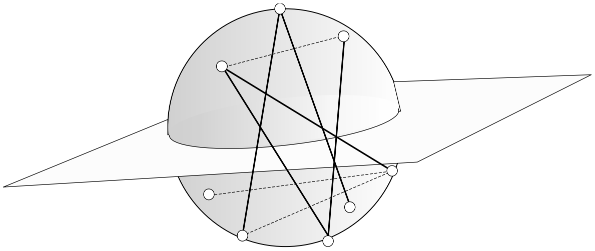


Figure 6.2. A cut in the graph given by a random hyperplane

To be more precise, let $ij \in E$ and let $u_i, u_j \in S^{n-1}$ be the corresponding vectors in the embedding constructed above. It is easy to see that the probability that a random hyperplane H through 0 separates u_i and u_j is α_{ij}/π, where $\alpha_{ij} = \arccos u_i^\mathsf{T} u_j$ is the angle between u_i and u_j. It is not difficult to verify that if $-1 \le t \le 1$, then $\arccos t \ge 1.38005(1-t)$. Thus the expected number of edges intersected by H is

$$\sum_{ij\in E} \frac{\arccos u_i^\mathsf{T} u_j}{\pi} \ge \sum_{ij\in E} 1.38005\frac{1-u_i^\mathsf{T} u_j}{\pi} = \frac{1.38005}{\pi} 2(-\mathcal{E}) \ge .878\mathcal{MC}.$$

(One objection to the above algorithm could be that it uses random numbers. In fact, the algorithm can be *derandomized* by well established but non-trivial techniques. We do not consider this issue in these notes; see e.g. [5], Chapter 15 for a survey of derandomization methods.)

6.2 Preliminaries

We collect some of the basic results from linear programming, linear algebra, and polyhedral combinatorics that we will use. While this is all textbook material, it will be convenient to have this collection of results for the purposes of notation, reference and comparison. [88] is a reference for linear algebra, and a [79], for linear programming.

6.2.1 Linear algebra

As the title of these lecture notes suggests, we'll be concerned with semidefinite matrices; to get to these, we start with a review of eigenvalues, and in particular eigenvalues of symmetric matrices.

Let A be an $n\times n$ real matrix. An *eigenvector* of A is a vector such that Ax is parallel to x; in other words, $Ax = \lambda x$ for some real or complex number λ. This number λ is called the *eigenvalue* of A belonging to eigenvector v. Clearly λ is an eigenvalue iff the matrix $A - \lambda I$ is singular, equivalently, iff $det(A - \lambda I) = 0$. This is an algebraic equation of degree n for λ, and hence has n roots (with multiplicity).

The *trace* of the (square) matrix $A = (A_{ij})$ is defined as

$$\mathrm{tr}(A) = \sum_{i=1}^{n} A_{ii}.$$

The trace of A is the sum of the eigenvalues of A, each taken with the same multiplicity as it occurs among the roots of the equation $det(A - \lambda I) = 0$.

If the matrix A is symmetric, then its eigenvalues and eigenvectors are particularly well behaved. All the eigenvalues are real. Furthermore, there is an orthogonal basis $v_1, \ldots, v_n$ of the space consisting of eigenvectors of A, so that the corresponding eigenvalues $\lambda_1, \ldots, \lambda_n$ are precisely the roots of $det(A - \lambda I) = 0$. We may assume that $|v_1| = \ldots = |v_n| = 1$; then A can be written as

$$A = \sum_{i=1}^{n} \lambda_i v_i v_i^{\mathsf{T}}.$$

Another way of saying this is that every symmetric matrix can be written as $U^{\mathsf{T}} D U$, where U is an orthogonal matrix and D is a diagonal matrix. The eigenvalues of A are just the diagonal entries of D.

To state a further important property of eigenvalues of symmetric matrices, we need the following definition. A *symmetric minor* of A is a submatrix B obtained by deleting some rows and the *corresponding* columns.

Theorem 6.2.1 (Interlacing eigenvalues) *Let A be an $n\times n$ symmetric matrix with eigenvalues $\lambda_1 \geq \ldots \geq \lambda_n$. Let B be an $(n-k)\times(n-k)$ symmetric minor of A with eigenvalues $\mu_1 \geq \ldots \geq \mu_{n-k}$. Then*

$$\lambda_i \leq \mu_i \leq \lambda_{i+k}.$$

Now we come to the definition that is crucial for our lectures. A symmetric $n\times n$ matrix A is called *positive semidefinite*, if all of its eigenvalues are nonnegative. This property is denoted by $A \succeq 0$. The matrix is *positive definite*, if all of its eigenvalues are positive.

There are many equivalent ways of defining positive semidefinite matrices, some of which are summarized in the Proposition below.

Proposition 6.2.2 *For a real symmetric $n\times n$ matrix A, the following are equivalent:*

(i) *A is positive semidefinite;*

(ii) *the quadratic form $x^T Ax$ is nonnegative for every $x \in \mathbb{R}^n$;*

(iii) *A can be written as the Gram matrix of n vectors $u_1, ..., u_n \in \mathbb{R}^m$ for some m; this means that $a_{ij} = u_i^\mathsf{T} u_j$. Equivalently, $A = U^\mathsf{T} U$ for some matrix U;*

(iv) *A is a nonnegative linear combination of matrices of the type xx^T;*

(v) *The determinant of every symmetric minor of A is nonnegative.*

Let me add some comments. The least m for which a representation as in (iii) is possible is equal to the rank of A. It follows e.g. from (ii) that the diagonal entries of any positive semidefinite matrix are nonnegative, and it is not hard to work out the case of equality: all entries in a row or column with a 0 diagonal entry are 0 as well. In particular, the trace of a positive semidefinite matrix A is nonnegative, and $\mathrm{tr}(A) = 0$ if and only if $A = 0$.

The sum of two positive semidefinite matrices is again positive semidefinite (this follows e.g. from (ii) again). The simplest positive semidefinite matrices are of the form aa^T for some vector a (by (ii): we have $x^\mathsf{T}(aa^\mathsf{T})x = (a^\mathsf{T}x)^2 \geq 0$ for every vector x). These matrices are precisely the positive semidefinite matrices of rank 1. Property (iv) above shows that every positive semidefinite matrix can be written as the sum of rank-1 positive semidefinite matrices.

The product of two positive semidefinite matrices A and B is not even symmetric in general (and so it is not positive semidefinite); but the following can still be claimed about the product:

Proposition 6.2.3 *If A and B are positive semidefinite matrices, then* $\mathrm{tr}(AB) \geq 0$, *and equality holds iff $AB = 0$.*

Property (v) provides a way to check whether a given matrix is positive semidefinite. This works well for small matrices, but it becomes inefficient very soon, since there are many symmetric minors to check. An efficient method to test if a symmetric matrix A is positive semidefinite is the following algorithm. Carry out 2-sided Gaussian elimination on A, pivoting always on diagonal entries ("2-sided" means that we eliminate all entries in both the row and the column of the pivot element).

If you ever find a negative diagonal entry, or a 0 diagonal entry whose row contains a non-zero, stop: the matrix is not positive semidefinite. If you obtain an all-zero matrix (or eliminate the whole matrix), stop: the matrix is positive semidefinite.

If this simple algorithm finds that A is not positive semidefinite, it also provides a certificate in the form of a vector v with $v^\mathsf{T}Av < 0$. Assume that the i-th diagonal entry of the matrix $A^{(k)}$ after k steps is negative. Write $A^{(k)} = E_k^\mathsf{T} \dots E_1^\mathsf{T} A E_1 \dots E_k$, where E_i are elementary matrices. Then we can take the vector $v = E_1 \dots E_k e_i$. The case when there is a 0 diagonal entry whose row contains a non-zero is similar.

It will be important to think of $n \times n$ matrices as vectors with n^2 coordinates. In this space, the usual inner product is written as $A \cdot B$. This should not be confused with the matrix product AB. However, we can express the inner product of two $n \times n$ matrices A and B as follows:

$$A \cdot B = \sum_{i=1}^{n} \sum_{j=1}^{n} A_{ij} B_{ij} = \mathrm{tr}(A^{\mathsf{T}} B).$$

Positive semidefinite matrices have some important properties in terms of the geometry of this space. To state these, we need two definitions. A *convex cone* in $\mathbb{R}^n$ is a set of vectors which along with any vector, also contains any positive scalar multiple of it, and along with any two vectors, also contains their sum. Any system of homogeneous linear inequalities

$$a_1^{\mathsf{T}} x \geq 0, \quad \dots \quad a_m^{\mathsf{T}} x \geq 0$$

defines a convex cone; convex cones defined by such (finite) systems are called *polyhedral.*

For every convex cone C, we can form its *polar cone* C^*, defined by

$$C^* = \{x \in \mathbb{R}^n : x^{\mathsf{T}} y \geq 0 \ \forall y \in C\}.$$

This is again a convex cone. If C is closed (in the topological sense), then we have $(C^*)^* = C$.

The fact that the sum of two such matrices is again positive semidefinite (together with the trivial fact that every positive scalar multiple of a positive semidefinite matrix is positive semidefinite), translates into the geometric statement that *the set of all positive semidefinite matrices forms a convex closed cone* P_n *in* $\mathbb{R}^{n \times n}$ *with vertex* 0. This cone P_n is important, but its structure is quite non-trivial. In particular, it is non-polyhedral for $n \geq 2$; for $n = 2$ it is a nice rotational cone (Figure 6.3; the fourth coordinate x_{21}, which is always equal to x_{12} by symmetry, is suppressed). For $n \geq 3$ the situation becomes more complicated, because P_n is neither polyhedral nor smooth: any matrix of rank less than $n - 1$ is on the boundary, but the boundary is not differentiable at that point.

The polar cone of $\mathcal{P}$ is itself; in other words,

Proposition 6.2.4 *A matrix A is positive semidefinite iff $A \cdot B \geq 0$ for every positive semidefinite matrix B.*

We conclude this little overview with a further basic fact about nonnegative matrices.

Theorem 6.2.5 (Perron-Frobenius) *If an $n \times n$ matrix has nonnegative entries then it has a nonnegative real eigenvalue λ which has maximum absolute value among all eigenvalues. This eigenvalue λ has a nonnegative real eigenvector. If, in addition, the matrix has no block-triangular decomposition (i.e., it does not contain a $k \times (n-k)$ block of* 0*-s disjoint from the diagonal), then λ has multiplicity* 1 *and the corresponding eigenvector is positive.*

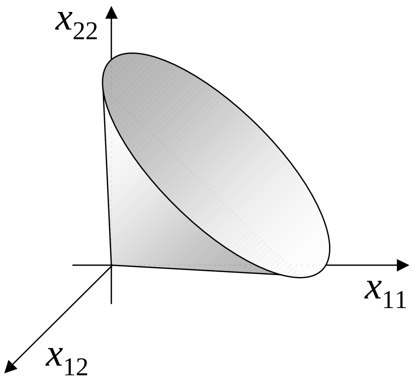


Figure 6.3. The semidefinite cone for $n = 2$.

6.2.2 Linear programming

Linear programming is closely related to (in a sense the same as) the study of systems of linear inequalities. At the roots of this theory is the following basic lemma.

Lemma 6.2.6 (Farkas Lemma) *A system of linear inequalities $a_1^T x \le b_1, \ldots, a_m^T x \le b_m$ has no solution iff there exist $\lambda_1, \ldots, \lambda_m \ge 0$ such that $\sum_i \lambda_i a_i = 0$ and $\sum_i \lambda_i b_i = -1$.*

Let us make a remark about the computational complexity aspect of this. The solvability of a system of linear inequalities is in NP ("just show the solution"; to be precise, one has to argue that there is a rational solution with small enough numerators and denominators so that it can be exhibited in space polynomial in the input size; but this can be done). One consequence of the Farkas Lemma (among many) is that this problem is also in co-NP ("just show the λ's").

A closely related statement is the following:

Lemma 6.2.7 (Farkas Lemma, inference version) *Let $a_1, \ldots, a_m, c \in \mathbb{R}^n$ and $b_1, \ldots, b_m, d \in \mathbb{R}$. Assume that the system $a_1^T x \le b_1, \ldots, a_m^T x \le b_m$ has a solution. Then $c^T x \le d$ for all solutions of $a_1^T x \le b_1, \ldots, a_m^T x \le b_m$ iff there exist $\lambda_1, \ldots, \lambda_m \ge 0$ such that $c = \sum_i \lambda_i a_i$ and $d \ge \sum_i \lambda_i b_i$.*

This again can be put into a general context: there is a semantical notion of a linear inequality being a consequence of others (it holds whenever the others do), and a syntactical (it is a linear combination of the others with non-negative coefficients). The lemma asserts that these two are equivalent. We'll see that e.g. for quadratic inequalities, the situation is more complicated.

Now we turn to linear programming. A typical linear program has the following form.

$$\begin{array}{lrcl} \text{maximize} & c^\mathsf{T} x & & \\ \text{subject to} & a_1^\mathsf{T} x & \le & b_1, \\ & & \vdots & \\ & a_m^\mathsf{T} x & \le & b_m, \end{array} \tag{6.1}$$

where $a_1, \dots, a_m$ are given vectors in $\mathbb{R}^n$, $b_1, \dots, b_m$ are real numbers, and $x = (x_1, \dots, x_n)$ is a vector of n unknowns. These inequalities can be summed up in matrix form as $Ax \le b$, where A is a matrix with m rows and m columns and $b \in \mathbb{R}^m$.

It is very fruitful to think of linear programs geometrically. The solution of the constraint system $Ax \le b$ (also called *feasible solutions*) form a convex polyhedron P in $\mathbb{R}^n$. For the following discussion, let us assume that P is bounded and has an internal point. Then each facet ($(n-1)$-dimensional faces) of P corresponds to one of the inequalities $a_i^\mathsf{T} x \le b_i$ (there may be other inequalities in the system, but those are redundant). The *objective function* $c^\mathsf{T} x$ can be visualized as a family of parallel hyperplanes; to find its maximum over P means to translate this hyperplane in the direction of the vector c as far as possible so that it still intersects P. If P is bounded, then these "ultimate" common points will form a face (a vertex, an edge, or higher dimensional face) P, and there will be at least one vertex among them (see Figure 6.4).

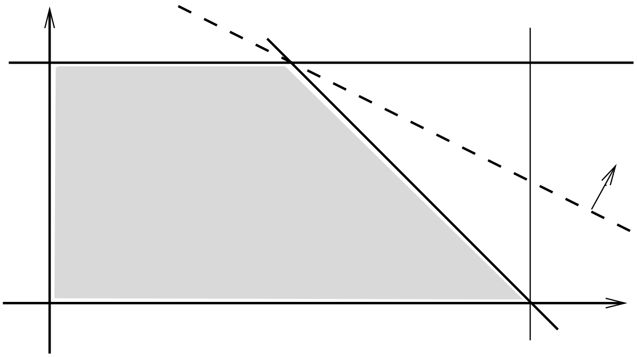


Figure 6.4. The feasible domain and optimum solution of the linear program: maximize $x_1 + 2x_2$, subject to $0 \le x_1 \le 2$, $0 \le x_2 \le 1$, and $x_1 + x_2 \le 2$.

There are many alternative ways to describe a linear program. We may want to maximize instead of minimize; we may have equations, and/or inequalities of the form $\ge$. Sometimes we consider only nonnegative variables; the inequalities $x_i \ge 0$ may be included in (6.1), but it may be advantageous to separate them. All these versions are easily reduced to each other.

The *dual* of (6.1) is the linear program

$$\begin{array}{lrcl} \text{minimize} & b^{\mathsf{T}}y & & \\ \text{subject to} & A^{\mathsf{T}}y & = & c, \\ & y & \geq & 0. \end{array} \tag{6.2}$$

The crucial property of this very important construction is the following.

Theorem 6.2.8 (Duality Theorem) *If either one of the primal and dual programs has an optimum solution, then so does the other and the two optimum values are equal.*

The primal program is infeasible if and only if the dual is unbounded. The dual program is infeasible iff the primal is unbounded.

The primal and dual linear programs are related to each other in many ways. The following theorem describes the relationship between their optimal solutions.

Theorem 6.2.9 (Complementary Slackness Theorem) *Let x be a solution of the primal program and y, a solution of the dual program. Then both x and y are optimal if and only if for every j with $y_j > 0$, the j-th constraint of the primal problem (6.1) is satisfied with equality.*

Linear programs are solvable in polynomial time. The classical, and still very well usable algorithm to solve them is the *Simplex Method*. This is practically quite efficient, but can be exponential on some instances. The first polynomial time algorithm to solve linear programs was the *Ellipsoid Method*; this is, however, impractical. The most efficient methods known today, both theoretically and practically, are *Interior Point Methods*.

6.2.3 Polyhedral combinatorics: the stable set polytope

The basic technique of applying linear programming in discrete optimization is polyhedral combinatorics. Instead of surveying this broad topic, we illustrate it by recalling some results on the stable set polytope. A detailed account can be found e.g. in [43].

Let $G = (V, E)$ be a graph; it is convenient to assume that it has no isolated nodes. The *Stable Set Problem* is the problem of finding $\alpha(G)$. This problem is NP-hard.

The basic idea in applying linear programming to study the stable set problem is the following. For every subset $S \subseteq V$, let $\chi^S \in \mathbb{R}^V$ denote its incidence vector, i.e., the vector defined by

$$\chi_i^S = \begin{cases} 1, & \text{if } i \in S, \\ 0, & \text{otherwise,} \end{cases}$$

The *stable set polytope* $\mathrm{STAB}(G)$ of G is the convex hull of incidence vectors of all stable sets.

There is a system of linear inequalities whose solution set is exactly the polytope STAB(G), and if we can find this system, then we can find $\alpha(G)$ by optimizing the linear objective function $\sum_i x_i$. Unfortunately, this system is in general exponentially large and very complicated. But if we can find at least some linear inequalities valid for the stable set polytope, then using these we get an upper bound on $\alpha(G)$, and for special graphs, we get the exact value.

Let us survey some classes of known constraints.

Non-negativity constraints:

$$x_i \geq 0 \quad (i \in V). \tag{6.3}$$

Edge constraints:

$$x_i + x_j \leq 1 \quad (ij \in E). \tag{6.4}$$

These inequalities define a polytope FSTAB(G). The integral points in FSTAB(G) are exactly the incidence vectors of stable sets, but FSTAB(G) may have other non-integral vertices, and is in general larger than STAB(G) (see Figure 6.5).

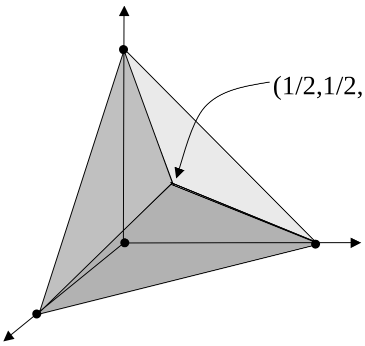


Figure 6.5. The fractional stable set polytope of the triangle. The black dots are incidence vectors of stable sets; the vertex (1/2, 1/2, 1/2) (closest to us) is not a vertex of STAB(K_3).

Proposition 6.2.10 (a) STAB(G) = FSTAB(G) *iff G is bipartite.*
(b) *The vertices of* FSTAB(G) *are half-integral.*

A *clique* is a maximal complete subgraph.

Clique constraints:

$$\sum_{i \in B} x_i \leq 1, \quad \text{where } B \text{ is a clique.} \tag{6.5}$$

Inequalities (6.3) and (6.5) define a polytope QSTAB(G), which is contained in FSTAB(G), but is in general larger than STAB(G).

A graph G is called *perfect* if $\chi(G') = \omega(G')$ for every induced subgraph G' of G. If G is perfect then so is $\overline{G}$ [62]. See [39, 43, 65] for the theory of perfect graphs.

Theorem 6.2.11 [Fulkerson–Chvatal] *$STAB(G)$ = QSTAB(G) iff G is perfect.*

A *convex corner* in $\mathbb{R}^V$ is a full-dimensional, compact, convex set P such that $x \in P$, $0 \le y \le x$ implies $y \in P$. The *antiblocker* of a convex corner P is defined as $P^* = \{x \in \mathbb{R}_+^V : x^\mathsf{T} y \le 1 \text{ for all } y \in P\}$. P^* is a convex corner and $P^{**} = P$. Figure 6.6 illustrates this important notion in 2 dimensions.

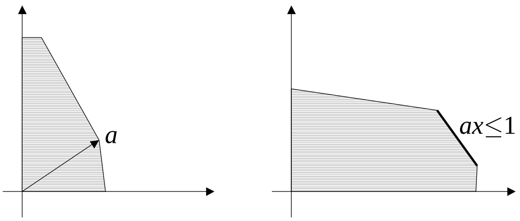


Figure 6.6. A pair of antiblocking convex corners. The vertex a on the left corresponds to the facet $ax \le 1$ on the right.

Proposition 6.2.12 *For every graph G,*

$$\text{QSTAB}(G) = \text{STAB}(\overline{G})^*.$$

Odd hole constraints:

$$\sum_{i\in C} x_i \le \frac{|C|-1}{2}, \quad \text{where } C \text{ induces a cordless odd cycle.} \tag{6.6}$$

A graph is called *t-perfect* if (6.3), (6.4) and (6.6) suffice to describe STAB(G), and *h-perfect* if (6.3), (6.5) and (6.6) suffice to describe STAB(G).

Odd antihole constraints:

$$\sum_{i\in B} x_i \le 2, \quad \text{where } B \text{ induces the complement of a cordless odd cycle.} \tag{6.7}$$

How strong are these inequalities? An inequality valid for a (for simplicity, full-dimensional) polytope $P \subseteq \mathbb{R}^n$ is called a *facet* if there are n affine independent vertices of P that satisfy it with equality. Such an inequality must occur in every description of P by linear inequalities (up to scaling by a positive number). The clique constraints are all facets, the odd hole and antihole inequalities are facets if $B = V$, and in many other cases. (If there are nodes not occurring in the inequality then they may sometimes be added to the constraint with non-zero coefficient; this is called *lifting*.)

All the previous constraints are special cases of the following construction. Let G_S denote the subgraph of G induced by $S \subseteq V$.

RANK CONSTRAINTS:

$$\sum_{i \in S} x_i \leq \alpha(G_S).$$

For this general class of constraints, however, we cannot even compute the right hand side efficiently. Another of their shortcomings is that we don't know when they are facets (or can be lifted to facets). An important special case when at least we know that they are facets was described by Chvátal [23]. A graph G is called α-critical if it has no isolated nodes, and deleting any edge e, $\alpha(G)$ increases. These graphs have an interesting and non-trivial structure theory; here we can only include figure 6.7 showing some of them.

Theorem 6.2.13 *Let $G = (V, E)$ be an α-critical graph. Then the inequality $\sum_{i \in V} x_i \leq \alpha(G)$ defines a facet of* STAB(G).

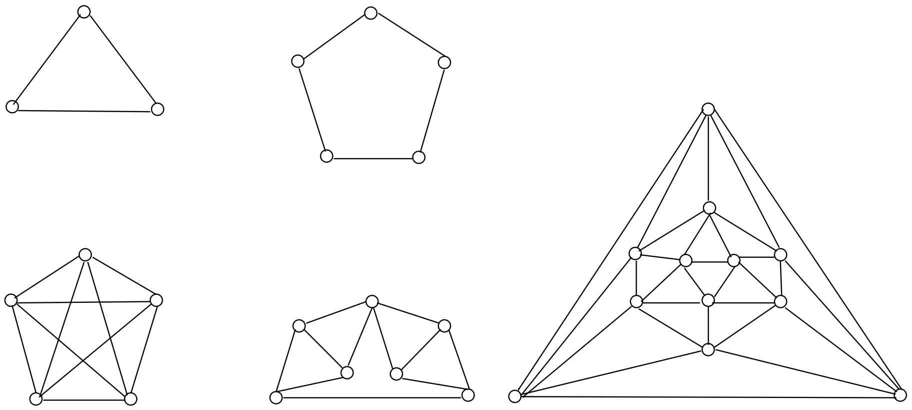


Figure 6.7. Some α-critical graphs.

6.3 Semidefinite programs

A semidefinite program is an optimization problem of the following form:

$$\begin{aligned} &\text{minimize} && c^{\mathsf{T}} x \\ &\text{subject to} && x_1 A_1 + \ldots x_n A_n - B \succeq 0 \end{aligned} \tag{6.8}$$

Here $A_1, \dots, A_n, B$ are given symmetric $m \times m$ matrices, and $c \in \mathbb{R}^n$ is a given vector. We can think of $X = x_1 A_1 + \dots x_n A_n - B$ as a matrix whose entries are linear functions of the variables.

As usual, any choice of the values x_i that satisfies the given constraint is called a *feasible solution.* A solution is *strictly feasible*, if the matrix X is positive definite. We denote by v_{primal} the supremum of the objective function.

The special case when $A_1, \dots, A_n, B$ are diagonal matrices is just a "generic" linear program, and it is very fruitful to think of semidefinite programs as generalizations of linear programs. But there are important technical differences. The following example shows that, unlike in the case of linear programs, the supremum may be finite but not a maximum, i.e., not attained by any feasible solution.

Example 6.3.1 Consider the semidefinite program

$$\begin{array}{ll} \text{minimize} & x_1 \\ \text{subject to} & \begin{pmatrix} x_1 & 1 \\ 1 & x_2 \end{pmatrix} \succeq 0 \end{array}$$

The semidefiniteness condition boils down to the inequalities $x_1, x_2 \geq 0$ and $x_1 x_2 \geq 1$, so the possible values of the objective function are all positive real numbers. Thus $v_{\text{primal}} = 0$, but the supremum is not attained.

As in the theory of linear programs, there are a large number of equivalent formulations of a semidefinite program. Of course, we could consider minimization instead of maximization. We could stipulate that the x_i are nonnegative, or more generally, we could allow additional linear constraints on the variables x_i (inequalities and/or equations). These could be incorporated into the form above by extending the A_i and B with new diagonal entries.

We could introduce the entries of A as variables, in which case the fact that they are linear functions of the original variables translates into linear relations between them. Straightforward linear algebra transforms (6.8) into an optimization problem of the form

$$\begin{array}{lrcl} \text{maximize} & C \cdot X & & \\ \text{subject to} & X & \succeq & 0 \\ & D_1 \cdot X & = & d_1 \\ & & \vdots & \\ & D_k \cdot X & = & d_k, \end{array} \tag{6.9}$$

where $C, D_1, \dots, D_k$ are symmetric $m \times m$ matrices and $d_1, \dots, d_k \in \mathbb{R}$. Note that $C \cdot X$ is the general form of a linear combination of entries of X, and so $D_i \cdot X = d_i$ is the general form of a linear equation in the entries of X.

It is easy to see that we would not get any substantially more general problem if we allowed linear inequalities in the entries of X in addition to the equations.

6.3.1 Fundamental properties of semidefinite programs

We begin with the semidefinite version of the Farkas Lemma:

Lemma 6.3.2 [Homogeneous version] *Let $A_1, \ldots, A_n$ be symmetric $m \times m$ matrices. The system*

$$x_1 A_1 + \ldots + x_n A_n \succ 0$$

has no solution in $x_1, \ldots, x_n$ if and only if there exists a symmetric matrix $Y \neq 0$ such that

$$\begin{aligned} A_1 \cdot Y &= 0 \\ A_2 \cdot Y &= 0 \\ &\vdots \\ A_n \cdot Y &= 0 \\ Y &\succeq 0\,. \end{aligned}$$

Proof. As discussed in section 6.2.1, the set $\mathcal{P}_m$ of $m \times m$ positive semidefinite matrices forms a closed convex cone. If

$$x_1 A_1 + \ldots + x_n A_n \succ 0$$

has no solution, then the linear subspace L of matrices of the form $x_1 A_1 + \ldots x_n A_n$ is disjoint from the interior of this cone PSD^m. It follows that this linear space is contained in a hyperplane that is disjoint from the interior of PSD^m. This hyperplane can be described as $\{X : \; Y \cdot X = 0\}$, where we may assume that $X \cdot Y \geq 0$ for every $X \in PSD^m$. Then $Y \neq 0$, $Y \succeq 0$ by Lemma 6.2.4, and $A_i \cdot Y = 0$ since the A_i belong to L. $\square$

By similar methods one can prove:

Lemma 6.3.3 [Inhomogeneous version] *Let $A_1, \ldots, A_n, B$ be symmetric $m \times m$ matrices. The system*

$$x_1 A_1 + \ldots x_n A_n - B \succ 0$$

has no solution in $x_1, \ldots, x_n$ if and only if there exists a symmetric matrix $Y \neq 0$ such that

$$\begin{aligned} A_1 \cdot Y &= 0 \\ A_2 \cdot Y &= 0 \\ &\vdots \end{aligned}$$

$$\begin{aligned} A_n \cdot Y &= 0 \\ B \cdot Y &\geq 0 \\ Y &\succeq 0\,. \end{aligned}$$

Given a semidefinite program (6.8), one can formulate the *dual program*:

$$\begin{array}{lrcl} \text{maximize} & B \cdot Y & & \\ \text{subject to} & A_1 \cdot Y & = & c_1 \\ & A_2 \cdot Y & = & c_2 \\ & & \vdots & \\ & A_n \cdot Y & = & c_m \\ & Y & \succeq & 0. \end{array} \tag{6.10}$$

Note that this too is a semidefinite program in the general sense. We denote by v_{dual} the supremum of the objective function.

With this notion of duality, the Duality Theorem holds in the following sense (see e.g. [96, 93, 94]):

Theorem 6.3.4 *Assume that both the primal and the dual semidefinite programs have feasible solutions. Then $v_{\text{primal}} \geq v_{\text{dual}}$. If, in addition, the primal program (say) has a strictly feasible solution, then the dual optimum is attained and $v_{\text{primal}} = v_{\text{dual}}$. In particular, if both programs have strictly feasible solutions, then the supremum resp. infimum of the objective functions are attained.*

Proof. Let $x_1, \ldots, x_n$ be any solution of (6.8) and Y, any solution of (6.10). By Proposition 6.2.3, we have

$$\sum_i c_i x_i - B \cdot Y = \operatorname{tr}(Y(\sum_i x_i A_i - B)) \geq 0,$$

which shows that $v_{\text{primal}} \geq v_{\text{dual}}$.
Moreover, the system

$$\begin{aligned} \sum_i c_i x_i &< v_{\text{primal}} \\ \sum_i x_i A_i &\succeq B \end{aligned}$$

has no solution in the x_i, by the definition of v_{primal}. Thus if we define the matrices

$$A_i' = \begin{pmatrix} -c_i & 0 \\ 0 & A_i \end{pmatrix}, \qquad B' = \begin{pmatrix} -v_{\text{primal}} & 0 \\ 0 & B \end{pmatrix},$$

then the system

$$x_1 A_1' + \ldots x_n A_n' - B' \succ 0$$

has no solution. By Lemma 6.3.3, there is a positive semidefinite matrix $Y' \neq 0$ such that

$$A_i' \cdot Y' = 0 \quad (i = 1, \dots, n), \qquad B' \cdot Y' \geq 0.$$

Writing

$$Y' = \begin{pmatrix} y_{00} & y^{\mathsf{T}} \\ y & Y \end{pmatrix},$$

we get that

$$A_i \cdot Y = y_{00} c_i \quad (i = 1, \dots, n), \qquad B \cdot Y \geq y_{00} v_{\text{primal}}.$$

We claim that $y_{00} \neq 0$. Indeed, if $y_{00} = 0$, then $y = 0$ by the semidefiniteness of Y', and since $Y' \neq 0$, it follows that $Y \neq 0$. The existence of Y would imply (by Lemma 6.3.3 again) that $x_1 A_1 + \dots x_n A_n - B \succ 0$ is not solvable, which is contrary to the hypothesis about the existence of a strictly feasible solution.

Thus $y_{00} \neq 0$, and clearly $y_{00} > 0$. By scaling, we may assume that $y_{00} = 1$. But then Y is a feasible solution of the dual problem (6.10), with objective value $B \cdot Y \geq v_{\text{primal}}$, proving that $v_{\text{dual}} \geq v_{\text{primal}}$, and completing the proof.

The following *complementary slackness conditions* also follow from this argument.

Proposition 6.3.5 *Let x be a feasible solution of the primal program and Y, a feasible solution of the dual program. Then $v_{\text{primal}} = v_{\text{dual}}$ and both x and Y are optimal solutions if and only if $Y(\sum_i x_i A_i - B) = 0$.*

The following example shows that the somewhat awkward conditions about the strictly feasible solvability of the primal and dual programs cannot be omitted (see [83] for a detailed discussion of conditions for exact duality).

Example 6.3.6 Consider the semidefinite program

$$\begin{array}{ll} \text{minimize} & x_1 \\ \text{subject to} & \begin{pmatrix} 0 & x_1 & 0 \\ x_1 & x_2 & 0 \\ 0 & 0 & x_1 + 1 \end{pmatrix} \succeq 0 \end{array}$$

The feasible solutions are $x_1 = 0$, $x_2 \geq 0$. Hence v_{primal} is assumed and is equal to 0. The dual program is

$$\begin{array}{lrcl} \text{maximize} & -Y_{33} & & \\ \text{subject to} & Y_{12} + Y_{21} + Y_{33} & = & 1 \\ & Y_{22} & = & 0 \\ & Y & \succeq & 0\,. \end{array}$$

The feasible solutions are all matrices of the form

$$\begin{pmatrix} a & 0 & b \\ 0 & 0 & 0 \\ b & 0 & 1 \end{pmatrix}$$

where $a \geq b^2$. Hence $v_{\mathrm{dual}} = -1$.

6.3.2 Algorithms for semidefinite programs

There are two essentially different algorithms known that solve semidefinite programs in polynomial time: the *ellipsoid method* and *interior point/barrier methods*. Both of these have many variants, and the exact technical descriptions are quite complicated; so we restrict ourselves to describing the general principles underlying these algorithms, and to some comments on their usefulness. We ignore numerical problems, arising from the fact that the optimum solutions may be irrational and the feasible regions may be very small; we refer to [82, 83] for discussions of these problems.

The first polynomial time algorithm to solve semidefinite optimization problems in polynomial time was the ellipsoid method. Let K be a convex body (closed, compact, convex, full-dimensional set) in $\mathbb{R}^N$. We set $S(K,t) = \{x \in \mathbb{R}^N : \; d(x,K) \leq t\}$, where d denotes euclidean distance. Thus $S(0,t)$ is the ball with radius t about 0.

A *(weak) separation oracle* for a convex body $K \subseteq \mathbb{R}^N$ is an oracle (a subroutine which is handled as a black box; one call on the oracle is counted as one step only) whose input is a rational vector $x \in \mathbb{R}^N$ and a rational $\varepsilon > 0$; the oracle either asserts that $x \in S(K,\varepsilon)$ or returns an "almost separating hyperplane" in the form of a vector $0 \neq y \in \mathbb{R}^N$ such that $y^\mathsf{T} x > y^\mathsf{T} z - \varepsilon|y|$ for all $z \in K$.

If we have a weak separation oracle for a convex body (in practice, any subroutine that realizes this oracle) then we can use the ellipsoid method to optimize any linear objective function over K [43]:

Theorem 6.3.7 *Let K be a convex body in $\mathbb{R}^n$ and assume that we know two real numbers $R > r > 0$ such that $S(0,r) \subseteq K \subseteq S(0,R)$. Assume further that we have a weak separation oracle for K. Let a (rational) vector $c \in \mathbb{R}^n$ and an error bound $0 < \varepsilon < 1$ be also given. Then we can compute a (rational) vector $x \in \mathbb{R}^n$ such that $x \in K$ and $c^\mathsf{T} x \geq c^T z - \varepsilon$ for every $y \in K$. The number of calls on the oracle and the number of arithmetic operations in the algorithm are polynomial in* $\log(R/r) + \log(1/\varepsilon) + n$.

This method can be applied to solve semidefinite programs in polynomial time, modulo some technical conditions. (Note that some complications arise already from the fact that the optimum value is not necessarily a rational number, even if all parameters are rational. A further warning is example 6.3.6.)

Assume that we are given a semidefinite program (6.8) with rational coefficients and a rational error bound $\varepsilon > 0$. Also assume that we know a rational, strictly feasible solution $\tilde{x}$, and a bound $R > 0$ for the coordinates of an optimal solution. Then the set K of feasible solutions is a closed, convex, bounded, full-dimensional set in $\mathbb{R}^n$. It is easy to compute a small ball around x_0 that is contained in K.

The key step is to design a separation oracle for K. Given a vector x, we need only check whether $x \in K$ and if not, find a separating hyperplane. Ignoring numerical problems, we can use the algorithm described in section 6.2.1 to check whether the matrix $Y = \sum_i x_i A_i - B$ is positive semidefinite. If it is, then $x \in K$. If not, the algorithm also returns a vector $v \in \mathbb{R}^m$ such that $v^\mathsf{T} Y v < 0$. Then $\sum_i x_i v^\mathsf{T} A_i v = v^\mathsf{T} B v$ is a separating hyperplane. (Because of numerical problems, the error bound in the definition of the weak separation oracle is needed.)

Thus using the ellipsoid method we can compute, in time polynomial in $\log(1/\varepsilon)$ and in the number of digits in the coefficients and in x_0, a feasible solution x such that the value of the objective function is at most $v_{\text{primal}} + \varepsilon$.

Unfortunately, the above argument gives an algorithm which is polynomial, but hopelessly slow, and practically useless. Still, the flexibility of the ellipsoid method makes it an inevitable tool in proving the *existence* (and not much more) of a polynomial time algorithm for many optimization problems.

Semidefinite programs can be solved in polynomial time and also *practically efficiently* by interior point methods [77, 1, 2]. The key to this method is the following property of the determinant of positive semidefinite matrices.

Lemma 6.3.8 *The function F defined by*

$$F(Y) = -\log \det\, (Y)$$

is convex and analytic in the interior of the semidefinite cone P_n, and tends to ∞ at the boundary.

The algorithm can be described very informally as follows. The feasible domain of our semidefinite optimization problem is of the form $K = \mathsf{P}_n \cap A$, where A is an affine subspace of symmetric matrices. We want to minimize a linear function $C \cdot X$ over $X \in K$. The good news is that K is convex. The bad news is that the minimum will be attained on the boundary of K, and this boundary can have a very complicated structure; it is neither smooth nor polyhedral. Therefore, neither gradient-type methods nor the methods of linear programming can be used to minimize $C \cdot X$.

The main idea of barrier methods is that instead of minimizing $C^\mathsf{T} X$, we minimize the function $F_C(X) = F(X) + \lambda C^\mathsf{T} X$ for some $\lambda > 0$. Since F_λ tends to infinity on the boundary of K, the minimum will be attained in the interior. Since F_λ is convex and analytic in the interior, the minimum can be very efficiently computed by a variety of numerical methods (conjugate gradient etc.)

Of course, the point we obtain this way is not what we want, but if λ is large it will be close. If we don't like it, we can increase λ and use the minimizing point for the old F_λ as the starting point for a new gradient type algorithm. (In practice, we can increase λ after each iteration of this gradient algorithm.)

One can show that (under some technical assumptions about the feasible domain) this algorithm gives an approximation of the optimum with relative error ε in time polynomial in $\log(1/\varepsilon)$ and the size of the presentation of the program. The proof of this depends on a further rather technical property of the determinant, called "self-concordance". We don't go into the details, but refer to the articles [2, 93, 94] and the book [76].

6.4 Obtaining semidefinite programs

How do we obtain semidefinite programs? It turns out that there are a number of considerations from which semidefinite programs, in particular semidefinite relaxations of combinatorial optimization problems arise. These don't always lead to different relaxations; in fact, the best known applications of semidefinite programming seem to be very robust in the sense that different methods for deriving their semidefinite relaxations yields the same, or almost the same, result. However, these different methods seem to have different heuristic power.

6.4.1 Unit distance graphs and orthogonal representations

We start with some semidefinite programs arising from geometric problems. A *unit distance representation* of a graph $G = (V, E)$ is a mapping $u : V \to \mathbb{R}^d$ for some $d \geq 1$ such that $|u_i - u_j| = 1$ for every $ij \in E$ (we allow that $|u_i - u_j| = 1$ for some $ij \in \overline{E}$). Figure 6.8 shows a 2-dimensional unit distance representation of the Petersen graph [31].

There are many questions one can ask about the existence of unit distance representations: what is the smallest dimension in which it exists? what is the smallest radius of a ball containing a unit distance representation of G (in any dimension)? In this paper, we are only concerned about the last question, which can be answered using semidefinite programming (for a survey of other aspects of such geometric representations, see [73]). Considering the Gram matrix $A = (u_i^\mathsf{T} u_j)$, it is easy to obtain the following reduction to semidefinite programming:

Proposition 6.4.1 *A graph G has a unit distance representation in a ball of radius R (in some appropriately high dimension) if and only if there*

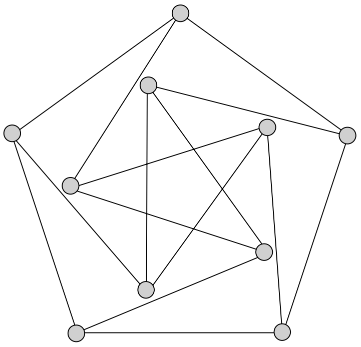


Figure 6.8. A unit distance representation of the Petersen graph.

exists a positive semidefinite matrix A such that

$$\begin{aligned} A_{ii} &\le R^2 \quad (i \in V) \\ A_{ii} - 2Aij + A_{jj} &= 1 \quad (ij \in E). \end{aligned}$$

In other words, the smallest radius R is the square root of the optimum value of the semidefinite program

$$\begin{array}{lrcl} \text{minimize} & w & & \\ \text{subject to} & A & \succeq & 0 \\ & A_{ii} & \le & w \quad (i \in V) \\ & A_{ii} - 2Aij + A_{jj} & = & 1 \quad (ij \in E). \end{array}$$

The unit distance embedding of the Petersen graph in Figure 6.8 is not an optimal solution of this problem. Let us illustrate how semidefinite optimization can find the optimal embedding by determining this for the Petersen graph. In the formulation above, we have to find a 10×10 positive semidefinite matrix A satisfying the given linear constraints. For a given w, the set of feasible solutions is convex, and it is invariant under the automorphisms of the Petersen graph. Hence there is an optimum solution which is invariant under these automorphisms (in the sense that if we permute the rows and columns by the same automorphism of the Petersen graph, we get back the same matrix).

Now we know that the Petersen graph has a very rich automorphism group: not only can we transform every node into every other node, but also every edge into every other edge, and every nonadjacent pair of nodes into every other non-adjacent pair of nodes. A matrix invariant under these automorphisms has only 3 different entries: one number in the diagonal, another number in positions corresponding to edges, and a third number in positions corresponding to nonadjacent pairs of nodes. This means that this optimal matrix A can be written as

$$A = xP + yJ + zI,$$

where P is the adjacency matrix of the Petersen graph, J is the all-1 matrix, and I is the identity matrix. So we only have these 3 unknowns x, y and z to determine.

The linear conditions above are now easily translated into the variables x, y, z. But what to do with the condition that A is positive semidefinite? Luckily, the eigenvalues of A can also be expressed in terms of x, y, z. The eigenvalues of P are well known (and easy to compute): they are 3, 1 (5 times) and -2 (4 times). Here 3 is the degree, and it corresponds to the eigenvector $\mathbf{1} = (1, \dots, 1)$. This is also an eigenvector of J (with eigenvalue 10), and so are the other eigenvectors of P, since they are orthogonal to $\mathbf{1}$, and so are in the nullspace of J. Thus the eigenvalues of $xP + yJ$ are $3x + 10y$, x, and $-2x$. Adding zI just shifts the spectrum by z, so the eigenvalues of A are $3x + 10y + z$, $x + z$, and $-2x + z$. Thus the positive semidefiniteness of A, together with the linear constraints above, gives the following linear program for x, y, z, w:

$$\begin{array}{lrcl}
\text{minimize} & w & & \\
\text{subject to} & 3x + 10y + z & \geq & 0, \\
& x + z & \geq & 0, \\
& -2x + z & \geq & 0, \\
& y + z & \leq & w, \\
& 2z - 2x & = & 1.
\end{array}$$

It is easy to solve this: clearly the optimum solution will have $w = y + z$, and $y = (-3x - z)/10$. We can also substitute $x = z - 1/2$, which leaves us with a single variable. The solution is $x = -1/4$, $y = 1/20$, $z = 1/4$, and $w = 3/10$. Thus the smallest radius of a ball in which the Petersen graph has a unit distance representation is $\sqrt{3/10}$. The corresponding matrix A has rank 4, so this representation is in 4 dimension.

It would be difficult to draw a picture of this representation, but I can offer the following nice matrix, whose columns will realize this representation (the center of the smallest ball containing it is not at the origin!):

$$\begin{pmatrix}
1/2 & 1/2 & 1/2 & 1/2 & 0 & 0 & 0 & 0 & 0 & 0 \\
1/2 & 0 & 0 & 0 & 1/2 & 1/2 & 1/2 & 0 & 0 & 0 \\
0 & 1/2 & 0 & 0 & 1/2 & 0 & 0 & 1/2 & 1/2 & 0 \\
0 & 0 & 1/2 & 0 & 0 & 1/2 & 0 & 1/2 & 0 & 1/2 \\
0 & 0 & 0 & 1/2 & 0 & 0 & 1/2 & 0 & 1/2 & 1/2
\end{pmatrix} \qquad (6.11)$$

(This matrix reflects the fact that the Petersen graph is the complement of the line-graph of K_5.)

It turns out that from a graph theoretic point of view, it is more interesting to modify the question and require that the nodes all lie on the surface of the sphere (in our example this happened automatically, due to the symmetries of the Petersen graph). In other words, we are interested in the smallest sphere (in any dimension) on which a given graph G can be drawn so that the euclidean distance between adjacent nodes is 1 (of course, we could talk here about any other given distance instead of 1, or spherical distance instead of euclidean, without essentially changing the problem). Again, by considering the Gram matrix $A = (u_i^\top u_j)$, we find that this smallest radius $t(G)$ is given by the square root of the optimum value of the following semidefinite program:

$$\begin{array}{lrcll} \text{minimize} & z & & & \\ \text{subject to} & A & \succeq & 0 & \\ & A_{ii} & = & z & (i \in V) \\ & A_{ii} - 2Aij + A_{jj} & = & 1 & (ij \in E). \end{array} \tag{6.12}$$

Since $A = \mathrm{diag}(1/2, \ldots, 1/2)$ is a solution, it follows that the optimal z satisfies $z \le 1/2$.

Another way of looking at this question is to add a further dimension. Think of a unit distance representation of the graph on the sphere with radius t as lying in a “horizontal” hyperplane. Choose the origin above the center of the sphere so that the vectors pointing to adjacent nodes of the graph are orthogonal (the distance of the origin to the hyperplane will be $\sqrt{(1/2) - z}$). It is worth scaling up by a factor of $\sqrt{2}$, so that the vectors pointing to the nodes of the graph become unit vectors. Such a system of vectors is called an *orthonormal representation* of the complementary graph $\overline{G}$ (the complementation is, of course, just a matter of convention). The matrix (6.11) above is an orthogonal representation of the complement of the Petersen graph, which is related to its unit distance representation by this construction, up to a change in coordinates.

In the introduction, we constructed an orthonormal representation of the pentagon graph (Figure 6.1). This is not the simplest case (in a sense, it is the smallest interesting orthogonal representation). Figure 6.9 below shows that it if we add a diagonal to the pentagon, then a much easier orthogonal representation in 2 dimensions can be constructed.

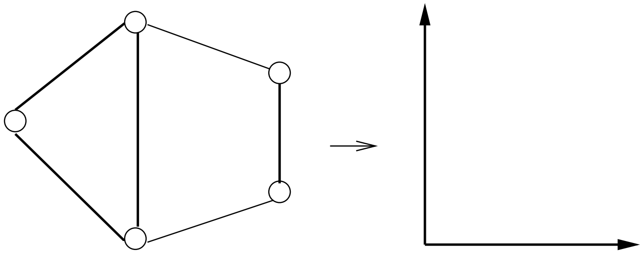


Figure 6.9. An (almost) trivial orthogonal representation

Orthogonal representations of graphs have several applications in graph theory. In particular, it turns out that the quantity $1/(1 - 2t(G)^2)$ is just $\vartheta(\overline{G})$ introduced before (for the complementary graph $\overline{G}$. We’ll return to it in sections 6.5.1 and 6.6.1.

6.4.2 Discrete linear and quadratic programs

Consider a typical 0-1 optimization problem:

$$\begin{aligned} \text{maximize} \quad & c^t x \\ \text{subject to} \quad & \begin{cases} Ax \le b \\ x \in \{0,1\}^n. \end{cases} \end{aligned} \tag{6.13}$$

We get an equivalent problem if we replace the last constraint by the quadratic equation

$$x_i^2 = x_i \qquad (i = 1, \dots, n). \tag{6.14}$$

Once we allow quadratic equations, many things become much simpler. First, we can restrict ourselves to homogeneous quadratic equations, by introducing a new variable x_0, and setting it to 1. Thus (6.14) becomes

$$x_i^2 = x_0 x_i \qquad (i = 1, \dots, n). \tag{6.15}$$

Second, we don't need inequalities: we can just replace $F \ge 0$ by $F - x^2 = 0$, where x is a new variable. Third, we can often replace constraints by simpler and more powerful constraints. For example, for the stable set problem (section 6.2.3), we could replace the edge constraints by the quadratic equations

$$x_i x_j = 0 \qquad (ij \in E). \tag{6.16}$$

Trivially, the solutions of (6.14) and (6.16) are precisely the incidence vectors of stable sets. If we are interested in $\alpha(G)$, we can consider the objective function $\sum_{i=1}^n x_0 x_i$.

Unfortunately, this also shows that even the solvability of such a simple system of quadratic equations (together with a linear equation $\sum_i x_i = \alpha$) is NP-hard.

The trick to obtain a polynomially solvable relaxation of such problems is to think of the x_i as vectors in $\mathbb{R}^k$ (and multiplication as inner product). For $k = 1$, we get back the original 0-1 optimization problem. For $k = 2, 3 \dots$, we get various optimization problems with geometric flavor, which are usually not any easier than the original. For example, for the stable set problem we get the *vector relaxation*

$$\begin{aligned} \text{maximize} \quad & \sum_{i \in V} v_0^\mathsf{T} v_i \\ \text{subject to} \quad & v_i \in \mathbb{R}^k \\ & v_0^\mathsf{T} v_i = |v_i|^2 \quad (i \in V) && (6.17) \\ & v_i^\mathsf{T} v_j = 0 \quad (ij \in E). && (6.18) \end{aligned}$$

But if we take $k = n$, then we get a relaxation which is polynomial time solvable. Indeed, we can introduce new variables $Y_{ij} = v_i^\mathsf{T} v_j$ and then the constraints and the objective function become linear, and if in addition we impose the condition that $Y \succeq 0$, then we get a semidefinite optimization problem. If we solve this problem, and then write Y as a Gram matrix, we obtain an optimum solution of the vector relaxation.

The conditions on vector relaxations often have useful geometric content. For example, (6.17) (which is common to the vector relaxations of all 0-1 programs) can be written in the following two forms:

$$(v_0 - v_i)^{\mathsf{T}} v_i = 0; \qquad \left| v_i - \frac{1}{2} v_0 \right|^2 = \frac{1}{4}.$$

This says that the vectors v_i and $v_0 - v_i$ are orthogonal to each other, and all the points v_i lie on the sphere with radius 1/2 centered at $(1/2)v_0$. (6.18) says that the v_i form an orthogonal representation of the complement of G.

For discrete linear or quadratic programs with variables from $\{-1, 1\}$, (6.14) becomes even simpler:

$$x_i^2 = 1, \tag{6.19}$$

i.e., the vectors are unit vectors. In the case of the Maximum Cut Problem for a graph $G = (V, E)$, we can think of a 2-coloring as an assignment of 1's and -1's to the nodes, and the number of edges in the cut is

$$\sum_{ij \in E} \frac{1}{4}(x_i - x_j)^2.$$

The vector relaxation of this problem has the nice physical meaning given in the introductory example (energy-minimization).

One can add further constraints. For example, if the variables x_i are 0-1, then we have

$$(x_i - x_j)(x_i - x_k) \geq 0$$

for any three variables. We may add these inequalities as quadratic constraints, and then get a vector relaxation that satisfies, besides the other constraints, also

$$(v_i - v_j)^{\mathsf{T}}(v_i - v_k) \geq 0.$$

Geometrically, this means that every triangle spanned by the vectors v_i is acute; this property is sometimes useful to have.

A further geometric property that can be exploited in some cases is *symmetry.* Linear systems always have solutions invariant under the symmetries of the system, but quadratic systems, or discrete linear systems do not. For example, if G is a cycle, then the system (6.14)-(6.16) is invariant under rotation, but its only solution invariant under rotation is the trivial all-0 vector. One advantage of the semidefinite relaxation is that it restores symmetric solvability.

Assume that we start with a quadratic system such that both the constraint set and the objective function are invariant under some permutation group Γ acting on the variables (for example, it can be invariant under the cyclic shift of indices). It may be that no optimal solution of the quadratic system is invariant under these permutations: For example, no maximal stable set in a cycle is invariant under cyclic shifts. However, in a semidefinite program feasible solutions define convex sets in the space of matrices, and the objective function is linear. Hence by averaging, we can assert that there exists an optimum solution Y which itself is invariant under all permutations of the indices under which the semidefinite program is. In other words, the semidefinite relaxation of the quadratic system has an optimal solution $Y \succeq 0$, such that if $\gamma \in \Gamma$, then

$$Y_{\gamma(i),\gamma(j)} = Y_{ij}. \tag{6.20}$$

Now we go over to the vector relaxation: this is defined by $Y_{ij} = v_i^T v_j$, where $v_i \in \mathbb{R}^d$ for some $d \le n$. We may assume that the v_i span $\mathbb{R}^d$. Let $\gamma \in \Gamma$. (6.20) says that $v_{\gamma(i)}^T v_{\gamma(j)} = v_i^T v_j$. In other words, the permutation $v_i \mapsto v_{\gamma(i)}$ preserves the length of the u_i and all the angles between them, and hence there is an orthogonal matrix M_γ such that $u_{\gamma(i)} = M_\gamma u_i$. Since the u_i span the space, this matrix M_γ is uniquely determined, and so we get a representation of Γ in $\mathbb{R}^d$. The vector solution (v_i) is invariant under this representation.

6.4.3 Spectra of graphs

Let $G = (V, E)$ be a graph. We denote by $\overline{G} = (V, \overline{E})$ its complement and set $\Delta = \{ii : \ i \in V\}$. The adjacency matrix A_G of G is defined by

$$(A_G)_{ij} = \begin{cases} 1, & \text{if } ij \in E, \\ 0, & \text{if } ij \in \overline{E} \cup \Delta. \end{cases}$$

Let $\lambda_1 \ge \ldots \ge \lambda_n$ be the eigenvalues of A_G. It is well known and easy to show that if G is d-regular than $\lambda_1 = d$. Since the trace of A_G is 0, we have $\lambda_1 + \ldots + \lambda_n = 0$, and hence if $E \ne \emptyset$ then $\lambda_1 > 0$ but $\lambda_n < 0$.

There are many useful connections between the eigenvalues of a graph and its combinatorial properties. The first of these follows easily from interlacing eigenvalues.

Proposition 6.4.2 *The maximum size $\omega(G)$ of a clique in G is at most $\lambda_1 + 1$. This bound remains valid even if we replace the non-diagonal 0's in the adjacency matrix by arbitrary real numbers.*

The following bound on the chromatic number is due to Hoffman.

Proposition 6.4.3 *The chromatic number $\chi(G)$ of G is at least $1 - (\lambda_1/\lambda_n)$. This bound remains valid even if we replace the 1's in the adjacency matrix by arbitrary real numbers.*

The following bound on the maximum size of a cut is due to Delorme and Poljak [28, 29, 75, 81], and was the basis for the Goemans-Williamson algorithm discussed in the introduction.

Proposition 6.4.4 *The maximum size $\gamma(G)$ of a cut in G is at most $|E|/2 - (n/4)\lambda_n$. This bound remains valid even if we replace the diagonal 0's in the adjacency matrix by arbitrary real numbers.*

Observation: to determine the best choice of the "free" entries in 6.4.2, 6.4.3 and 6.4.4 takes a semidefinite program. Consider 6.4.2 for example: we fix the diagonal entries at 0, the entries corresponding to edges at 1, but are free to choose the entries corresponding to non-adjacent pairs of vertices (replacing the off-diagonal 1's in the adjacency matrix). We want to minimize the largest eigenvalue. This can be written as a semidefinite program:

$$\begin{array}{lrcll} \text{minimize} & t & & & \\ \text{subject to} & tI - X & \succeq & 0, & \\ & X_{ii} & = & 0 & (\forall i \in V), \\ & X_{ij} & = & 1 & (\forall ij \in E). \end{array}$$

It turns out that the semidefinite program constructed for 6.4.3 is just the dual of this, and their common optimum value is the parameter $\vartheta(G)$ introduced before. The program for 6.4.4 gives the approximation used by Goemans and Williamson (for the case when all weights are 1, from which it is easily extended). See [50] for a similar method to obtain an improved bound on the mixing rate of random walks.

6.4.4 Engineering applications

Semidefinite optimization has many applications in stability problems of dynamical systems and optimal control. Since this is not in the main line of these lecture notes, we only illustrate this area by a simple example; see chapter 14 of [98] for a detailed survey.

Consider a "system" described by the differential equation

$$\frac{dx}{dt} = A(t)x(t), \tag{6.21}$$

where $x \in \mathbb{R}^n$ is a vector describing the state of the system, and $A(t)$ is an $n \times n$ matrix, about which we only know that it is a linear combination of m given matrices $A_1, \ldots, A_m$ with nonnegative coefficients (an example of this situation is when we know the signs of the matrix entries). Is the zero solution $x(t) \equiv 0$ asymptotically stable, i.e., is it true that for every initial value $x(0) = x_0$, we have $x(t) \to 0$ as $t \to \infty$?

Suppose first that $A(t) = A$ is a constant matrix, and also suppose that we know from the structure of the problem that it is symmetric. Then the basic theory of differential equations tells us that the zero solution is asymptotically stable if and only if A is negative definite.

But semidefinite optimization can be used even if $A(t)$ can depend on t, and is not necessarily symmetric, at least to establish a sufficient condition for asymptotic stability. We look for a *quadratic Lyapunov function* $x^\mathsf{T} Px$, where P is a positive definite $n \times n$ matrix, such that

$$\frac{d}{dt} x(t)^T P x(t) < 0 \tag{6.22}$$

for every non-zero solution of the differential equation. If we find such a matrix P, then Lyapunov's theorem implies that the trivial solution is asymptotically stable.

Now the left hand side of (6.22) can be written as

$$\frac{d}{dt} x(t)^T P x(t) = \dot{x}^\mathsf{T} P x + x^\mathsf{T} P \dot{x} = x^\mathsf{T}(A^\mathsf{T} P + PA)x.$$

Thus (6.22) holds for every solution and every t if and only if $A^\mathsf{T} P + PA$ (which is a symmetric matrix) is negative semidefinite. We don't explicitly know $A(t)$, but we do know that it is a linear combination of $A_1, \dots, A_m$; so it suffices we require that the matrices $A_i^\mathsf{T} P + PA_i$, $i = 1, ..., m$ are negative semidefinite.

To sum up, we see that a sufficient condition for the asymptotic stability of the zero solution of (6.21) is that the semidefinite system

$$\begin{array}{rcl} P & \succ & 0, \\ -A^\mathsf{T} P - PA & \succ & 0 \quad (i = 1, \dots, m) \end{array}$$

has a solution in P.

6.5 Semidefinite programming in proofs

6.5.1 More on stable sets and the Shannon capacity

An *orthogonal representation* of a graph $G = (V, E)$ is a mapping (labeling) $u : V \to \mathbb{R}^d$ for some d such that $u_i^\mathsf{T} u_j = 0$ for all $ij \in \overline{E}$. An *orthonormal representation* is an orthogonal representation with $|u_i| = 1$ for all i. The *angle* of an orthonormal representation is the smallest half-angle of a rotational cone containing the representing vectors.

Proposition 6.5.1 *The minimum angle ϕ of any orthogonal representation of G is given by* $\cos^2 \phi = 1/\vartheta(G)$.

In what follows we collect some properties of ϑ, mostly from [64] (see also [57] for a survey).

We start with a formula that expresses $\vartheta(G)$ as a maximum over orthogonal representations of the *complementary* graph. Let the *leaning* of an orthonormal representation of G be defined as $\sum_{i\in V}(e_1^{\mathsf{T}}u_i)^2$.

Proposition 6.5.2 *The maximum leaning of an orthonormal representation of G is $\vartheta(\overline{G})$.*

The "umbrella" construction given in the introduction shows, by Proposition 6.5.1, that $\vartheta(C_5) \le \sqrt{5}$, and by Proposition 6.5.2, that $\vartheta(\overline{C_5}) \ge \sqrt{5}$. Hence $\vartheta(C_5) = \sqrt{5}$.

Proposition 6.5.2 is a "duality" result, which is in fact a consequence of the Duality Theorem of semidefinite programs (Theorem 6.3.4). To see the connection, let us give a "semidefinite" formulation of ϑ. This formulation is by no means unique; in fact, several others come up in these lecture notes.

Proposition 6.5.3 *$\vartheta(G)$ is the optimum of the following semidefinite program:*

$$\begin{array}{lrcll} \textit{minimize} & t & & & \\ \textit{subject to} & Y & \succeq & 0 & \\ & Y_{ij} & = & -1 & (\forall\, ij \in E(\overline{G})) \\ & Y_{ii} & = & t-1 & \end{array} \tag{6.23}$$

It is also the optimum of the dual program

$$\begin{array}{lrcll} \textit{maximize} & \sum_{i\in V}\sum_{j\in V} Z_{ij} & & & \\ \textit{subject to} & Z & \succeq & 0 & \\ & Z_{ij} & = & 0 & (\forall\, ij \in E(G)) \\ & \mathrm{tr}(Z) & = & 1 & \end{array} \tag{6.24}$$

Any stable set S provides a feasible solution of (6.24), by choosing $Z_{ij} = 1/|S|$ if $i, j \in S$ and 0 otherwise. Similarly, any k-coloring of $\overline{G}$ provides a feasible solution of (6.23), by choosing $Y_{ij} = -1$ if i and j have different colors, $Y_{ii} = k-1$ and $Y_{ij} = 0$ otherwise. These explicit solutions imply the following.

Theorem 6.5.4 [Sandwich Theorem] *For every graph G,*

$$\omega(G) \le \vartheta(\overline{G}) \le \chi(G).$$

The *fractional chromatic number* $\chi^*(G)$ is defined as the least t for which there exists a family $(A_j : j = 1, \dots, p)$ of stable sets in G, and nonnegative weights $(\tau_j : j = 1, \dots, p)$ such that $\sum\{\tau_j : A_j \ni i\} \ge 1$ for all $i \in V$ and $\sum_j \tau_j = t$. Note that the definition χ^* can be considered as a linear program. By linear programming duality, $\chi^*(G)$ is equal to the largest s for which there exist weights $(\sigma_i : i \in V)$ such that $\sum_{i\in A}\sigma_i \le 1$ for every stable set A and $\sum_i \sigma_i = s$.

Clearly $\omega(G) \le \chi^*(G) \le \chi(G)$.

Proposition 6.5.5 $\vartheta(G) \le \chi^*(\overline{G})$.

Returning to orthogonal representations, it is easy to see that not only the angle, but also the dimension of the representation yields an upper bound on $\alpha(G)$. This is, however, not better that ϑ:

Proposition 6.5.6 *Suppose that G has an orthonormal representation in dimension d. Then $\vartheta(G) \le d$.*

On the other hand, if we consider orthogonal representations over fields of finite characteristic, the dimension may be a better bound than ϑ [44, 6]. This, however, goes outside the ideas of semidefinite optimization.

To relate ϑ to the Shannon capacity of a graph, the following is the key observation:

Proposition 6.5.7 *For any two graphs,*

$$\vartheta(G \cdot H) = \vartheta(G)\vartheta(H)$$

and

$$\vartheta(\overline{G \cdot H}) = \vartheta(\overline{G})\vartheta(\overline{H}).$$

It is now easy to generalize the bound for the Shannon capacity of the pentagon, given in the introduction, to arbitrary graphs.

Corollary 6.5.8 *For every graph,*

$$\Theta(G) \le \vartheta(G).$$

Does equality hold here? Examples by Haemers [44], and more recent much sharper examples by Alon [6] show that the answer is negative in general. But we can derive at least one interesting class of examples from the general results below.

Proposition 6.5.9 *For every graph G,*

$$\vartheta(G)\vartheta(\overline{G}) \ge n.$$

If G has a vertex-transitive automorphism group, then equality holds.

Corollary 6.5.10 *If G is a self-complementary graph on n nodes with a node-transitive automorphism group, then*

$$\Theta(G) = \vartheta(G) = \sqrt{n}.$$

An example to which this corollary applies is the *Paley graph*: for a prime $p \equiv 1 \pmod 4$, we take the $\{0, 1, \dots, p-1\}$ as vertices, and connect two of them iff their difference is a quadratic residue. Thus we get an infinite family for which the Shannon capacity is non-trivial (i.e., $\Theta > \alpha$), and can be determined exactly.

The Paley graphs are quite similar to random graphs, and indeed, for random graphs ϑ behaves similarly:

Theorem 6.5.11 (Juhász [49]) *If G is a random graph on n nodes then $\sqrt{n} < \vartheta(G) < 2\sqrt{n}$ with probability $1 - o(1)$.*

It is not known how large the Shannon capacity of a random graph is.

We conclude this section by using semidefinite optimization to add further constraints to the stable set polytope (continuing the treatment in section 6.2.3). For every orthonormal representation $(v_i :\ i \in V)$ of $\overline{G}$, we consider the linear constraint

$$\sum_{i \in V} (e_1^\mathsf{T} v_i)^2 x_i \le 1. \tag{6.25}$$

It is easy to see that these inequalities are valid for STAB(G); we call them *orthogonality constraints.* The solution set of non-negativity and orthogonality constraints is denoted by TSTAB(G). It is clear that TSTAB is a closed, convex set. The incidence vector of any stable set A satisfies (6.25). Indeed, it then says that

$$\sum_{i \in A} (e_1^\mathsf{T} v_i)^2 \le 1.$$

Since the v_i $(i \in A)$ are mutually orthogonal, the left hand side is just the squared length projection of e_1 onto the subspace spanned by these e_i, and the length of this projection is at most the length of e_1, which is 1.

Furthermore, every clique constraint is an orthogonality constraint. Indeed,

$$\sum_{i \in B} x_i \le 1$$

is the constraint derived from the orthogonal representation

$$i \mapsto \begin{cases} e_1, & \text{if } i \in A, \\ e_i, & \text{if } i \notin A. \end{cases}$$

Hence we have

$$\text{STAB}(G) \subseteq \text{TSTAB}(G) \subseteq \text{QSTAB}(G)$$

for every graph G.

There is a dual characterization of TSTAB [42], which can be derived from semidefinite duality. For every orthonormal representation $(u_i :\ i \in V)$, consider the vector $x[u] = (e_1^\mathsf{T} u_i)^2 :\ i \in V) \in \mathbb{R}^V$.

Theorem 6.5.12 TSTAB$(G) = \{x[u] :\ u$ is an orthonormal representation of $G\}$.

Not every orthogonality constraint is a clique constraint; in fact, the number of essential orthogonality constraints is infinite in general:

Theorem 6.5.13 TSTAB(G) *is polyhedral if and only if the graph is perfect. In this case* TSTAB = STAB = QSTAB.

While TSTAB is a rather complicated set, in many respects it behaves much better than, say, STAB. For example, it has a very nice connection with graph complementation:

Theorem 6.5.14 TSTAB($\overline{G}$) *is the antiblocker of* TSTAB(G).

Maximizing a linear function over STAB(G) or QSTAB(G) is NP-hard; but, surprisingly, TSTAB behaves much better:

Theorem 6.5.15 *Every linear objective function can be maximized over* TSTAB(G) *(with arbitrarily small error) in polynomial time.*

The maximum of $\sum_i x_i$ over TSTAB(G) is the familiar function $\vartheta(G)$.

6.5.2 Discrepancy and number theory

Let $\mathcal{F}$ be a family of subsets of $\{0, 1, \dots, n-1\}$. We want to find a sequence $x = (x_0, x_1, \dots, x_{n-1})$ of ± 1's so that each member of $\mathcal{F}$ contains about as many 1's as -1's. More exactly, we define the *discrepancy of the sequence* x by

$$\max_{A \in \mathcal{F}} \left| \sum_{i \in A} x_i \right|,$$

and the *discrepancy of the family* $\mathcal{F}$ by

$$\Delta(\mathcal{F}) = \min_{x \in \{-1,1\}^n} \max_{A \in \mathcal{F}} \left| \sum_{i \in A} x_i \right|.$$

We can also consider the "average discrepancy" in various versions. For our purposes, we only need the ℓ_2*-discrepancy*

$$\Delta_2(\mathcal{F}) = \min_{x \in \{-1,1\}^n} \frac{1}{|\mathcal{F}|} \sum_{A \in \mathcal{F}} \left(\sum_{i \in A} x_i \right)^2.$$

It is clear that $\Delta_2 \le \Delta^2$. (We refer to [17] and [18] for an exposition of combinatorial discrepancy theory.)

Clearly, $\Delta(\mathcal{F})$ can be thought of as the optimum of a linear program in $\{-1, 1\}$-variables:

$$\begin{array}{lrcl} \text{minimize} & t & & \\ \text{subject to} & -t & \le & \sum_{i \in A} x_i \le t \\ & x_i \in \{-1, 1\}, & & \end{array} \qquad (6.26)$$

while Δ_2 is optimum of a quadratic function in $\{-1,1\}$-variables (but otherwise unconstrained). So both quantities have natural semidefinite relaxations. We only formulate the second:

$$\begin{array}{lrcl} \text{minimize} & \frac{1}{|\mathcal{F}|}\sum_{A\in\mathcal{F}}\sum_{i\in A}\sum_{j\in A} Y_{ij} & & \\ \text{subject to} & Y & \succeq & 0, \\ & Y_{ii} & = & 1 \;\; (\forall\, i \in V). \end{array} \tag{6.27}$$

We show how to use the semidefinite relaxation to estimate $\Delta(\mathcal{F})$ in the case when $\mathcal{F}$ is the family of arithmetic progressions in $\{0,1,\dots,n-1\}$ [68]. One way of looking at this particular question is to think of the x_i in the definition of discrepancy as the output of a pseudorandom number generator, and of the discrepancy, as a randomness test (a quantitative version of von Mises' test). If the x_i are truly random, we expect this discrepancy to be about $n^{1/2}$. Most "bad" sequences one encounters fail by producing a larger discrepancy. Can a sequence fail by producing a discrepancy that is too small?

The theorem of Roth [85] below shows that the discrepancy $\Delta(\mathcal{F})$ cannot be smaller than $\Omega(n^{1/4})$; this allows sequences to have substantially smaller discrepancy than a random sequence. One might expect that the lower bound in the theorem can be strengthened to about $\Omega(n^{1/2})$ (so that the random sequences would have, at least approximately, the smallest discrepancy), but it was shown by Beck [16] that Roth's estimate is sharp up to a logarithmic factor. Recently, even this logarithmic factor was removed by Matoušek and Spencer [74].

Theorem 6.5.16 *For every sequence $(x_0,\dots,x_{n-1})$, $x_i \in \{-1,1\}$, there is an arithmetic progression $A \subseteq \{0,\dots,n-1\}$ such that*

$$\left|\sum_{i\in A} x_i\right| > \frac{1}{14}n^{1/4}.$$

All proofs of this theorem establish more: one has such an arithmetic progression A with difference at most $8k$ and length exactly k, where $k = \lfloor\sqrt{n/8}\rfloor$. We consider arithmetic progressions modulo n, *i.e.*, we let them wrap around. (Of course, in this case it may happen that the progression with the large discrepancy is wrapped; but since $(k-1)(8k) < n$, it wraps over n at most once, and so it is the union of two unwrapped arithmetic progressions, one of which has discrepancy at least half the original.) Let $\mathcal{H}$ denote the family of such arithmetic progressions. Clearly $|\mathcal{H}| = 8kn$.

Following Roth, we prove the stronger result that the ℓ_2-discrepancy of arithmetic progressions in $\mathcal{H}$ is at least $(1/49)n^{1/2}$; even stronger, we prove that the optimum of its semidefinite relaxation is large: the minimum of

$$\frac{1}{|\mathcal{H}|}\sum_{A\in\mathcal{H}}\sum_{i\in A}\sum_{j\in A} Y_{ij} \tag{6.28}$$

subject to

$$\begin{aligned} Y &\succeq 0, & (6.29)\\ Y_{ii} &= 1 \ \ (1 \le i \le n) & (6.30)\end{aligned}$$

is at least $(1/49)n^{1/2}$.

The next step is to notice that both (6.30) and (6.29) are invariant under the cyclic shift of indices. Hence by our discussions in section 6.4.2, we have an optimal vector solution $(u_0, \dots, u_n)$, and an orthogonal matrix M such that $M^n = I$ and $u_i = M^i u_0$.

Elementary group representation theory tells us that the space decomposes into the direct sum of 1- and 2-dimensional subspaces invariant under M. In other words, if we choose a basis appropriately, M has a block-diagonal form

$$M = \begin{pmatrix} M_1 & 0 & \dots & 0 \\ 0 & M_2 & \dots & 0 \\ \vdots & & & \vdots \\ 0 & 0 & \dots & M_d \end{pmatrix}$$

where each M_t is a 1×1 or 2×2 real matrix of order n.

We show that the statement is true if M has only one block (thus $d = 1$ or 2). The general case then follows easily by adding up the lower bounds on the objective function for all diagonal blocks. We treat the case $d = 2$; the case $d = 1$ is trivial.

The matrix M defines a rotation in the plane with an angle $2\pi a/n$ for some $1 \le a \le n$. By Dirichlet's Theorem, there are integers $1 \le q \le 8k$ and p such that $|q(a/n) - p| < 1/(8k)$. This implies that for every arithmetic progression A of difference q and length k, the vectors $M^j u_0$ $(j \in A)$ point in almost the same direction: the maximum angle between them is less than $(k-1)(2\pi/(8k)) < \pi/4$. Hence

$$\left| \sum_{j \in A} M^j u_0 \right|^2 > \frac{k^2}{2}.$$

Since there are n arithmetic progressions in $\mathcal{H}$ with this difference, we get

$$\frac{1}{8kn} \sum_{A \in \mathcal{H}} \left| \sum_{j \in A} M^j u_0 \right|^2 > \frac{1}{8kn} \frac{k^2 n}{2} = \frac{k}{16} > \frac{n^{1/2}}{49},$$

as claimed.

6.6 Semidefinite programming in approximation algorithms

The algorithm of Goemans and Williamson, discussed in the introduction, was a breakthrough which showed that semidefinite optimization can lead to approximation algorithms with very good approximation ratio. Since then, many other applications have been developed; a couple of these are discussed below.

6.6.1 Stable sets, cliques, and chromatic number

The Sandwich Theorem 6.5.4 implies that $\vartheta(\overline{G})$ can be considered as an approximation of the clique size $\omega(G)$, which is at least as good as the natural upper bound $\chi(G)$. Note that both quantities $\omega(G)$ and $\chi(G)$ are NP-hard, but $\vartheta(\overline{G})$, which is "sandwiched" between them, is polynomial time computable.

The most important algorithmic consequence of theorem 6.5.4 is that *for perfect graphs, $\omega(G) = \chi(G)$ is polynomial time computable* [41]. Of course, by complementation it follows that $\alpha(G)$ is also polynomial time computable. It is not hard to see how to use this algorithm to compute a maximum stable set and (with more work) an optimum coloring. The surprising fact is that there is no algorithm known to find a maximum stable set in a perfect graph without the use of semidefinite optimization. (For another application of this result to complexity theory, see [90].)

How good an approximation does ϑ provide for α? Unfortunately, it can be quite bad. First, consider the case when α is very small. Koniagin [55] constructed a graph that has $\alpha(G) = 2$ and $\vartheta(G) = \Omega(n^{1/3})$. This is the largest $\vartheta(G)$ can be; in fact, Alon and Kahale [8], improving results of Kashin and Koniagin [54], proved that if $\alpha(G) \leq k$ then $\vartheta(G) < Cn^{(k-1)/(k+1)}$, for some absolute constant C.

Once α is unbounded, very little is true. Feige [32] showed that there are graphs for which $\alpha(G) = n^{o(1)}$ and $\vartheta(G) = n^{1-o(1)}$; in other words, ϑ/α can be larger than $n^{1-\varepsilon}$ for every $\varepsilon > 0$. (The existence of such graphs also follows from the results of Håstad [46] showing that it is NP-hard to determine $\alpha(G)$ with a relative error less than $n^{1-\varepsilon}$, where $n = |V|$.) By results of Szegedy [89], this also implies that $\vartheta(\overline{G})$ does not approximate the chromatic number within a factor of $n^{1-\varepsilon}$.

Let us consider the other end of the scale, when $\vartheta(\overline{G})$ is small. Suppose first that $\vartheta(\overline{G}) = 2$, then $\vartheta(\overline{G}) = 2$. Then it is not hard to see that G is bipartite, and hence perfect, and hence $\vartheta(G) = \alpha(G)$.

For the case when $\vartheta(\overline{G})$ is larger than 2 but bounded, the following (much weaker) positive result was proved by Karger, Motwani and Sudan [51]:

Theorem 6.6.1 *Let $k = \lceil \vartheta(\overline{G}) \rceil$, then $\alpha(G) \geq (1/2)n^{3/(k+1)}/\sqrt{\ln n}$. Furthermore, a stable set of this size can be found in randomized polynomial time.*

Note that we have $\vartheta(G) \geq n/k$ by Proposition 6.5.9. It is not known how large a stable set follows from the assumption $\vartheta(G) \geq n/k$.

Let us sketch the algorithm. If $k = 2$ then a stronger bound holds, as discussed above, so suppose that $k > 2$.

We first treat the case when the maximum degree of the graph is $\Delta > n^{k/(k+1)}$. Let G' be the subgraph induced by the neighbors of a node with maximum degree. It is easy to see that $\vartheta(G') \leq k-1$, and so (by induction on k) we can find in G' a stable set of size at least $\Delta^{3/k}/\sqrt{\ln \Delta} \geq n^{3/(k+1)}/\sqrt{\ln n}$.

So suppose that $\Delta \leq n^{k/(k+1)}$. Compute the optimum solution of (6.12) for the complementary graph $\overline{G}$, and the corresponding vector representation. Thus we get unit vectors $u_i \in \mathbb{R}^d$ such that for every edge $ij \in E$, we have $u_i^{\mathsf{T}} u_j = -1/(k-1)$.

Next, we take a random vector $w \in \mathbb{R}^d$ from the standard normal distribution in $\mathbb{R}^d$, and consider the set S of nodes i such that $w^{\mathsf{T}} u_i \geq c$, where $c = \sqrt{2(\ln n)(k-2)/k}$. The probability that a given node belongs to S is

$$\frac{1}{\sqrt{\pi}} \int_c^{\infty} e^{-t^2/2}\, dt \geq n^{-(k-2)/(k+1)}/\sqrt{\ln n},$$

and hence the expected size of S is at least $n^{3/(k+1)}/\sqrt{\ln n})$. On the other hand, the probability that both endpoints u_i and u_j of an edge belong to S can be estimated as follows:

$$\mathsf{P}(w^{\mathsf{T}} u_i \geq c,\ w^{\mathsf{T}} u_j \geq c) \leq \mathsf{P}(w^{\mathsf{T}}(u_i + u_j) \geq 2c).$$

The conditions on the vector solution imply that

$$|u_i + u_j| = \sqrt{2(k-2)/(k-1)},$$

and using this a more elaborate computation shows that the expected number of edges spanned by S is less than $|S|/2$. Hence we can delete at most half of the nodes of S and get a stable set of the desired size.

The previous algorithm has an important application to a coloring problem. Suppose that somebody gives a graph and guarantees that the graph is 3-colorable, without telling us its 3-coloring. Can we find this 3-coloring? (This may sound artificial, but this kind of situation does arise in cryptography and other data security applications; one can think of the hidden 3-coloring as a "watermark" that can be verified if we know where to look.)

It is easy to argue that knowing that the graph is 3-colorable does not help: it is still NP-hard to find the 3-coloration. But suppose that we would be satisfied with finding a 4-coloration, or 5-coloration, or $(\log n)$-coloration; is this easier? It is known that to find a 4-coloration is still NP-hard, but little is known above this. Improving earlier results, Karger, Motwani and Sudan [51] gave a polynomial time algorithm that, given a 3-colorable graph, computes a coloring with $O(n^{1/4}(\ln n)^{3/2})$ colors. More recently, this was improved by Blum and Karger [20] to $O(n^{3/14})$.

The algorithm of Karger, Motwani and Sudan starts with computing $\vartheta(\overline{G})$, which is at most 3 by Theorem 6.5.4. Using Theorem 6.6.1, they find a stable set of size $\Omega(n^{3/4}/\sqrt{\ln n})$. Deleting this set from G and iterating, they get a coloring of G with $O(n^{1/4}(\ln n)^{3/2})$ colors.

6.6.2 Satisfiability

One of the most fundamental problems in computer science is satisfiability. Let $x_1, \dots, x_n$ be Boolean variables. A *literal* is a variable x_i or the negation of a variable $\overline{x_i}$. A *clause* is a disjunction (OR) of literals; a *conjunctive normal form* is a conjunction (AND) of clauses. In standard logics notation, the following formula is an example of a conjunctive normal form:

$$(x_1 \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_4) \wedge (x_4 \vee \overline{x_5}) \wedge (x_2 \vee \overline{x_3} \vee x_5).$$

The Satisfiability Problem (SAT) is the problem of deciding whether there is an assignment of values 0 or 1 to the variables that satisfies a given conjunctive normal form. The restricted case when we assume that each clause in the input has at most k literals is called k-SAT (the example above is an instance of 3-SAT). k-SAT is polynomial time solvable by a rather easy algorithm if $k = 2$, but NP-hard if $k > 2$.

Suppose that the given conjunctive normal form is not satisfiable; then we may want to find an assignment that satisfies as many clauses as possible; this optimization problem is called the MAX-SAT problem (we could assign weights to the clauses, and try to maximize the total weight of satisfied clauses; but we keep our discussion simple by assuming that all clauses are equally valuable). The restricted case of MAX-k-SAT is defined in the natural way. MAX-k-SAT is NP-hard already when $k = 2$; indeed, it is easy to see that MAX CUT is a special case.

Can we extend the semidefinite programming method so successful for MAX CUT to obtain good approximation algorithms for MAX-k-SAT? This idea was exploited already by Goemans and Williamson [38], who showed how to obtain for MAX-2-SAT the same approximation ratio .878 as for the MAX CUT problem; this was improved by Feige and Goemans [34] to .931.

We do not survey all the developments for various versions of the Satisfiability Problem, only the case of MAX-3-SAT. An important special case will be *exact MAX-3-SAT*, when all clauses contain exactly 3 literals.

In the negative direction, Håstad [45] proved that for the exact MAX-3-SAT problem no polynomial time approximation algorithm can have an approximation ratio better than 7/8 (unless P=NP). This approximation ratio is easy to achieve, since if we randomly assign values to the variables, we can expect to satisfy 7/8-th of all clauses.

Can this optimal approximation ratio be achieved in the more general case of MAX-3-SAT (when the clauses may contain 1, 2 or 3 literals)? Of course, Håstad's negative result remains valid. Using semidefinite optimization, Karloff and Zwick [53] (cf. also [99]) showed that this bound can be attained:

Theorem 6.6.2 *There is a polynomial time approximation algorithm for MAX-3-SAT with an approximation ratio of* $7/8$*.*

Let us sketch this algorithm. First, we give a quadratic programming formulation. Let $x_1, \dots, x_n$ be the original variables, where we consider TRUE=1 and FALSE=0. Let $x_{n+i} = 1 - x_i$ $(i = n+1, \dots, 2n)$ be their negations. Let x_0 be a further variable needed for homogenization, which is set to $x_0 = 1$. We also introduce a variable $z_C \in 0, 1$ for the logical value of each clause C. Then we can relate z_C algebraically to the x_i as follows. For a clause $C = x_i$, we have $z_C = x_i$. For a clause $C = x_i \vee x_j$, we have $z_C = x_i + x_j - x_i x_j$. So far, this is all linear or quadratic, but clauses with 3 literals are a bit more difficult. If $C = x_i \vee x_j \vee x_k$, then clearly

$$z_C = x_i + x_j + x_k - x_i x_j - x_i x_k - x_j x_k + x_i x_j x_k.$$

unfortunately, this is cubic. We could get an upper bound on z_C if we omitted the last term, but as we will see, we need a lower bound. So we delete the cubic term and one of the quadratic terms; then we do get a lower bound. But which quadratic term should we delete? The trick is to create three inequalities, deleting one at a time:

$$\begin{aligned} z_C &\geq x_i + x_j + x_k - x_i x_j - x_i x_k \\ z_C &\geq x_i + x_j + x_k - x_i x_j - x_j x_k \\ z_C &\geq x_i + x_j + x_k - x_i x_k - x_j x_k \end{aligned}$$

Writing these expressions in a homogeneous form, we get the following optimization problem:

$$\begin{aligned} x_0 x_i + x_0 x_j + x_0 x_k - x_i x_j - x_i x_k \geq z_C &\qquad \forall \text{ clause } C = x_i \vee x_j \vee x_k \\ x_0 x_i + x_0 x_j - x_i x_j = z_C &\qquad \forall \text{ clause } C = x_i \vee x_j \\ x_i = z_C &\qquad \forall \text{ clause } C = x_i \qquad (6.31) \\ x_{n+i} = x_0 - x_i &\qquad \forall\, 1 \leq i \leq n, \\ x_i, z_C \in \{0, 1\}. & \end{aligned}$$

It is easy to see that every assignment of the variables x_i and the values z_C determined by them give a solution of this system, and vice versa. Thus the value M of the MAX-3-SAT problem is the maximum of $\sum_C z_C$, subject to (6.31).

Now we consider the semidefinite relaxation where we replace the x_i by unit vectors; the variables z_C are relaxed to real values satisfying $0 \leq z_C \leq 1$. Using semidefinite programming, this can be solved in polynomial time (with an arbitrarily small error, which causes some complications to be ignored here).

Next, similarly as in the Goemans–Williamson algorithm, we take a random hyperplane H through the point $(1/2)v_0$, and set $x_i = 1$ if x_i is separated from 0 by H, and $x_i = 0$ otherwise. A clause with at most 2 variables will be satisfied with probability at least $.878z_C > (7/8)z_C$ (which follows similarly as in the case of the Maximum Cut problem). A clause with 3 variables will be satisfied with probability at least $(7/8)z_C$ (this is quite a bit more difficult to show). Hence the expected number of clauses that are satisfied is at least

$$\sum_C \frac{7}{8}z_C = \frac{7}{8}M.$$

6.7 Constraint generation and quadratic inequalities

6.7.1 Example: the stable set polytope again

Recall the stable set polytope of a graph $G = (V, E)$ is the convex hull of integer solutions of the following system of linear inequalities:

$$x_i \geq 0 \qquad (\forall\ i \in V) \tag{6.32}$$

$$x_i + x_j \leq 1 \qquad (\forall\ ij \in E) \tag{6.33}$$

Without the integrality condition, however, this system describes the larger polytope FSTAB. We discussed above how to add new faces to get a sufficiently large set of inequalities for certain classes of graphs. The additional constraints were obtained by *ad hoc* combinatorial considerations. We show now that many of them (in fact, all those mentioned above) can also be derived by algebraic arguments ([71, 72]; see also [67]).

The trick is to go quadratic. As we have seen, the fact that the variables are 0-1 valued implies that for every node i,

$$x_i^2 = x_i, \tag{6.34}$$

and the fact that x is the incidence vector of a stable set can be expressed as

$$x_i x_j = 0 \qquad (ij \in E). \tag{6.35}$$

Now we can start deriving inequalities, using only (6.34) and (6.35). We have

$$x_i = x_i^2 \geq 0,$$

and

$$1 - x_i - x_j = 1 - x_i - x_j + x_i x_j = (1 - x_i)(1 - x_j) \geq 0, \tag{6.36}$$

so (6.32) and (6.33) follow. These are rather trivial, so let us consider the odd hole constraint associated with a pentagon $(1,2,3,4,5)$. Then we have

$$\begin{aligned}1-x_1-x_2-x_3+x_1x_3 &= 1-x_1-x_2-x_3+x_1x_2+x_1x_3\\ &= (1-x_1)(1-x_2-x_3)\geq 0,\end{aligned}$$

and similarly

$$1-x_1-x_4-x_5+x_1x_4\geq 0.$$

Furthermore,

$$x_1-x_1x_3-x_1x_4=x_1(1-x_3-x_4)\geq 0$$

Summing these inequalities, we get the odd hole constraint

$$2-x_1-x_2-x_3-x_4-x_5\geq 0. \tag{6.37}$$

One obtains all odd hole constraints in a similar way.

We can also derive the clique constraints. Assume that nodes 1,2,3,4,5 induce a complete 5-graph. Then

$$\begin{aligned}0 &\leq (1-x_1-x_2-x_3-x_4-x_5)^2 = 1+\sum_{i=1}^{5}x_i^2-2\sum_{i=1}^{5}x_i+2\sum_{i\neq j}x_ix_j\\ &= 1-x_1-x_2-x_3-x_4-x_5,\end{aligned}$$

by (6.34) and (6.35). All clique constraints, and in fact all orthogonality constraints can be derived similarly. Odd antihole constraints can be derived from the clique constraints in a way similar to the derivation of the odd hole constraints.

6.7.2 Strong insolvability of quadratic equations

We describe the procedures behind the computations in the previous section in a general context. We consider quadratic inequalities in n real variables $x_1,\dots,x_n$. Unfortunately, for quadratic inequalities there is no full analogue of the Farkas Lemma or of the efficient algorithms of linear programming. In fact, the system consisting of the quadratic equations (6.14) and (6.16), and a single linear equation $\sum_i x_i = k$ has a solution if and only if $\alpha(G)\geq k$. This reduction shows:

Proposition 6.7.1 *It is NP-hard to decide whether a system of quadratic inequalities has a real solution.*

However, using a semidefiniteness test for matrices, at least the case of a single inequality is solvable:

Proposition 6.7.2 *We can decide in polynomial time whether a single quadratic inequality is solvable. In fact, the quadratic polynomial*

$$q(x) = x^\mathsf{T} Ax + b^\mathsf{T} x + c$$

(where A is an $n \times n$ symmetric matrix, $b \in \mathbb{R}^n$ and $c \in \mathbb{R}$) is everywhere positive if and only if

(a) $A \succeq 0$,

(b) $b = Ah$ *for some* $h \in \mathbb{R}^n$, *and*

(c) *for this* h, $h^\mathsf{T} b < 4c$.

These conditions are easy to verify.

A system of quadratic inequalities is *strongly unsolvable* if there is a single unsolvable quadratic inequality that can be obtained as a linear combination of the given inequalities. By the Farkas Lemma, the analogous condition for the solvability of a system of linear inequalities is necessary and sufficient. In the quadratic case, there are unsolvable but not strongly unsolvable systems. A nice example is given by the quadratic equations (6.14) and (6.16), and the linear equation $\sum_i x_i = k$. As we noted, this system is unsolvable for $k > \alpha(G)$. However, it can be shown that it is strongly unsolvable only for $k > \theta(G)$. So if we take G to be the pentagon and $k = 2.1$, we get an unsolvable, but not strongly unsolvable system.

Using semidefinite optimization, we get a solution for a very special but important case:

Theorem 6.7.3 *It is decidable in polynomial time whether a system of quadratic inequalities is strongly unsolvable.*

6.7.3 Inference rules

An *inference rule* for algebraic inequalities is a procedure that, given a system $\alpha_1 \geq 0, \ldots, \alpha_m \geq 0$ of algebraic inequalities in n variables, determines a new algebraic inequality $\alpha \geq 0$, which is a *logical consequence* of the given system in the sense that every vector $x \in \mathbb{R}^n$ satisfying $\alpha_1(x) \geq 0, \ldots, \alpha_m(x) \geq 0$ also satisfies $\alpha(x) \geq 0$. Perhaps the simplest inference rule is the following.

LINEAR COMBINATION RULE:

$$\alpha_1 \geq 0, \ldots, \alpha_m \geq 0 \implies c_0 + c_1\alpha_1 + \ldots c_m\alpha_m \geq 0 \quad (c_0, c_1, \ldots, c_m \geq 0). \tag{6.38}$$

The Farkas Lemma asserts that among linear inequalities, this single rule generates *all* logical consequences. As we have mentioned, it is not sufficient once we have quadratic inequalities; however, in this case we can formulate other inference rules.

Multiplication rule:

$$\alpha_1 \geq 0,\ \alpha_2 \geq 0 \implies \alpha_1\alpha_2 \geq 0. \tag{6.39}$$

Assume that the linear inequalities $0 \leq x_i \leq 1$ as well as the quadratic equations $x_i^2 = x_i$ are present. Under this assumption, one can formulate the following
Restricted multiplication rule:

$$\alpha \geq 0 \implies x_i\alpha \geq 0,\ (1 - x_i)\alpha \geq 0. \tag{6.40}$$

The following rule will provide the connection with semidefinite optimization:
Square rule:

$$\alpha \geq 0 \implies \alpha + \beta_1^2 + \ldots + \beta_m^2 \geq 0 \tag{6.41}$$

(where the β_i are arbitrary polynomials). We can consider the Restricted square rule where all the β_i are linear.

Finally, let us formulate one other rule:
Division rule:

$$\alpha_1 \geq 0,\ (1 + \alpha_1)\alpha_2 \geq 0 \implies \alpha_2 \geq 0. \tag{6.42}$$

A further restriction is obtained when we are not allowed to use the commutativity of the variables. We'll only consider this in connection with the restricted multiplication and linear rules.

Artin's Theorem (see below) implies that these rules are sufficient to derive all consequences of a system of algebraic inequalities. In the case of interest for us, namely linear consequences of linear programs with 0-1 variables, we don't need all these rules to generate all the logical consequences of our starting system. In fact, the following is true [71, 72, 12]:

Theorem 6.7.4 *Starting with any system of linear inequalities and the equations $x_i^2 = x_i$, repeated application of the Linear rule and the Restricted multiplication rule (even with the further non-commutativity restriction) generates all linear inequalities valid for the 0-1 solutions, in at most n iterations.*

6.7.4 Deriving facets of the stable set polytope

Deriving a facet in n iterations (as guaranteed by Theorem 6.7.4) gives little information about it. We have seen in section 6.7.1 that the most important facets of the stable set polytope can be derived in just one or two iterations. It turns out that (for the stable set polytope) one can obtain reasonably good bounds on the number of iterations needed to derive a facet, in terms of other useful parameters.

Let $\sum_i a_i x_i \leq b$ be an inequality defining a facet of STAB(G); we assume that it is scaled so that the a_i are relatively prime integers. We define its *defect* as $\sum_i a_i - 2b$. The defect of an odd hole constraint is 1; the defect of a clique constraint (6.5) is $|B| - 2$. In the case of a facet defined by an α-critical graph G, this value is the *Gallai class number* $\delta(G) = |V(G)| - 2\alpha(G)$ of the graph.

Lemma 6.7.5 [72] *Let* $\sum_i a_i x_i \le b$ *be a facet of* $\mathrm{STAB}(G)$. *Then*

$$\max\left\{\sum_i a_i x_i : x \in \mathrm{FSTAB}(G)\right\} = \frac{1}{2}\sum_i a_i.$$

It follows that the defect is non-negative, and in fact it can be characterized as twice the *integrality gap* between optimizing over STAB and FSTAB:

Corollary 6.7.6 *The defect of a facet* $\sum_i a_i x_i \le b$ *satisfies*

$$\begin{aligned}\sum_i a_i - 2b \quad &= \quad 2\max\left\{\sum_i a_i x_i : x \in \mathrm{FSTAB}(G)\right\} \\ &- \quad 2\max\left\{\sum_i a_i x_i : x \in \mathrm{STAB}(G)\right\}.\end{aligned}$$

Graphs that are α-critical with bounded Gallai class number have a finite classification [63]. There is a similar classification of facets of $\mathrm{STAB}(G)$ with bounded defect [61].

The following theorem can be proved by calculations similar to those given in section 6.7.1 above.

Theorem 6.7.7 [71, 72] *Let G any graph, and let F be a facet of* $\mathrm{STAB}(G)$, *defined by the inequality* $\sum_i a_i x_i \le b$, *with defect* δ.

(a) *Starting with the non-negativity constraints* (6.3) *and the edge constraints* (6.4), *the facet F can be derived, using the Linear and Restricted Multiplication rules, in at most δ steps.*

(b) *Starting with the non-negativity constraints* (6.3) *and the edge constraints* (6.4), *the facet F can be derived, using the Linear, Restricted Multiplication, and Restricted Square rules, in at most b steps.*

If we also use the square rule, then the derivation may be much faster. For example, to derive a k-clique constraint using the Linear and Restricted multiplication rules takes $k - 2$ steps; with the Restricted square rule, it takes only one. It seems that all the known “nice” (polynomially separable, see below) classes of facets of the stable set polytope, with the exception of the ”Edmonds facets” in the case of the matching polytope, can be derived by one or two rounds of applications of the Linear, Restricted Multiplication, and Square Rules.

6.7.5 A bit of real algebraic geometry

Finally, let us put these considerations into a more general context. A fundamental theorem in real algebraic geometry is Artin's Theorem:

Theorem 6.7.8 *A polynomial $f \in \mathbb{R}[x_1, \dots, x_n]$ is nonnegative for all $(x_1, \dots, x_n) \in \mathbb{R}^n$ if and only if it is a sum of squares of rational functions.*

One might expect that the term "rational functions" can be replaced by "polynomials", but this cannot be guaranteed in general. In special cases of combinatorial interest, however, we do get a simpler representation.

Let $G = (V, E)$ be a graph and let $I(G)$ denote the polynomial ideal generated by the polynomials $x_i^2 - x_i$ $(i \in V)$ and $x_i x_j$ $(ij \in E)$. Obviously, the roots of this ideal are the incidence vectors of stable sets. We write $f \geq 0$ $(\mathrm{mod}\ I(G))$ iff $f(x) \geq 0$ for every root of the ideal $I(G)$.

Proposition 6.7.9 *For any polynomial f, we have $f \geq 0$ $(\mathrm{mod}\ I(G))$ iff there exist polynomials $g_1, \dots, g_N$ such that $f \equiv g_1^2 + \dots + g_N^2$ $(\mathrm{mod}\ I(G))$.*

From theorem 6.5.13 it is easy to derive the following characterization of perfect graphs:

Theorem 6.7.10 *A graph G is perfect if and only if the following holds: For any linear polynomial f, we have $f \geq 0$ $(\mathrm{mod}\ I(G))$ iff there exist linear polynomials $g_1, \dots, g_N$ such that $f \equiv g_1^2 + \dots + g_N^2$ $(\mathrm{mod}\ I(G))$.*

6.7.6 Algorithmic aspects of inference rules

Let $\mathcal{L}$ be a possibly infinite system of linear inequalities in n variables, associated to a finite structure (e.g., a graph). We say that $\mathcal{L}$ is *polynomially separable*, if for every vector $x \in \mathbb{R}^n$, we can decide in polynomial time whether x satisfies every member of $\mathcal{L}$, and if it does not, we can find a violated member.

Let $\mathbf{R}$ be any inference rule, and let $\mathbf{R}\mathcal{L}$ denote the set of all linear inequalities produced by one application of $\mathbf{R}$ to members of $\mathcal{L}$. We say that the rule is *polynomial*, if $\mathbf{R}\mathcal{L}$ is polynomially separable whenever $\mathcal{L}$ is.

Using the ellipsoid method combined with semidefinite optimization, we get:

Lemma 6.7.11 *The Linear Rule* (6.38)*, the Restricted Multiplication Rule* (6.40) *and the Restricted Square Rule* (6.41) *are polynomial.*

It follows that if for some class of graphs, all facets of the stable set polytope can be derived by a bounded number of "rounds" of these three rules, then the stable set problem is polynomial for the class. In particular, we have the following consequences [42, 71, 72].

Corollary 6.7.12 *The Stable Set Problem can be solved for perfect, t-perfect and h-perfect graphs in polynomial time.*

Corollary 6.7.13 *Assume that for a class of graphs either the right hand side or the defect of each facet of the stable set polytope is bounded. Then the Stable Set Problem can be solved polynomially for this class.*

6.8 Extensions and problems

6.8.1 Small dimension representations and rank minimization

If we consider a semidefinite relaxation of a discrete optimization problem (say, a 0-1 linear program), then typically the original solutions correspond to semidefinite matrices of rank 1. In linear programming, there are special but useful conditions that guarantee that the solutions of the relaxed linear problem are also solutions of the original integer problem (for example, perfectness, or total unimodularity).

Problem 6.8.1 *Find combinatorial conditions that guarantee that the semidefinite relaxation has a solution of rank 1.*

This question can be interesting for special combinatorial semidefinite relaxations. For example,

Problem 6.8.2 *Which graphs are "max-cut-perfect?"*

Theorem 6.7.10 suggests an algebraic question:

Problem 6.8.3 *Which polynomial ideals I are "perfect" in the sense that for any linear polynomial f, we have $f \geq 0 \pmod{I}$ iff there exist linear polynomials $g_1, \dots, g_N$ such that $f \equiv g_1^2 + \dots + g_N^2 \pmod{I}$? Of course, there is a lot of room to modify the question by replacing "linear" with "bounded degree", etc.*

Coming back to semidefinite programs, if we find a solution that has, instead of rank 1, some other small rank, (i.e., a vector solution in low dimension), then this may decrease the error of the rounding methods, used to extract approximate solutions to the original problems. Thus the version of problem 6.8.1 with "low rank" instead of "rank 1" also seems very interesting. One result in this direction is the following (discovered in many versions [14, 36, 80, 59]; see also [27], section 31.5, and [15]):

Theorem 6.8.4 *The semidefinite system*

$$X \succeq 0$$

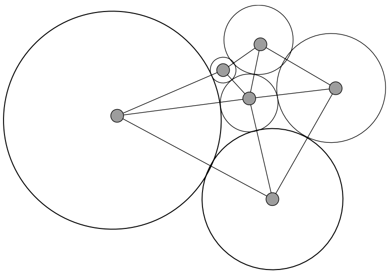


Figure 6.10. Representing a planar graph by touching circles

$$\begin{aligned} D_1 \cdot X &= d_1 \\ &\vdots \\ D_k \cdot X &= d_k, \end{aligned}$$

has a solution of rank at most $\lceil \sqrt{2k} \rceil$.

Also from a geometric point of view, it is natural to consider unit distance (orthogonal, etc.) representations in a fixed small dimension. Without control over the rank of the solutions of semidefinite programs, this additional condition makes the use of semidefinite optimization methods very limited. On the other hand, several of these geometric representations of graphs are connected to interesting graph-theoretic properties, and some of them are related to semidefinite optimization. This connection is largely unexplored.

Let us mention a few examples where we do have some information about low rank solutions. A vector labeling $V \to \mathbb{R}^d$ is *generic* if any d labels are linearly independent. Let $\kappa(G)$ denote the node-connectivity of G. The following was proved in [69] (see also [70]):

Theorem 6.8.5 *The minimum dimension in which a graph G has a generic orthogonal representation is $n - \kappa(G)$.*

In other words, the smallest d for which the semidefinite constraints

$$\begin{aligned} Y &\succeq 0 \\ Y_{ij} &= 0 \qquad \forall\, ij \notin E,\ i \neq j \end{aligned}$$

have a solution of rank d such that every $d \times d$ subdeterminant is non-zero, is exactly $n - \kappa(G)$.

A classical result of Koebe [58] (see also [9, 91, 86], asserts that every planar graph can be represented in the plane by touching circular disks (Figure 6.10. One of the many extensions of this theorem characterizes triangulations of the plane that have a representation by orthogonal circles: more exactly, circles representing adjacent nodes must intersect at 90°, other pairs, at > 90° (i.e., their centers must be farther apart) [9, 91, 56] (Figure 6.11.

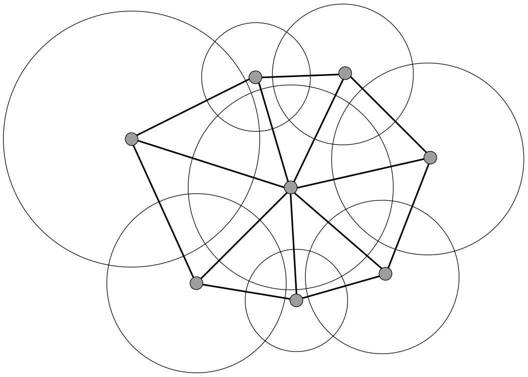


Figure 6.11. Representing a planar graph by orthogonal circles

Such a representation, if it exists, can be projected to a representation by orthogonal circles on the unit sphere; with a little care, one can do the projection so that each disk bounded by one of the circles is mapped onto a "cap" which covers less than half of the sphere. Then each cap has a unique *pole*: the point in space from which the part of the sphere you see is exactly the given cap. The key observation is that *two circles are orthogonal if and only if the corresponding poles have inner product* 1 (Figure 6.12). This translates a representation with orthogonal circles into a representation by vectors of length larger than 1, where adjacent nodes are represented by vectors with inner product 1, non-adjacent nodes by vectors with inner product less than 1.

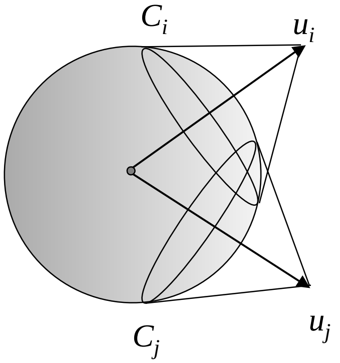


Figure 6.12. Poles of circles

This in turn can be translated into semidefinite matrices. We only state the final result of these transformations. Consider the following two sets of semidefinite constraints:

$$\begin{aligned} Y &\succeq 0 \\ Y_{ij} &= 1 \qquad \forall\, ij \in E, \\ Y_{ij} &< 1 \qquad \forall\, ij \notin E,\ i \neq j, \\ Y_{ii} &> 1 \end{aligned} \tag{6.43}$$

and the weaker set of constraints

$$\begin{aligned} Y &\succeq 0 \\ Y_{ij} &= 1 \qquad \forall\, ij \in E, \\ Y_{ij} &< 1 \qquad \forall\, ij \notin E,\ i \neq j, \end{aligned} \tag{6.44}$$

$$\tag{6.45}$$

To formulate the theorem, we need two simple definitions. A cycle C in a graph G is called *separating*, if $G \setminus V(C)$ has at least two connected components, where any chord of C is counted as a connected component here. The cycle C is called *strongly separating*, if $G \setminus V(C)$ has at least two connected components, each of which has at least 2 nodes. If G is a 3-connected planar map, then its non-separating cycles are exactly the boundaries of the faces.

Theorem 6.8.6 *Let G be a 3-connected graph*

(a) *If (6.44) has a solution of rank 3, then G is planar.*

(b) *Assume that G is a maximal planar graph. Then (6.43) has a solution of rank 3 if and only if G has no separating 3- and 4-cycles.*

(c) *Assume that G is a maximal planar graph. Then (6.44) has a solution with rank 3 if and only if G has no strongly separating 3- and 4-cycles.*

Colin de Verdière [24] introduced an interesting spectral invariant of graphs that is related to topological properties. Kotlov, Lovász and Vempala [56] showed that this invariant can be defined in terms of the minimum rank of a "non-degenerate" solution of (6.44) (see [3] for the definition and theory of non-degeneracy in semidefinite programs).

Tutte [92] constructed a straight-line embedding in the plane of a 3-connected planar graph by fixing the vertices of a face to the vertices of a convex polygon, replacing the edges by "rubber bands", and letting the other nodes find their equilibrium (Figure 6.13). A similar construction was used in [60] to characterize k-connectivity of a graph, and to design an efficient randomized k-connectivity test. There is an obvious similarity with our description of the Goemans-Williamson algorithm in the introduction, and we could obtain the equilibrium situation through a semidefinite program. But in Tutte's case the sum of squares of edge lengths is to be minimized, rather than maximized; since this function is concave, this makes a substantially better behaved optimization problem, which can be solved

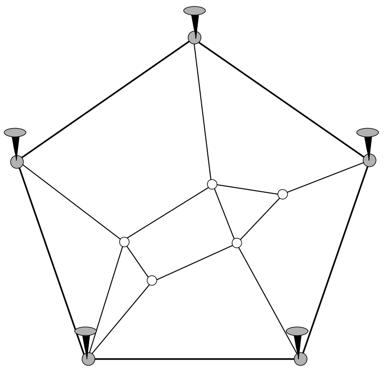


Figure 6.13. Tutte's "rubber band" representation of planar graphs

efficiently in every fixed dimension. What is important for us, however, is that this is an example of a semidefinite program whose solution has fixed small rank.

Rubber band problems form a special class of semidefinite optimization problems which can be solved by direct means. Further such problems are described in [95]. It would be interesting to understand the structure of such special classes.

A final remark: many problems in graph theory, matroid theory, electrical engineering, statics etc. can be formulated as *maximizing* the rank of a matrix subject to linear constraints (see [84, 66]). Such problems can be solved by an obvious polynomial time randomized algorithm, by substituting random numbers for the variables. Unlike in the case of the randomized algorithms described above for the Max Cut and other problems, it is not known whether these rank maximization problems can be solved in deterministic polynomial time.

6.8.2 Approximation algorithms

The most important open question is: can the randomized "rounding" method of Goemans–Williamson and Karger–Motwani–Sudan be generalized to semidefinite relaxations of more general problems? Can other, different rounding techniques be found?

There are many candidate problems, the most interesting is the "class of the factor 2". We have seen that the Maximum Cut problem has a trivial factor 2 approximation algorithm. There are several other such optimization problems; here are three very fundamental examples:

The Node Cover problem: given a graph G, find a minimum set of nodes covering all edges.

The Acyclic Subgraph problem: given a directed graph, find the maximum number of edges that form no directed cycle.

The Overdetermined Binary Equations problem: given a system of linear equations over GF(2), find an assignment of the variables that satisfies as many of them as possible.

We leave it to the reader to find the easy algorithms that give suboptimal solutions off by a factor of 2 or less. In all cases it is known that we cannot bring this error factor arbitrarily close to 1.

Problem 6.8.7 Can we do better than the trivial factor of 2?

In the case of the Maximum Cut problem, we saw that the answer is positive. Surprisingly, for the Overdetermined Binary Equations problem (which is in fact a generalization of the Maximum Cut problem) Håstad [45] showed that the answer is negative: the factor of 2 is optimal. For the Node Cover and Acyclic Subgraph problems the question is open. The most promising technique to attack these questions is semidefinite optimization, even though the attempts by many have not been successful so far.

There are many open questions about approximating the stability number (or equivalently, the largest clique), and the chromatic number (whether or not semidefinite optimization can be used in answering these is not clear):

Problem 6.8.8 Can the ratio ϑ/α be estimated by $n^{1-\varepsilon}$ for special classes of graphs? Are there interesting classes of graphs for which the ϑ can be bounded by some function (or small function) of α?

Problem 6.8.9 Can $\alpha(G)$ be approximated better than the error factor $n/(\log n)^2$ (this is achieved in [21]).

Problem 6.8.10 Is there a polynomial time algorithm that outputs an upper bound $\phi(G)$ for $\alpha(G)$ such that there is a function $f : \mathbb{Z}_+ \to \mathbb{Z}_+$ with $\phi(G) < f(\alpha(G))$ (f is independent of the size of the graph)?

Problem 6.8.11 Is is true that for every $\varepsilon > 0$ there exists an algorithm that computes $\alpha(G)$ in time $O((1+\varepsilon)^n)$?

Problem 6.8.12 Suppose that G is a graph with chromatic number 3. Can G be k-colored in polynomial time, where (a) $k = n^{o(1)}$; (b) $k = \log n$; (c) $k = O(1)$?

6.8.3 Inference rules

We discussed strong insolvability of systems of quadratic equations. Barvinok [13] gives a polynomial time algorithm to decide whether a system of a bounded number of quadratic equations is solvable (over the real field). This suggests a hierarchy of extensions of strong insolvability: produce a fixed number k of quadratic equations by linear combination which are collectively unsolvable.

Problem 6.8.13 Can one decide in polynomial time the k-th version of strong insolvability? Is this a real hierarchy? Are there any natural problems in higher classes?

Problem 6.8.14 Are the multiplication rule (6.39) and the division rule (6.42) polynomial? Are they polynomial if we restrict ourselves to quadratic inequalities? If not, does the division rule have a natural and useful restriction that is polynomial?

Problem 6.8.15 Are there other combinatorial optimization problems for which interesting classes of facets can be derived using the division rule?

Problem 6.8.16 Are there other inference rules that are worth considering? Can any interesting discrete programming problem be attacked using polynomials of higher degree?

Problem 6.8.17 How to implement the restricted multiplication rule (6.40) efficiently? Is there a way to use interior point methods, in a way parallel to Alizadeh's application of interior point methods to semidefinite programming?

Problem 6.8.18 If a graph G contains no subdivision of K_4, then it is series-parallel, and hence t-perfect [22]. This means that every facet of STAB(G) has defect at most 1. Is there an analogous simple graph-theoretic condition that guarantees that every facet has defect at most 2, 3, etc.?

Acknowledgement. My thanks are due to András Frank and to Bruce Reed for organizing two series of talks on this subject. I am particularly indebted to Miklós Újváry for pointing out several errors in an earlier version of these notes, and to Wayne Barrett and the anonymous referees of this version for suggesting many improvements.

References

[1] F. Alizadeh: Combinatorial optimization with semi-definite matrices, in: *Integer Programming and Combinatorial Optimization* (Proceedings of IPCO '92), (eds. E. Balas, G. Cornuéjols and R. Kannan), Carnegie Mellon University Printing (1992), 385–405.

[2] F. Alizadeh, Interior point methods in semidefinite programming with applications to combinatorial optimization, *SIAM J. Optim.* **5** (1995), 13–51.

[3] F. Alizadeh, J.-P. Haeberly, and M. Overton: Complementarity and nondegeneracy in semidefinite programming, in: *Semidefinite Programming*, *Math. Programming* Ser. B, **77** (1997), 111–128.

[4] N. Alon, R. A. Duke, H. Lefmann, V. Rödl and R. Yuster: The algorithmic aspects of the Regularity Lemma, *Proc. 33rd Annual Symp. on Found. of Computer Science*, IEEE Computer Society Press (1992), 473–481.

[5] N. Alon and J.H. Spencer: *The Probabilistic Method*, Wiley, New York, 1992.

[6] N. Alon, The Shannon capacity of a union, *Combinatorica* **18** (1998), 301–310.

[7] N. Alon: Explicit Ramsey graphs and orthonormal labelings, *The Electronic Journal of Combinatorics* **1** (1994), 8pp.

[8] N. Alon and N. Kahale: Approximating the independence number via the ϑ-function, *Math. Programming* **80** (1998), Ser. A, 253–264.

[9] E. Andre'ev, On convex polyhedra in Lobachevsky spaces, *Mat. Sbornik*, Nov. Ser. **81** (1970), 445–478.

[10] S. Arora, C. Lund, R. Motwani, M. Sudan, M. Szegedy: Proof verification and hardness of approximation problems *Proc. 33rd FOCS* (1992), 14–23.

[11] R. Bacher and Y. Colin de Verdière, Multiplicités de valeurs propres et transformations étoile-triangle des graphes, *Bull. Soc. Math. France* **123** (1995), 101-117.

[12] E. Balas, S. Ceria and G. Cornuéjols, A lift-and-project cutting plane algorithm for mixed $0-1$ programs, *Mathematical Programming* **58** (1993), 295–324.

[13] A.I. Barvinok: Feasibility testing for systems of real quadratic equations, *Discrete and Comp. Geometry* **10** (1993), 1–13.

[14] A.I. Barvinok: Problems of distance geometry and convex properties of quadratic maps, *Discrete and Comp. Geometry* **13** (1995), 189–202.

[15] A.I. Barvinok: A remark on the rank of positive semidefinite matrices subject to affine constraints, *Discrete and Comp. Geometry* **25** (2001), 23–31.

[16] J. Beck: Roth's estimate on the discrepancy of integer sequences is nearly sharp, *Combinatorica* **1** (1981) 327–335.

[17] J. Beck and W. Chen: *Irregularities of Distribution,* Cambridge Univ. Press (1987).

[18] J. Beck and V.T. Sós: Discrepancy Theory, Chapter 26 in: *Handbook of Combinatorics* (ed. R.L. Graham, M. Grötschel and L. Lovász), North-Holland, Amsterdam (1995).

[19] M. Bellare, O. Goldreich, M. Sudan: Free bits, PCPs and non-approximability — towards tight results, *Proc. 36th FOCS* (1996), 422–431.

[20] A. Blum and D. Karger: An $O(n^{3/14})$-coloring for 3-colorable graphs, *Inform. Process. Lett.* **61** (1997), 49–53.

[21] R. Boppana and M. Haldórsson: Approximating maximum independent sets by excluding subgraps, *BIT* **32** (1992), 180–196.

[22] M. Boulala and J.-P. Uhry: Polytope des indépendants d'un graphe série-parallèle, *Discrete Math.* **27** (1979), 225–243.

[23] V. Chvátal: On certain polytopes associated with graphs, *J. of Combinatorial Theory (B)* **18** (1975), 138–154.

[24] Y. Colin de Verdière, Sur la multiplicité de la première valeur propre non nulle du laplacien, *Comment. Math. Helv.* **61** (1986), 254–270.

[25] Y. Colin de Verdière, Sur un novel invariant des graphes at un critère de planarité, *J. Combin. Theory B* **50** (1990) 11–21.

[26] Y. Colin de Verdière, On a new graph invariant and a criterion for planarity, in: *Graph Structure Theory* (Robertson and P. D. Seymour, eds.), Contemporary Mathematics, Amer. Math. Soc., Providence, RI (1993), 137–147.

[27] M. Deza and M. Laurent: *Geometry of Cuts and Metrics*, Springer Verlag, 1997.

[28] C. Delorme and S. Poljak: Combinatorial properties and the complexity of max-cut approximations, *Europ. J. Combin.* **14** (1993), 313–333.

[29] C. Delorme and S. Poljak: Laplacian eigenvalues and the maximum cut problem, *Math. Programming* **62** (1993)

[30] P. Erdős: Gráfok páros körüljárású részgráfjairól (On bipartite subgraphs of graphs, in Hungarian), *Mat. Lapok* **18** (1967), 283–288.

[31] P. Erdős, F. Harary and W.T. Tutte, On the dimension of a graph *Mathematika* **12** (1965), 118–122.

[32] U. Feige: Randomized graph products, chromatic numbers, and the Lovász ϑ-function, *Combinatorica* **17** (1997), 79–90.

[33] U. Feige: Approximating the Bandwidth via Volume Respecting Embeddings, Tech. Report CS98-03, Weizmann Institute (1998).

[34] U. Feige and M. Goemans, Approximating the value of two-prover proof systems, with applications to MAX-2SAT and MAX-DICUT, in: Proc. 3rd Israel Symp. on Theory and Comp. Sys., Tel Aviv, Isr. (1995), 182–189.

[35] U. Feige and L. Lovász: Two-prover one-round proof systems: Their power and their problems. *Proc. 24th ACM Symp. on Theory of Computing* (1992), 733-744.

[36] S. Friedland and R. Loewy, Subspaces of symmetric matrices containing matrices with multiple first eigenvalue, *Pacific J. Math.* **62** (1976), 389–399.

[37] M. X. Goemans and D. P. Williamson: .878-Approximation algorithms for MAX CUT and MAX 2SAT, *Proc. 26th ACM Symp. on Theory of Computing* (1994), 422-431.

[38] M. X. Goemans and D. P. Williamson: Improved approximation algorithms for maximum cut and satisfiablity problems using semidefinite programming, *J. ACM* **42** (1995), 1115–1145.

[39] M. C. Golumbic: *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York (1980).

[40] M. Grötschel, L. Lovász and A. Schrijver: The ellipsoid method and its consequences in combinatorial optimization, *Combinatorica* **1** (1981), 169-197.

[41] M. Grötschel, L. Lovász and A. Schrijver: Polynomial algorithms for perfect graphs, *Annals of Discrete Math.* **21** (1984), 325-256.

[42] M. Grötschel, L. Lovász and A. Schrijver: Relaxations of vertex packing, *J. Combin. Theory B* **40** (1986), 330-343.

[43] M. Grötschel, L. Lovász and A. Schrijver: *Geometric Algorithms and Combinatorial Optimization*, Springer, Heidelberg, 1988.

[44] W. Haemers: On some problems of Lovász concerning the Shannon capacity of a graph, *IEEE Trans. Inform. Theory* **25** (1979), 231–232.

[45] J. Håstad: Some optimal in-approximability results, *Proc. 29th ACM Symp. on Theory of Comp.*, 1997, 1–10.

[46] J. Håstad: Clique is hard to approximate within a factor of $n^{1-\varepsilon}$, *Acta Math.* **182** (1999), 105–142.

[47] H. van der Holst, A short proof of the planarity characterization of Colin de Verdière, Preprint, CWI Amsterdam, 1994.

[48] H. van der Holst, L. Lovász and A. Schrijver: On the invariance of Colin de Verdière's graph parameter under clique sums, *Linear Algebra and its Applications*, **226–228** (1995), 509–518.

[49] F. Juhász: The asymptotic behaviour of Lovász' ϑ function for random graphs, *Combinatorica* **2** (1982) 153–155.

[50] N. Kahale: A semidefinite bound for mixing rates of Markov chains, DIMACS Tech. Report No. 95-41.

[51] D. Karger, R. Motwani, M. Sudan: Approximate graph coloring by semidefinite programming, *Proc. 35th FOCS* (1994), 2–13; full version: *J. ACM* **45** (1998), 246–265.

[52] H. Karloff: How good is the Goemans-Williamson MAX CUT algorithm? *SIAM J. Comput.* **29** (1999), 336–350.

[53] H. Karloff and U. Zwick: A 7/8-approximation algorithm for MAX 3SAT? in: *Proc. of the 38th Ann. IEEE Symp. in Found. of Comp. Sci.* (1997), 406–415.

[54] B. S. Kashin and S. V. Konyagin: On systems of vectors in Hilbert spaces, *Trudy Mat. Inst. V.A.Steklova* **157** (1981), 64–67; English translation: *Proc. of the Steklov Inst. of Math.* (AMS 1983), 67–70.

[55] V. S. Konyagin, Systems of vectors in Euclidean space and an extremal problem for polynomials, *Mat. Zametky* **29** (1981), 63–74. English translation: *Math. Notes of the Academy USSR* **29** (1981), 33–39.

[56] A. Kotlov, L. Lovász, S. Vempala, The Colin de Verdière number and sphere representations of a graph, *Combinatorica* **17** (1997) 483–521.

[57] D. E. Knuth: The sandwich theorem, *The Electronic Journal of Combinatorics* **1** (1994) 48 pp.

[58] P. Koebe: Kontaktprobleme der konformen Abbildung, *Berichte uber die Verhandlungen d. Sächs. Akad. d. Wiss.*, Math.–Phys. Klasse, **88** (1936) 141–164.

[59] M. Laurent and S. Poljak: On the facial structure of the set of correlation matrices, *SIAM J. on Matrix Analysis and Applications* **17** (1996), 530–547.

[60] N. Linial, L. Lovász, A. Wigderson: Rubber bands, convex embeddings, and graph connectivity, *Combinatorica* **8** (1988), 91–102.

[61] L. Lipták, L. Lovász: Facets with fixed defect of the stable set polytope, *Math. Programming*, Series A **88** (2000), 33–44.

[62] L. Lovász: Normal hypergraphs and the perfect graph conjecture, *Discrete Math.* **2** (1972), 253-267.

[63] L. Lovász: Some finite basis theorems in graph theory, in: *Combinatorics*, Coll. Math. Soc. J. Bolyai **18** (1978), 717-729.

[64] L. Lovász: On the Shannon capacity of graphs, *IEEE Trans. Inform. Theory* **25** (1979), 1–7.

[65] L. Lovász: Perfect graphs, in: *More Selected Topics in Graph Theory* (ed. L. W. Beineke, R. L. Wilson), Academic Press (1983), 55-67.

[66] L. Lovász: Singular spaces of matrices and their applications in combinatorics, *Bol. Soc. Braz. Mat.* **20** (1989), 87–99.

[67] L. Lovász: Stable sets and polynomials, *Discrete Math.* **124** (1994), 137–153.

[68] L. Lovász: Integer sequences and semidefinite programming *Publ. Math. Debrecen* **56** (2000) 475–479.

[69] L. Lovász, M. Saks and A. Schrijver: Orthogonal representations and connectivity of graphs, *Linear Alg. Appl.* **114/115** (1989), 439–454.

[70] L. Lovász, M. Saks and A. Schrijver: A correction: orthogonal representations and connectivity of graphs (with M. Saks and A. Schrijver) *Linear Algebra Appl.* **313** (2000) 101–105.

[71] L. Lovász and A. Schrijver: Cones of matrices and set-functions, and 0-1 optimization, *SIAM J. on Optimization* **1** (1990), 166-190.

[72] L. Lovász and A. Schrijver: Matrix cones, projection representations, and stable set polyhedra, in: *Polyhedral Combinatorics*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science I, Amer. Math. Soc., Providence (1990), 1–17.

[73] L. Lovász and K. Vesztergombi: Geometric representations of graphs, in: *Paul Erdős and his Mathematics*

[74] J. Matoušek and J. Spencer, Discrepancy in arithmetic progressions, *J. Amer. Math. Soc.* **9** (1996) 195–204.

[75] B. Mohar and S. Poljak: Eigenvalues and the max-cut problem, *Czechoslovak Mathematical Journal* **40** (1990), 343–352.

[76] Yu. E. Nesterov and A. Nemirovsky: *Interior-point polynomial methods in convex programming,* Studies in Appl. Math. **13**, SIAM, Philadelphia, 1994.

[77] M. L. Overton: On minimizing the maximum eigenvalue of a symmetric matrix, *SIAM J. on Matrix Analysis and Appl.* **9** (1988), 256–268.

[78] M. L. Overton and R. Womersley: On the sum of the largest eigenvalues of a symmetric matrix, *SIAM J. on Matrix Analysis and Appl.* **13** (1992), 41–45.

[79] M. Padberg: *Linear optimization and extensions.* Second, revised and expanded edition, Algorithms and Combinatorics **12**, Springer-Verlag, Berlin, 1999.

[80] G. Pataki: On the rank of extreme matrices in semidefinite programs and the multiplicity of optimal eigenvalues, *Math. of Oper. Res.* **23** (1998), 339–358.

[81] S. Poljak and F. Rendl: Nonpolyhedral relaxations of graph-bisection problems, DIMACS Tech. Report 92-55 (1992).

[82] L. Porkoláb and L. Khachiyan: On the complexity of semidefinite programs, *J. Global Optim.* **10** (1997), 351–365.

[83] M. Ramana: An exact duality theory for semidefinite programming and its complexity implications, in: *Semidefinite programming. Math. Programming* Ser. B, **77** (1997), 129–162.

[84] A. Recski: *Matroid Theory and its Applications in Electric Network Theory and Statics*, Akadémiai Kiadó–Springer-Verlag (1989).

[85] K.F. Roth: Remark concerning integer sequences, *Acta Arith.* **35**, 257–260.

[86] O. Schramm: How to cage an egg, *Invent. Math.* **107** (1992), 543–560.

[87] H.D. Sherali and W.P. Adams (1990): A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems, *SIAM J. on Discrete Math.* bf 3, 411–430.

[88] G. Strang: *Linear algebra and its applications*, Second edition, Academic Press, New York–London, 1980.

[89] M. Szegedy: A note on the θ number of Lovász and the generalized Delsarte bound, *Proc. 35th FOCS* (1994), 36–39.

[90] É. Tardos: The gap between monotone and non-monotone circuit complexity is exponential, *Combinatorica* **8** (1988), 141–142.

[91] W. Thurston: *The Geometry and Topology of Three-manifolds*, Princeton Lecture Notes, Chapter 13, Princeton, 1985.

[92] W.T. Tutte: How to draw a graph, *Proc. London Math. Soc.* **13** (1963), 743-768.

[93] L. Vandeberghe and S. Boyd: Semidefinite programming, in: *Math. Programming: State of the Art* (ed. J. R. Birge and K. G. Murty), Univ. of Michigan, 1994.

[94] L. Vandeberghe and S. Boyd: Semidefinite programming. SIAM Rev. 38 (1996), no. 1, 49–95.

[95] R.J. Vanderbei and B. Yang: The simplest semidefinite programs are trivial, *Math. of Oper. Res.* **20** (1995), no. 3, 590–596.

[96] H. Wolkowitz: Some applications of optimization in matrix theory, *Linear Algebra and its Applications* **40** (1981), 101–118.

[97] H. Wolkowicz: Explicit solutions for interval semidefinite linear programs, *Linear Algebra Appl.* **236** (1996), 95–104.

[98] H. Wolkowicz, R. Saigal and L. Vandenberghe: *Handbook of semidefinite programming. Theory, algorithms, and applications.* Int. Ser. Oper. Res. & Man. Sci., **27** (2000) Kluwer Academic Publishers, Boston, MA.

[99] U. Zwick: Outward rotations: a tool for rounding solutions of semidefinite programming relaxations, with applications to MAX CUT and other problems, *Proc. 31th STOC* (1999), 679–687.

7 Approximability of NP-Optimization Problems

A. Steger

7.1 Introduction

Probably every computer science student knows the short comics story from the introduction of the fundamental textbook by Garey and Johnson [35]. It indicates in a pictorial way why an a priori seemingly theoretical concept as the notion of NP-completeness has been so successful. Till today, the first attempt of every student, researcher, algorithm designer with a new problem for which he can't find a polynomial-time algorithm immediately is to try proving that it is NP-complete.

However, from a practical point of view this is often not sufficient. A proof that a problem is NP-complete just tells that we should not look for a polynomial-time algorithm that solves the problem optimally. But what if we relax this optimality constraint a bit and just aim at finding a "nearly" optimal solution? For many applications this might still be sufficient. As a theoretician we are thus asked to investigate the trade-off between the quality of approximate solutions and the necessary running time of the algorithm. Unfortunately, this turns out to be rather difficult.

Approximation algorithms were first considered even before the theory of NP-completeness was established [38]. In the following years work by several people [31, 33, 43, 59] formalized the notion of approximation algorithms and provided the first non-approximability results, which are based on the assumption $\mathcal{P} \neq \mathcal{NP}$. But all in all the achieved results remained more or less singular events and were out-shined by the success of NP-completeness. A main reason was that the techniques for proving upper bounds, that is, for actually constructing approximation algorithms became more and more sophisticated, but there were basically no techniques available for proving lower bounds. In particular, one was missing an analogue to the comparison techniques between problems according to the motto "this problem is as difficult as this one, which is already accepted to be difficult", which made the theory of NP-completeness so successful. A first major progress in this direction was the introduction of L-reductions and MaxSNP-completeness by Papadimitriou and Yannakakis [55]. The next key step was achieved by Condon [21] and Feige, Goldwasser, Lovász, Safra, Szegedy [29] who linked the proof of hardness results of some optimization problems to the existence of certain interactive proofs. Thereby establishing a connection between computational complexity theory and approximability of optimization problems, which allowed statements of the form "Unless this and that collapse between complexity classes takes place, this problem cannot be approximated within a certain factor.". Their ideas culminated in the introduction of a whole hierarchy of complexity classes by Arora, Safra [8] based on probabilistically checkable proofs, and the proof of the so-called PCP-Theorem by Arora, Lund, Motwani, Sudan, and Szegedy [7]. Altogether, these results opened the door for scientifically highly productive years yielding not only numerous strong non-approximability results, but also the development of a natural and elegant hierarchy which classifies optimization problems according to their approximability.

The aim of this survey is twofold. Firstly, we want to introduce the reader to the development sketched above. Secondly, we aim at providing an overview of the known techniques for constructing reductions between optimization problems. We stress that throughout this survey we are interested in providing the reader with a gentle introduction to these topics. Our survey is neither meant to be comprehensive nor complete. Readers may turn to the survey articles [6], [11] and the books [9], [36], and [52] for more in-depth discussions of these areas.

The survey is organized as follows. In the remainder of this section we state basic definitions from complexity theory and provide a list of examples which will be used further on. In Section 7.2 we describe various tools, techniques, and results for proving lower bounds on the approximability of optimization problems. In Section 7.3 we will then use these results to show that the class $\mathcal{NPO}$ can be divided into various subclasses which resemble problems of similar approximability. Finally, we discuss in Section 7.4 several methods for proving lower bounds for an optimization problem and provide some examples.

7.1.1 Decision Problems

In this section we summarize a few basic definitions and results from complexity theory. Let $\Sigma = \{0, 1\}$. As usual, we let Σ^* denote the set of all finite words over the alphabet Σ and use $|x|$ to denote the length of a word $x \in \Sigma^*$.

Every *instance* of a problem can be encoded as a word $x \in \Sigma^*$. The set $\mathcal{I}$ of all instances of a problem will usually form a *proper* subset of Σ^*. For arbitrary subsets $\mathcal{I} \subseteq \Sigma^*$, the problem of deciding whether a word $x \in \Sigma^*$ belongs to $\mathcal{I}$ can be highly non-trivial or even undecidable. In the context of decision problems the set $\mathcal{I}$ just corresponds to proper encodings of, for example, a graph, a Boolean formula or any other combinatorial instance. It should therefore be plausible that it is usually trivial to check whether a given word x encodes a proper instance. Formally, we call a set $\mathcal{I} \subseteq \Sigma^*$ *recognizable in linear time*, if there exists an algorithm $\mathcal{A}$ that stops for every string $x \in \Sigma^*$ after at most $\mathcal{O}(|x|)$ steps and returns ACCEPT if $x \in \mathcal{I}$ and REJECT if $x \in \Sigma^* \setminus \mathcal{I}$. In this chapter we will only consider sets $\mathcal{I} \subseteq \Sigma^*$ that are recognizable in linear time.

Definition 7.1 *A* decision problem Π *is a pair* $\langle \mathcal{I}, \mathcal{S}ol \rangle$ *such that*

- $\mathcal{I} \subseteq \Sigma^*$ *is a set of* instances *that is recognizable in linear time.*
- *For every instance* $I \in \mathcal{I}$, $\mathcal{S}ol(I) \subseteq \Sigma^*$ *denotes the set of* solutions *of* I.

An algorithm $\mathcal{A}$ is said to *solve* a decision problem $\Pi = \langle \mathcal{I}, \mathcal{S}ol \rangle$ if the algorithm stops for all instances $I \in \mathcal{I}$ and returns ACCEPT if $\mathcal{S}ol(I) \neq \emptyset$ and REJECT otherwise. Note that we do not require that the algorithm finds a member of $\mathcal{S}ol(I)$. It simply has to decide whether $\mathcal{S}ol(I)$ is empty or not.

Definition 7.2 *The class $\mathcal{P}$ contains all problems which can be solved by a polynomial-time algorithm.*

For many decision problems no polynomial-time algorithm is known. Nevertheless many of these problems have a property which is not inherent to every decision problem. Namely, that there exists an algorithm, which may not *find* a solution in polynomial time, but which can at least *check* in polynomial time, whether a given string x is a solution. Decision problems with this property form the class $\mathcal{NP}$. This abbreviation comes from *$\mathcal{N}$ondeterministic $\mathcal{P}$olynomial time.*

Definition 7.3 *A decision problem $\Pi = \langle \mathcal{I}, \mathcal{S}ol \rangle$ belongs to the class $\mathcal{NP}$ iff*

- *The size of the solutions is polynomially bounded in the length of I, i.e., there exists a polynomial p such that*

$$|x| \leq p(|I|) \qquad \text{for all } I \in \mathcal{I} \text{ and } x \in \mathcal{S}ol(I).$$

- *There exists an algorithm $\mathcal{A}$ and a polynomial q such that for every $I \in \mathcal{I}$ the following is true:*
  - *If $\mathcal{S}ol(I) \neq \emptyset$ then there exists $x_0 \in \mathcal{S}ol(I)$ such that $\mathcal{A}(I, x_0)$ accepts in time $q(|I|)$.*
  - *If $\mathcal{S}ol(I) = \emptyset$ then for every $x \in \Sigma^*$ the algorithm $\mathcal{A}(I, x)$ rejects in time $q(|I|)$.*

An immediate consequence of this definition is that $\mathcal{P} \subseteq \mathcal{NP}$. The question whether the converse inclusion is also true, i.e., the question

$$\mathcal{P} \stackrel{?}{=} \mathcal{NP}$$

is one of the central open problems in complexity theory. There are thousands of important and well studied problems in $\mathcal{NP}$ and so far no polynomial-time algorithm is known which solves a single one of them. A reasonable strategy to attack the $\mathcal{P}$ versus $\mathcal{NP}$ question is to study the most difficult problems in $\mathcal{NP}$.

Definition 7.4 *A decision problem* $\Pi = \langle \mathcal{I}, \mathcal{S}ol \rangle$ *is said to be (Karp-)reducible to another decision problem* $\Pi^* = \langle \mathcal{I}^*, \mathcal{S}ol^* \rangle$*, written as* $\Pi \leq_p \Pi^*$*, if there exists a function* $f : \mathcal{I} \to \mathcal{I}^*$ *computable in polynomial time such that*

$$\mathcal{S}ol(I) \neq \emptyset \quad \Longleftrightarrow \quad \mathcal{S}ol^*(f(I)) \neq \emptyset \qquad \textit{for all } I \in \mathcal{I}.$$

A decision problem Π^* *is called* NP-complete *if and only if* $\Pi^* \in \mathcal{NP}$ *and* $\Pi \leq_p \Pi^*$ *for all problems* $\Pi \in \mathcal{NP}$.

The use of the less or equal sign in the above definition is justified by the following observation. If $\Pi \leq_p \Pi^*$ then the fact that there exists a polynomial-time algorithm for Π^* implies that there exists one for Π as well. In particular, the existence of a polynomial-time algorithm for an NP-complete problem thus implies that $\mathcal{P} = \mathcal{NP}$. Generalizing this idea to arbitrary problems, e.g. optimization problems, we say that a problem Π is NP-*hard*, iff the existence of a polynomial-time algorithm for Π implies that $\mathcal{P} = \mathcal{NP}$.

The notion of NP-completeness was introduced independently by Stephen Cook [22] and Leonid Levin [50] in the beginning of the 70ies. They also showed that the problem

SAT: Given a Boolean formula in conjunctive normal form. Does there exist a satisfying assignment?

is NP-complete. (See Section 7.1.3 for a more detailed definition of the problem SAT.)

Theorem 7.1.1 SAT *is* NP-*complete.* □

For further use in Section 7.3.2 we also state the following corollary, which follows immediately from the proof of Theorem 7.1.1.

Corollary 7.1.2 *Let* $\mathcal{F}$ *denote the set of instances of* SAT*, that is, the set of Boolean formulae in conjunctive normal form. Then for every* $\Pi = \langle \mathcal{I}, \mathcal{S}ol \rangle \in \mathcal{NP}$ *there exist mappings* f *and* g *which are computable in polynomial time such that* f *maps any instance* $I \in \mathcal{I}$ *to a Boolean formula* $f(I) \in \mathcal{F}$ *with variables* $y_1, \ldots, y_n, z_1, \ldots, z_m$ *(where* n *and* m *depend on* I*) such that*

$$a = a_1 \cdots a_{n'} \in \mathcal{S}ol(I) \quad \Longleftrightarrow \quad \exists a_{n'+1}, \ldots, a_n, b_1, \ldots, b_m \in \{0,1\} \textit{ such that}$$

$(a_1, \ldots, a_n, b_1, \ldots, b_m)$ is a satisfying assignment for $f(I)$.

Note that the relation $\leq_p$ is clearly transitive. In order to show that a certain problem $\Pi \in \mathcal{NP}$ is NP-complete, Theorem 7.1.1 implies that it suffices to show that SAT $\leq_p \Pi$. A first list of twelve natural NP-complete problems was presented by Karp [46]. A few years later Garey and Johnson [35] wrote their seminal book "*Computers and Intractability, a Guide to the Theory of NP-Completeness*" which contains a list of hundreds of NP-complete problems.

7.1.2 Optimization Problems

In the above definition of a decision problem, $\mathcal{S}ol(I)$ is the set of all certificates which "prove" that an instance $I \in \mathcal{I}$ has to be accepted. In the context of optimization problems, we assume that the set $\mathcal{S}ol(I)$ is always nonempty. Instead we are given an objective function $\mathrm{val}(I, x)$ which measures the quality of the solution. The task is to find for a given instance I a solution $x \in \mathcal{S}ol(I)$ such that $\mathrm{val}(I, x)$ is as small (or large) as possible.

Definition 7.5 *An optimization problem* Π *is a four-tuple* $\langle \mathcal{I}, \mathcal{S}ol, \mathrm{val}, \mathrm{goal}\rangle$ *such that*

- $\mathcal{I} \subseteq \Sigma^*$ *is the set of* instances*;*
- *For every instance* $I \in \mathcal{I}$*,* $\mathcal{S}ol(I) \subseteq \Sigma^*$ *denotes the set of* solutions *of* I *and is non-empty.*
- *For every instance* I *and solution* $x \in \mathcal{S}ol(I)$*, the value* $\mathrm{val}(I, x)$ *is a positive integer. The function* $\mathrm{val}(\cdot, \cdot)$ *is called the* objective function.
- $\mathrm{goal} \in \{\min, \max\}$.

The aim of an optimization problem is to find, for a given instance I, a solution $x_{\mathrm{opt}} \in \mathcal{S}ol(I)$ such that

$$\mathrm{val}(I, x_{\mathrm{opt}}) = \begin{cases} \min\{\mathrm{val}(I, x) \mid x \in \mathcal{S}ol(I)\} & \text{if goal} = \min, \\ \max\{\mathrm{val}(I, x) \mid x \in \mathcal{S}ol(I)\} & \text{if goal} = \max. \end{cases}$$

We abbreviate the value of the optimal solution by $\mathrm{opt}(I) := \mathrm{val}(I, x_{\mathrm{opt}})$. Note that $\mathrm{opt}(I)$ is well defined, as $\mathcal{S}ol(I)$ is, by definition, nonempty for all $I \in \mathcal{I}$. An algorithm is said to solve an optimization problem Π, if the algorithm stops for all instances $I \in \mathcal{I}$ with a solution $y \in \mathcal{S}ol(I)$ such that $\mathrm{val}(I, y) = \mathrm{opt}(I)$.

Definition 7.6 *An optimization problem belongs to the class* $\mathcal{NPO}$ *iff*

- $\mathcal{I} \subseteq \Sigma^*$ *is a set of* instances *that is recognizable in linear time.*
- *The size of the solutions is polynomially bounded in the length of* I*, i.e., there exists a polynomial* p *such that*

$$|x| \leq p(|I|) \qquad \textit{for all } I \in \mathcal{I} \textit{ and } x \in \mathcal{S}ol(I).$$

- *The question* "Is $x \in \mathcal{S}ol(I)$?" *is decidable in polynomial time.*
- *The function* $\mathrm{val}(\cdot,\cdot)$ *is computable in polynomial time.*

A (polynomial-time) *approximation algorithm* for an optimization problem $\Pi = \langle \mathcal{I}, \mathcal{S}ol, \mathrm{val}, \mathrm{goal}\rangle$ is an algorithm $\mathcal{A}$ which computes for each instance $I \in \mathcal{I}$ (in polynomial time) a solution $x_{\mathcal{A}} \in \mathcal{S}ol(I)$. We will use the notation $\mathcal{A}(I) := \mathrm{val}(I, x_{\mathcal{A}})$ to denote the value of the objective function for the solution obtained by algorithm $\mathcal{A}$.

Definition 7.7 *Let* $\Pi = \langle \mathcal{I}, \mathcal{S}ol, \mathrm{val}, \mathrm{goal}\rangle$ *be an optimization problem. An* approximation algorithm with performance ratio $\varrho \geq 1$ *is an algorithm* $\mathcal{A}$ *such that*

$$\frac{1}{\varrho} \leq \frac{\mathcal{A}(I)}{\mathrm{opt}(I)} \leq \varrho \qquad \text{for all } I \in \mathcal{I}.$$

An algorithm $\mathcal{A}$ *is said to be a* polynomial-time approximation scheme *if* $\mathcal{A}$ *returns for every instance* $I \in \mathcal{I}$ *and every rational* $\varepsilon > 0$ *a solution* $x_{\mathcal{A}} \in \mathcal{S}ol(I)$ *such that* $\mathcal{A}(I,\varepsilon) := \mathrm{val}(I, x_{\mathcal{A}})$ *satisfies*

$$\frac{1}{1+\varepsilon} \leq \frac{\mathcal{A}(I,\varepsilon)}{\mathrm{opt}(I)} \leq 1+\varepsilon.$$

For every fixed $\varepsilon > 0$ *the running time of* $\mathcal{A}$ *has to be polynomially bounded in* $|I|$. *If the running time is in fact bounded by a polynomial in* $|I|$ and $1/\varepsilon$ *then* $\mathcal{A}$ *is called a* fully polynomial-time approximation scheme.

Remark 7.1.3 *In our formulation of Definition 7.7* ϱ *is a fixed constant. This is, however, just for simplicity, as in this survey we restrict our attention to these cases. In general, one also considers the case that the ratio* $\mathcal{A}(I)/\mathrm{opt}(I)$ *is bounded by* $\varrho(|I|)$, *where* $\varrho(\cdot)$ *is some slowly growing function. For various problems this is in fact also necessary, as one can show that, unless* $\mathcal{P} = \mathcal{NP}$, *no polynomial-time approximation algorithm with constant performance ratio can exist. Prominent examples in this context are the chromatic number and the clique number of a graph* $G = (V,E)$ *(see below for definitions) and the set-covering problem. In the first two cases Feige and Kilian [30] and Håstad [39] showed that it is unlikely that an approximation algorithm with performance ratio* $n^{1-\varepsilon}$ *exists, where* $n = |V|$ *denotes the number of vertices of the graph* G *and* $\varepsilon > 0$ *is an arbitrarily small constant. For the set covering problem Feige [28] showed that unless* $\mathcal{P} = \mathcal{NP}$ *the performance ratio* $(1-o(1))\ln(n)$ *of the greedy algorithm [43, 51] is essentially best possible.*

Based on Definition 7.7 we can subdivide the set $\mathcal{NPO}$ into classes of problems which share the existence of certain approximation algorithms.

Definition 7.8 $\mathcal{PO}$ *is the set of all optimization problems in* $\mathcal{NPO}$ *which can be solved optimally in polynomial time.*

Definition 7.9 *$\mathcal{FPTAS}$ is the set of all optimization problems in $\mathcal{NPO}$ which admit a fully polynomial-time approximation scheme.*

Definition 7.10 *$\mathcal{PTAS}$ is the set of all optimization problems in $\mathcal{NPO}$ which admit a polynomial-time approximation scheme.*

Definition 7.11 *$\mathcal{APX}$ is the set of all optimization problems in $\mathcal{NPO}$ which admit a polynomial-time approximation algorithm with performance ratio ϱ for some constant $\varrho \geq 1$.*

From the definition it follows immediately that

$$\mathcal{P} \subseteq \mathcal{FPTAS} \subseteq \mathcal{PTAS} \subseteq \mathcal{APX} \subseteq \mathcal{NPO}.$$

In Section 7.3 we will see that, unless $\mathcal{P} = \mathcal{NP}$, all these inclusions are in fact strict.

7.1.3 Examples and Reference Problems

In this section we list a couple of optimization problems which will be used as examples and reference problems in the latter sections of this survey.

We start with some optimization variants of satisfiability problems. A *Boolean formula* for the set of variables $X = \{x_1, \ldots, x_n\}$ is recursively defined as follows. Every variable x_i is a Boolean formula. For Boolean formulas F_1 and F_2, also the negation $\neg(F_1)$ and the two expressions $F_1 \wedge F_2$ and $F_1 \vee F_2$ are Boolean formulas. The formulas x_i and $\neg x_i$ are also called *literals.* A *Boolean formula* is said to be in *conjunctive normal form* if it is a conjunction of clauses C_i where each of these clauses C_i is a disjunction of literals. A truth assignment is a mapping $\tau : X \to \{0, 1\}$ which assigns to every variable the Boolean value TRUE or FALSE. A truth assignment is said to be a *satisfying assignment* for a Boolean formula F if F evaluates to TRUE when we replace each variable x_i by $\tau(x_i)$. Recall that SAT denotes the decision problem "Given a Boolean formula in conjunctive normal form. Does there exist a satisfying assignment?". It is NP-complete according to Theorem 7.1.1. It is well known that also the special case in which each clause contains at most 3 literals and each variable appears in at most 3 clauses (negated or unnegated) is NP-complete [35]. With E3SAT we denote the version where each clause contains exactly 3 literals. It is NP-complete as well [35]. There are a couple of natural optimization problems related to satisfiability problems.

MAXSAT:	Given a Boolean formula in conjunctive normal form, find an assignment that satisfies as many clauses as possible.
MAXEkSAT:	Subcase of MAXSAT in which every clause contain exactly k literals.

MAXkSAT:	Subcase of MAXSAT in which every clause contains at most k literals.
MAXSAT(ℓ):	Subcase of MAXSAT in which every variable appears in at most ℓ clauses. With MAXkSAT(ℓ) we denote the combination of the last two restrictions.

A set of optimization problems that are syntactically closely related to satisfiability problems are systems of linear equations of the form $x_{i_1} + \ldots + x_{i_k} = b_i$. In this survey we restrict our considerations to the case that all computations are done modulo 2, i.e. in the field $\mathbb{F}_2 := \{0, 1\}$. Note that the decision version "Given a system of linear equations. Does there exist an assignment that satisfies all equations simultaneously?" is solvable in polynomial time by Gauss elimination. The optimization variants, however, will turn out to be not so easy.

MAXLINEQ:	Given a system of linear equations over $\mathbb{F}_2$. Find an assignment that satisfies as many equations as possible.
MAXEkLINEQ:	Subcase of MAXLINEQ in which every equation contains exactly k variables. With MAXEkLINEQ(ℓ) we denote the variant in which every variable appears in at most ℓ equations.

In graph theory the chromatic number $\chi(G)$ is a well studied parameter. It is defined as follows. A legal k-coloring of a graph $G = (V, E)$ is a mapping $c : V \to \{1, \ldots, k\}$ such that $c(x) \neq c(y)$ for all $\{x, y\} \in E$. The chromatic number $\chi(G)$ is defined as the minimum k for which there exists a legal k-coloring of G. The decision problem "Given a graph G and an integer k, is $\chi(G) \leq k$?" is known to be NP-complete. It even remains NP-complete if $k \geq 3$ is a fixed integer which is not part of the input. There are two natural related optimization problems.

MINCOL:	Given a graph $G = (V, E)$, find a legal k-coloring such that k is as small as possible. The subcase which contains only those graphs G with chromatic number $\chi(G) \leq \ell$ is denoted by MINCOL(ℓ).
MAXkCOL:	Given a graph G, find a k-coloring c that maximizes the number of edges $\{x, y\}$ in G for which $c(x) \neq c(y)$. Similarly as above, MAXkCOL(ℓ) denotes the subcase in which the input is restricted to graphs which are ℓ-colorable.

A closely related concept to the chromatic number is the *clique number* $\omega(G)$. It is defined as the cardinality of the largest clique which is contained in G as a subgraph. (For the relation to the chromatic number observe that $\chi(G) \geq \omega(G)$ and $\chi(G) \geq |V|/\omega(\overline{G})$, where $\overline{G}$ denotes the complement of G.) The decision problem "Given a graph G and an integer k, is $\omega(G) \geq k$?" is again known to be NP-complete. Note, however, that for the clique number the special case where k is a fixed constant that is not part of the input is solvable in polynomial time. (Consider an enumeration algorithm which checks all subsets of V of size k.) In this case there exists just one natural optimization problem.

MAXCLIQUE: Given a graph $G = (V, E)$, find a subset $X \subseteq V$ such that X induces a clique in G and such that $|X|$ is as large as possible.

For a graph $G = (V, E)$ and a subset $X \subset V$ we denote by $cut(X)$ the number of edges from E which have exactly one endpoint in X. It is well known, see e.g. [54], that one can find in polynomial time a set $\emptyset \neq X \subset V$ such that $cut(X)$ is minimized. The maximization variant, however, is NP-hard. If we restrict X to sets which contain exactly half the vertices, also the minimization variant becomes NP-hard [34].

MAXCUT: Given a graph $G = (V, E)$, find a subset $X \subset V$ such that $cut(X)$ is maximized.

MINBISECTION: Given a graph G, find a subset $X \subset V$, $|X| = \lfloor \frac{1}{2} V \rfloor$, such that $cut(X)$ is minimized.

MAXBISECTION: Given a graph G, find a subset $X \subset V$, $|X| = \lfloor \frac{1}{2} V \rfloor$, such that $cut(X)$ is maximized.

As we will see in the subsequent sections, there is a fundamental difference in the approximability of MINCOL and MAXkCOL. Intuitively, this is due to the following difference in the definition of the two problems. In the problem MINCOL the solutions consist only of legal colorings – and finding a legal coloring is a global problem where we do have to consider the whole graph simultaneously. In MAXkCOL on the other hand *every* k-coloring is a solution and the objective function just counts the number of "good" edges. Deciding whether an edge is good (has both of its vertices colored with different colors) can be done locally, without considering other edges. Thus MAXkCOL is in a sense a *local* optimization problem, while MINCOL is a *global* optimization problem.

Observe also that MAXkCOL is not only a local problem, it is also very similar to MAXkSAT. In both problems we are given a set of objects (vertices, variables) which should be colored with a given number of colors such that an objective function which counts the number of local configurations which do satisfy a certain property is maximized. Generalizing this idea leads to constraint satisfaction problems, which we define next.

Constraint satisfaction problems

A (k-ary) constraint function is a Boolean function $f : \{0,1\}^k \to \{0,1\}$. Given a set of variables $X = \{x_1, \ldots, x_n\}$, a *constraint* for X is a pair $C = (f, (i_1, \ldots, i_k))$ such that f is a constraint function and $1 \leq i_1, \ldots, i_k \leq n$. The constraint is said to be satisfied by an assignment $a = (a_1, \ldots, a_n)$ for the variables $(x_1, \ldots, x_n)$ if $f(a_{i_1}, \ldots, a_{i_k}) = 1$. A *constraint family* $\mathcal{F}$ is a finite collection of constraint functions. A constraint $C = (f, (i_1, \ldots, i_k))$ is said to be from $\mathcal{F}$ if $f \in \mathcal{F}$. With this notation at hand we can now define a *constraint satisfaction problem* formally.

MAX$\mathcal{F}$: Given a set of variables $X = \{x_1, \ldots, x_n\}$ and a collection $C_1, \ldots, C_m$ of constraints for X from $\mathcal{F}$. Find an assignment for $x_1, \ldots, x_n$ which maximizes the number of satisfied constraints.

By choosing appropriate sets $\mathcal{F}$, one easily observes that MAX$\mathcal{F}$ is a generalization of many well known problems.

- Let $SAT_k : \{0,1\}^k \to \{0,1\}$ with $SAT_k(x_1, \ldots, x_k) = x_1 \vee \ldots \vee x_k$. Then MAX$k$SAT is equivalent to MAX$\mathcal{F}$, if we set $\mathcal{F} = \{SAT_1, \ldots, SAT_k\}$.
- Let $CUT : \{0,1\}^2 \to \{0,1\}$ with $CUT(x,y) = x \oplus y$, where $\oplus$ denotes the XOR between the two variables. Then the graph problem MAXCUT is equivalent to MAX$\mathcal{F}$, if we set $\mathcal{F} = \{CUT\}$.
- Let $COL_4 : \{0,1\}^2 \to \{0,1\}$ with $COL_4(x_1, x_2, y_1, y_2) = (x_1 \oplus y_1) \vee (x_2 \oplus y_2)$. Then the graph coloring problem MIN4COL is equivalent to MAX$\mathcal{F}$, if we set $\mathcal{F} = \{COL_4\}$.

Remark 7.1.4 *Note that the problem* MAX$\mathcal{F}$ *belongs (for all constraint families $\mathcal{F}$ that do not contain a function that is identically* 0*) to the class $\mathcal{APX}$. This is easily seen as follows. A random assignment of the variables satisfies a given constraint with probability at least* 2^{-k}*, where k is the maximum arity of a constraint function in $\mathcal{F}$. That is, a randomized algorithm which just guesses an assignment has an (expected) performance ratio of at least* 2^k*. As this algorithm can easily be derandomized using the method of conditional probabilities (see e.g. [53]), this implies that* MAX$\mathcal{F}$ *can always be approximated within a ratio of* 2^k*. The reader is invited to observe that the constant* 2^k *can often be decreased dramatically, if the constraint functions satisfy additional properties. E.g. for SAT_k we obtain* $2^k/(2^k - 1)$*, for CUT we obtain* 2*, and for COL_4 we obtain* $4/3$.

Weighted problems

In principle, any of the above optimization problems can be turned into a weighted version. In a weighted version the instances are enlarged by some weight function w, and the objective function measures the solution also with respect to this weight function. For example, for MAXSAT we could assign a weight to each clause and then ask for an assignment that maximizes the total weight of all satisfied clauses. In Section 7.4.2 we will show that for a wide range of optimization problems the approximability of weighted and unweighted problems is very similar. We therefore desist from specifying weighted versions for all problems mentioned so far. Instead we just define weighted constraint satisfaction problems.

MaxWeight$\mathcal{F}$: Given a set of variables $X = \{x_1, \ldots, x_n\}$, a collection $C_1, \ldots, C_m$ of constraints for X from $\mathcal{F}$, and weights $w_1, \ldots, w_m \in \mathbb{Z}^+$. Find an assignment for $x_1, \ldots, x_n$ which maximizes the total weight of all satisfied constraints.

7.2 Proving Lower Bounds

In this section we survey some techniques which have been used in proving lower bounds.

7.2.1 *Adversary*

Adversary arguments have been used successfully in proving lower bounds on the running time of certain algorithms. Perhaps the most well known example is the proof of the fact that every comparison based sorting algorithm needs at least $\lceil \log_2(n!) \rceil$ many comparisons. As the total number of linear orderings on n numbers is $n!$, an adversary can always answer a comparison "*Is* $a[i] \leq a[j]$*?*" of the algorithm in such a way that after k comparisons $n!/2^k$ linear orders are still consistent with all answers given so far. This implies that the algorithm may have to perform $\lceil \log_2(n!) \rceil$ many comparisons before the true linear ordering is identified.

Another area where adversary arguments have been used very successfully, is in proving lower bounds on the competitive ratio of online algorithms. See e.g. the textbook by Borodin and El-Yaniv [20] for details.

7.2.2 *Bootstrapping*

Bootstrapping is a technique which permits to obtain algorithms of increasingly better quality by just applying the same algorithm to suitably changed inputs. The following example from [32] illustrates this idea.

Theorem 7.2.1 *If there exists an approximation algorithm for* MaxClique *with performance ratio ϱ_0 for some ϱ_0, then there also exist approximation algorithms with performance ratio ϱ for any $\varrho > 1$.*

Proof. Given a graph $G = (V, E)$, we let $G \times G$ denote the graph with vertex set $V \times V$ in which two vertices (x_1, y_1) and (x_2, y_2) are connected by an edge if and only either $x_1 = x_2$ and $\{y_1, y_2\} \in E$ or $\{x_1, x_2\} \in E$. One easily checks that $\omega(G \times G) = \omega(G)^2$ and that every clique C in $G \times G$ induces a clique C' in G of size at least $\sqrt{|C|}$.

Assume that $\mathcal{A}$ is an approximation algorithm for MaxClique with performance ratio ϱ_0. If we apply $\mathcal{A}$ to $G \times G$ the algorithm has to return a clique C of size at least $\omega(G \times G)/\varrho_0$. $\mathcal{A}$ can thus be used to find a clique in G of size at least $\sqrt{\omega(G \times G)/\varrho_0} = \omega(G)/\sqrt{\varrho_0}$, yielding an an approximation algorithm $\mathcal{A}'$ for MAXCLIQUE with performance ratio $\sqrt{\varrho_0}$. Now repeat this argument for $\sqrt{\varrho_0}$ instead of ϱ_0 to obtain an approximation algorithm with performance ratio $(\varrho_0)^{1/4}$, and so on. As $(\varrho_0)^{1/2^k}$ tends to 1 for every $\varrho_0 > 1$, this concludes the proof of the theorem. □

A slight variant of the bootstrapping techniques is the proof that optimization problems where the value of the optimal solution is polynomially bounded in the input size cannot belong to the class $\mathcal{FPTAS} \setminus \mathcal{PO}$.

Theorem 7.2.2 *Let* $\Pi = \langle \mathcal{I}, \mathcal{S}ol, \text{val}, \text{goal} \rangle$ *be an optimization problem in* $\mathcal{NPO}$. *Assume that there exists a polynomial* p *such that* $|\text{opt}(I)| \le p(|I|)$ *for all instances* $I \in \mathcal{I}$. *Then* $\Pi \notin \mathcal{FPTAS} \setminus \mathcal{PO}$, *unless* $\mathcal{P} = \mathcal{NP}$.

Proof. Assume $\Pi \in \mathcal{FPTAS}$. Then there exists an approximation scheme $\mathcal{A}$ which returns, for given $I \in \mathcal{I}$ and rational $\varepsilon > 0$, in time polynomial in $|I|$ and $1/\varepsilon$ a solution $\mathcal{A}(I, \varepsilon) \in \mathcal{S}ol(I)$ whose performance ratio is bounded by $1 + \varepsilon$.

Apply the algorithm with $\varepsilon_0 = 1/(p(|I|) + 1)$. Then $1/\varepsilon_0$ is bounded by a polynomial in $|I|$ and according to the assumption on $\mathcal{A}$ we have

$$\frac{1}{1 + \frac{1}{p(|I|)+1}} \le \frac{\text{val}(I, \mathcal{A}(I, \varepsilon_0))}{\text{opt}(I)} \le 1 + \frac{1}{p(|I|) + 1}.$$

As $1 - \frac{1}{p(|I|)+1} \le \frac{1}{1+\frac{1}{p(|I|)+1}}$ this implies that

$$|\text{val}(I, \mathcal{A}(I, \varepsilon_0)) - \text{opt}(I)| \le \frac{|\text{opt}(I)|}{p(|I|) + 1} < 1,$$

where the last inequality follows from the assumption that $|\text{opt}(I)| \le p(|I|)$ for all instances I. As $\text{val}(I, \mathcal{A}(I, \varepsilon_0))$ and $\text{opt}(I)$ are both integers this therefore implies that $\text{val}(I, \mathcal{A}(I, \varepsilon_0)) = \text{opt}(I)$. That is, we have constructed an algorithm which solves Π optimally and whose running time is bounded by a polynomial in $|I|$. But this can only happen if $\Pi \in \mathcal{PO}$ or $\mathcal{P} = \mathcal{NP}$. □

A similar result holds for problems for which the optimum is bounded by a polynomial in $|I|$ and $Max(I)$ (the maximum number contained in the input) if the underlying decision problem is strongly NP-complete [33].

7.2.3 The Gap Technique

In this section we first introduce a simple idea of deducing non-approximability results from NP-completeness results. Subsequently we then define so-called gap reductions, which can be used to get improved non-approximability results.

As a first example consider the problem MINCOL(3). The underlying decision problem "Given a graph G, is G 3-colorable?" is known to be

NP-complete [34]. We can thus deduce that MinCol(3) cannot be approximated within a factor of $4/3 - \varepsilon$, unless $\mathcal{P} = \mathcal{NP}$. To see this observe that an approximation algorithm with performance ratio $4/3 - \varepsilon$ has to find a 3-coloring for every 3-colorable graph.

We now formalize this idea. Let Π be a minimization problem. Suppose furthermore that we are given two functions $t : \mathcal{I} \to \mathbb{N}$ and $g : \mathcal{I} \to \mathbb{Q}^{>1}$. Then $\Pi_{t,g}$ denotes the restriction of Π to those instances $I \in \mathcal{I}$ such that either $\mathrm{opt}(I) \le t(I)$ or $\mathrm{opt}(I) \ge t(I) \cdot g(I)$. Every such problem $\Pi_{t,g}$ gives rise to a natural decision problem $\Pi^d_{t,g}$ if we set $\mathcal{S}ol^d(I) := \{y \in \mathcal{S}ol(I) \mid \mathrm{val}(I, y) \le t(I)\}$. If $\Pi^d_{t,g}$ is NP-complete, we easily deduce a non-approximability result for Π.

Theorem 7.2.3 *Let Π be a minimization problem. Suppose furthermore that $t : \mathcal{I} \to \mathbb{N}$ and $g : \mathcal{I} \to \mathbb{Q}^{>1}$ are two functions computable in polynomial time such that $\Pi^d_{t,g}$ is* NP*-complete. Then Π cannot be approximated within a factor of $g(I) - \varepsilon$ for any $\varepsilon > 0$, unless $\mathcal{P} = \mathcal{NP}$.*

Proof. Assume $\mathcal{A}$ is a polynomial-time approximation algorithm for Π with performance ratio $g(I) - \varepsilon$. Then $\mathcal{A}$ has to return for every instance $I \in \mathcal{I}$ such that $\mathrm{opt}(I) \le t(I)$ a solution with value at most $t(I) \cdot (g(I) - \varepsilon)$. On the other hand by assumption every instance I such that $\mathrm{opt}(I) > t(I)$ in fact satisfies $\mathrm{opt}(I) \ge t(I) \cdot g(I)$. The approximation algorithm $\mathcal{A}$ can thus be used to decide $\Pi^d_{t,g}$. This is only possible if $\mathcal{P} = \mathcal{NP}$. □

Let us return to the problem MinCol(3). Setting $t(G) = 3$ and $g(G) = 4/3$ yields the non-approximability factor of $4/3 - \varepsilon$ mentioned above. But even more is true. Khanna, Linial and Safra [47] described a (polynomial-time) reduction, which transforms every 3-colorable graph into another 3-colorable graph, but every non-3-colorable graph into a graph which is also not 4-colorable. Similarly as above, we can thus deduce that MinCol(3) cannot be approximated within a factor of $5/3 - \varepsilon$, unless $\mathcal{P} = \mathcal{NP}$.

We again formalize this idea. Let $\Pi_{t,g}$ and $\Pi^*_{t^*,g^*}$ be two minimization problems. A *gap preserving* reduction from $\Pi_{t,g}$ to $\Pi^*_{t^*,g^*}$ – in symbols $\Pi_{t,g} \le_{\mathsf{gap}} \Pi^*_{t^*,g^*}$ – is a function $f : \mathcal{I} \to \mathcal{I}^*$ computable in polynomial time such that the following property is satisfied for all instances $I \in \mathcal{I}$.

$$\begin{aligned} \mathrm{opt}(I) \le t(I) \quad &\Longrightarrow \quad \mathrm{opt}(f(I)) \le t^*(I) \\ \mathrm{opt}(I) \ge t(I) \cdot g(I) \quad &\Longrightarrow \quad \mathrm{opt}(f(I)) \ge t^*(I) \cdot g^*(I). \end{aligned}$$

Then the following theorem can be proven in a similar way as Theorem 7.2.3.

Theorem 7.2.4 *Assume $\Pi_{t,g} \le_{\mathsf{gap}} \Pi^*_{t^*,g^*}$. Then the fact that $\Pi^d_{t,g}$ is* NP*-complete implies that $\Pi^{*d}_{t^*,g^*}$ is* NP*-complete.*

Proof. Just observe that the gap preserving reduction implies the existence of a reduction between the two decision problems $\Pi^d_{t,g}$ and $\Pi^{*d}_{t^*,g^*}$. □

Remark 7.2.5 *In this section we only considered minimization problems so far. This was just for convenience. It should be clear that similar definitions hold for maximization problems as well, if we interchange $\leq$ and $\geq$ and change $t(I) \cdot g(I)$ to $t(I)/g(I)$.*

Unfortunately, it turns out that an application of Theorem 7.2.4 to maximization problems or, more general, to problems Π in which the optimum tends to infinity if the size of the problem tends to infinity is usually non-trivial. The reason is that in this case it is a priori not clear how to obtain suitable functions t and g. Consider for example satisfiability problems. Given an instance I it is NP-complete to distinguish whether I is satisfiable or not. For MaxSat this just means that we could set $t(I) = m$, where m denotes the number of clauses, and $g(I) = 1 + \frac{1}{m-1}$ (so that $t(I)/g(I) = m - 1$). As $g(I)$ tends to one with $|I|$ tending to infinity, Theorem 7.2.3 gives no proper non-approximability result. But of course we could try to apply Theorem 7.2.4. That is, construct a gap preserving reduction with the property that satisfiable instances are mapped to satisfiable instances, and non-satisfiable instances are mapped to instances where at most some constant fraction of all clauses can be satisfied simultaneously. At first sight this may sound like an easily solvable combinatorial problem. Unfortunately, this is not so. In order to construct such a reduction we need some notions and deep results from computational complexity theory.

7.2.4 Probabilistically Checkable Proofs

The connection between the existence of certain interactive proofs and hardness results for optimization problems, first established in [21, 29], has turned out to be immensely fruitful and productive over the last years. In this section we briefly survey some of the key ideas and results.

Let us first informally recall the definition of the class $\mathcal{NP}$. A decision problem is in the class $\mathcal{NP}$ if there exists an algorithm which can check in polynomial time whether a string x is a solution for a given instance I. Recall that it is important that the algorithm just has to *check* a given solution x. It need not *find* such a solution. It therefore makes sense to call such a solution $x \in \mathcal{S}ol(I)$ a *proof* for the fact that $\mathcal{S}ol(I) \neq \emptyset$. The fact that a decision problem is in the class $\mathcal{NP}$ then simply means that these proofs can be checked (or, as we will henceforth also call it, verified) in polynomial time.

For many problems in $\mathcal{NP}$ the "proofs" are simple and straightforward. For example, they just consist of a satisfying assignment, a coloring or a clique of a given size. It is important to note here that these proofs do not contain much redundant information. For example, to distinguish a satisfying assignment from one that satisfies all but one clause, one usually has to read the truth value of every variable, i.e., the whole proof.

This is not the case with *probabilistically checkable proofs.* Probabilistically checkable proofs are inspected by *verifiers* (polynomial-time algorithms) which proceed as follows. After reading the instance I they generate a couple of random bits. Based on these random bits they decide which bits (positions) of the proof they want to read. Subsequently, they either accept the instance I or reject it — only on the knowledge of the (few) queried bits! A decision problem is said to have a probabilistically checkable proof if for all YES-instances I there exists a proof π_0 which the verifier accepts for *all* possible outcomes of the random bits, while for all NO-instances the verifier rejects *all* proofs with probability at least one half.

At first it may seem impossible to construct probabilistically checkable proofs for problems in $\mathcal{NP}$, which can be checked by reading only a constant number of bits. (Try it!) Surprisingly enough this is, however, not too difficult. At least not if we allow the proof to be arbitrarily long. Highly non-trivial on the other hand is the fact that every problem in $\mathcal{NP}$ has a proof of polynomial length with the same property. More precisely, every YES-instance I of length $n := |I|$ admits a proof of length polynomial in n which can be checked probabilistically by reading only a constant number of bits from it. Note that in order to choose a bit from a proof of length t we just need to specify $\lceil \log_2 t \rceil$ many bits. That is, in order to specify constantly many positions of a proof of length polynomial in n it suffices to generate $\mathcal{O}(\log n)$ many random bits. We now make these definitions precise.

The concept of probabilistically checkable proofs or, more precisely, of interactive proofs, were introduced independently by Goldwasser, Micale, Rackoff [37] and Babai [10, 14] and generalized to multi-prover protocols by Ben-Or, Goldwasser, Kilian, Wigderson [16]. The following definition is due to Arora and Safra [8], who also coined the term *probabilistically checkable proofs.*

Definition 7.12 *Let $r(n)$ and $q(n)$ be positive integer functions. An* $(r(n), q(n))$-restricted verifier *for a decision problem $\langle \mathcal{I}, \mathcal{S}ol \rangle$ is an algorithm $\mathcal{V}$ that has access to an input I, a string τ of random bits, and a proof π such that for every input I of length $n := |I|$ the verifier $\mathcal{V}$ reads only the first $\mathcal{O}(r(n))$ many bits from τ and reads at most $\mathcal{O}(q(n))$ positions of the proof π.*

Such a verifier is said to decide $\langle \mathcal{I}, \mathcal{S}ol \rangle$ *if for every input I of length $n := |I|$ the verifier $\mathcal{V}$ returns in time polynomial in n either* ACCEPT *or* REJECT *such that*

$$\mathcal{S}ol(I) \neq \emptyset \quad \Longrightarrow \quad \exists \pi_0 : \ \Pr[\mathcal{V}(I, \tau, \pi_0) = \text{ACCEPT}] = 1,$$

and

$$\mathcal{S}ol(I) = \emptyset \quad \Longrightarrow \quad \forall \pi : \ \Pr[\mathcal{V}(I, \tau, \pi) = \text{REJECT}] \geq \frac{1}{2}.$$

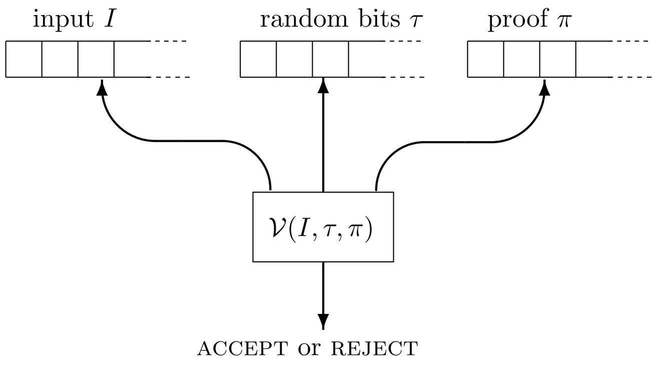


Figure 7.1. Illustration of a verifier for the classes $\mathcal{PCP}(\cdot,\cdot)$.

(Here the probability is with respect to the random string τ, assuming that all such 0-1 *strings are equally likely.)*

The class $\mathcal{PCP}(r(n),q(n))$ denotes the set of all decision problems $\langle\mathcal{I},\mathcal{S}ol\rangle$ that can be decided by an $(r(n),q(n))$-restricted verifier.

The functions $r(n)$ and $q(n)$ make the definition of the classes $\mathcal{PCP}(\cdot,\cdot)$ rather general. In particular, it contains well-known classes like $\mathcal{P}$ or $\mathcal{NP}$ as special cases. For example, one easily deduces that

$$\begin{aligned}\mathcal{P} &= \mathcal{PCP}(0,0)\\ \mathcal{NP} &= \mathcal{PCP}(0,\mathit{poly}(n)) := \bigcup_{k\geq 1}\mathcal{PCP}(0,n^k), \text{ and },\\ \mathit{co}\mathcal{RP} &= \mathcal{PCP}(\mathit{poly}(n),0) := \bigcup_{k\geq 1}\mathcal{PCP}(n^k,0).\end{aligned}$$

The next question arises quite naturally. How large is the class $\mathcal{PCP}(\mathit{poly}(n),\mathit{poly}(n))$? A result of Babai, Fortnow, Lund [13] implies that this class is indeed very large:

$$\mathcal{PCP}(\mathit{poly}(n),\mathit{poly}(n)) = \mathcal{NEXP}.$$

This indicates that $\mathcal{NP}$ should also be contained in $\mathcal{PCP}(r(n),q(n))$ for much smaller growing functions r and q. After several intermediate steps [12, 29, 8], Arora, Lund Motwani, Sudan, and Szegedy [7] finally succeeded in proving the following surprisingly strong characterization of $\mathcal{NP}$ in terms of probabilistically checkable proofs.

Theorem 7.2.6 [7] (PCP-Theorem) $\mathcal{NP} = \mathcal{PCP}(\log n, 1)$. □

The proof of Theorem 7.2.6 is highly non-trivial. Besides the original reference [7] the reader can also check [9, 41, 42] for complete proofs.

Starting from Theorem 7.2.6 it is now straightforward to deduce non-approximability results.

Theorem 7.2.7 [7] *There exists $\varepsilon_0 > 0$ and two functions f and g, computable in polynomial time, such that for every instance $\varphi \in$ SAT the following properties are fulfilled:*

(1) *$f(\varphi)$ is an instance of* 3SAT.

(2) *If φ is satisfiable, then there exists a truth assignment for $f(\varphi)$.*

(3) *If φ is not satisfiable, then every truth assignment satisfies at most $(1 - \varepsilon_0)m$ many clauses of $f(\varphi)$, where m denotes the number of clauses in $f(\varphi)$.*

(4) *Given a truth assignment τ which satisfies more than $(1 - \varepsilon_0)m$ clauses of $f(\varphi)$, $g(\tau)$ is a satisfying truth assignment of φ.*

Proof. As SAT belongs to $\mathcal{NP}$, Theorem 7.2.6 implies that there exists a $(\log n, 1)$-restricted verifier $\mathcal{V}$ for SAT. Consider an arbitrary instance of SAT, i.e., an arbitrary Boolean formula φ in conjunctive normal form. We will use $\mathcal{V}$ to construct the desired instance $f(\varphi)$ of 3SAT in polynomial time.

The definition of a $(\log n, 1)$-restricted verifier implies that there exist constants c and k such that $\mathcal{V}$ will use at most $c \log_2 |\varphi|$ many random bits and read at most k many bits of a given proof π. Clearly, we may assume without loss of generality that $\mathcal{V}$ always uses *exactly* $c \log_2 |\varphi|$ many random bits and reads *exactly* k many bits of the proof π. This implies in particular that $\mathcal{V}$ can access at most $k \cdot 2^{c \log_2 |\varphi|} = k \cdot |\varphi|^c$ different positions of π.

Assume the value of the ith bit from π is denoted by x_i. We identify these values with the variables $x_1, x_2, \ldots$ of the Boolean formula $f(\varphi)$. The clauses of $f(\varphi)$ are constructed as follows. For every string τ of length $c \log_2 |\varphi|$ we construct a 3SAT formula F_τ. Assume that $\mathcal{V}$ reads the bits $x_{\tau_1}, \ldots, x_{\tau_k}$ from the proof π, if the random bits are set to τ. Clearly, there are exactly 2^k different conjunctive(!) clauses which use each of the variable $x_{\tau_1}, \ldots, x_{\tau_k}$ exactly once (negated or unnegated). From these 2^k clauses we keep exactly those which correspond to an assignment for which the verifier accepts (identify TRUE with 1 and FALSE with 0). This Boolean formula can easily be transformed into a 3SAT formula F_τ which contains at most $k2^k$ many clauses.

Now consider $f(\varphi) := \bigwedge_\tau F_\tau$? It obviously can be constructed in polynomial time. Furthermore, by construction, every satisfying assignment of F corresponds to a proof π for which the verifier $\mathcal{V}$ accepts for *all* random strings τ. Similarly, every proof π, which is rejected by the verifier for at least half of all random strings, corresponds to an assignment that does not satisfy at least half of all formulas F_τ. In other words, for any such assignment, at least $\frac{1}{2} c \log_2 |\varphi|$ many clauses of $f(\varphi)$ are not satisfied. As $f(\varphi)$ contains at most $k2^k \cdot c \log_2 |\varphi|$ many clauses, this shows that all properties of Theorem 7.2.7 are satisfied if we let $\varepsilon_0 := \frac{1}{k2^{k+1}}$. □

Theorem 7.2.4 allows us to rephrase Theorem 7.2.7 as follows.

Corollary 7.2.8 *There exists an* $\varepsilon_0 > 0$ *such that it is* NP*-hard to distinguish satisfiable* 3SAT *instances from those in which at most* $1 - \varepsilon_0$ *of all clauses can be satisfied simultaneously.* □

Corollary 7.2.9 MAX3SAT $\notin \mathcal{PTAS}$*, unless* $\mathcal{P} = \mathcal{NP}$*.* □

Historically, Theorem 7.2.6 was the starting point for a whole sequence of stronger and stronger non-approximability results. See [6, 3] for surveys. One line of research was to improve the constant ε_0 in Theorem 7.2.7 by constructing slightly different verifiers and tightening the analysis, see e.g. [61] for an overview and references. This race for better and better constants was brought to an end by Håstad [39, 40]. He combined Theorem 7.2.6 with Raz's proof [57] of the so-called "parallel repetition theorem" to obtain essentially optimal non-approximability results.

Theorem 7.2.10 [40] *For every* $\varepsilon > 0$*, it is* NP*-hard to distinguish satisfiable* E3SAT *instances from those in which at most* $7/8 + \varepsilon$ *of all clauses can be satisfied simultaneously.*

Theorem 7.2.11 [40] *For every* $\varepsilon > 0$*, it is* NP*-hard to distinguish between* E3LIN *instances in which at least* $(1-\varepsilon)$ *of all clauses can be satisfied simultaneously from those in which at most* $(1 + \varepsilon)/2$ *of all equations can be satisfied simultaneously.*

Remark 7.2.12 *The fact, that these results are essentially best possible follows from Remark 7.1.4. Karloff and Zwick [45] could even show that there also exists an approximation algorithm for* MAX3SAT *with performance ratio* $8/7$*.*

Theorem 7.2.13 [39] *For every* $\varepsilon > 0$*,* MAXCLIQUE *is not approximable within a factor of* $|V|^{1-\varepsilon}$*, unless* $\mathcal{P} = \mathcal{NP}$*.*

7.3 A Hierarchy for NP-Optimization Problems

The results of the last section enable us to explain a hierarchy for the problems in $\mathcal{NPO}$, which categorize optimization problems according to their approximability. There are three steps for achieving this. Firstly, we need a suitable notion of reductions between optimization problems. Secondly, we need a notion of "complete" problems. Thirdly, we need to show that these two notions fit together properly so that we are able to identify a first complete problem (similar as in the Cook-Levin Theorem 7.1.1).

7.3.1 Reducing optimization problems

In their seminal paper [55] Papadimitriou and Yannakakis introduced the so called L-reduction and the notion of MaxSNP-hardness. In the subsequent years the L-reduction was used very successfully to establish the MaxSNP-hardness of many natural optimization problems. Unfortunately, it turned out that the syntactically-defined class MaxSNP is difficult to compare with the computationally-defined class $\mathcal{APX}$. Moreover, the L-reduction seemed to be to weak in order to allow identification of complete problems within $\mathcal{APX}$. Generalizing the L-reduction solved this problem [25, 27, 48].

Before we formally state definitions, let us first point out why reductions for optimization problems are more complicated than those for decision problems. A key feature of a reduction is that it allows to transform one problem to another in such a way that it suffices to solve the new problem in order to obtain an answer for the original problem. Note that this requires that we do have a way for transforming the answers. In case of decision problems this is trivial, as the answer consists just of ACCEPT or REJECT. In case of optimization problems, however, the answer is a solution, which satisfies some properties, e.g. meets a desired performance ratio. So the notion of a reduction has to provide also a way of transforming solutions of the new problem into solutions of the original problem. As there is some flexibility in the precise realization of such a transformation there do exists several possibilities for defining reductions between optimization problems.

In this section we introduce the notion of AP-reductions due to Crescenzi, Kann, Silvestri, and Trevisan [24], which is by now widely accepted as the standard reduction for optimization problems. Subsequently, we then also introduce the concept of L-reductions, as they are often easier to construct. For a discussion of other notions of reductions which were considered in the literature we refer the reader to [23].

Definition 7.13 *An optimization problem* $\Pi = \langle \mathcal{I}, \mathcal{S}ol, \mathrm{val}, \mathrm{goal}\rangle$ *is AP-reducible to an optimization problem* $\Pi^* = \langle \mathcal{I}^*, \mathcal{S}ol^*, \mathrm{val}^*, \mathrm{goal}^*\rangle$ *– in symbols* $\Pi \leq_{AP} \Pi^*$ *– if and only if there exist functions f and g and a constant $\alpha > 0$ such that:*

(AP1) *For any $\delta > 0$, for any $I \in \mathcal{I}$, $f(I, \delta) \in \mathcal{I}^*$.*

(AP2) *For any $\delta > 0$, for any $I \in \mathcal{I}$, and $y \in \mathcal{S}ol^*(f(I,\delta))$, $g(I, y, \delta) \in \mathcal{S}ol(I)$.*

(AP3) *For any fixed $\delta > 0$, the functions f and g are computable in polynomial time.*

(AP4) *For any $I \in \mathcal{I}$, for any $\delta > 0$, and for any $y \in \mathcal{S}ol^*(f(I,\delta))$,*

$$\frac{1}{1+\delta} \leq \frac{\mathrm{val}(f(I,\delta), y)}{\mathrm{opt}(f(I,\delta))} \leq 1+\delta \quad \Longrightarrow$$

$$\frac{1}{1+\alpha\cdot\delta} \le \frac{\mathrm{val}(I, g(I,y,\delta))}{\mathrm{opt}(I)} \le 1+\alpha\cdot\delta.$$

The triple (f, g, α) is an AP-reduction from Π to Π^*. If we want to emphasize that there exists an AP-reduction (f, g, α) for some specific value of α, we also write $\Pi \le_{AP}^{\alpha} \Pi^*$.

The following lemmas capture important properties of AP-reductions. We leave the easy proofs of the first two lemmas to the reader.

Lemma 7.3.1 *Let Π_0, Π_1 and Π_2 be optimization problems. If $\Pi_0 \le_{AP} \Pi_1$ and $\Pi_1 \le_{AP} \Pi_2$ then $\Pi_0 \le_{AP} \Pi_2$.* □

Lemma 7.3.2 *Let $\Pi, \Pi^* \in \mathcal{NPO}$. If $\Pi^* \in \mathcal{APX}$ and $\Pi \le_{AP} \Pi^*$ then $\Pi \in \mathcal{APX}$.* □

Lemma 7.3.3 *Let $\Pi, \Pi^* \in \mathcal{NPO}$. If $\Pi^* \in \mathcal{PTAS}$ and $\Pi \le_{AP} \Pi^*$ then $\Pi \in \mathcal{PTAS}$.*

Proof. Let $\Pi = \langle \mathcal{I}, \mathcal{S}ol, \mathrm{val}, \mathrm{goal}\rangle$ and $\Pi^* = \langle \mathcal{I}^*, \mathcal{S}ol^*, \mathrm{val}^*, \mathrm{goal}^*\rangle$. We have to show that for every $\varepsilon > 0$ there exists a polynomial-time approximation algorithm for Π with performance ratio $1+\varepsilon$. Fix some $\varepsilon > 0$.

Let (f, g, α) be an AP-reduction from Π to Π^* and choose $\delta = \varepsilon \cdot \alpha^{-1}$. Since $\Pi^* \in \mathcal{PTAS}$ there exists a polynomial-time approximation algorithm, say $\mathcal{A}_\delta$, for Π^* with performance ratio $1+\delta$. Let $I \in \mathcal{I}$ and consider the instance $f(I,\delta) \in \mathcal{I}^*$ which can be computed in polynomial time. $\mathcal{A}_\delta$, then, computes in polynomial time a solution $y \in \mathcal{S}ol^*(f(I,\delta))$ such that

$$\frac{1}{1+\delta} \le \frac{\mathrm{val}(f(I,\delta), y)}{\mathrm{opt}(f(I,\delta))} \le 1+\delta.$$

Starting from y we can then compute, again in polynomial time, a solution $g(I,y,\delta) \in \mathcal{S}ol(I)$. By condition (AP4) and the choice of δ we deduce that this solution satisfies

$$\frac{1}{1+\varepsilon} \le \frac{\mathrm{val}(f(I,\delta), y)}{\mathrm{opt}(f(I,\delta))} \le 1+\varepsilon,$$

as desired. □

Lemma 7.3.4 *Let $\Pi \in \mathcal{NPO}$ and $\varepsilon > 0$ and assume* MaxE3Sat $\le_{AP}^{\alpha}$ Π *for some constant $\alpha > 0$. Then there cannot exist a polynomial-time approximation algorithm for Π with performance ratio at most $1 + \frac{1}{7\alpha} - \varepsilon$, unless $\mathcal{P} = \mathcal{NP}$.*

Proof. According to Theorem 7.2.10 it is NP-hard to distinguish between satisfiable E3Sat instances and those for which at most $(\frac{7}{8}+\varepsilon)$ of all clauses can be satisfied simultaneously. Assume there exists an approximation algorithm $\mathcal{A}$ for Π with performance ratio $1+\delta$. Consider what happens if we apply $\mathcal{A}$ to $f(I,\delta)$, where I is a satisfiable E3SAT instance. By property (AP4) we know that

$$\mathrm{val}(I, g(I, \mathcal{A}(f(I,\delta),\delta))) \geq \frac{\mathrm{opt}(I)}{1+\alpha\delta}.$$

That is, whenever $1/(1+\alpha\delta) > 7/8$ the algorithm $\mathcal{A}$ can be used to decide an NP-hard problem. □

Note that an important feature of Definition 7.13 is that the reduction may depend on the desired performance ratio δ. In other words, an instance $I \in \mathcal{I}$ may be mapped to different instances $I^* \in \mathcal{I}^*$, depending on the parameter δ. According to present knowledge, this seems to be necessary in order to show that $\mathcal{APX}$ contains natural complete problems. On the other hand, reductions between different problems in $\mathcal{APX}$ will often not need this freedom. From a technical point of view, this is quite fortunate, as the dependence on the parameter δ makes AP-reductions usually difficult to describe. We therefore also introduce the notion of the simpler L-reductions and show that they are indeed weaker than AP-reductions.

Definition 7.14 *An optimization problem* $\Pi = \langle \mathcal{I}, \mathcal{S}ol, \mathrm{val}, \mathrm{goal}\rangle$ *is L-reducible to an optimization problem* $\Pi^* = \langle \mathcal{I}^*, \mathcal{S}ol^*, \mathrm{val}^*, goal^*\rangle$ *– in symbols* $\Pi \leq_L \Pi^*$ *– if and only if there exist functions* f *and* g *and constants* $\beta, \gamma > 0$ *such that:*

(L1) *For any* $I \in \mathcal{I}$, $f(I) \in \mathcal{I}^*$ *is computable in polynomial time.*

(L2) *For any* $I \in \mathcal{I}$ *and for any* $y \in \mathcal{S}ol^*(f(I))$, $g(I,y) \in \mathcal{S}ol(I)$ *is computable in polynomial time.*

(L3) *For any* $I \in \mathcal{I}$, $\mathrm{opt}(f(I)) \leq \beta \cdot \mathrm{opt}(I)$.

(L4) *For any* $I \in \mathcal{I}$ *and for any* $y \in \mathcal{S}ol^*(f(I))$,

$$|\mathrm{opt}(I) - \mathrm{val}(I, g(I,y))| \leq \gamma \cdot |\mathrm{opt}(f(I)) - \mathrm{val}(f(I),y)|.$$

The quadruple (f,g,β,γ) *is an L-reduction from* Π *to* Π^*.

Lemma 7.3.5 *Assume* Π *is a problem in* $\mathcal{APX}$ *and* Π^* *is an arbitrary problem in* $\mathcal{NPO}$. *Then*

$$\Pi \leq_L \Pi^* \qquad \Longrightarrow \qquad \Pi \leq_{AP} \Pi^*.$$

Proof. Let (f,g,β,γ) be an L-reduction from Π to Π^*. Then the functions f and g trivially satisfy conditions (AP1)-(AP3). So we only have to show that condition (AP4) holds as well. That is, assume $I \in \mathcal{I}$ and $y \in \mathcal{S}ol^*(f(I))$ are such that

$$\frac{1}{1+\delta} \leq \frac{\mathrm{val}(f(I),y)}{\mathrm{opt}(f(I))} \leq 1+\delta \qquad (7.1)$$

for some $\delta > 0$. We have to show that this implies that

$$\frac{1}{1+\alpha\cdot\delta} \le \frac{\mathrm{val}(I, g(I,y))}{\mathrm{opt}(I)} \le 1+\alpha\cdot\delta \tag{7.2}$$

for an appropriate constant $\alpha > 0$. Unfortunately, for maximization problems Π (and large values of δ) this will in general not be true. To cover these case we have to use the assumption that Π is contained in $\mathcal{APX}$. We thus know that there exists an approximation algorithm $\mathcal{A}$ for Π with performance ratio ϱ for some $\varrho \ge 1$. This allows us to define

$$g'(I,y,\delta) := \begin{cases} g(I,y), & \text{if } \delta \le 1/(2\beta\gamma) \\ \mathcal{A}(I), & \text{otherwise} \end{cases}$$

We claim that (f, g', α) is an AP-reduction from Π to Π^*, if we set $\alpha := 2\beta\gamma\varrho$. Note that (AP1) - (AP3) are still trivially satisfied. To show (AP4) we distinguish four cases. Consider first the case that Π is a maximization problem and that Π^* is a minimization problem. Then

$$\begin{aligned} \mathrm{val}(I, g(I,y)) &\overset{(L4)}{\ge} \mathrm{opt}(I) - \gamma\cdot(\mathrm{val}(f(I),y) - \mathrm{opt}(f(I))) \\ &\overset{(7.1)}{\ge} \mathrm{opt}(I) - \gamma\cdot\delta\cdot\mathrm{opt}(f(I)) \\ &\overset{(L3)}{\ge} \mathrm{opt}(I) - \beta\gamma\delta\cdot\mathrm{opt}(I). \end{aligned}$$

For $\delta \le 1/(2\beta\gamma)$ this implies

$$\begin{aligned} \frac{\mathrm{val}(I, g'(I,y,\delta))}{\mathrm{opt}(I)} &= \frac{\mathrm{val}(I, g(I,y)}{\mathrm{opt}(I)} \ge 1-\beta\gamma\delta \ge \frac{1}{1+2\beta\gamma\delta} \\ &\ge \frac{1}{1+2\beta\gamma\varrho\delta} = \frac{1}{1+\alpha\delta} \end{aligned}$$

by choice of α. If on the other hand $\delta > 1/(2\beta\gamma)$, we use the definition of g' and the fact that $\mathcal{A}$ is an approximation algorithm with performance ratio ϱ to deduce that also in this case

$$\frac{\mathrm{val}(I, g'(I,y,\delta))}{\mathrm{opt}(I)} = \frac{\mathrm{val}(I, \mathcal{A}(I))}{\mathrm{opt}(I)} \ge \frac{1}{\varrho} = \frac{1}{1+\alpha/(2\beta\gamma)} \ge \frac{1}{1+\alpha\delta}.$$

The case that both Π and Π^* are maximization problems, and the two cases that Π is a minimization problem are treated similarly and are left to the reader. □

7.3.2 APX-completeness

Let us recall some facts about NP-completeness. The fact that a problem Π is NP-complete means that it is at least as difficult as any other problem in $\mathcal{NP}$. In other words, that the existence of a polynomial-time algorithm for Π implies the existence of polynomial-time algorithms for every problem in $\mathcal{NP}$. The definition of APX-completeness transfers this idea to optimization problems. The main difference is that we substitute "existence of a polynomial-time algorithm" by "existence of a polynomial-time approximation scheme".

Definition 7.15 *An optimization problem* $\Pi \in \mathcal{APX}$ *is* $\mathcal{APX}$*-complete if for any other problem* $\Pi^* \in \mathcal{APX}$ *we have* $\Pi^* \leq_{AP} \Pi$.

As the AP-reduction is transitive, cf. Lemma 7.3.1, the identification of a first APX-complete problem will drastically simplify the task of proving the APX-completeness of other problems. The problem which we will first show to be APX-complete is the optimization variant MaxSat of the problem Sat which was shown to be NP-complete by Cook, cf. Theorem 7.1.1.

Theorem 7.3.6 [48] MaxSat *is* APX-*complete.*

For the proof of Theorem 7.3.6 we follow [60]. First we adapt a result from [48] that, informally speaking, guarantees that minimization problems in $\mathcal{APX}$ are not harder than maximization problems.

Lemma 7.3.7 *For every minimization problem* $\Pi \in \mathcal{APX}$ *there exists a maximization problem* $\Pi^* \in \mathcal{APX}$ *such that* $\Pi \leq_{AP} \Pi^*$.

Proof. Let $\Pi = \langle \mathcal{I}, \mathcal{S}ol, \text{val}, \min\rangle$ be a minimization problem in $\mathcal{APX}$. By definition of $\mathcal{APX}$, there exists some constant $\varrho \geq 1$ such that Π admits a polynomial-time approximation algorithm $\mathcal{A}$ with performance ratio ϱ. Without loss of generality we assume that $\varrho \in \mathbb{N}$.

For $I \in \mathcal{I}$, we denote by $\mathcal{A}(I)$ the solution $x \in \mathcal{S}ol(I)$ which is generated by $\mathcal{A}$. We define a maximization problem $\Pi^* = \langle \mathcal{I}^*, \mathcal{S}ol^*, \text{val}^*, \max\rangle$ by $\mathcal{I}^* := \mathcal{I}$, $\mathcal{S}ol^* := \mathcal{S}ol$, and

$$\text{val}^*(I, x) := \max\{1, (1+\varrho)\text{val}(I, \mathcal{A}(I)) - \varrho \cdot \text{val}(I, x)\}.$$

As val is computable in polynomial time, val* is also computable in polynomial time, implying that $\Pi^* \in \mathcal{NPO}$. To see that in fact $\Pi^* \in \mathcal{APX}$, we have to exhibit an approximation algorithm with constant performance ratio. We claim that algorithm $\mathcal{A}$ is such an algorithm. To see this, observe that

$$\text{val}^*(I, \mathcal{A}(I)) = \max\{1, (1+\varrho)\text{val}(I, \mathcal{A}(I)) - \varrho \cdot \text{val}(I, \mathcal{A}(I))\} = \text{val}(I, \mathcal{A}(I))$$

(recall that val(·) is by definition positive) and that

$$\text{opt}^*(I) = (1+\varrho)\text{val}(I, \mathcal{A}(I)) - \varrho \cdot \text{opt}(I) \leq (1+\varrho)\text{val}(I, \mathcal{A}(I)) \qquad (7.3)$$

(as opt(I) is positive). This implies

$$\frac{\text{val}^*(I, \mathcal{A}(I))}{\text{opt}^*(I)} \geq \frac{1}{1+\varrho}.$$

That is, $\mathcal{A}$ is an approximation algorithm for Π^* with performance ratio $1+\varrho$.

It remains to show that $\Pi \le_{AP} \Pi^*$. We claim that $(f, g, 1+\varrho)$ where

$$\begin{array}{rcl} f(I,\delta) & := & I, \quad \text{and} \\ g(I,y,\delta) & := & \begin{cases} y, & \text{if } \mathrm{val}(I,y) \le \mathrm{val}(I,\mathcal{A}(I)) \\ \mathcal{A}(I), & \text{otherwise} \end{cases} \end{array}$$

is an AP-reduction from Π to Π^*. Conditions (AP1)-(AP3) are obvioulsy satisfied. So it remains to verify that (AP4) holds as well. That is, we have to verify that for every $\delta > 0$:

$$\frac{1}{1+\delta} \le \frac{\mathrm{val}^*(I,y)}{\mathrm{opt}^*(I)} \qquad \Longrightarrow \qquad \frac{\mathrm{val}(I,g(I,y,\delta))}{\mathrm{opt}(I)} \le 1+(1+\varrho)\delta.$$

To see this observe first that the definitions of val* and g imply

$$\mathrm{val}^*(I,y) \le (1+\varrho)\mathrm{val}(I,\mathcal{A}(I)) - \varrho \cdot \mathrm{val}(I,g(I,y,\delta)).$$

From the assumption $\mathrm{val}^*(I,y)/\mathrm{opt}^*(I) \ge 1/(1+\delta)$ we can therefore deduce that

$$\begin{array}{rcl} \dfrac{\mathrm{val}(I,g(I,y,\delta))}{\mathrm{opt}(I)} & < & \dfrac{1+\delta}{\varrho} \cdot \dfrac{\varrho \cdot \mathrm{val}(I,g(I,y,\delta))}{\mathrm{opt}(I)} \\ & \le & \dfrac{1+\delta}{\varrho} \cdot \dfrac{(1+\varrho)\mathrm{val}(I,\mathcal{A}(I)) - \mathrm{val}^*(I,y)}{\mathrm{opt}(I)} \\ & \le & \dfrac{1+\delta}{\varrho} \cdot \left[\dfrac{(1+\varrho)\mathrm{val}(I,\mathcal{A}(I))}{\mathrm{opt}(I)} - \dfrac{\mathrm{opt}^*(I)}{(1+\delta)\mathrm{opt}(I)}\right] \\ & \overset{(7.3)}{=} & 1 + \dfrac{\delta(1+\varrho)}{\varrho} \dfrac{\mathrm{val}(I,\mathcal{A}(I))}{\mathrm{opt}(I)} \ \le \ 1+\delta(1+\varrho), \end{array}$$

where the last inequality follows from the fact that $\mathcal{A}$ is an approximation algorithm for Π with performance ratio ϱ. This proves that $(f, g, 1+\varrho)$ is indeed an AP-reduction. □

Using Lemma 7.3.7 and Theorem 7.2.7 we are now able to prove Theorem 7.3.6.

Proof of Theorem 7.3.6. By Lemma 7.3.7 and the transitivity of $\le_{AP}$, it suffices to prove that for any maximization problem $\Pi = \langle \mathcal{I}, \mathcal{S}ol, \mathrm{val}, \max\rangle \in \mathcal{APX}$ we have that $\Pi \le_{AP}$ Max3Sat. So consider an arbitrary maximization problem $\Pi = \langle \mathcal{I}, \mathcal{S}ol, \mathrm{val}, \max\rangle$. By definition of $\mathcal{APX}$, we know that there exists an approximation algorithm $\mathcal{A}$ for Π with performance ratio ϱ for some constant $\varrho \ge 1$. Let $\varepsilon_0 > 0$ be the constant from Theorem 7.2.7. Based on these two constants we will later define a suitable constant $\alpha = \alpha(\varrho, \varepsilon_0)$. Let $\delta > 0$ be given. We have to construct suitable functions f and g. First assume that δ is large enough so that $\varrho \le 1 + \alpha \cdot \delta$. In this case the approximation algorithm $\mathcal{A}$ already yields the required performance ratio. To be formally correct, we may map every instance I to some trivial MAX3SAT instance, say $f(I,\delta) \equiv x_1$ and define $g(I,y,\delta) := \mathcal{A}(I)$. Then

$$\frac{\mathrm{opt}(I)}{\mathrm{val}(I, g(I,y,\delta))} = \frac{\mathrm{opt}(I)}{\mathrm{val}(I, \mathcal{A}(I))} \leq \varrho \leq 1 + \alpha \cdot \delta,$$

and therefore the AP-condition is satisfied.

In the following we may therefore assume that $\delta > 0$ is given such that $\varrho > 1 + \alpha \cdot \delta$. To simplify notation, put $c := 1 + \alpha \cdot \delta$, $k := \lceil \log_c \varrho \rceil$ and $\mathrm{val}(\mathcal{A}_I) := \mathrm{val}(I, \mathcal{A}(I))$. We partition the interval $[\mathrm{val}(\mathcal{A}_I), \varrho \mathrm{val}(\mathcal{A}_I)]$ into k subintervals as follows:

$$[\mathrm{val}(\mathcal{A}_I), c \cdot \mathrm{val}(\mathcal{A}_I)], [c \cdot \mathrm{val}(\mathcal{A}_I), c^2 \cdot \mathrm{val}(\mathcal{A}_I)], \ldots, [c^{k-1} \cdot \mathrm{val}(\mathcal{A}_I), \varrho \cdot \mathrm{val}(\mathcal{A}_I)]$$

Since $\mathrm{val}(\mathcal{A}_I) \leq \mathrm{opt}(I) \leq \varrho \cdot \mathrm{val}(\mathcal{A}_I)$, the optimum value $\mathrm{opt}(I)$ belongs to one of the above subintervals. For $i = 0, \ldots, k-1$ consider the $\mathcal{NP}$-problem Π_i of deciding whether

$$\mathrm{opt}(I) \geq c^i \cdot \mathrm{val}(\mathcal{A}_I).$$

Since $\Pi_i \leq_p$ 3SAT, Corollary 7.1.2 implies that we can compute for every $I \in \mathcal{I}$ in polynomial time a 3SAT instance $\varphi_i := \varphi_i(I)$ such that given a truth assignment σ_i satisfying φ_i, we can compute in polynomial time a solution $x \in \mathcal{S}ol(I)$ so that $\mathrm{val}(I,x) \geq c^i \mathrm{val}(\mathcal{A}_I)$. Next we use Theorem 7.2.7 in order to compute for every φ_i another 3SAT formula $\psi_i := f(\varphi_i)$. Finally, we define a 3SAT formula $f(I,\delta)$ as

$$f(I,\delta) := \bigwedge_{i=0}^{k-1} \psi_i.$$

Notice that, since k is constant, $\psi := f(I,\delta)$ can be computed in polynomial time. In the following we assume that each ψ_i contains the same number, say m, of clauses. We can always achieve this, by taking sufficiently many copies of each ψ_i. Furthermore, we denote with i_0 the maximum index i such that ψ_i is satisfiable. Note that by our construction this implies that

$$c^{i_0} \mathrm{val}(\mathcal{A}_I) \leq \mathrm{opt}(I) < c^{i_0+1} \mathrm{val}(\mathcal{A}_I).$$

Let τ be any truth assignment for the variables of ψ such that

$$\frac{\mathrm{opt}(\psi)}{\mathrm{val}(\psi, \tau)} \leq 1 + \delta. \tag{7.4}$$

Assume that for some index i the restriction τ_i of the assignment τ to the variables in ψ_i satisfies

$$\mathrm{val}(\psi_i, \tau_i) \geq (1 - \varepsilon_0) m. \tag{7.5}$$

According to Theorem 7.2.7 this can only happen, if ψ_i is satisfiable (i.e., if $i \leq i_0$). Moreover, Theorem 7.2.7 also implies that starting from τ_i we can compute in polynomial time a satisfying assignment σ_i for φ_i. As already mentioned above, Cook's Theorem implies that we can then compute, again in polynomial time, a solution $x \in \mathcal{S}ol(I)$ such that $\text{val}(I,x) \geq c^i \text{val}(\mathcal{A}_I)$.

That is, if we can show that (7.5) holds for $i = i_0$, we can compute in polynomial time a solution $x = g(I,\tau,\delta) \in \mathcal{S}ol(I)$ such that

$$\frac{\text{opt}(I)}{\text{val}(I,x)} \leq \frac{c^{i_0+1}\text{val}(\mathcal{A}_I)}{c^{i_0}\text{val}(\mathcal{A}_I)} = c = 1 + \alpha\cdot\delta,$$

i.e., condition (AP4) would be satisfied.

Thus, in order to complete the proof, it remains to show that $\text{val}(\psi_{i_0}, \tau_{i_0}) \geq (1-\varepsilon_0)m$. According to (7.4) τ is a truth assignment for the variables of ψ such that

$$\text{opt}(\psi) - \text{val}(\psi,\tau) \leq \frac{\delta}{1+\delta}\,\text{opt}(\psi) \leq \frac{\delta}{1+\delta}\,k\cdot m.$$

On the other hand, defining ξ by $\text{val}(\psi_{i_0},\tau_{i_0}) = (1-\xi)m = (1-\xi)\text{opt}(\psi_{i_0})$, we get

$$\text{opt}(\psi) - \text{val}(\psi,\tau) = \underbrace{\sum_{i\neq i_0}(\text{opt}(\psi_i) - \text{val}(\psi_i,\tau_i))}_{\geq 0} + \underbrace{\text{opt}(\psi_{i_0}) - \text{val}(\psi_{i_0},\tau_{i_0})}_{=\xi m}.$$

Combining these two inequalities, we obtain

$$\xi \leq \frac{\delta}{1+\delta}\cdot k.$$

It thus suffices to show that we can define α (and thus $k = \lceil \ln\varrho/\ln(1+\alpha\delta)\rceil \leq 2\ln\varrho/\ln(1+\alpha\delta)$) in such a way that

$$\frac{\delta}{1+\delta}\cdot k \leq \frac{\delta}{1+\delta}\cdot\frac{2\ln\varrho}{\ln(1+\alpha\delta)} < \varepsilon_0 \qquad \text{for all } \delta > 0.$$

As

$$\frac{2\delta}{1+\delta}\cdot\frac{\ln\varrho}{\ln(1+\alpha\delta)} < \varepsilon_0 \qquad\Longleftrightarrow\qquad \frac{\varrho^{\frac{2\delta}{\varepsilon_0(1+\delta)}} - 1}{\delta} < \alpha$$

and $\frac{1}{x}(\varrho^{\frac{2x}{\varepsilon_0(1+x)}} - 1)$ is monotone decreasing for all sufficiently large x and converges to a constant for $x \to 0$, such an α obviously exists. This completes the proof of Theorem 7.3.6. □

7.4 Constructing Reductions

In this section we survey some techniques and results which are used in constructing reductions between optimization problems. We start by considering some specific examples.

7.4.1 Examples: Constraint Satisfaction Problems

Reductions between constraint satisfaction problems are particularly easy to construct. In this section we present three examples. In Section 7.4.3 we will then outline how such reductions can be constructed automatically. The reductions in this section are from Trevisan, Sorkin, Sudan, Williamson [62].

Theorem 7.4.1 MaxE3Sat $\leq_L$ MaxE2Sat.

Proof. Given an E3Sat formula F we construct an E2Sat formula F' as follows. We replace each clause $x \vee y \vee z$ in a given 3Sat instance by eight 2Sat clauses

$$x \vee z, \quad \neg x \vee \neg z, \quad x \vee \neg\xi, \quad \neg x \vee \xi, \quad z \vee \neg\xi, \quad \neg z \vee \xi, \quad y \vee \xi, \quad y \vee \xi,$$

using a new variable ξ for each clause. (Note that the last two clauses are identical!) By case checking one easily verifies that these 8 clauses have the following property:
– At most 7 of the 8 clauses can be satisfied simultaneously.
– If $x \vee y \vee z$ is satisfied than there exists an assignment for ξ such that 7 clauses are satisfied.
– If $x \vee y \vee z$ is not satisfied than at most 5 clauses can be satisfied simultaneously.
– If $x \vee y \vee z$ is not satisfied than there exists an assignment for ξ such that 5 clauses are satisfied.
One easily checks that for this construction properties (L1)-(L4) of Definition 7.14 are satisfied. (Use Remark 7.1.4 in order to see that (L3) is satisfied.) □

With some more care one can show that the above reduction also implies a non-approximability result. We defer the precise statement of such a result to Section 7.4.3.

Theorem 7.4.2 MaxE3Lin $\leq_L$ MaxE2Lin.

Proof. Given an instance I from MaxE3Lin we construct an instance $f(I)$ from MaxE2Lin as follows. We add one new variable A and, for each equation in I, 4 additional new variables. For each equation we then introduce equations as indicated in Figure 7.2: a black edge between two vertices u and v corresponds to an equation of the form $u \oplus v = 1$, while a gray edge corresponds to an equation of the form $u \oplus v = 0$. (Note that we use the notation $u \oplus v$ in order to emphasize that we consider the equations modulo two.)

Observe that MaxE2Lin is symmetric in the sense that for every assignment a of the variables the complement $\overline{a}$ satisfies the same number of equations. That is, we may assume without loss of generality that $A = 0$.

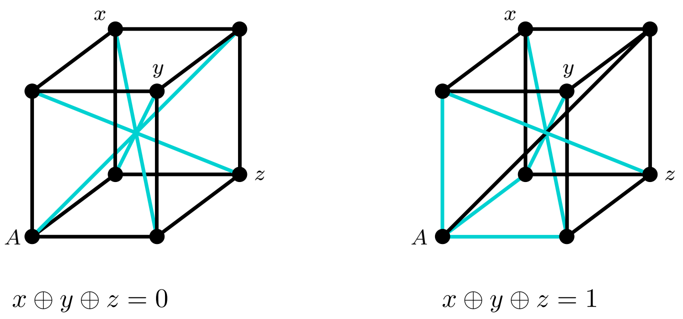


Figure 7.2. Reducing MaxE3Lin to MaxE2Lin.

Then the following properties of the construction are easily verified by case checking.

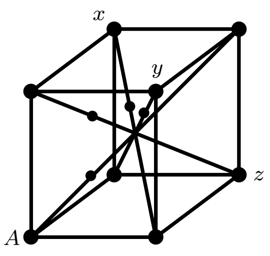
The construction for $x \oplus y \oplus z = 0$ is such that an assignment for x, y, z that satisfies the equation can be extended to an assignment of the additional (unnamed) variables such that 12 equations are satisfied, while an assignment for x, y, z that does not satisfy the equation can only be extended to an assignment that satisfies 10 equations.

Similarly, the construction for $x \oplus y \oplus z = 1$ also has the property that an assignment for x, y, z that satisfies the equation can be extended to an assignment of the additional (unnamed) variables such that 12 equations are satisfied, while an an assignment for x, y, z that does not satisfy the equation can only be extended to an assignment that satisfies 10 equations.

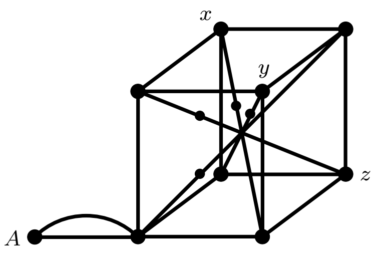
Note that this implies that $\text{opt}(f(I)) = 10|I| + 2\text{opt}(I)$, where $|I|$ denotes the number of equations in I. For every assignment a' for the variables in $f(I)$ we can also "construct" (by just restricting a' to the variables which also occur in I) an assignment a for the variables in I such that $\text{val}(f(I), a') = 10|I| + 2\text{val}(I, a)$. The conditions of Definition 7.14 are thus all satisfied. (Again we use Remark 7.1.4 in order to see that (L3) is satisfied.) □

Theorem 7.4.3 MaxE3Lin $\leq_L$ MaxCut.

Proof. Observe that MaxCut is essentially identical to MaxE2Lin, except that we may use only equations with right hand side equal to 1. That is, we can essentially use the same reduction as in the proof of Theorem 7.4.2 except that we have to get rid of the gray edges. This is easily achieved by subdividing each such edge (introducing a new variable) as indicated in the left picture in Figure 7.3. For the equation $x \oplus y \oplus z = 1$ we could, in principle, proceed similarly. However, in order to keep the number of additional variables small, we use instead the construction shown on the right hand side of Figure 7.3. The properties of the reduction remain essentially the same, only the numbers change slightly. In particular, for each former gray edge one of the two new equations is now always satisfied. While both



$x \oplus y \oplus z = 0$



$x \oplus y \oplus z = 1$

Figure 7.3. Reducing MaxE3Lin to MaxCut.

equations are satisfied if and only if the equation corresponding to the gray edge is satisfied.

That is, the construction for $x \oplus y \oplus z = 0$ is such that an assignment for x, y, z that satisfies the equation can be extended to an assignment of the additional (unnamed) variables such that 16 equations are satisfied, while an assignment for x, y, z that does not satisfy the equation can only be extended to an assignment that satisfies 14 equations.

Similarly, the construction for $x \oplus y \oplus z = 1$ is such that an assignment for x, y, z that satisfies the equation can be extended to an assignment of the additional (unnamed) variables such that 18 equations are satisfied, while an an assignment for x, y, z that does not satisfy the equation can only be extended to an assignment that satisfies 16 equations.

That is, we obtain $\mathrm{opt}(f(I)) = (14n_0 + 16n_1) + 2\mathrm{opt}(I)$, where n_i denotes the number of equations in I with right hand side i. For every assignment a' for the variables in $f(I)$ we can again construct an assignment a for the variables in I such that $\mathrm{val}(f(I), a') \leq (14n_0 + 16n_1) + 2\mathrm{val}(I, a)$. The conditions of Definition 7.14 are thus again all satisfied. □

Note that we have been slightly sloppy in the above proofs. Namely, we ignored the fact that in the constructions we obtained multiple clauses resp. edges. There are two ways to deal with this problem: either one can assume that the definition tacitly allows the use of multiple clauses or, if not, we have to add a second reduction which gets rid of multiple clauses/edges. This can in fact be done, as we will show next.

7.4.2 Weighted vs. Unweighted

The aim of this section is to show that for a wide range of optimization problems weighted versions are not harder than unweighted versions. As we will see in the subsequent section, these results will turn out to be extremely useful for designing reductions between optimization problems. The results of this section are due to Crescenzi, Silvestri, and Trevisan [26, 60].

We start by showing how to restrict arbitrary weights to those which are polynomially bounded in the input size. We will show such a result for a rather large class of problems, namely, the so-called *subset* problems. In order to first get some feeling for the upcoming definition consider the (weighted) satisfiability problem as an example. It consists of a set of variables, a set of clauses, and weight function which assigns a weight to every clause. The objective is to find an assignment of the variables such that sum of the weights of the clauses which are satisfied by this assignment is maximized. Clearly, by changing, for example, the weight function we obtain another weighted satisfiability problem. The definition of a subset problem generalizes these ideas.

An optimization problem Π is called a *subset* problem, if every instance $I \in \mathcal{I}$ consists of a tuple $I = (I_0, S, w)$, where S is a finite set, $w : S \to \mathbb{N}$ is a weight function, and every solution $x \in \mathcal{S}ol(I)$ uniquely defines a set S_x such that

$$\mathrm{val}(I, x) = \sum_{s \in S_x} w(s).$$

In addition, we require that Π is "complete" in the sense that just changing the weight function w to some other function w' leads to another legal instance of Π.

Theorem 7.4.4 *Assume $\Pi \in \mathcal{APX}$ is a subset problem. Let Π^p denote the restriction of Π to those instances $I = (I_0, S, w)$ which satisfy $w(s) \le p(|I|)$ for all $s \in S$. Then there exists a polynomial p_0 such that $\Pi \le_{AP}^{\alpha} \Pi^{p_0}$ for all $\alpha > 1$.*

Proof. We assume without loss of generality that Π is a maximization problem. (For minimization problems the proof is very similar.) Let $\mathcal{A}$ be an approximation algorithm for Π with performance ratio ϱ, where ϱ is an arbitrary constant. Such an algorithm exists, as we assumed that $\Pi \in \mathcal{APX}$. We will use $\mathcal{A}$ to construct an AP-reduction from Π to Π^{p_0}, where $p_0(n) = \varrho n^2 + 1$.

First we define the function f. For an instance $I = (I_0, S, w)$ we define a new weight function $\tilde{w} : S \to \mathbb{N}$ as follows:

$$\tilde{w}(s) = \begin{cases} \lfloor \frac{w(s) \cdot |S|^2}{\mathcal{A}(I)} \rfloor & \text{if } w(s) \le \varrho \mathcal{A}(I) \\ \varrho |S|^2 + 1 & \text{otherwise} \end{cases}$$

and let $f(I) := (I_0, S, \tilde{w})$. Clearly, $f(I) \in \Pi^{p_0}$. (Note that according to the definition of an AP-reduction the function f may also depend on an additional parameter δ. Here we do not use this freedom.) We denote the objective function of $f(I)$ by $\widetilde{\mathrm{val}}(I, x) = \sum_{s \in S_x} \tilde{w}(s)$.

Before we continue let us provide some intuition for this definition. Observe that the fact that $\mathcal{A}$ is an approximation algorithm with performance ratio ϱ implies that $\mathrm{opt}(I) \le \varrho \cdot \mathcal{A}(I)$. As Π is a maximization problem, this implies that there cannot exist a solution $x \in \mathcal{S}ol(I)$ such that S_x contains an element s of weight $w(s) > \varrho \cdot \mathcal{A}(I)$. The second case of the definition of $\tilde{w}$ thus specifies the value of elements in S which are in fact irrelevant.

Next we define the function g. Let x be an arbitrary solution in $\mathcal{S}ol(f(I)) = \mathcal{S}ol(I)$. We let

$$g(I, x, \delta) := \begin{cases} x_{\mathrm{opt}} & \text{if } |I| \le ((\alpha - 1)\delta)^{-1} \\ \mathcal{A}(I) & \text{if } \mathrm{val}(I, x) \le \mathcal{A}(I) \\ x & \text{otherwise.} \end{cases}$$

Here x_{opt} denotes an optimal solution. Clearly, g can be computed in polynomial time for any fixed $\delta > 0$. Note that this definition implies that

$$\mathrm{val}(I, g(I, x, \delta)) \ge \max\{\mathrm{val}(I, x), \mathcal{A}(I)\}. \tag{7.6}$$

It remains to show that condition (AP4) is satisfied. Consider an arbitrary $x \in \mathcal{S}ol(I)$. Then

$$\widetilde{\mathrm{val}}(I, x) = \sum_{s \in S_x} \tilde{w}(s) \le \sum_{s \in S_x} \frac{w(s) \cdot |S|^2}{\mathcal{A}(I)} = \frac{|S|^2}{\mathcal{A}(I)} \cdot \mathrm{val}(I, x) \tag{7.7}$$

and

$$\begin{aligned} \mathrm{val}(I, x) = \sum_{s \in S_x} w(s) &\le \frac{\mathcal{A}(I)}{|S|^2} \cdot \sum_{s \in S_x} \left(\left\lfloor \frac{w(s) \cdot |S|^2}{\mathcal{A}(I)} \right\rfloor + 1 \right) \\ &\le \frac{\mathcal{A}(I)}{|S|^2} \cdot \left(\widetilde{\mathrm{val}}(I, x) + |S| \right). \end{aligned}$$

Note that the last inequality implies in particular that

$$\begin{aligned} \mathrm{opt}(I) = \mathrm{val}(I, x_{\mathrm{opt}}) &\le \frac{\mathcal{A}(I)}{|S|^2} \cdot \left(\widetilde{\mathrm{val}}(I, x_{\mathrm{opt}}) + |S| \right) \\ &\le \frac{\mathcal{A}(I)}{|S|^2} \cdot \left(\mathrm{opt}(f(I)) + |S| \right). \end{aligned}$$

With these observations at hand we are now ready to verify that (AP4) holds. If $|S| \le ((\alpha - 1)\delta)^{-1}$ there is nothing to show, as in this case g computes an optimal solution anyway. So we may assume $|S| > ((\alpha - 1)\delta)^{-1}$. Then

$$\begin{aligned} \frac{\mathrm{opt}(I)}{\mathrm{val}(I, g(I, x, \delta))} &\le \frac{(\mathcal{A}(I)/|S|^2) \cdot (\mathrm{opt}(f(I)) + |S|)}{\mathrm{val}(I, g(I, x, \delta))} \\ &\overset{(7.6)}{\le} \frac{(\mathcal{A}(I)/|S|^2) \cdot \mathrm{opt}(f(I))}{\mathrm{val}(I, x)} + \frac{\mathcal{A}(I)/|S|}{\mathrm{val}(I, g(I, x, \delta))} \\ &\overset{(7.7)}{\le} \frac{\mathrm{opt}(f(I))}{\widetilde{\mathrm{val}}(I, x)} + \frac{\mathcal{A}(I)/|S|}{\mathrm{val}(I, g(I, x, \delta))} \end{aligned}$$

$$\overset{(7.6)}{\leq} \quad \frac{\text{opt}(f(I))}{\widetilde{\text{val}}(I,x)} + \frac{1}{|S|} \ \leq\ 1 + \alpha\delta$$

$$\text{whenever } \text{opt}(f(I)) \leq (1+\delta)\widetilde{\text{val}}(I,x),$$

which concludes the proof of the theorem. □

Our next aim is to reduce weighted constraint satisfaction problems to unweighted ones. Observe that Theorem 7.4.4 implies that we only have to consider weights which are polynomially bounded in the number of clauses and variables. This allows us to construct in polynomial time a new instance in which we replace every variable x_i by N copies x_i^j, $1 \leq j \leq N$, where N is a sufficiently large integer which depends polynomially on the weights of the clauses. We then replace every clause of weight w which uses variables $x_1, \ldots, x_k$, say, by w copies using variables $x_1^{j_1}, \ldots, x_k^{j_k}$. Here it is important that the tuples $(j_1, \ldots, j_k)$ are carefully chosen so that we may in fact deduce that every "good" solution of the transformed problem allows us to construct also a "good" assignment for the variables x_i. The following lemma will be useful in achieving this.

Lemma 7.4.5 *For every $k \geq 2$ and $\varepsilon > 0$ there exists an integer n_0 such that we can construct in (random) polynomial time for every pair of integers N, w, so that $N \geq n_0$ and $N^{3/2} \leq w \leq N^k$, a subset $S \subset [N]^k$ such that $|S| = w$ and*

$$(1-\varepsilon)\cdot w \cdot \frac{|A_1 \times \ldots \times A_k|}{N^k} \leq |S \cap A_1 \times \ldots \times A_k| \leq (1+\varepsilon)\cdot w \cdot \frac{|A_1 \times \ldots \times A_k|}{N^k} \tag{7.8}$$

for all subsets $A_i \subseteq [N]$ such that $|A_i| \geq \varepsilon N$.

Proof. The randomized algorithm which we are about to construct is straightforward indeed. We just choose a set S randomly and then add or delete some arbitrary elements so that S satisfies the condition $|S| = w$.

More precisely, the random construction is done as follows. We add every element in $[N]^k$ to the set S independently with probability $p = w/N^k$. In order to show that such a randomly chosen set has some nice properties, we first fix some notation. For sets $A_i \subseteq [N]$ we let $A := A_1 \times \ldots \times A_k$. Furthermore, we let $\mu := \frac{\varepsilon^{k+1}}{1+\varepsilon^k}$.

We claim that, whenever N is sufficiently large, a randomly chosen set S satisfies

$$\mathbb{P}[\,||S \cap A| - w\tfrac{|A|}{N^k}| \leq \mu w \tfrac{|A|}{N^k} \text{ for all } A_i \subseteq [N] \text{ s.t. } |A_i| \geq \varepsilon N\,] \ \geq\ 1 - 2^{-N}. \tag{7.9}$$

In order to show this, consider arbitrary sets $A_1, \ldots, A_k \subseteq [N]$ satisfying $|A_i| \geq \varepsilon N$. We apply Chernoff's inequality, see e.g. [53], to bound $|S \cap A|$. More precisely, we let X denote the size of $|S \cap A|$. Then X is the sum of $|A|$ many disjoint Bernoulli experiments with probability $p = w/N^k$. Hence,

$\mathbb{E}[X] = w \cdot |A|/N^k \geq \varepsilon^k \cdot w \geq \varepsilon^k N^{3/2}$ and Chernoff's inequality implies

$$\begin{aligned}\mathbb{P}[\,||S \cap A| - w\tfrac{|A|}{N^k}| \geq \mu w \tfrac{|A|}{N^k}\,] &= \mathbb{P}[\,|X - \mathbb{E}[X]| \geq \mu \cdot \mathbb{E}[X]\,] \\ &\leq 2e^{-\frac{1}{4}\mu^2 \mathbb{E}[X]} \leq 2e^{-\frac{1}{4}\mu^2 \varepsilon^k N^{3/2}}.\end{aligned}$$

As there are at most 2^{kN} many choices for the sets A_i, $1 \leq i \leq k$, this implies that

$$\mathbb{P}[\,\exists A_i \text{ s.t. } ||S \cap A| - w\tfrac{|A|}{N^k}| \geq \mu w \tfrac{|A|}{N^k}\,] \leq 2^{kN} \cdot 2e^{-\frac{1}{4}\mu^2 \varepsilon^k N^{3/2}}.$$

As the term tends to zero for any fixed $k \geq 2$ whenever N tends to infinity, this implies that n_0 can be chosen such that S satisfies (7.9) for all $N \geq n_0$.

To conclude the proof, assume that S is a "good" set. That is, one for which the condition of (7.9) is satisfied. Observe that this implies in particular that the set S satisfies $w(1-\mu) \leq |S| \leq w(1+\mu)$. Starting from S we can thus construct a set S' of size exactly $|S'| = w$ by arbitrarily adding or deleting at most μw elements. For all sets $A_i \subseteq [N]$ s.t. $|A_i| \geq \varepsilon N$ the new set S' then satisfies

$$\begin{aligned}|S' \cap A| &\leq |S \cap A| + |(S \setminus S') \cup (S' \setminus S)| \leq \frac{(1+\mu)w|A|}{N^k} + \mu w \\ &\leq \frac{(1+\mu)w|A|}{N^k} + \mu w \cdot \frac{|A|}{\varepsilon^k N^k} = (1+\varepsilon) \cdot \frac{w|A|}{N^k},\end{aligned}$$

where the last equality follows from the definition of μ. Similarly, we also deduce $|S' \cap A| \geq (1-\varepsilon) \cdot \frac{w|A|}{N^k}$, concluding the proof of the lemma. □

Remark 7.4.6 *Trevisan [60, 26] showed that Lemma 7.4.5 also holds if we replace "random polynomial time" by "deterministic polynomial time".*

Theorem 7.4.7 *For all $\alpha > 1$, MAXWEIGHT$\mathcal{F} \leq_{AP}^{\alpha}$ MAX$\mathcal{F}$.*

Proof. Note that Theorem 7.4.4 can be applied to MAXWEIGHT$\mathcal{F}$. As the AP-reduction is transitive, it therefore suffices to show that MAXWEIGHT$\mathcal{F}^p \leq_{AP}^{\alpha}$ MAX$\mathcal{F}$ for all polynomials p. Fix any such polynomial p. Recall that in the definition of an AP-reduction the functions f and g may depend on a parameter δ. Here we will use this possibility. In the following we describe the reduction for an arbitrary, but fixed $\delta > 1$. First we choose a parameter $\varepsilon > 0$ such that

$$\frac{1}{(1+\delta)(1+\varepsilon)} - \frac{(1-(1-\varepsilon)^{\ell+1}) \cdot 2^\ell}{1+\varepsilon} \geq \frac{1}{1+\alpha\delta}.$$

(Observe that such an ε exists, as the left hand side tends to $1/(1+\delta)$ for $\varepsilon \to 0$ and α is a constant greater than one.) Let ℓ be the largest arity of a constraint in $\mathcal{F}$. Choose n_0 large enough so that Lemma 7.4.5 holds for the ε chosen above, all $N \geq n_0$, and all $k = 2, \ldots, \ell$.

Consider an instance I of MAXWEIGHT$\mathcal{F}^p$. Assume I consists of n variables $x_1, \ldots, x_n$ and m constraints $C_1, \ldots, C_m$ with weights $w_1, \ldots, w_m$.

Let $w_{\max} := \max_i w_i$. Note that $w_{\max} \leq p(n+m)$. Observe that we may assume without loss of generality that $w_i \geq (n_0)^{3/2} \cdot (w_{\max})^{3/4}$ for all $1 \leq i \leq n$. (If this is not the case a priori, consider instead the scaled problem with weights $w_i' := \xi \cdot w_i$. For $\xi \geq n_0^6 \cdot (w_{\max})^3$ we then trivially have $w_i' \geq \xi \geq n_0^{3/2} \cdot (\xi \cdot w_{\max})^{3/4} = n_0^{3/2} \cdot (w_{\max}')^{3/4}$.) Let $N := n_0 \cdot \sqrt{w_{\max}}$. Then $N^{3/2} \leq w_i \leq N^2$ for all $1 \leq i \leq n$. That is, we may apply Lemma 7.4.5 for all w_i.

We now describe how to transform the instance I in an unweighted instance $\tilde{I}$ of MAX$\mathcal{F}$. First we replace each variable x_i by a set of variables

$$X_i := \{x_i^1, \ldots, x_i^N\}.$$

Then we replace each constraint C_j of weight w_j by w_j suitably defined constraints $C_j^1, \ldots, C_j^{w_j}$. In order to define the constraints C_j^k let us assume for simplicity of notation that $C_j = (f, x_1, \ldots, x_h)$, where $f \in \mathcal{F}$ is an h-ary function. Then the C_j^k are chosen such that

$$\{C_j^1, \ldots, C_j^{w_j}\} = \{(f, x_1^{j_1}, \ldots, x_h^{j_h}) \mid (j_1, \ldots, j_h) \in S_j\},$$

where $S_j \subset [N]^h$ denotes the set S according to Lemma 7.4.5 for the parameters ε, h, N and w_j.

Consider an arbitrary assignment $a = (a_i)$ for the variables x_i. Let $w(a)$ denote the total weight of all satisfied clauses in I. By letting $x_i^j := a_i$ for all i and j, we obtain an assignment $\tilde{a}$ for the variables in $\tilde{I}$ such that exactly $w(a)$ constraints in $\tilde{I}$ are satisfied. In particular we therefore have

$$\operatorname{opt}(\tilde{I}) \geq \operatorname{opt}(I). \tag{7.10}$$

Now consider an arbitrary assignment $\tilde{a} = (a_i^j)$ for the variables x_i^j in $\tilde{I}$. Our aim is to construct an assignment for I such that $w(a) \geq \tilde{w}(\tilde{a})/(1+\varepsilon)$, where $\tilde{w}(\tilde{a})$ denotes the number of satsified constraints in $\tilde{I}$.

In every set X_i we arbitrarily switch the value of εN variables x_i^j in such way that afterwards there exist at least εN many variables with value TRUE and at least εN many variables with value FALSE. Let $\tilde{a}'$ be the resulting assignment for the variables in $\tilde{I}$. In order to bound the difference $\tilde{w}(\tilde{a}) - \tilde{w}(\tilde{a}')$ we consider a constraint $C_j = (f, x_1, \ldots, x_h)$. Let $A_i \subseteq X_i$ denote the variables for which the truth value was *not* switched, then $|A_i| \geq (1-\varepsilon)N$. Lemma 7.4.5 thus implies

$$|S_j \cap A_1 \times \ldots \times A_h| \geq (1-\varepsilon) \cdot w_j \cdot (1-\varepsilon)^h \geq (1-\varepsilon)^{\ell+1} \cdot w_j,$$

as ℓ was chosen to denote the maximum arity of a constraint function in $\mathcal{F}$.

That is, at most $|S_j| - |S_j \cap A_1 \times \ldots \times A_h| \leq (1-(1-\varepsilon)^{\ell+1}) \cdot w_j$ many constraints C_j^i may be satisfied for $\tilde{a}$ but not for $\tilde{a}'$ or vice versa. Hence

$$|\tilde{w}(\tilde{a}') - \tilde{w}(\tilde{a})| \leq \sum_{j=1}^{m} (1-(1-\varepsilon)^{\ell+1}) \cdot w_j \leq (1-(1-\varepsilon)^{\ell+1}) \cdot 2^\ell \operatorname{opt}(\tilde{I}), \tag{7.11}$$

where the last inequality follows from Remark 7.1.4. (A random assignment satisfies at least a fraction of $1/2^\ell$ of the $\sum_{j=1}^m w_j$ many clauses in $\tilde{I}$.)

Starting from $\tilde{a}' = (a'^j_i)$ we now construct an assignment a for I such that $w(a) \geq \tilde{w}(\tilde{a}')$. In a first step we construct such an assignment randomly. Later we will see that this random construction can in fact be derandomized. Let

$$T_i := \{1 \leq j \leq N \mid a'^j_i = \text{TRUE}\} \qquad \text{and} \qquad p_i := \frac{|T_i|}{N}$$

and set the variable x_i to TRUE with probability p_i. What can we say about the expectation of the weight $w(a)$ of the resulting assignment a? Consider again a constraint $C_j = (f, x_1, \ldots, x_h)$. The probability that C_j is satisfied is equal to the sum of the probabilities $\mathbb{P}[x_1 = b_1 \wedge \ldots \wedge x_h = b_h]$, where the sum is over all tuples $\vec{b} = (b_1, \ldots, b_h)$ such that $f(\vec{b}) = f(b_1, \ldots, b_h) = 1$. Observe that by construction

$$\mathbb{P}[x_1 = b_1 \wedge \ldots \wedge x_h = b_h] = \frac{|B_1 \times \ldots \times B_h|}{N^h},$$
$$\text{where } B_i = \begin{cases} T_i, & b_i = 1 \\ [N] \setminus T_i, & \text{otherwise.} \end{cases}$$

As $|B_i| \geq \varepsilon N$, we know from Lemma 7.4.5 that the set S_j satisfies

$$|S_j \cap (B_1 \times \ldots \times B_h)| \leq (1+\varepsilon) \cdot w_j \cdot \frac{|B_1 \times \ldots \times B_h|}{N^h}$$

and hence

$$\mathbb{P}[x_1 = b_1 \wedge \ldots \wedge x_h = b_h] \geq \frac{|S_j \cap (B_1 \times \ldots \times B_h)|}{w_j \cdot (1+\varepsilon)}. \tag{7.12}$$

As $\tilde{w}(\cdot)$ just counts the number of satisfied constraints, we also have

$$\tilde{w}(\tilde{a}') = \sum_{j=1}^m \sum_{\vec{b}: f(\vec{b})=1} |S_j \cap (B_1 \times \ldots \times B_h)|. \tag{7.13}$$

Combining these observations we conclude

$$\begin{aligned}
\mathbb{E}[w(a)] &= \sum_{j=1}^m w_j \cdot \mathbb{P}[C_j \text{ is satisfied}] \\
&= \sum_{j=1}^m w_j \cdot \sum_{\vec{b}: f(\vec{b})=1} \mathbb{P}[x_1 = b_1 \wedge \ldots \wedge x_h = b_h] \\
&\overset{(7.12)}{\geq} \sum_{j=1}^m \sum_{\vec{b}: f(\vec{b})=1} \frac{|S_j \cap (B_1 \times \ldots \times B_h)|}{1+\varepsilon} \\
&\overset{(7.13)}{=} \frac{\tilde{w}(\tilde{a}')}{1+\varepsilon}
\end{aligned}$$

$$\overset{(7.11)}{\geq} \frac{\tilde{w}(\tilde{a}) - (1-(1-\varepsilon)^{\ell+1}) \cdot 2^{\ell}\mathrm{opt}(\tilde{I})}{1+\varepsilon}.$$

Using the method of conditional probabilities (cf. e.g. [53]) we can also *construct* in polynomial time an assignment a_0 for I such that

$$w(a_0) \geq \mathbb{E}[w(a)] \geq \frac{\tilde{w}(\tilde{a}) - (1-(1-\varepsilon)^{\ell+1}) \cdot 2^{\ell}\mathrm{opt}(\tilde{I})}{1+\varepsilon}.$$

Rewriting the last inequality we deduce

$$\frac{w(a_0)}{\mathrm{opt}(I)} \overset{(7.10)}{\geq} \frac{\tilde{w}(\tilde{a})}{(1+\varepsilon)\cdot \mathrm{opt}(\tilde{I})} - \frac{(1-(1-\varepsilon)^{\ell+1}) \cdot 2^{\ell}}{1+\varepsilon}.$$

In particular,

$$\frac{\tilde{w}(\tilde{a})}{\mathrm{opt}(\tilde{I})} \geq \frac{1}{1+\delta} \implies \frac{w(a_0)}{\mathrm{opt}(I)} \geq \frac{1}{(1+\delta)(1+\varepsilon)} - \frac{(1-(1-\varepsilon)^{\ell+1}) \cdot 2^{\ell}}{1+\varepsilon} \geq \frac{1}{1+\alpha\cdot\delta}$$

by the choice of ε. □

7.4.3 Constructing optimal gadgets

A major advantage of constraint satisfaction problems is that it is very easy to construct reductions between such problems. The reason is that we only have to consider the underlying constraint functions, but not the global structure of the problem. This allows the construction of reductions by "local replacements". In the context of non-approximability for optimization problems such an approach was first successfully pursued by Bellare, Goldreich and Sudan [15] and then extended and further improved by Trevisan, Sorkin, Sudan, and Williamson [62]. Here we largely follow their exposition.

Definition 7.16 *Let f be a k-ary constraint function and $\mathcal{F}$ be a constraint family, where typically $f \notin \mathcal{F}$. An (α, β)-*gadget *reducing f to $\mathcal{F}$ is a finite collection $\mathcal{G}ad = (f_1, \ldots, f_r)$ of constraint functions f_i from $\mathcal{F}$ involving (primary) variables $x_1, \ldots, x_k$ and auxiliary variables $y_1, \ldots, y_s$ such that the following properties are satisfied:*

$$\forall a \in \{0,1\}^k \text{ s.t. } f(a) = 1: \quad \max\{\sum_{i=1}^{r} f_i(a,b) \mid b \in \{0,1\}^s\} = \alpha \quad (7.14)$$

$$\forall a \in \{0,1\}^k \text{ s.t. } f(a) = 0: \quad \max\{\sum_{i=1}^{r} f_i(a,b) \mid b \in \{0,1\}^s\} \leq \alpha - \beta \quad (7.15)$$

The gadget is called strict *if* (7.15) *holds with equality.*

A weighted α-gadget *is a gadget* $\mathcal{G}ad = (f_1, \dots, f_r)$ *together with weights* $(w_1, \dots, w_r)$ *such that*

$$\forall a \in \{0,1\}^k \ s.t. \ f(a) = 1 : \qquad \max\{\sum_{i=1}^{r} w_i f_i(a,b) \mid b \in \{0,1\}^s\} = \alpha \tag{7.16}$$

$$\forall a \in \{0,1\}^k \ s.t. \ f(a) = 0 : \qquad \max\{\sum_{i=1}^{r} w_i f_i(a,b) \mid b \in \{0,1\}^s\} = \alpha - 1 \tag{7.17}$$

Note that every strict (α, β)-gadget induces a weighted $\alpha/(\alpha-\beta)$-gadget, if we let $w_i := 1/(\alpha - \beta)$.

Gadgets are a very useful tool for constructing reductions between constraint satisfaction problems. We exemplify this for the case that the constraint family consists of just a single function, i.e. $\mathcal{F} = \{h\}$.

Lemma 7.4.8 *Assume* $\mathcal{G}ad$ *is a strict* (α,β)*-gadget between the* k*-ary constraint function* h *and a constraint family* $\mathcal{F}$*. Then*

$$\textsc{Max}\{h\} \leq_L \textsc{Max}\mathcal{F}.$$

Proof. Consider an instance I from $\textsc{Max}\{h\}$. We let $f(I)$ be the instance which is obtained by replacing every constraint C from $\textsc{Max}\{h\}$ by the collection of constraints from $\textsc{Max}\mathcal{F}$ given by the gadget $\mathcal{G}ad$, where we choose as primary variables the variables from C. Note that the auxiliary variables are different for each constraint C. If we let $|I|$ denote the number of constraints in I, then one easily checks that the following two properties are satisfied:

$$\text{opt}(f(I)) = \beta \cdot |I| + (\alpha - \beta)\text{opt}(I).$$

and

$$\text{val}(f(I), a) \leq \beta \cdot |I| + (\alpha - \beta)\text{val}(I, a'),$$

where a' denotes the restriction of a to the variables occurring in I. Recalling (cf. Remark 7.1.4) that there exists a constant $c = c(h)$ such that $\text{opt}(I) \geq c \cdot |I|$ we conclude that the conditions of Definition 7.14 are indeed all satisfied. □

Remark 7.4.9 *The reader is invited to check that Theorems 7.4.1, 7.4.2, and 7.4.3 were in fact proven by constructing appropriate gadgets.*

Using the non-approximability results from Section 7.2.4 the existence of a gadget also carries over to non-approximability bounds. We only examplify this for $\textsc{MaxE3Lin}$.

Lemma 7.4.10 *Let* $\mathcal{F}$ *be some constraint family. Assume there exist an* α_0*-gadget reducing* $x \oplus y \oplus z = 0$ *to* $\mathcal{F}$*, and an* α_1*-gadget reducing* $x \oplus y \oplus z =$

1 to $\mathcal{F}$. Then MAX$\mathcal{F}$ *cannot be approximated within* $\frac{\alpha_0+\alpha_1}{\alpha_0+\alpha_1-1} - \varepsilon$ *for all* $\varepsilon > 0$.

Proof. We aim at applying Theorem 7.2.11. For conciseness we assume that $\alpha_0 \leq \alpha_1$. The other case is treated similarly. Consider an instance I of MAXE3LIN. Let n_i denote the number of equations with right hand side i. Note that we may assume without loss of generality that $n_0 \geq n_1$. (Otherwise we just replace each right hand side of 1 by 0 and vice versa, and observe that the complement of an assignment a satisfies a new equation if and only if a satisfies the old equation.)

Now use the reduction from Lemma 7.4.8. Then $\text{opt}(f(I)) = (\alpha_0-1)n_0 + (\alpha_1-1)n_1 + \text{opt}(I)$. If $\text{opt}(I) \geq (1-\varepsilon)|I| = (1-\varepsilon)(n_0+n_1)$, then $\text{opt}(f(I)) \geq \alpha_0 n_0 + \alpha_1 n_1 - \varepsilon|I|$. If $\text{opt}(I) \leq \frac{1}{2}(1+\varepsilon)|I|$, then $\text{opt}(f(I)) \leq (\alpha_0 - \frac{1}{2})n_0 + (\alpha_1 - \frac{1}{2})n_1 + \frac{1}{2}\varepsilon|I|$. As

$$\begin{aligned}\frac{\alpha_0 n_0 + \alpha_1 n_1 - \varepsilon|I|}{(\alpha_0 - \frac{1}{2})n_0 + (\alpha_1 - \frac{1}{2})n_1 + \frac{1}{2}\varepsilon|I|} &= 1 + \frac{\frac{1}{2}n_0 + \frac{1}{2}n_1 - \frac{3}{2}\varepsilon|I|}{(\alpha_0 - \frac{1}{2})n_0 + (\alpha_1 - \frac{1}{2})n_1 + \frac{1}{2}\varepsilon|I|} \\ &\geq 1 + \frac{1-3\varepsilon}{\alpha_0+\alpha_1-1+\varepsilon}\end{aligned}$$

the claim of the lemma follows from Theorem 7.2.11. □

Using the gadgets from Theorems 7.4.2, and 7.4.3 we obtain the following non-approximability results for MAXE2LIN and MAXCUT.

Corollary 7.4.11 *For every $\varepsilon > 0$,* MAXE2LIN *is not approximable within a factor of* $12/11 - \varepsilon$, *unless* $\mathcal{P} = \mathcal{NP}$.

Corollary 7.4.12 *For every $\varepsilon > 0$,* MAXCUT *is not approximable within a factor of* $17/16 - \varepsilon$, *unless* $\mathcal{P} = \mathcal{NP}$.

In the remainder of this section we will see that for a large class of constraint families the search for good gadgets can be computerized.

Definition 7.17 *A constraint family $\mathcal{F}$ is called* hereditary *if for any k-ary function $f \in \mathcal{F}$ and any two indices $1 \leq i < j \leq k$ the function f when restricted to $x_i \equiv x_j$ and considered as a function of $k-1$ variables is identical to some other function $f' \in \mathcal{F}$.*

Lemma 7.4.13 *Assume $\mathcal{G}ad$ is a strict (α,β)-gadget between the k-ary constraint function h and a hereditary constraint family $\mathcal{F}$. Then there also exists an (α,β)-gadget $\mathcal{G}ad'$ between the k-ary constraint function h and a constraint family $\mathcal{F}$ that uses at most 2^{2^k} many auxiliary variables.*

Proof. The characterizing property of a strict gadget is that every assignment a for the primary variables can be extended by an assignment b of the auxiliary variables such that equations (7.14) or (7.15) hold with equality.

For every a we fix one such assignment b_a. Assume $\mathcal{G}ad$ contains more than 2^{2^k} auxiliary variables. As there exactly 2^k different assignments for the primary variables, this implies that there have to exist two different auxiliary variables y_i and y_j that have the same values in b_a for all assignments of the primary variables. As $\mathcal{F}$ was assumed to be hereditary, we may thus replace every occurrence of the variable y_j by y_i to obtain a gadget which uses one less auxiliary variable. Repeating this argument until the number of auxiliary variables is at most 2^{2^k} concludes the proof of the lemma. □

Lemma 7.4.13 implies that within the search of a good α-gadget for a given k-ary function f we may restrict our attention to a finite number of constraints. Namely, those which are built from a function from $\mathcal{F}$ and use variables $x_1, \ldots, x_k, y_1, \ldots, y_{2^{2^k}}$. If we denote these constraints by C_j, it thus suffices to compute the optimal weights w_j. This is easily done by solving the following linear program with variables w_j and α.

$$
\begin{array}{lllrcl}
\text{(LP)} & \text{minimize } \alpha & & & & \\
& \text{subject to} & & & & \\
& \forall a : f(a) = 1, \forall b: & & \sum_j w_j C_j(a,b) & \leq & \alpha \\
& \forall a : f(a) = 1: & & \sum_j w_j C_j(a,b_a) & = & \alpha \\
& \forall a : f(a) = 0, \forall b: & & \sum_j w_j C_j(a,b) & \leq & \alpha - 1 \\
& \forall a : f(a) = 0: & & \sum_j w_j C_j(a,b_a) & = & \alpha - 1 \\
& & & \alpha & \geq & 0 \\
& \forall j: & & w_j & \geq & 0.
\end{array}
$$

Theorem 7.4.14 *Every optimal solution to (LP) corresponds to an optimal α-gadget, i.e., an α-gadget where α is as small as possible.* □

Note that the above phrase "This is easily done by solving the following linear program..." should of course be handled with care. Theoretically, this is indeed easy. In practice, the fact that the number of auxiliary variables grows double exponentially in the arity of the function h, makes the task of solving the LP not quite so easy. Nevertheless the linear programs have been solved for many specific functions h and constraint families $\mathcal{F}$. In particular, Theorem 7.4.14 has been used to show [62] that the gadgets which were used in the proofs of Section 7.4.1 are in fact optimal.

7.4.4 Randomized reductions

For many reductions it is essential to construct instances which have certain properties. Thereby it sometimes happens that one can show that instances satisfying a given property *exist*, but one does not know how to *construct* such an instance efficiently (cf. Section 7.4.5 for examples). In the case that there do exist sufficiently many "good" instances, it is sometimes a good strategy to just choose an instance "randomly" hoping that one does indeed hit a "good" instance. Proceeding in such a way, leads to randomized reductions and non-approximability results based on the assumption $\mathcal{RP} \neq \mathcal{NP}$. In the remainder of this section we make these ideas precise. We assume the reader to be familiar with the notion of randomized algorithms and refer to the textbook by Motwani and Raghavan [53] for additional background on this topic.

Definition 7.18 *A* randomized AP-reduction *is defined as in Definition 7.13, except that we replace conditions (AP3) and (AP4) by*

(AP3') *For any fixed $\delta > 0$ the function g is computable in polynomial time, and the function f is computable in polynomial time by a randomized algorithm.*

(AP4') *For any $I \in \mathcal{I}$, for any $\delta > 0$, $f(I, \delta)$ satisfies with probability at least $1/2$ the following property for all $y \in \mathcal{S}ol^*(f(I, \delta))$:*

$$\begin{aligned} \frac{1}{1+\delta} &\le \frac{\mathrm{val}(f(I,\delta),y)}{\mathrm{opt}(f(I,\delta))} \le 1+\delta \quad \Longrightarrow \\ \frac{1}{1+\alpha\cdot\delta} &\le \frac{\mathrm{val}(g(I,y,\delta))}{\mathrm{opt}(I)} \le 1+\alpha\cdot\delta. \end{aligned}$$

We use the notation $\Pi \le_{rAP} \Pi^*$ to indicate that Π is reducible to Π^* by a randomized AP-reduction. If we want to emphasize that there exists a randomized AP-reduction for some specific value of α, we also write $\Pi \le_{rAP}^{\alpha} \Pi^*$.

Reductions have the property that they carry over the existence or non-existence of algorithms of particular properties like running time or approximation ratio. Due to the use of randomization within the construction of the instance $f(I)$ the existence of a randomized AP-reduction allow only statements about existence or non-existence of *randomized* algorithms. With this difference in mind we could now rephrase all facts and lemmas previously obtained for AP-reductions also for randomized AP-reductions. We examplify this for just two cases.

Lemma 7.4.15 *Let $\Pi, \Pi^* \in \mathcal{NPO}$. If $\Pi^* \in \mathcal{PTAS}$ and $\Pi \le_{rAP} \Pi^*$ then $\Pi \in \mathcal{PRAS}$. ($\mathcal{PRAS}$ denotes the set of all optimization problems in $\mathcal{NPO}$ which admit a randomized polynomial-time approximation scheme.)* □

Lemma 7.4.16 *Let $\Pi \in \mathcal{NPO}$ and $\varepsilon > 0$. If* MaxE3Sat$\le_{rAP}^{\alpha}$ Π *then there cannot exist a (random) polynomial time approximation algorithm for Π with performance ratio $1 + \frac{1}{7\alpha} - \varepsilon$, unless $\mathcal{RP} = \mathcal{NP}$.* □

The advantage of randomized reductions is that it is sometimes much easier to construct gadgets by a probabilistic method than in a deterministic fashion. Consider e.g. the construction in Lemma 7.4.5. Constructing the desired set S was easy indeed. On the other hand, the deterministic construction mention in Remark 7.4.6 is seemingly more complicated. At the end of the next section we will also state some results using randomized reductions, where a matching deterministic counterpart has not been found up to now.

7.4.5 Expander

For decision problems the satisfiability problem SAT plays a major rôle. Not only because it was the first problem shown to be NP-complete, but also because over the years it has shown to be very often a good candidate for constructing reductions in order to show the NP-completeness of other decision problems.

For optimization problems a natural way to construct AP- or L-reductions is to reconsider the reduction for the corresponding decision problem – in the hope that it can be transformed into an AP-reduction. In some cases such an approach works in a straightforward way. In many cases, however, one runs into the following problem: the reductions start not from SAT or 3SAT directly, but from a variant where the number of occurrences of each variable is bounded by some constant. We therefore would need a result saying that the corresponding problem MAX3SAT(d) is APX-complete as well.

For decision problems the reduction from SAT to SAT(3) is very simple, indeed: one just replaces each variable x_i by k variables $x_i^1, \dots, x_i^k$, where k is the number of occurrences of variable x_i, replaces the jth occurrence of x_i by x_i^j, and adds additional clauses

$$\bar{x}_i^1 \vee x_i^2, \quad \bar{x}_i^2 \vee x_i^3, \quad \dots, \quad \bar{x}_i^{k-1} \vee x_i^k, \quad \bar{x}_i^k \vee x_i^1.$$

As one easily checks, that the additional clauses have the property that they are all satisfied if and only if all x_i^j's are set to the same truth value, this construction obviously has the desired properties.

Unfortunately, the attempt to reuse this reduction for showing that MAX3SAT $\leq_{AP}$ MAX3SAT(d) fails. To see why, just consider what happens if k is large and we set e.g. the first ℓ of the x_i^j's to true and the remaining ones to false. What one needs is a construction which adds only a "few" additional clauses, which nevertheless ensure that in all "reasonable" solutions all variables x_i^j are set to the same value. The following lemma explains such a construction for a special case.

Lemma 7.4.17 *For all $k \geq 2$,* MAXkSAT(5) $\leq_L$ MAXkSAT(3).

Proof. Let F be an instance from MAXkSAT(5). Starting from F we construct an instance from MAXkSAT(3) as follows. For each variable x_i in F we introduce 5 new variables $x_i^1, \dots, x_i^5$ and replace the jth occurrence of x_i by x_i^j. In addition, we add for each variable 10 additional new variables and 20 new clauses as indicated in Figure 7.4. For an edge directed from u

to v we add a clause $\bar{u} \vee v$. Let F' be the resulting 2SAT instance. From the construction it is immediately clear that each variable occurs in at most 3 clauses.

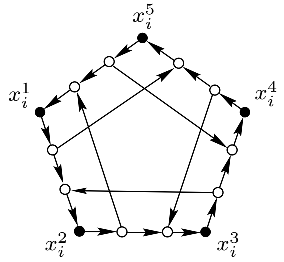


Figure 7.4. Reducing MAXkSAT(5) to MAXkSAT(3).

Consider an arbitrary assignment for F' in which for some i some of the five variables $x_1^1, \ldots, x_i^5$ are set to true and some are set to false. Observe what happens if we set for this i all variables (the x_i^j's and the additional variables according to Figure 7.4) to the majority of the truth values occurring among the x_i^j. This will change the value of one or two variables of the x_i^j's, implying that at most one resp. two of the "old" clauses might not be satisfied any more. On the other hand, straightforward case checking shows that there always will be at least one resp. two "new" clauses which are satisfied now, but hadn't been satisfied before. Thus, $\mathrm{opt}(F') = \mathrm{opt}(F) + 20n$ and for every assignment a' for F' we can construct (in polynomial time) an assignment a for F such that $\mathrm{opt}(F') - \mathrm{val}(F', a') = \mathrm{opt}(F) - \mathrm{val}(F, a)$. □

Can we extend the construction from Lemma 7.4.17 in order to show that MAX3SAT $\leq_L$ MAX3SAT(3)? In principle, this is plausible. We would just need to show that there exist digraphs similar to the one in Figure 7.4 for arbitrarily many x_i^j's. Namely, graphs $D_k = (U \cup V, A)$ with the following properties: $|U| = k$, $deg(x) = 2$ for all $x \in U$, $deg(x) = 3$ for all $x \in V$ and

$$\forall X \subseteq U \cup V, |X \cap U| \leq \tfrac{1}{2}|U| : cut^+(X) \geq |X|,$$

where $cut^+(X)$ denotes the number of edges (u, v) such that $u \in X$ and $v \notin X$.

Constructing such graphs is, however, a very difficult task – which is far from being completely solved. In the remainder of this chapter we state some notions and results which have been used in attacking this problem and comment their applicability.

Definition 7.19 *A graph $G = (V, E)$ is called a c-expander if it satisfies the following condition:*

$$|\Gamma(X)| \geq c \cdot |X| \qquad \forall X \subseteq V, |X| \leq \tfrac{1}{2}|V|.$$

Of course, it is trivial to come up with a c-expander for every $c \leq 1$: every complete graph is such an expander. It is also not too difficult to check that a random graph $G_{n,p}$ with edge probability $p = d/n$ is with probability $1 - o(1)$ a c-expander, whenever $d = d(c)$ is a sufficiently large constant. In addition, Ajtai [1] describes a deterministic, polynomial-time construction for a 3-regular c_0-expander, where $0 < c_0 < 1$ is an appropriate constant. More details on the construction of expanders can e.g. be found in [2, 44].

Definition 7.20 *A graph $G = (V, E)$ is called an amplifier for a set $S \subseteq V$ if it satisfies the following condition:*

$$|cut(X)| \geq |X \cap S| \qquad \forall X \subseteq V, |X \cap S| \leq \tfrac{1}{2}|S|.$$

Amplifiers are the type of graphs which are most useful in constructing reductions for optimization problems. Unfortunately, the quality of the reduction (or, equivalently, the resulting non-approximability result) depends heavily on two properties of the amplifier: the maximum degree and the relation of the cardinality of the set S to the number of vertices in $V \setminus S$. The following lemma exemplifies this.

Lemma 7.4.18 *Assume there exists an algorithm that constructs for a given set S in polynomial time an amplifier $G = (S \cup T, E)$ for S such that every vertex in S has degree $d-1$, every vertex in T has degree d and such that $|T| = c \cdot |S|$. Then there cannot exist a polynomial-time approximation algorithm for* MaxE2Lin(d) *with performance ratio $\frac{16d(c+1)-4}{16d(c+1)-5} - \varepsilon$, unless $\mathcal{P} = \mathcal{NP}$.*

Proof. First we reconsider the proof of Lemma 7.4.2. Observe that this reduction can be viewed as a gap-reduction in the sense of Theorem 7.2.4: it transforms a MaxE3Lin instance with n equations such that either at least $(1-\varepsilon)n$ or at most $\frac{1}{2}(1+\varepsilon)n$ many equations are satisfiable simultaneously (cf. Theorem 7.2.11), into a MaxE2Lin instance with 16n equations such that either at least $12(1-\varepsilon)n + 10\varepsilon n = 12(1-\frac{1}{6}\varepsilon)n$ or at most $12\frac{1}{2}(1+\varepsilon)n + 10\frac{1}{2}(1-\varepsilon) = 11(1+\frac{1}{11}\varepsilon)n$ many equations can be satisfied simultaneously.

Starting from this MaxE2Lin instance we now use the amplifier to construct a MaxE2Lin(d) instance. We do this by using a similar construction as in the proof of Lemma 7.4.17. That is, we replace every variable x_i by a set of variables $x_i^1, \ldots, x_i^k$, where k is the number of occurrences of variable x_i, and replace the jth occurrence of x_i by x_i^j. In addition, we construct for each i an amplifier for the set $\{x_i^1, \ldots, x_i^k\}$ and add an equation $u \oplus v = 0$ for each edge $\{u, v\}$ of the amplifier. Observe that the defining properties of an amplifier imply that changing an arbitrary assignment for the x_i^j's to one where all x_i^j (and all additional variables in the amplifier) are set to the value of the majority of x_i^j's will never decrease the number of satisfied equations. That is, without loss of generality we only have to consider assignments such that for each amplifier all variables have the same value –

implying that all equations, which correspond to an edge of an expander, are satisfied.

The following table summarizes the properties of these two reductions:

	MaxE3Lin	MaxE2Lin	MaxE2Lin(d)
equations	n	$16n$	$16n + 32n \cdot \frac{1}{2}(d-1+cd)$
satisfiable			$= 16(d+cd)n$
either $\geq$	$(1-\varepsilon)n$	$12(1-\frac{1}{6}\varepsilon)n$	$12(1-\frac{1}{6}\varepsilon)n + 16(cd+d-1)n$
			$\geq (16d(c+1)-4)(1-\varepsilon)n$
or $\leq$	$\frac{1}{2}(1+\varepsilon)$	$11(1+\frac{1}{11}\varepsilon)n$	$11(1+\frac{1}{11}\varepsilon)n + 16(cd+d-1)n$
			$\leq (16d(c+1)-5)(1+\varepsilon)n$

The claimed result follows immediately from Theorems 7.2.11 and 7.2.4. □

In [55] Papadimitriou and Yannakakis describe a construction which transforms a c-expander $G = (S, E)$ with maximum degree d into an amplifier $G' = (S \cup T, E')$ for S such that the maximum degree of G' is $d + 1$ and $|T| \approx 2|S|/c$. Combining this with a variant of Lemma 7.4.18 (for Max3Sat instead of MaxE3Lin) and subsequently applying Lemma 7.4.17 shows that Max3Sat $\leq_{AP}$ Max3Sat(3), implying in particular that Max3Sat(3) is APX-complete as well.

Unfortunately, the detour of using expanders (which measure vertex expansion) in order to construct amplifiers (which measure edge expansion) does not give very good results with respect to the maximum degree and the size of the set $V \setminus S$. In [17] Berman and Karpinski pursue another approach: they construct an amplifier directly. More precisely, they construct a graph similarly as in Figure 7.4. Namely, they start with a circle which contains alternatingly one vertex from S and then 6 vertices from T. In a second step they add a random matching between the vertices in T. Finally, they show that the resulting graph is with high probability an amplifier for the set S. Plugging the numbers into Lemma 7.4.18 gives the following result.

Theorem 7.4.19 [17] *There cannot exist a polynomial-time approximation algorithm for* MaxE2Lin(3) *with performance ratio* $332/331 - \varepsilon$, *unless* $\mathcal{RP} = \mathcal{NP}$. □

By slightly modifying this construction Berman and Karpinski also obtain non-approximability results for several other optimization problems. Here we just state the corresponding result for MaxE2Sat.

Theorem 7.4.20 [17] *There cannot exist a polynomial-time approximation algorithm for* MaxE2Sat(3) *with performance ratio* $2012/2011 - \varepsilon$, *unless* $\mathcal{RP} = \mathcal{NP}$. □

7.5 Open Problems

We close this survey by stating three famous problems whose approximability status is still largely unknown.

MinCol(3)

Given a 3-colorable graph $G = (V, E)$, find a legal k-coloring such that k is as small as possible.

The best lower bound is due to Khanna, Linial, and Safra [47] who showed that is NP-hard to color 3-colorable graphs with 4 colors. On the other hand, the best polynomial-time approximation algorithm, due to Blum, Karger [19], has a performance ratio of $\mathcal{O}(|V|^{3/8} \log^{\mathcal{O}(1)} |V|)$.

MinBisection

Given a graph G, find a subset $X \subset V$, $|X| = \lfloor \frac{1}{2} V \rfloor$, such that $cut(X)$ is minimized.

Leighton, Rao [49] approximate a closely related problem. Arora, Karger, Karpinski [5] developed an approximation scheme for sufficiently dense instances. Feige, Krauthgamer, Nissim [63] designed an $O(\sqrt{n} \log n)$ approximation algorithm for the general problem.

PlanarSteinerTree

Given a planar graph $G = (V, E)$ and a subset $K \subseteq V$, find a connected subgraph $T = (V_T, E_T)$ of G with $K \subseteq V_T$ such that $|E_T|$ is minimized.

The problem trivially belongs to $\mathcal{APX}$, as the unrestricted Steiner tree problem belongs to that class. See [58] for the best known approximation algorithm for the general problem. The general case is known to be APX-complete [18]. For the Euclidean case Arora [4] gave a polynomial-time approximation scheme. The status of the planar case is still open.

Acknowledgment. Special thanks are due to my colleague Hans Jürgen Prömel for his contribution to this survey. I benefited enormously from our discussions and collaborations on various topics related to this article. Moreover, I have freely plundered and adapted material from our joint book [56].

References

[1] M. Ajtai.Recursive construction for 3-regular expanders.In *28th Annual Symposium on Foundations of Computer Science*, pages 295–304, 1987.

[2] N. Alon.Eigenvalues and expanders.*Comb*, 6(2):83–96, 1986.

[3] S. Arora.The approximability of NP-hard problems.In *28th Annual Symposium on Theory of Computing*, pages 337–348, 1998.

[4] S. Arora.Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems.*JACM*, 45(5):753–782, Sept. 1998.

[5] S. Arora, D. Karger, and M. Karpinski.Polynomial time approximation schemes for dense instances of graph problems.*JCSS*, 2000, to appear.Preliminary version in STOC'95.

[6] S. Arora and C. Lund.Hardness of approximations.In D. Hochbaum, editor, *Approximation Algorithms for NP-Hard Problems.* PWS Publishing Company, 1995.

[7] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy.Proof verification and the hardness of approximation problems.*JACM*, 45(3):501–555, 1998.Preliminary version in FOCS'92.

[8] S. Arora and S. Safra.Probabilistic checking of proofs: a new characterization of NP.*JACM*, 45(1):70–122, 1998.Preliminary version in FOCS'92.

[9] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi.*Complexity and Approximation.*Springer-Verlag, Berlin, 1999.

[10] L. Babai.Trading group theory for randomness.In *Proceedings of the 17th Annual Symposium on Theory of Computing*, pages 421–429, 1985.

[11] L. Babai.Transparent proofs and limits to approximations.In *First European Congress of Mathematicians*, pages 31–91. Birkhäuser, Basel, 1994.

[12] L. Babai, L. Fortnow, L. Levin, and M. Szegedy.Checking computations in polylogarithmic time.In *Proceedings of the 23rd Annual Symposium on Theory of Computing*, pages 21–31, 1991.

[13] L. Babai, L. Fortnow, and C. Lund.Non-deterministic exponential time has two-prover interactive protocols.*Computational Complexity*, 1:3–40, 1991.

[14] L. Babai and S. Moran.Arthur-Merlin games: a randomized proof system, and a hierarchy of complexity classes.*JCSS*, 36:254–276, 1988.

[15] M. Bellare, S. Goldwasser, and M. Sudan.Free bits, PCPs and non-approximability – Towards tight results.*SIAMCOMP*, 27:804–915, 1998.Preliminary version in FOCS'95.

[16] M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson.Multi-prover interactive proofs: How to remove intractability assumptions.In *Proceedings of the 20th Annual Symposium on Theory of Computing*, pages 113–131, 1988.

[17] P. Berman and M. Karpinski.On some tighter inapproximability results.In *24th International Colloquium on Automata, Languages and Programming*, LNCS1644, pages 200–209, Berlin, 1999. Springer-Verlag.

[18] M. Bern and P. Plassmann.The Steiner problem with edge lengths 1 and 2.*InfLet*, 32:171–176, 1989.

[19] A. Blum and D. Karger.An $\tilde{O}(n^{3/14})$-coloring algorithm for 3-colorable graphs.*InfLet*, 61:49–53, 1997.

[20] A. Borodin and R. El-Yaniv.*Online Computation and Competitive Analysis.*Cambridge University Press, 1998.

[21] A. Condon.The complexity of the max word problem and the power of one-way interactive proof systems.*Computational Complexity*, 3:292–305, 1993.Preliminary version in STACS'91.

[22] S. Cook.The complexity of theorem-proving procedure.In *3rd Annual Symposium on Foundations of Computer Science*, pages 151–158. IEEE, 1971.

[23] P. Crescenzi.A short guide to approximation preserving reductions.In *12th Annual Conference on Computational Complexity*, pages 262–273, 1997.

[24] P. Crescenzi, V. Kann, R. Silvestri, and L. Trevisan.Structure in approximation classes.*SIAM Journal on Computing*, 28:1759–1782, 1999.

[25] P. Crescenzi and A. Panconesi.Completeness in approximation classes.*Information and Computation*, 93:241–262, 1991.

[26] P. Crescenzi, R. Sivestri, and L. Trevisan.To weight or not to weight: where is the question?In *4th Israel Symposium on Theory of Computing and Systems*, pages 68–77, 1996.

[27] P. Crescenzi and L. Trevisan.On approximation scheme preserving reducibility and its applications.*Theory of Computing Systems*, 33:1–16, 2000.

[28] U. Feige.A threshold of ln n for approximating set cover.*JACM*, 45(4):634–652, 1998.Preliminary version in STOC'96.

[29] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy.Interactive proofs and the hardness of approximating cliques.*JACM*, 43(2):268–292, 1996.Preliminary version in FOCS'91.

[30] U. Feige and J. Kilian.Zero knowledge and the chromatic number.*JCSS*, 57(2):187–199, 1998.Preliminary version in CCC'96.

[31] M. Garey, R. Graham, and D. Johnson.Worst case analysis of memory allocation algorithms.In *Proceedings of the 4th Annual Symposium on Theory of Computing*, pages 143–150, 1972.

[32] M. Garey and D. Johnson.Approximation algorithms for combinatorial problems: an annotated bibliography.In J. Traub, editor, *Algorithms and Complexity: New Directions and Recent Results*, pages 41–52. Academic Press, New York, 1976.

[33] M. Garey and D. Johnson.Strong NP-completeness results: motivation, examples, and implications.*JACM*, 25:499–508, 1978.

[34] M. Garey, D. Johnson, and L. Stockmeyer.Some simplified NP-complete graph problems.*TCS*, 1:237–267, 1976.

[35] M. R. Garey and D. S. Johnson.*Computers and Intractability: A Guide to the Theory of NP-Completeness*.W. H. Freeman, New York, NY, 1979.

[36] O. Goldreich.*Modern cryptography, probabilistic proofs and pseudo-randomness*.Springer-Verlag, Berlin, 1999.

[37] S. Goldwasser, S. Micali, and C. Rackoff.The knowledge complexity of interactive proof-systems.*SIAMCOMP*, 418:186–208, 1989.

[38] R. Graham.Bounds for certain multiprocessing anomalies.*Bell System Technical Journal*, 45:1563–1581, 1966.

[39] J. Håstad.Clique is hard to approximate within $n^{1-\varepsilon}$.In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, pages 627–636, 1996.

[40] J. Håstad.Some optimal inapproximability results.In *Proceedings of the 29th Annual Symposium on Theory of Computing*, pages 1–10, 1997.

[41] V. Heun, W. Merkle, and U. Weigand.Proving the PCP-Theorem.In E. Mayr, H. Prömel, and A. Steger, editors, *Lectures on Proof Verification and Approximation Algorithms*, pages 83–160. Springer-Verlag, Berlin, 1998.

[42] S. Hougardy, H. Prömel, and A. Steger.Probabilistically checkable proofs and their consequences for approximation algorithms.*DM*, 136:175–223, 1994.

[43] D. Johnson.Approximation algorithms for combinatorial problems.*JCSS*, 9:256–278, 1974.

[44] N. Kahale.*Expander Graphs*.PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institut of Technology, 1993.

[45] H. Karloff and U. Zwick.A 7/8-approximation algorithm for MAX3SAT?In *38th Annual Symposium on Foundations of Computer Science*, pages 406–415, 1997.Remark: According to the authors Conjectures 4.3 and 4.5 are now proven.

[46] R. Karp.Reducibility among combinatorial problems.In J. Thatcher and R. Miller, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972.

[47] S. Khanna, N. Linial, and S. Safra.On the hardness of approximating the chromatic number.In *Proceedings of the 2nd Israel Symposium on Theory of Computing and Systems*, pages 250–260, Natanya, Israel, 1993. IEEE Comp. Soc. Press.

[48] S. Khanna, R. Motwani, M. Sudan, and U. Vazirani.On syntactic versus computational views of approximability.*SIAMCOMP*, 28:164–191, 1998.Preliminary Version in FOCS'94.

[49] Leighton and Rao.Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms.*JACM*, 46:787–832, 1999.

[50] L. Levin.Universal search problems (in Russian).*Problemy Peredaci Informatsii*, 9:115–116, 1973.English translation in *Problems of Information Transmission* **9**:265-266.

[51] L. Lovasz.On the ratio of the optimal integral and fractional covers.*Discrete Mathematics*, 13:383–390, 1975.

[52] E. Mayr, H. Prömel, and A. Steger, editors.*Lectures on Proof Verification and Approximation Algorithms*.LNCS1367. Springer-Verlag, Berlin, 1998.

[53] R. Motwani and P. Raghavan.*Randomized Algorithms*.Cambridge University Press, 1995.

[54] C. Papadimitriou and K. Steiglitz.*Combinatorial Optimization. Algorithms and Complexity*.Prentice-Hall, 1982.

[55] C. Papadimitriou and M. Yannakakis.Optimization, approximation, and complexity classes.*JCSS*, 43:425–440, 1991.

[56] H. Prömel and A. Steger.*The Steiner Tree Problem. A Tour Through Graphs, Algorithms and Complexity*.Vieweg Verlag, Wiesbaden, 2001, to appear.

[57] R. Raz.A parallel repetition theorem.*SIAMCOMP*, 27:763–803, 1998.Preliminary version in STOC'95.

[58] G. Robins and Z. A.Improved Steiner tree approximation in graphs.In *Proceedings 11th Symposium on Discrete Algorithms*, pages 770–779, 2000.

[59] S. Sahni and T. Gonzales.P-complete approximation problems.*JACM*, 23:555–565, 1976.

[60] L. Trevisan.*Reductions and (Non)-Approximability.*PhD thesis, Computer Science Department, University of Rome“La Sapienza”, 1997.

[61] L. Trevisan.Interactive and probabilistic proof-checking.*Annals of Pure and Applied Logic*, 2000, to appear.

[62] L. Trevisan, G. Sorkin, M. Sudan, and D. Williamson.Gadgets, approximation, and linear programming.In *Proceedings of the 37th Symposium on Foundations of Computer Science*, pages 617–626, 1996.

[63] K. N. U. Feige, R. Krauthgamer.Approximating minimum bisection size.In *30th Annual Symposium on Theory of Computing*, pages 530–536, 2000.

8

Pattern Inference under many Guises

M.-F. Sagot
Y. Wakabayashi[1]

8.1 Introduction

In a world of constant changes, conserved patterns of any kind are objects of interest for various reasons. Some are prosaic. If one has to perform a given operation on a set of objects and some of these objects are identical one may sometimes economize by performing just one operation for each group of identical objects. If the objects are not identical but almost (there are just a very limited number of well-characterized differences between them), one could perhaps adjust the operation to a smaller number of steps than starting from scratch for each object among the group of almost identical ones.

Another reason for being interested in conserved patterns is deeper: things that do not change, or change a little, or less than others, are objects upon which strong constraints are potentially acting. The chances are great that these objects may perform a function, possibly an important one. This is typically the case in biology. Some portions of a DNA or protein sequence, corresponding to string patterns, are conserved through evolution because the portions represent in fact segments of a molecule which will interact in a biochemical way with another molecule. Such interaction will be essential for some fundamental biological function (such as protein synthesis) to happen, and thus the organism to survive.

What aspects or properties of a pattern are preserved, and how conserved they must be depends on the area of investigation one is concerned with. In many cases, this is a matter of debate even among specialists of the area.

[1]Partially supported by CAPES-COFECUB (project 272/99-II), PRONEX project 107/97 (MCT/FINEP/CNPQ), and CNPQ (proc. 464114/00-4 and proc. 304527/89-0)

The area that will interest us in this paper is molecular biology. Among the objects which may model biological entities, we shall consider strings only. These may correspond, among others, to nucleic acid sequences (DNA or RNA) or protein sequences, or protein structures. A string can be thought as a labelled path. More generally, labelled graphs (sometimes directed) are important for describing biological entities or modelling biological problems. A special type of graphs, the trees, are used to represent, among others, phylogenies and some macromolecular structures (*e.g.* RNA secondary structures). We shall not treat such general graphs in this paper; they will however be encountered as an useful expository tool.

We concentrate upon one aspect related to pattern conservation; namely, given one or more strings identify all patterns that have conserved some well-defined properties (for instance, they appear in the string(s) with a maximum number of substitutions, insertions and deletions). This is called "inferring patterns". The properties will vary depending on the biological problem. Without going into detailed discussion on the biological subtleties, the properties are presented, specially in what they may influence the inference. Combinatorial methods for performing such inference are then surveyed (references to statistical methods for solving the same problems may be found in [34] and [45]). These include other persons work as well as our own.

In the next section we present some biological motivations for the study of the topics we address here. Then, in Section 3 we discuss the notions of similarity we shall be considering. These include the identity, a non-transitive relation, allowing for errors, a non-transitive relation with errors and, finally, a word instead of a symbol-based similarity. We then introduce the two main types of pattern inference we shall address: *common pattern inference* and *repeat identification*. When the repetition of interest appears dispersed in a string, the problem is quite similar to identifying single patterns common to a set of strings. In fact, the second may be easily derived from the first. We just need to concatenate the set of strings into a single long one and insert a different character to distiguish the concatenation point. The case of tandem repeats, and of other forms of structured patterns, that is, of patterns composed of various parts at non-random distances from one another, requires a different treatment. Finally, we survey some algorithms for solving the various kinds of pattern inference problems under these different notions of similarity.

8.2 Biological motivation

As is by now well known, biological sequences, whether DNA, RNA or proteins, may be represented as strings over an alphabet of 4 letters (DNA/RNA) or 20 (proteins). Some of the basic problems encountered in classical text analysis have their counterpart when the texts are biological, among them pattern matching. However, this problem, as well as others, comes with a twist once we are in the realm of biology: exact patterns hardly make sense in this case.

By *exact* above, we mean *identical*; and there are in fact at least two types of "non-identical" matchings one must consider in biology. One comes from looking at what "hides" behind each letter of the DNA/RNA or protein alphabet while the other corresponds to the more familiar notion of "errors". The errors concern mutational events that may affect a molecule during DNA replication. Those that will be of interest to us in this paper are *point mutations*, that is, mutations operating on single letters of a biological sequence: *substitution*, *insertion* or *deletion*. Considering substitutions only will sometimes be enough for dealing with a given problem.

There is another important difference between classical text analysis and biological sequence analysis. In the latter case, the most interesting question is often not testing whether a specific known pattern has matches in one or more strings, but rather determining which (initially unknown) patterns match the string(s) often enough to have a "chance" of representing an interesting biological entity. This entity may correspond to a binding site, *i.e.* to a (in general small) part of a molecule that will interact with another, or it may represent an element that is repeated in a dispersed or periodic fashion (for instance, tandemly, that is adjacently). The role played by a repetition of whatever type is often unknown: some repeats, in particular small tandem ones, have been related to a number of genetic diseases and are also interesting for the purposes of studying polymorphism; other types of repeats, such as short inverted ones, seem seem to indicate hotspots for recombination (roughly, the exchange of genetic material) intra and inter-species.

8.3 Notions of similarity

If s is a string of length $|s| = n$ over an alphabet Σ, that is, $s \in \Sigma^n$, its individual elements (the letters composing it) will be denoted by s_i, $1 \leq i \leq n$. A nonempty word $u \in \Sigma^*$ in a string s is a factor $s_i s_{i+1} \ldots s_j$ for a given pair (i, j) such that $1 \leq i \leq j \leq n$. The empty word trivially is a factor of all strings. It is denoted by λ.

8.3.1 Identity

Although identity is seldom an appropriate notion of similarity to consider when working with biological objects, it may sometimes be of interest.

This is a straightforward notion we nevertheless define properly as this will allow us to introduce some notations that will be used throughout the paper.

The identity concerns words in a string and we therefore adopt Karp *et al.* [20] identification of such words by their start position in the string. To facilitate exposition, this and all other notions of similarity are given for words in a single string. As mentioned, it is straightforward to adapt them to the case of more than one string. Let us denote by E the identity relation on the alphabet Σ (the letter E stands for "Equivalence").

Relation E between elements of Σ may then be extended to a relation E_k between words of length k in a string s in the following way.

Definition 8.3.1 *Given a string $s \in \Sigma^n$ and two positions i, j in s such that $i, j \leq n - k + 1$, then*

$$i\ E_k\ j \Leftrightarrow s_{i+l}\ E\ s_{j+l} \text{ for all } l \text{ such that } 0 \leq l \leq k-1.$$

For each $k \geq 1$, E_k establishes an equivalence relation that corresponds to the identity relation between positions in a string s: two positions i and j are in the relation E_k if and only if the words of length k in s starting at positions i and j are identical. This gives us a first definition of similarity between such words. Indeed, each class of E_k of cardinality at least two represents a set of exactly repeated words in s.

8.3.2 Non-transitive relation

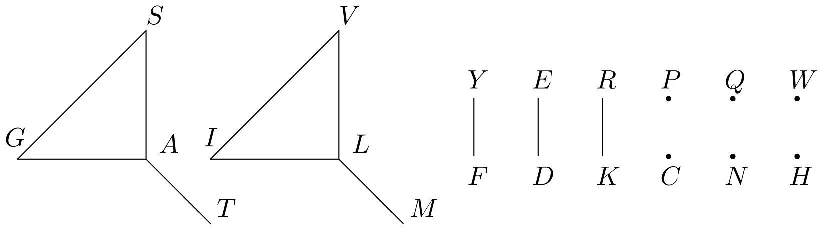
When dealing with biological strings one has to consider that the "letters" represented by such strings are complex biological objects with physico-chemical properties such as, for instance, electrical charge, polarity, size, different levels of acidity, etc. Some of these (but seldom all) properties may be shared by two or more of the objects. This applies more to proteins than to DNA/RNA but is true to some extent of both.

A more realistic relation to establish between the letters of the protein or DNA/RNA alphabet (respectively called amino acids and nucleotides) would therefore be reflexive and symmetric but non-transitive [40]. An example of such a relation, denoted by R, is given in Figure 8.1. It may be represented by a graph. The nodes of the graph are the elements of Σ. An edge links two nodes if the elements of Σ labelling the nodes correspond to biological objects sharing enough physico-chemical properties to be considered related.

As in the previous section, a relation R between elements of Σ may easily be extended to a relation R_k between words of length k in a string s.

Definition 8.3.2 *Given a string $s \in \Sigma^n$ and two positions i, j in s such that $i, j \leq n - k + 1$, then*

Let $\Sigma = \{A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y\}$ be the alphabet of the amino acids and R be the relation of similarity between these amino acids given by the following graph:



The maximal cliques of R are the sets: $\{A, S, G\}$, $\{A, T\}$, $\{I, L, V\}$, $\{L, M\}$, $\{F, Y\}$, $\{D, E\}$, $\{K, R\}, \{C\}, \{P\}, \{N\}, \{Q\}, \{H\}, \{W\}$.

Figure 8.1. Example of a relation of similarity between the letters of the protein alphabet (called amino acids).

$$i\ R_k\ j \Leftrightarrow s_{i+l}\ R\ s_{j+l} \ \textit{ for all } l \textit{ such that } \ 0 \le l \le k-1.$$

An important concept in our setting will be that of a (maximal) clique of a non-transitive relation.

Definition 8.3.3 *Given an alphabet Σ and a non-transitive relation R on Σ, a set C of elements of Σ is a clique of relation R if $\alpha\ R\ \beta$ for all $\alpha, \beta \in C$. If C is a clique and $C \bigcup \{\gamma\}$ is not a clique for all $\gamma \in \Sigma \setminus C$, then C is called a maximal clique.*

Definition 8.3.4 *Given a string $s \in \Sigma^n$, a set C_k of positions in s is a clique of relation R_k if $i\ R_k\ j$ for all $i, j \in C_k$. If C_k is a clique and $C_k \bigcup \{l\}$ is not a clique for all $l \in [1..n] \setminus C_k$, then C_k is called a maximal clique of R_k.*

Maximal cliques of R_k give us then a second way of establishing a definition of similarity between words of length k in a string. If the similarity relation is transitive, the strings can be translated into a smaller alphabet and R is an identity relation E.

As we describe in what follows, a non-transitive relation and its maximal cliques are a particularly appropriate tool for analyzing protein structures when such structures are coded as linear sequences of internal coordinates.

Indeed, the 3D structure of a protein is determined by the spatial arrangement of the atoms of its amino acids or residues. The amino acids are linked together in a chain and a representation of the structure that preserves the linear order of the residues allows us to consider working with strings. When studying such structure, it is possible to focus attention on its backbone only. It is well known [4] that the local conformation of the backbone can be defined at each residue by three internal coordinates usually referred to as three dihedral angles: Φ, Ψ and ω [30]. For chemical reasons, ω is fixed (to 0 or 180°) and can usually be forgotten, and the internal coordinates of the backbone are therefore in general represented on a two-dimensional map Φ, Ψ called a Ramachandran map [30] (see Figure 8.2). The structure of the backbone can then be uniquely defined by the linear succession of the pairs of angles (Φ, Ψ) along the backbone (see Figure 8.3). Since these pairs of angles represent pairs of real values, they have to be recoded into discrete values so that we can work with an alphabet of discrete symbols. In order to do it, a grid of mesh $\varepsilon°$ is constructed on the Ramachandran map [28, 37] (see again Figure 8.2). Note that what is shown in the figure as flat is actually the surface of a sphere. All the following considerations about squares on this map will hence implicitly assume that these squares actually wrap around the edges. The center of each small square becomes a node of a square lattice. Each node of the lattice corresponds then to a symbol of a new alphabet. Any real valued pair of angles is thus coded into the symbol represented by the small square inside which the pair is plotted. A relation R is then defined between the new symbols (nodes) of the map in the following way:

$$\forall(\alpha,\beta) \in \Sigma^2,\ \alpha\ R\ \beta \Leftrightarrow \exists \text{ a square of side } 2K\varepsilon \text{ enclosing } \alpha \text{ and } \beta,$$

where α and β are nodes of the lattice and K is a parameter (called margin) that can be adjusted to broaden or narrow the matching precision. Reducing the values of ε and K, alphabets and relations of increasing sizes and degeneracies are produced (for instance, for a mesh of 5° the alphabet has 5182 symbols, and for $K = 1$ each symbol belongs to 9 distinct cliques of R).

In terms of angles, the previous definition simply means that two pairs of angles match if their corresponding symbols lie in a square of side $2K\varepsilon$ of the Ramachandran map. The relation R is intrinsically non-transitive. The maximal cliques of R on Σ are, by definition, all (big) squares of side $K\varepsilon$ centered on each node.

It is important to note that the maximal cliques of R_k represent then all structural contiguous motifs of length k of a set of protein structures.

A similar approach has also been developed to look for structural motifs in true three-dimensional space, that is, for motifs that are not necessarily composed of contiguous amino acids [9, 18].

8.3.3 Allowing for errors

Let us initially assume the sole errors authorized are substitutions. In view of the definitions established in the previous sections, one would be tempted

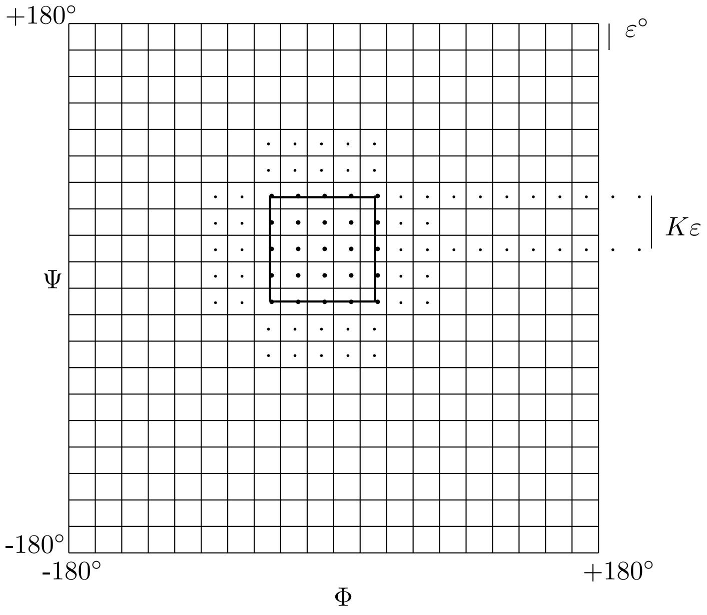


Figure 8.2. Sampled Ramachandran map: one square of side ε = one symbol of the alphabet.

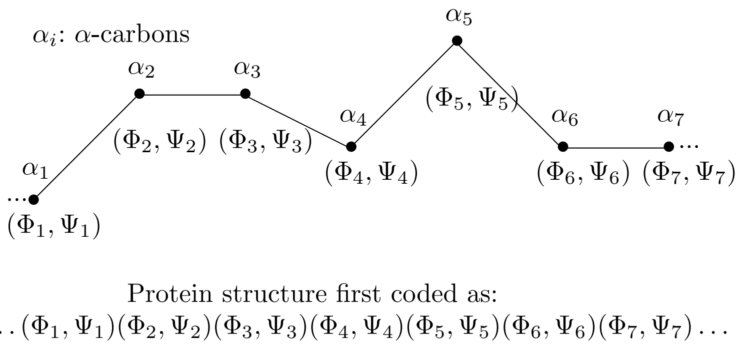


Figure 8.3. Structure of backbone defined as a linear succession of pairs of angles (Φ, Ψ).

to define a relation of similarity H between two words of length k in a string s, that is, between two positions i and j in s, in the following way.

Definition 8.3.5 *Given a string $s \in \Sigma^n$ and two positions i, j in s such that $i, j \leq n - k + 1$, then*

$$i\ H_k\ j \Leftrightarrow dist_H(s_i \dots s_{i+k-1}, s_j \dots s_{j+k-1}) \leq e,$$

where $dist_H(u, v)$ is the Hamming distance between u and v (that is, the minimum number of substitutions to be performed on u to obtain v) and e is a nonnegative integer that is fixed.

Parameter e corresponds to the maximum number of substitutions one wishes to tolerate. In the same way as in Section 8.3.2, maximal cliques of H_k would provide us with another possible definition of similarity between words of length k in a string.

One could consider now how to adapt the above definition to the case of a Levenshtein distance L (which, in the case of strings, is the minimum number of substitutions, insertions and deletions necessary to obtain one string from another), or any other type of distance where insertions and deletions are permitted besides substitutions. This is not completely trivial: indeed, given two words u and v respectively starting at positions i and j in s and such that iL_kj, what is the meaning of k ? Before even trying, one may intuitively note that calculating H_k (and, *a fortiori*, L_k) is no longer as easy as calculating E_k or R_k.

The reason is that, although the definitions given in Sections 8.3.1 and 8.3.2 involve pairs of positions in a string s, it is possible to rewrite them in such a way that, given a position i in s and a length k (that of the words in s one is currently considering), it is immediate to determine to which class or clique(s) i belongs. Indeed, the class or clique(s) can be uniquely identified just by "reading" $s_i \dots s_{i+k-1}$. Let us consider first the simpler case of an identity. Straightforwardly, position i will belong to the class whose label is $s_i \dots s_{i+k-1}$. In the case of a non-transitive relation R between the letters of Σ, let us name C the set of (maximal) cliques of R and denote by $clique_R(\alpha)$ the cliques of R to which a letter α belongs. Then, position i will belong to the sets of R_k whose labels may be spelled from the regular expression $clique_R(s_i) \dots clique_R(s_{i+k-1})$ and that are maximal under R_k. Note the small difference here with an identity relation: maximality of a validly labelled set has to be checked [40]; a class is always "maximal".

No such easy rewriting and verification are possible in the case of the definition of H_k (or L_k) if we wish to build the notion of similarity between words in a string upon that of the cliques of H_k. Indeed, obtaining such cliques requires comparing (a possibly large number of) pairs of positions between themselves. This is expensive.

One may, however, rewrite the definition of H_k in such a way that it refers to labels as we did above for E_k and R_k although such labels are no longer as immediately identifiable. A possible definition (still for the case where substitutions only are considered) would be the following.

Definition 8.3.6 *Given a string $s \in \Sigma^n$ and two positions i, j in s such that $i, j \leq n - k + 1$, then*

$$i \; H_k \; j \Leftrightarrow \exists m \in \Sigma^k \text{ such that } dist_H(m, s_i \dots s_{i+k-1}) \leq e \text{ and } dist_H(m, s_j \dots s_{j+k-1}) \leq e,$$

where $dist_H(u, v)$ and e are as before.

Generalizing this, we would then have:

Definition 8.3.7 *A set S_k of positions in s represents a set of words in s of length k that are all similar between themselves if, and only if, there exists (at least) one string $m \in \Sigma^k$ such that, for all elements i in S_k, $dist_H(m, s_i \dots s_{i+k-1}) \leq e$ and, for all $j \in [1..n] \setminus S_k$, $dist_H(m, s_j \dots s_{j+k-1}) > e$.*

Observe that extension of both definitions to a Levenshtein distance becomes now straightforward. We reproduce below, after modification, just the last definition.

Definition 8.3.8 *A set S_k of positions in s represents a set of words that are all similar between themselves if, and only if, there exists (at least) one string $m \in \Sigma^k$ such that, for all elements i in S_k, $dist_L(m, s_i \dots) \leq e$ and, for all $j \in [1..n] \setminus S_k$, $dist_L(m, s_j \dots) > e$.*

Since the length of similar strings (with respect to m) may now be different from that of m (it will vary between $|m| - e$ and $|m| + e$ where $|m|$ is the length of m), we denote it $(s_i \dots)$ leaving undefined its right-end point.

Observe also that it remains possible, given a position i in s and a length k, to obtain the label of the group(s) (we shall see that this is no longer a clique and is obviously not a class unless e is zero) of the relation H_k (or L_k) i belongs to. Such labels are represented by all strings $m \in \Sigma^k$ such that $dist_H(m, s_i \dots s_{i+k-1}) \leq e$ (or $dist_L(m, s_i \dots) \leq e$), that is, such that their distance from the word starting at position i in s is no more than e.

We call *models* such group labels. Positions in s indicating the start of a word of length k are *e-occurrences* (or simply *occurrences* where there is no ambiguity) of a model m if $dist(m, s_i \dots) \leq e$, where $dist$ is either the Hamming or Levenshtein distance. Observe that a model m may never be present exactly in s.

When e is zero, models represent the classes of an identity relation. In the case where e is greater than zero, models such as defined in the previous section are neither classes nor cliques nor yet another well-defined mathematical object: Steiner strings. We recall that, given a distance *dist* and a set of strings S, s is a *Steiner string* of the strings in S if $\sum_{u \in S} dist(s,u)$ is minimal.

Although Definition 8.3.6 could lead us to believe that models correspond to cliques, this is not the case. Consider $s \in \Sigma^* = \{\text{A, B, C}\}^*$ equal to ABACACABAAAA. If at most one substitution is allowed, then positions 1 and 5 (strings s are indexed from 1 to $|s|$) are related by H_4 because of models $m = ABAB$ and $m = ACAC$; so are positions 1 and 9 because of models $m = ABAA$ and $m = AAAC$ and, finally, positions 5 and 9 because of models $m = ACAA$ and $m = AAAB$. The set $\{1, 5, 9\}$ forms therefore a clique (any two elements are related by H_4), but it is never the same model for every possible pair. Indeed, there is no single model such that all three positions are occurrences of it.

The case of a Steiner string also calls for an example. Let us consider the string $s \in \Sigma^* = \{\text{A, B, C}\}^*$ equal this time to AAAAAABBBBACAC, and let us assume that at most one substitution is allowed. Consider $O = \{1, 2, 3, 11\}$ a set of positions in s. There exist only two models $m \in \Sigma^4$ such that $dist_H(m, s_i \dots s_{i+3}) \leq e$ for all $i \in O$. These are models ACAA and AAAC. None is a Steiner string for O. The unique string that is a Steiner string for O is AAAA, which is not a model for O (it is at distance 2 from occurrence ACAC).

Nonetheless, models are interesting objects that provide a precise definition for the idea of conservation and will allow us, as we shall see later, to obtain reasonably efficient algorithms for identifying sets of conserved words in a string.

In what follows we extend the idea of models and introduce two other concepts of similarity.

Models allow us to considerably enrich the notion of similarity. For instance, they enable us to simultaneously consider a non-transitive relation between the letters of the alphabet (amino acids or nucleotides) and the possibility of errors. In order to do that, it suffices to permit the model to be written over an extended alphabet composed of a subset of the set of all subsets of Σ (denoted $\mathcal{P}(\Sigma)$), where Σ is the alphabet of amino acids or nucleotides. Such an alphabet could be, for instance, one defined by the maximal cliques of the relation R given in Figure 8.1. Definition 8.3.8 of Section 8.3.3 then becomes:

Definition 8.3.9 *A set S_k of positions in s represents a set of words of length k that are all similar between themselves if, and only if, there exists (at least) one element $M \in P^k$ with $P \subseteq \mathcal{P}(\Sigma)$ such that, for all elements i in S_k, $setdist(M, s_i \dots) \leq e$ and, for all $j \in [1..n] \setminus S_k$, $setdist(M, s_j \dots) > e$, where $setdist(M, v)$ for $M \in P^*$ and $v \in \Sigma^*$ is the minimum Hamming or Levenshtein distance between v and all $u \in M$.*

Let $\Sigma = \{A, B, C\}$ and

$$\text{score } (\alpha, \alpha) = 1, \forall \; \alpha \in \Sigma;$$
$$\text{score } (A, B) = \text{score } (B, A) = -1;$$
$$\text{score } (A, C) = \text{score } (C, A) = -1;$$
$$\text{score } (B, C) = \text{score } (C, B) = -1.$$

If we say that two words are similar if either

- the number of substitutions between them is at most 1; or
- their score is at least 1;

then by the first criterion the words $AABAB$ and $AACCB$ are not similar, while by the second criterion they are, the second substitution being allowed because the two words on average share enough resemblance.

Figure 8.4. Example of the greater flexibility allowed by scoring words rather than counting errors.

Among the subsets allowed in P, the alphabet of models, we may take $\{\Sigma\}$ itself, that is, the *wild card*. It is obvious that this may lead to trivial models. Alphabet P may then come with weights attached to each of its elements indicating how many times (possibly infinite) it may appear in a model of interest to us. Observe that another way of describing the alphabet P of models is as the set of edges of a (possibly weighted) hypergraph whose nodes are the elements of Σ.

When e is zero, we obtain a definition of similarity between words in the string that closely resembles that given in Section 8.3.2. Note however that, given two models M_1 and M_2, we may well have that the set of occurrences of M_1 is included in that of M_2. The cliques of Definition 8.3.4 will correspond to the sets of occurrences that are maximal.

Errors between a group of similar words and the model of which they are occurrences can either be counted as unitary events (possibly with different weights) as was done previously, or they can be given a score. The main idea behind scoring a resemblance between two objects is that it allows to average the differences that may exist between them. It may thus provide a more flexible function for measuring the similarity between words. A simple example illustrates this point in Figure 8.4.

In the example and in the definition of similarity introduced in this section, insertions and deletions are not allowed, only substitutions. This is done essentially for the sake of clarity. Insertions and deletions may, however, be authorized, the reader is referred to [38] for details.

Let a numerical matrix $\mathcal{M}$ of size $\mid \Sigma \mid \times \mid \Sigma \mid$ be given such that:

$$\mathcal{M}(a, b) = \text{score between } a \text{ and } b \text{ for all } a,\, b \in \Sigma.$$

If this score measures a similarity between a and b, we talk of a *similarity matrix* (two well-known examples of which in biology are PAM250 [7] and

Let $\Sigma = \{A, B, C\}$, $w = 3$ and $t = 6$. Let $\mathcal{M}$ be the following matrix:

	A	B	C
A	3	1	0
B	1	2	1
C	0	1	3

Given the three strings:

s1 = ABCBBABB$\underline{\text{BACABACB}}$BBBAB
s2 = $\underline{\text{CABACAAC}}$BACCABCACCACCC
s3 = BBBACACCABA$\underline{\text{BACABACA}}$BA

then the longest model that is present in all strings is CACACACC (at positions 9, 1 and 12 respectively).

Figure 8.5. Example of a model under a word-based relation of similarity.

BLOSUM62 [14]), while if the score measures a dissimilarity between a and b we talk of a *dissimilarity matrix*. A special case of this latter matrix is when the dissimilarity measure is a metric, that is when the scores obey, among other conditions, the triangular inequality. In that case, we talk of a *distance matrix* (an example of which is the matrix proposed by J.-L. Risler [31]).

In what follows, we consider $\mathcal{M}$ a similarity matrix.

Definition 8.3.10 *Given a string $u = u_1u_2 \dots u_k \in \Sigma^k$, a model $m = m_1m_2 \dots m_k \in \Sigma^k$ and a matrix $\mathcal{M}$, we define*

$$score_{\mathcal{M}}(m, u) = \sum_{i=1}^{k} \mathcal{M}(m_i, u_i).$$

Definition 8.3.11 *A set S_k of positions in s represents a set of words of length k that are similar if, and only if, given a positive integer w such that $w \leq k$ and a threshold value t, there exists (at least) one element $m \in \Sigma^k$ such that,*

1. *for all elements i in S_k and for all $j \in \{1, \dots, |m| - w + 1\}$, $score_{\mathcal{M}}(m_j \dots m_{j+w-1}, s_{i+j-1} \dots s_{(i+j-1)+(w-1)}) \geq t$;*
2. *for all $i \in [1..n] \setminus S_k$, there exists at least one $j \in \{1, \dots, |m| - w + 1\}$ such that $score_{\mathcal{M}}(m_j \dots m_{j+w-1}, s_{i+j-1} \dots s_{(i+j-1)+(w-1)}) < t$.*

An example is given in Figure 8.5.

8.4 Models and their properties

The previous sections presented various definitions of similarity between words in a string (easily extensible to words in a set of strings). The last definitions (from Section 8.3.3 on) introduced the notion of a model that was somewhat implicit in the earlier ones.

We discuss now further properties the models that will interest us must satisfy. Doing so, we also render such models more complex. To facilitate exposition, we henceforward denote models by a lower case m, whether this represents a word (*i.e.*, is defined over Σ) or a set of words (*i.e.*, is defined over $\mathcal{P}(\Sigma)$).

8.4.1 "Simple" models

Models such as given in the previous sections will be called *simple* models. There will be one important property such models (and, in some way or another, all other kinds of models) will have to satisfy and that is the following.

Property 8.4.1

Case of a single string. *Given a string s and a nonnegative integer q, a model m is said to be* valid *if it has at least q occurrences in s;*

Case of a set of strings. *Given a set of N strings and an integer q such that $1 \leq q \leq N$, a model m is said to be* valid *if it has at least one occurrence in at least q distinct strings of the set.*

Parameter q is called the *quorum* valid models must satisfy.

8.4.2 Structured models

Although the objects defined in the previous section can be reasonable, algorithmically tractable models for single binding sites, they do not take into account the fact that such sites are often not alone (in the case of eukaryotes, they may even come in clusters). Specially, they do not consider that the relative positions of such sites when more than one participates in a biological process are in general not random. This is particularly true for some DNA binding sites such as those involved in the transcription of DNA into RNA (*e.g.* the so-called promoter sequences).

There is therefore a need for defining biological models as objects that take such characteristics into account. This has the motivation just mentioned but presents also interesting algorithmical aspects: exploiting such characteristics could lead to algorithms that are both more sensitive and more efficient. Models that incorporate such characteristics are called *structured models.*

Formally, a *structured model* is a pair (m, d) where:

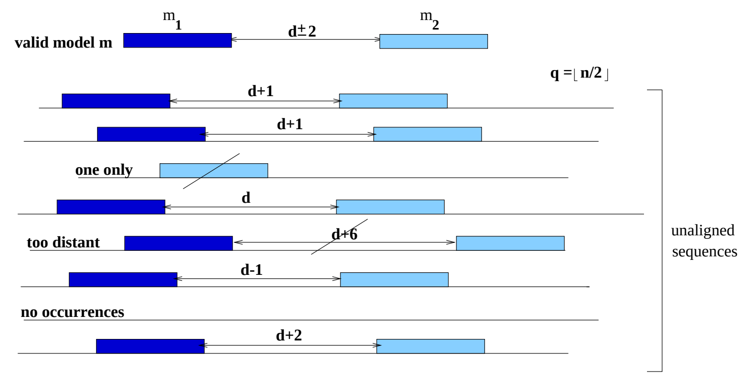


Figure 8.6. Example of a model with two boxes ($p = 2$).

- m is a p-tuple of simple models $(m_1, \ldots, m_p)$ (representing the p parts a structured model is composed of – we shall call these parts *boxes*);

- d is a $(p-1)$-tuple of triplets $((d_{min_1}, d_{max_1}, \delta_1), \ldots, (d_{min_{p-1}}, d_{max_{p-1}}, \delta_{p-1}))$ (representing the $p-1$ intervals of distance between two successive boxes in the structured model);

with p a positive integer, $m_i \in \Sigma^+$ and d_{min_i}, d_{max_i} $(d_{max_i} \geq d_{min_i})$, δ_i nonnegative integers.

Given a set of N strings $s_1, \ldots, s_N$ and an integer $1 \leq q \leq N$, a model (m, d) is said to be *valid* if, for all $1 \leq i \leq p-1$ and for all occurrences u_i of m_i, there exist occurrences $u_1, \ldots, u_{i-1}, u_{i+1}, \ldots, u_p$ of $m_1, \ldots, m_{i-1}, m_{i+1}, \ldots, m_p$ such that:

- $u_1, \ldots, u_{i-1}, u_i, u_{i+1}, \ldots, u_p$ belong to the same string of the set;

- there exists d_i, with $d_{min_i} + \delta_i \leq d_i \leq d_{max_i} - \delta_i$, such that the distance between the end position of u_i and the start position of u_{i+1} in the string is equal to $d_i \pm \delta_i$;

- d_i is the same for p-tuples of occurrences present in at least q distinct strings.

The term d_i represents a distance and $\pm\delta_i$ an allowed interval around that distance. When $\delta_i = (d_{max_i} - d_{min_i} + 1)/2$, then δ_i is omitted and d in a structured model (m, d) is denoted by a pair (d_{min_i}, d_{max_i}). An example of a model with $p = 2$ is given in Figure 8.6.

Observe that simple models are indeed but a special case of structured ones.

8.4.3 Models for tandem arrays (satellites)

Tandem arrays (called *tandem repeats* when there are only two units) are a sequence of repeats that appear adjacent in a string. In biology, such tandemly repeated units are divided into three categories depending on the length of the repeated element, the span of the repeat region and its location within the chromosome [2]. Repeats occurring in or near the centromeres and telomeres are called simply *satellites.* Their span is large, up to a million bases, and the length of the repeated element varies greatly, anywhere from 5 to a few hundreds of base pairs. In the remaining, euchromatic region of the chromosome the kinds of tandem repeats found are classified as either *micro* or *mini satellites*, according to the length of the repeated element. Micro satellites are composed of short units, of 2 to 5 base pairs, in copy numbers in general around 100. Mini satellites on the other hand involve slightly longer repeats, typically around 15 base pairs, in clusters of variable sizes, comprising between 30 and 2000 elements.

Satellites of whatever type ask for a more complex definition of models that may initially recall that of the structured models presented in Section 8.4.2. Indeed, in the case of satellites, the models themselves are simple. Indeed, some constraints are imposed on the relative positions of the occurrences of a satellite model in a way that is somewhat similar to what was done with structured models. However, the overall nature of such constraints is not the same as we show in what follows.

Satellite models

We have in fact two definitions related to a satellite model, one called *prefix model* and the other *consensus model.* The latter will concern satellite models strictly speaking while prefix models are in fact models for approximately periodic repetitions that are not necessarily tandem.

Formally, a *prefix model* of a satellite is a string $m \in \Sigma^*$ (or $\mathcal{P}(\Sigma)$) that approximately matches a train of wagons. A *wagon* of m is a word u in s such that $dist(m, u) \leq e$. A *train* of a satellite model m is a collection of wagons $u_1, u_2, \ldots, u_p$ ordered by their starting positions in s and such that:

Property 8.4.2 *$p \geq min\_repeat$, where min_repeat is a fixed parameter that indicates the minimum number of elements a repeating region must contain.*

Property 8.4.3 *$left_{u_{i+1}} - left_{u_i} \in JUMP$, where $left_u$ is the position of the left-end of wagon u in s and*

$$JUMP = \{y : y \in \cup_{x \in [1, max\_jump]}\, x \times [min\_range, max\_range]\},$$

with the three parameters min_range, max_range and max_jump fixed.

A prefix model m is said to be *valid* if there is at least one train of m in the string s. Similarly, a train, when viewed simply as a sequence of substrings of s, is valid if it is the train for some model m. A prefix model represents the invariant that must be true as we progressively search for our final goal, which is to arrive at a *consensus model*. This is a prefix model which further satisfies the following:

Property 8.4.4 *$left_{u_{i+1}} - right_{u_i} \in GAP$, where $right_u$ is the position of the right-end of wagon u, and $GAP = \{y : y \in \cup_{x\in[0,max\_jump-1]}\ x \times [min\_range, max\_range]\}$.*

Parameter *max_jump* allows us to deal with very badly conserved elements inside a satellite (by actually not counting them) while we require that the satellite be relatively well conserved overall. Fixing *max_jump* at a value strictly greater than one means we allow some wagons (the badly conserved ones) to be "jumped". This may be seen as "meta-errors", that is as errors involving not a letter inside a wagon but a wagon inside a train. Note that $0 \in GAP$. This guarantees that, when jumps are not authorized, the repeats found are effectively tandem.

Since mutations affecting a unit concern *indels* (that is, insertions and deletions) as well as substitutions, it is sometimes interesting to work with a variant of the above properties where *JUMP* and *GAP* are defined as

$$JUMP = \left\{ y : \begin{array}{l} y \in [min\_range, max\_range] \text{ or} \\ y \in \cup_{x\in[2,max\_jump]}\ x \times [min\_range - g, max\_range + g] \end{array} \right\}$$

$$GAP = \left\{ y : \begin{array}{l} y \in [min\_range, max\_range] \text{ or} \\ y \in \cup_{x\in[1,max\_jump]}\ x \times [min\_range - g, max\_range + g] \end{array} \right\},$$

where $g \geq e$ is a fixed value. The idea is to allow the length of the badly conserved elements to vary more than is permitted for the detected "good" wagons.

8.5 Algorithms

We now develop the main ideas behind the algorithms for inferring patterns under the various definitions of similarity and properties specified in the previous sections. We focus on one algorithm each time, or a class of equivalent algorithms, the one that appears the most performing or flexible for our purposes. This is in general the most easily adaptable to special cases or the introduction of additional constraints. Other methods are just mentioned. We also discuss (Section 8.5.3) the use of a special data structure for storing the string(s) itself (themselves).

We treat the case of one string only, except where considering more than one string increases the complexity of the algorithm. In this latter case, we address the multiple string problem instead. Once errors are permitted, we adopt the reference to models even though some of the earlier algorithms that handled errors did not explicitly make use of such external objects for extracting sets of similar words, or did not call them models.

Proofs are omitted and some details are skipped. Both may, in general, be found in the original papers. In all cases, we start by stating the problem that each specific algorithm intends to solve.

8.5.1 The simplest property: Identity

Perhaps the first classical algorithm for finding all exact repetitions in a string (henceforth called KMR) was elaborated by Karp, Miller and Rosenberg in 1972 [20]. Given a string s, KMR solves the following problems:

Problem 8.5.1 *Identify the positions of all words of a fixed length k that appear repeated in s.*

Problem 8.5.2 *Find the length k_{max} of the longest repeated word in s, and solve Problem 8.5.1 for $k = k_{max}$.*

KMR rests on the definition of the equivalence relation E_k given in Section 8.3.1. Problem 8.5.1 can then be formulated as the problem of finding the partition associated with E_k. Problem 8.5.2 requires finding the maximum value of k such that E_k is not the identity. The algorithm is based on an iterative construction of partitions E_l for $l \leq k$. The mechanism for performing such constructions rests on the following lemma.

Lemma 8.5.1 *Given two integers a, b with $1 \leq b \leq a$, and i, j two positions in a string s of length n, such that $i, j \leq n - (a + b) + 1$, then*

$$i \; E_{a+b} \; j \Leftrightarrow i \; E_a \; j \text{ and } (i + b) \; E_a \; (j + b).$$

The main idea behind the KMR algorithm is to use the lemma with $b = a$ for as long as possible. This means finding repeats of length $2a$ by using previouly acquired information on the repeats of length a that may become the prefixes and suffixes of those of length $2a$. If we are dealing with Problem 8.5.1, and if k is not a power of 2, we then use the lemma with $b < a$ in a last step in order to obtain E_k. If we are treating Problem 8.5.2, we may need more than one step to find the value of k_{max} such that $E_{k_{max}}$ is not the identity but $E_{k_{max+1}}$ is. The search for k_{max} from the smallest power of 2 that is bigger than k_{max}, let us say it is 2^p, can be done by applying the lemma with $b < a$ in a dichotomous fashion between 2^{p-1} and 2^p.

Constructing the partitions E_a basically corresponds to performing a set intersection operation. The intersections may be implemented using, for instance, stacks. More precisely, we need an array V_a of size n which stores, for each position i in s, the label of the class of E_a to which the a-long word starting at i belongs. The lemma is applied by means of two arrays of stacks P and Q. Stacks in P are filled by traversing V_a. Such stacks are in fact a dual of V_a. Each one corresponds to a class c of E_a and contains the positions i in s belonging to c. Array P serves therefore to sort the prefixes of length a of the repeats of length $2a$ one is trying to identify. The content of each stack of P in turn is then poured into the appropriate stack of Q. A division separates, within a same stack of Q, elements coming from different stacks of P. Like P, array Q has as many stacks as there are classes in E_a. It serves to sort the suffixes of length also a of the repeats of length $2a$. One then just needs to orderly pour Q into V_{2a} to obtain the classes of E_{2a}. In case the quorum is higher than 2, verifying that it is satisfied is a simple question of counting how many elements there are in each class.

Each partition construction takes $O(n)$ time, there are $O(\log k)$ such constructions, KMR time complexity is therefore $O(n \log k)$. When solving Problem 8.5.2, this leads to an $O(n \log n)$ complexity because of possible degenerate cases (such as that of a string s composed of a single letter). KMR space complexity is $O(n)$.

Another method for obtaining the same result in a more efficient way that does not make use of complex data structures as will be discussed in Section 8.5.3 has been shown in [5]. It is, however, not easily extensible to the case of a non-transitive relation.

8.5.2 More complex properties for single patterns: non-transitive relation without/with errors

Let us consider first the case of non-transitive relations without errors.

In this case problems on simple patterns can be solved by algorithms obtained from an adaptation of KMR to deal with a non-transitive relation R [40]. The problems solved will be the same as for KMR in the previous section.

Lemma 8.5.1 applies analogously, one just needs to substitute relation E by R.

Lemma 8.5.2 *Given two integers a, b with $1 \leq b \leq a$, and i, j two positions in a string s of length n, such that $i, j \leq n - (a + b) + 1$, then*

$$i \ R_{a+b} \ j \Leftrightarrow i \ R_a \ j \text{ and } (i + b) \ R_a \ (j + b).$$

Computing relations R_l for $l \leq k$ requires the same structures as for KMR, except that, as we saw, a set of positions pairwise-related by R_l is no longer a class but a clique. The algorithm was in consequence called KMRC (the “C” standing for Clique) [40]. In particular, a position may belong to two or more distinct cliques of R_l. Array V_l must therefore now be an array of stacks, like P and Q. It indicates, for each cell i corresponding to a position in s, the cliques of relation R_l to which i belongs.

The construction itself follows the same schema as indicated for KMR. Some of the sets of similar words obtained at the end of each step may not be maximal. A further operation is therefore needed to eliminate sets included in another one to obtain maximal cliques.

To analyse the complexity of the KMRC algorithm, we need to define a parameter g that measures the "degree of non-transitiveness" of relation R.

Definition 8.5.1 *Given R, a non-transitive relation on Σ, we call g the greatest number of cliques of R to which a symbol may belong, that is:*

$$g = Max\ \{g_a \mid a \in \Sigma,\ g_a = \textit{number of cliques to which a belongs}\}.$$

We call $\bar{g}$ the average value of g_a for $a \in \Sigma$, that is:

$$\bar{g} = \frac{\sum_a g_a}{n_c},$$

where n_c is the number of cliques of R.

If one does not count the set inclusion operations to eliminate non-maximal cliques, KMRC has time complexity $O(n \log k g^k)$ since each position i in s may belong to up to g^k (or, on average, $\bar{g}^k$) cliques of R_k. Inclusion tests based on comparing the positions contained in each set will take $O(n^2 g^{2k})$ time at the end of step k. At least another approach for testing set inclusion is possible and may result in a better theoretical time complexity (but not necessarily better in practice – this is discussed in [40]). Space complexity is $O(ng^k)$.

Another combinatorial approach for solving the problems addressed by KMRC uses an idea that is quite close to that of inferring models and will be discussed in the next section.

Let us now treat the case of non-transitive relations with errors.

Models are considered this time. The problem we wish to solve is then the following.

Problem 8.5.3 *Given a string s, an integer $e \geq 0$ and a quorum q, find all models m such that m is valid, that is, is present at least q times in s, each time with at most e errors.*

KMR's principle for finding all exact repetitions of length $2k$ in a string s is based on the idea that such repetitions can be decomposed into two adjacent repetitions of length k.

In a likewise manner, we could base the principle of an algorithm for finding all the models of length k that are present with errors in a string s

on an iterative construction that would double the length of the models at each step. Indeed, the set of occurrences of a model $m = m_1m_2$ with $|m| = 2 \times |m_1| = 2 \times |m_2| = 2k$ is the set of occurrences of m_1 which are adjacent to at least one occurrence of m_2. As for KMR, we need to obtain and stock the sets of occurrences of all the valid models of length k in order to obtain those of length $2k$.

If we consider the search space of the problem, which corresponds to the tree of all possible models (in fact a trie), KMR, KMRC and the algorithm we are proposing now thus perform a breadth-first exploration of such search tree. All levels are not visited since at each step a lot of pruning may be realized (cutting off whole subtrees whose root is not labelled by a valid model). However, all information concerning a given level, say k, is needed to build a deeper level, say $2k$. If errors are allowed, a breadth-first exploration of the tree may therefore consume a lot of memory in the earlier stages, when almost all models occur at almost all positions.

A second, more space-parsimonious approach to constructing such models is to traverse the tree depth-first, again with possible pruning along the way. Indeed, if instead of doubling the length of the models at each step, we extend each model separately to the right by just one unit at a time, then all we need to obtain the set of occurrences of a model m is the set of occurrences of the model m' of length $|m|$ - 1 that is its prefix, plus a look at what follows each such occurrence in the string. In terms of memory, all we need to stock at any time is therefore the sets of occurrences of all the models that are prefixes of the currently considered model m.

The lemma that is applied is the following. Observe that, to facilitate the "look at what follows in the string", occurrences are now identified by their right-end positions instead of left-ends. They come also accompanied by the number of errors that they have accumulated against the model. They are therefore represented by a pair (i, d) where i is a position in s and d is a distance.

Lemma 8.5.3 *Pair (i, d) is an occurrence of model $m\alpha$ with $\alpha \in \Sigma$ if, and only if, $d \leq e$ and at least one of the following is true:*

(match)	*$(i-1, d)$ is an occurrence of m and $s_i = \alpha$;*
(substitution)	*$(i-1, d-1)$ is an occurrence of m and $s_i \neq \alpha$;*
(deletion)	*$(i, d-1)$ is an occurrence of m;*
(insertion)	*$(i-1, d-1)$ is an occurrence of $m\alpha$.*

Note that applying the lemma corresponds to doing sparse dynamic programming between string s and the virtual trie $\mathcal{M}$ of all possible valid models. It is sparse because we are looking for models having occurrences at a maximum distance and therefore only a few cells of the usual dynamic programming matrix need to be kept.

Other, earlier approaches had been elaborated to infer patterns with errors. They implied generating all possible words of a given length and identifying their occurrences by simple pattern matching against the string [12, 29, 41, 47, 48, 49].

Extension of the algorithm to deal with models m defined over $\mathcal{P}(\Sigma)^*$ is straighforward [17, 16] (no error) [39]. There is, however, a notion of redundancy that appears at the level of models over $\mathcal{P}(\Sigma)^*$ that is not trivial to treat. The interested reader is referred to [27] and [36] for further details.

In the case of no error, the difference in time complexity between KMR/KMRC and the algorithm sketched above varies only in that a $\log k$ term for the first approach is changed into a k term for the last one.

Before giving the complexity for the case where errors are permitted, we need to introduce the notion of a *word neighbourhood.*

Definition 8.5.2 *Given a word of length k defined over Σ^k, its e-neighbourhood, denoted by $\mathcal{V}(e,k)$, is the number of words situated at a distance (Hamming or Levenshtein) at most e from it.*

In [33] and [39], we show that $\mathcal{V}(k,e)$ is bounded above in both cases by $k^e|\Sigma|^e$.

The neighbourhood measures the number of models $m \in \Sigma^k$ of which a position i in s may be an occurrence. In the case where models are in $\mathcal{P}(\Sigma)^k$ instead, $k^e|\Sigma|^e$ must be further multiplied by g^k. Since there are $O(n)$ positions, and the algorithm takes k steps, the time complexity of the algorithm sketched above is $O(nk\mathcal{V}(e,k))$ or $O(nkg^k\mathcal{V}(e,k))$. The space complexity is $O(n)$ or $O(ng^k)$.

It is worth pointing out that, when e is zero, we obtain algorithms that have time complexity $O(nk)$ and $O(nkg^k)$ respectively. As we saw, KMR and KMRC obtain better theoretical performances. Furthemore, at least for models defined over Σ and an identity relation, it is possible to have even better results, in particular but not exclusively, by making use of special data structures such as suffix trees. The reader will find extensive discussion of such use for the "no error" case in [13]. We discuss below the use of suffix trees for inferring models with errors. Since considering more than one input string makes a difference to the algorithm in terms of time and space, we address the case of multiple input strings. Deducing equivalent algorithms and their complexities for the case of just one string is straighforward and will not be detailed.

8.5.3 Introducing suffix trees

The idea behind using suffix trees, as will be developed below, comes from the observation that long strings, specially when they are defined over a small alphabet, may contain many exact repetitions. One does not want to compare such repeated parts more than once with the potentially valid models. One way of doing that is using a representation of the string s that allows to put together some of the repetitions, that is, using an index of s such as a suffix tree $\mathcal{T}$.

We do not describe the suffix tree construction, this can be found in either [25, 42] or (for a review of this and other data structures and text algorithms) [6] and [13]. We just recall some of the basic properties such structures have (these are taken from [25]).

Basic properties of the suffix tree $\mathcal{T}$ of a string s

Property 8.5.1 *An arc of $\mathcal{T}$ may represent any nonempty substring of s.*

Property 8.5.2 *Each node of $\mathcal{T}$ that is not a leaf, except for the root, must have at least two offspring arcs (compact version of the tree).*

Property 8.5.3 *The strings represented by sibling arcs of $\mathcal{T}$ must begin with different symbols of Σ.*

Observe that Property 8.5.2 means an arc of $\mathcal{T}$ may be labelled by an element of Σ^k for $k \geq 2$ (for space considerations, each arc of $\mathcal{T}$ is in fact labelled by a pair of numbers corresponding to the start and end positions in s of the substring it represents, or its start position and length). Furthermore, an edge links every node spelling αx with $\alpha \in \Sigma$ and $x \in \Sigma^*$ to the node spelling x. Such edges are called *suffix links* and are what allows the tree to be built in time linear with the length of the string.

The key feature of a suffix tree is that for any leaf i, the concatenation of the labels of the arcs on the path from the root to i spells the suffix of s starting at position i. Reciprocally, the path spelled by every suffix of s leads to a distinct leaf if we assume that the last symbol of s appears nowhere else in s. To achieve this, we just need to concatenate at the end of s a symbol not appearing in Σ.

Trees for representing all the suffixes of a set of strings $\{s_i,\ 1 \leq i \leq N$ for some $N \geq 2\}$ are called *generalized suffix trees* and are constructed in a way very similar to the construction of the suffix tree for a single string [1, 15]. We denote such generalized trees by $\mathcal{GT}$. They share all the properties of a suffix tree given in Section 8.5.3 with, in Property 8.5.1, string s substituted by strings $s_1, \ldots, s_N$.

In particular, a generalized suffix tree $\mathcal{GT}$ verifies the fact that every suffix of every string s_i in the set leads to a distinct leaf. When $p \geq 2$ strings have a same suffix, the generalized tree has therefore p leaves corresponding to this suffix, each associated with a different string. To achieve this property during construction, we just need to concatenate to each string s_i of the set a symbol that is not in Σ and is specific to that string.

To be able to spell valid models (*i.e.* models satisfying the quorum constraint), we need to add some information to the nodes of the suffix tree.

In the case where we are looking for repeats in a single string s, we just need to know, for each node x of $\mathcal{T}$, how many leaves are contained in the subtree rooted at x. Let us denote $leaves_x$ this number for each node x. Such information can be added to the tree by a simple traversal of it.

If we are dealing with $N \geq 2$ strings, and therefore a generalized suffix tree $\mathcal{GT}$, it is not enough anymore to know the value of $leaves_x$ for each node x in $\mathcal{GT}$ in order to be able to check whether a model remains valid. Indeed, for each node x, we need this time to know not the number of leaves in the subtree of $\mathcal{GT}$ having x as root, but that number for each different string the leaves refer to.

In order to do that, we must associate to each node x in $\mathcal{GT}$ an array, denoted $colours_x$, of dimension N, that is defined by:

$$colours_x[i] = \begin{cases} 1, & \text{if at least one leaf in the subtree rooted at } x \\ & \text{represents a suffix of } s_i; \\ 0, & \text{otherwise,} \end{cases}$$

for $1 \leq i \leq N$.

The array $colours_x$ for all x may also be obtained by a simple traversal of the tree with each visit to a node taking $O(N)$ time. The additional space required is $O(N)$ per node.

The main difference with the approach described in sections 8.5.1 and 8.5.2 is that occurrences are now grouped into classes and the "real" ones (that is, occurrences considered as individual words in the strings) are never directly manipulated. Occurrences of a model are thus in fact nodes of the suffix tree (we denote them by the term "node-occurrences") and are extended *in* the tree instead of in the string. Once the process of model spelling has ended, the start positions of the "real" occurrences of the valid models may be recovered by traversing the subtrees of the nodes reached so far and reading the labels of their leaves.

The algorithm is a development of the recurrence formula given in the lemma below where x denotes a node of the tree, $father(x)$ its father and d the number of errors between the label of the path going from the root to x as against a model m.

Lemma 8.5.4 *A pair (x, d) is a node-occurrence of $m' = m\alpha$ with $m \in \Sigma^k$ and $\alpha \in \Sigma$ if, and only if, $d \leq e$ and one of the following two conditions is verified:*

(match)	*$(father(x), d)$ is a node-occurrence of m and the label of the arc from $father(x)$ to x is α;*
(substitution)	*$(father(x), d-1)$ is a node-occurrence of m and the label of the arc from $father(x)$ to x is $\beta \neq \alpha$;*
(deletion)	*$(x, d-1)$ is a node-occurrence of m;*
(insertion)	*$(father(x), d-1)$ is a node-occurrence of $m\alpha$.*

If n is the average length of the strings and N their number, creating $colours_x$ for each node x of the tree takes time $O(nN^2)$, however manipulating it requires $O(N)$ time per model. Since there can be $O(nN\mathcal{V}(e,k))$ valid models in the worst case, the algorithm time complexity becomes $O(nN^2\ \mathcal{V}(e,k))$.

8.5.4 Structured models

Concerning structured models, solutions to variants of increasing generality of a same basic problem are proposed. Suffix trees are used in all cases. These variants may be stated as follows. Given a set of N strings $s_1, \ldots, s_N$, a nonnegative integer e and a positive integer q.

Problem 8.5.4 *Find all models* $((m_1, m_2), (d_{min_1}, d_{max_1}))$ *that are valid.*

Problem 8.5.5 *Find all models* $((m_1, \ldots, m_p),\ ((d_{min_1},\ d_{max_1}),\ \ldots,\ (d_{min_{p-1}},\ d_{max_{p-1}})))$ *that are valid, where* $p \geq 2$.

Problem 8.5.6 *Find all models* $((m_1, m_2), (d_{min_1}, d_{max_1}, \delta_1))$ *that are valid.*

Problem 8.5.7 *Find all models* $((m_1, \ldots, m_p),\ ((d_{min_1}, d_{max_1}, \delta_1), \ldots, (d_{min_{p-1}}, d_{max_{p-1}}, \delta_{p-1})))$ *that are valid, where* $p \geq 2$.

The last two problems represent situations where the exact intervals of distances separating the parts of a structured site are unknown, the only known fact being that these intervals cover a restricted range of values. How restricted is indicated by the δ_i parameters.

To simplify matters, we shall consider that, for $1 \leq i \leq p$, $m_i \in \Sigma^k$ where k is a positive integer, *i.e.* each single model m_i of a structured model (m, d) is of fixed, unique length k. In a likewise manner, we shall assume that each part m_i has the same substitution rate e and, when dealing with models composed of more than two boxes, that the d_{min_i}, d_{max_i} and, possibly, δ_i for $1 \leq i \leq p-1$ have identical values. We denote by d_{min}, d_{max} and δ these values. Problem 8.5.5 is then formulated as finding all models $((m_1, \ldots, m_p), (d_{min}, d_{max}))$ that are valid and Problem 8.5.7 as finding all valid models $((m_1, \ldots, m_p), (d_{min}, d_{max}, \delta))$.

Besides fixing a maximum substitution rate for each part in a structured model, one can also establish a maximum substitution rate for the whole model. Such a global error rate allows to consider in a limited way possible correlations between boxes in a model.

Another possible global, or local, constraint one may wish to consider for some applications concerns the composition of the boxes. One may, for instance, determine that the frequency of one or more nucleotide in a box (or among all boxes) be below or above a certain threshold. For structured models composed of more than p boxes, one may also establish that a box i is palindromic in relation to a box j for $1 \leq i < j \leq p$. In algorithmic terms, the two types of constraints just mentioned are not equivalent. The first type, box composition whether local or global, can in general be verified only *a posteriori* while the second type (palindromic boxes) will result in a, sometimes substantial, pruning of the virtual trie of models.

Introducing such additional constraints may in some cases require changes to the basic algorithms described below. The interested reader may find the details concerning such changes in the original paper [23, 24].

Algorithms for Problem 8.5.4

We start by presenting a naive approach then two algorithms that are efficient enough to tackle structured model extraction from big datasets. Since they will often be mentioned in what follows, we call them SMA$_1$ and SMA$_2$ (SM stands for Structured Model). The second algorithm has a better time complexity than the first but needs more space. The first is easier to understand and implement. Both are described in more detail than previous algorithms as structured models in some ways incorporate almost all other kinds of patterns we have been considering. The most notable exception concerns satellites that will be discussed in Section 8.5.5.

Other combinatorial approaches were developed for treating somewhat similar kinds of structured patterns. They either enumerate all possible (not just valid) patterns [43], do not allow for errors [17, 16] or are heuristics [11, 21].

A naive way of solving Problem 8.5.4 consists in extracting and storing all valid single models of length k (given q and e), and then, once this is finished, in checking which pairs of such models could represent valid structured models (given an interval of distance $[d_{min}, d_{max}]$).

The lemma used for building valid single models is the same as in Section 8.5.3 except that in practice, for most biological problems we wish to address [44, 45], substitutions only will in general be allowed. The lemma therefore becomes:

Lemma 8.5.5 *A pair (x, d) is a node-occurrence of $m' = m\alpha$ with $m \in \Sigma^k$ and $\alpha \in \Sigma$ if, and only if, $d \leq e$ and one of the following two conditions is verified:*

(match)	*$(father(x), d)$ is a node-occurrence of m and the label of the arc from $father(x)$ to x is α;*
(substitution)	*$(father(x), d-1)$ is a node-occurrence of m and the label of the arc from $father(x)$ to x is $\beta \neq \alpha$.*

One way of doing the verification profits from the simple observation that two single models m_1 and m_2 may form a structured one if, and only if, at least one occurrence of m_1 is at the right distance of at least one occurrence of m_2. Building an array of size nN where cell i contains the list of models having an occurrence starting at that position in $s = s_1 \ldots s_N$ allows to compare models in cell i to models in cells $i + d_{min}, \ldots, i + d_{max}$ only. If the sets of occurrences of models are ordered, this comparison may be done in an efficient way (in time proportional to the size of the sets of node-occurrences, which is upper-bounded by nN).

Algorithm SMA$_1$*: Jumping in the suffix tree*

A first non-naive approach to solving the problem starts by extracting single models of length k. Since we are traversing the trie of models in depth-first fashion (also in lexicographic order), models are recursively extracted one by one. At any time, a single model m (and its prefixes) is being considered. Once a valid model m_1 of length k is obtained together with its set of $\mathcal{T}$-node-occurrences V_1 (which are nodes located at level k in $\mathcal{GT}$), the extraction of all single models m_2 with which m_1 could form a structured model $((m_1, m_2), (d_{min}, d_{max}))$ starts. This is done with m_2 representing the empty word and having as node-occurrences the set V_2 given by:

$$\begin{aligned} V_2 \quad = \quad & \{(w, e_w = e_v) \mid \text{there exists } v \text{ in } V_1 \text{ ancestor of } w \text{ with} \\ & d_{min} \leq \; level(w) - \; level(v) \leq d_{max}\}, \end{aligned}$$

where level(v) indicates the level of node v in $\mathcal{GT}$. From a node-occurrence v in V_1, a jump is therefore made in $\mathcal{GT}$ to all potential start node-occurrences w of m_2. These nodes are the d_{min}- to d_{max}-generation descendants of v in $\mathcal{GT}$. Exactly the same recurrence formula given in Lemma 8.5.5 may be applied to the nodes w in V_2 to extract all single models m_2 that, together with m_1 could form a structured model verifying the conditions of the problem, for all valid m_1. An illustration is given in Figure 8.7 and a pseudo-code is presented below. The procedure ExtractModels is called with m the empty word having as sole node-occurrence the root of $\mathcal{GT}$ and with $i = 1$.

```
Algorithm SMA_1
procedure ExtractModels(Model $m$, Block $i$)
1.  for each node-occurrence $v$ of $m$ do
2.      if $i = 2$ then
3.          put in $PotentialStarts$ the children $w$ of $v$ at levels $k + d_{min}$
            to $k + d_{max}$
4.      else
5.          put $v$ (i.e., the root) in $PotentialStarts$
6.  for each model $m_i$ (and its occurrences) obtained by doing a
            recursive depth-first traversal from the root of the virtual
            model tree $\mathcal{M}$ while $\mathcal{M}$ while simultaneously traversing $\mathcal{GT}$
            from the node-occurrences in $PotentialStarts$ (Lemma 8.5.5
            and quorum constraint) do
```

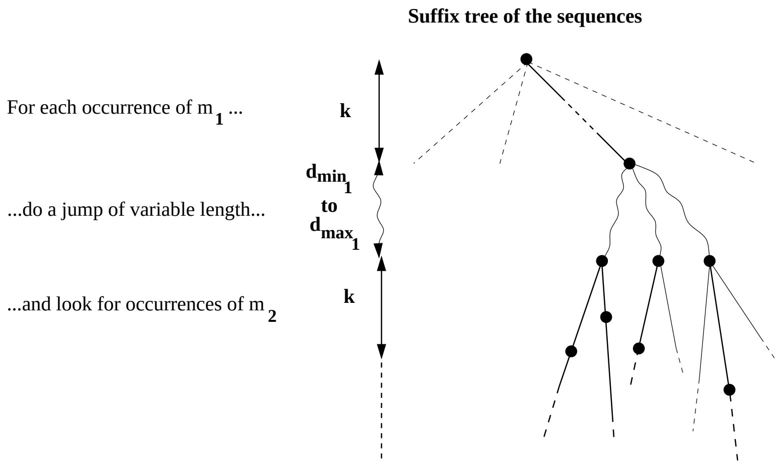

Suffix tree of the sequences
For each occurrence of m 1 ...
...do a jump of variable length...
...and look for occurrences of m 2
k
d min 1
to
d max 1
k

Figure 8.7. Extracting structured models (in the context of Problem 8.5.4) with a suffix tree – An illustration of Algorithm SMA$_1$.

```
7.          **if** $i = 1$ **then**
8.              ExtractModels($m = m_1$, $i + 1$)
9.          **else**
10.             report the complete model $m = ((m_1, m_2), (d_{min}, d_{max}))$
                as valid
```

Since the minimum and maximum length of a structured model (m, d) that may be considered are, respectively, $2k + d_{min}$ and $2k + d_{max}$, we need only build the tree of suffixes of length $2k + d_{min}$ or more, and for each such suffix to consider at most the first $2k + d_{max}$ symbols.

The observation made in the previous paragraph applies also to the second algorithm (to be described in what follows). Note that, in both cases, this implies $n_i \leq n_{i+1} \leq Nn$ for all $i \geq 1$ where n_i is the number of nodes at depth i in $\mathcal{GT}$.

Algorithm SMA$_2$*: Modifying the suffix tree*

Algorithm SMA$_2$ initially proceeds like Algorithm SMA$_1$: it starts by building single models of length k, one at a time. For each node-occurrence v of a first part m_1 considered in turn, a jump is made in $\mathcal{GT}$ down to the descendants of v situated at lower levels. This time however, the algorithm just passes through the nodes at these lower levels, grabs some information the nodes contain and jumps back up to level k again (in a way that will be explained in a short while). The information grabbed in passing is used to temporarily and partially modify $\mathcal{GT}$ and start, *from the root of* $\mathcal{GT}$, the extraction of the second part m_2 of a potentially valid structured model $((m_1, m_2), (d_{min}, d_{max}))$. Once the operation of extracting all possible companions m_2 for m_1 has ended, that part of $\mathcal{GT}$ that was modified is restored to its previous state. The construction of another single model m_1 of a structured model $((m_1, m_2), (d_{min}, d_{max}))$ then follows. The whole process then unwinds in a recursive way until all structured models satisfying the initial conditions are extracted.

More precisely, the operation between the spelling of models m_1 and m_2 locally alterates $\mathcal{GT}$ up to level k to a tree $\mathcal{GT}'$ that contains only the k-long prefixes of suffixes of $\{s_1, \ldots, s_N\}$ starting at a position between d_{min} and d_{max} from the end position in s_i of an occurrence of m_1. Tree $\mathcal{GT}'$ is, in a sense, the union of all the subtrees t of depth at most k rooted at nodes that represent start occurrences of a potential companion m_2 for m_1.

For each model m_1 obtained, before spelling all possible companions m_2 for m_1, the content of $colours_z$ for all nodes z at level k in $\mathcal{GT}$ are stored in an array L of dimension n_k (this is for later restoration of $\mathcal{GT}$). Tree $\mathcal{GT}'$ is then obtained from $\mathcal{GT}$ by considering all nodes w in $\mathcal{GT}$ that may be reached on a descent of, this time, $k + d_{min}$ to $k + d_{max}$ arcs down from the node-occurrences (v, e_v) of m_1. These correspond to all end node-occurrences (instead of start as in the first algorithm) of potentially valid models having m_1 as first part. The boolean arrays $colours_w$ for all w indicate to which input strings these occurrences belong. This is the information we grab in passing and take along the only path of suffix links in $\mathcal{GT}$ that leads back to a node z at level k in $\mathcal{GT}$. If it is the first time z is reached, $colours_z$ is set equal to $colours_w$, otherwise $colours_w$ is added (boolean "or" operation) to $colours_z$. Once all nodes v and w have been treated, the information contained in the nodes z that were reached during this operation are propagated up the tree from level k to the root (using normal tree arcs) in the following way: if $\bar{z}$ and $\hat{z}$ have same parent z, then $colours_z = colours_{\bar{z}} \cup colours_{\hat{z}}$. Any arc from the root that is not visited at least once in such a traversal up the tree is not part of $\mathcal{GT}'$, nor are the subtrees rooted at its end node.

The extraction of all second parts m_2 of a structured model (m, d) follows as for single models in the initial algorithm (Lemma 8.5.5).

Restoring the tree $\mathcal{GT}$ as it was before the operations described above requires restoring the value of $colours_z$ preserved in L for all nodes z at level k and propagating the information (state of boolean arrays) from z up to the root.

Since nodes w at level between $2k + d_{min}$ to $2k + d_{max}$ will be solicited for the same operation over and over again, which consists in following the unique suffix-link path from w to a node z at level k in $\mathcal{GT}$, $\mathcal{GT}$ is pretreated so that one single link has to be followed from z. Going from w to z takes then constant time.

A pictorial illustration of Algorithm SMA$_2$ is given in Figure 8.8.

A pseudo-code for the algorithm is as follows. The procedure Extract-Models is called, as in the previous algorithm, with m the empty word having as sole node-occurrence the root of $\mathcal{GT}$ and with $i = 1$.

```
**Algorithm** SMA$_2$
**procedure** ExtractModels(Model $m$, Block $i$)
1.  **for** each node-occurrence $v$ of $m$ **do**
2.      **if** $i = 2$ **then**
3.          put in $PotentialEnds$ the children $w$ at levels $2k + d_{min}$ to
            $2k + d_{max}$
4.          **for** each node-occurrence $w$ in $PotentialEnds$ **do**
5.              follow fast suffix-link to node $z$ at level $k$
6.              put $z$ in $L$
7.              **if** first time $z$ is reached **then**
8.                  initialize $colours_z$ with zero
9.                  put $z$ in $NextEnds$
10.             add $colours_w$ to $colours_z$
11.         do a depth-first traversal of $\mathcal{GT}$ to update the boolean arrays
                from the root to all $z$ in $NextEnds$ (let $\mathcal{GT}'$ be the $k$-deep
                tree obtained by such an operation)
12.     **if** $i = 1$ **then**
13.         $Tree = \mathcal{GT}$
14.     **else**
15.         $Tree = \mathcal{GT}'$
16. **for** each model $m_i$ (and its occurrences) obtained by doing a recursive
            depth-first traversal from the root of the virtual model tree
            $\mathcal{M}$ while simultaneously traversing $Tree$ from the root
            (Lemma 8.5.5 and quorum constraint) **do**
17.     **if** $i = 1$ **then**
18.         ExtractModels($m = m_1$, $i + 1$)
19.     **else**
20.         report the complete model $m = ((m_1, m_2), (d_{min}, d_{max}))$ as valid
21.         restore tree $\mathcal{GT}$ to its original state using $L$
```

Proposition 8.5.1 *The following two statements are true:*

- *$\mathcal{GT}'$ contains only the k-long prefixes of suffixes of $\{s_1, \ldots, s_N\}$ that start at a position between d_{min} and d_{max} of the end position in $\{s_1, \ldots, s_N\}$ of an occurrence of m_1;*
- *the above algorithm solves Problem 8.5.4.*

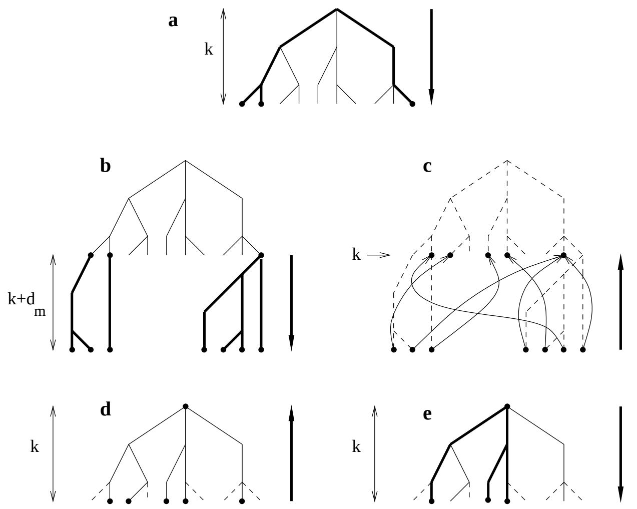


Figure 8.8. Extracting structured models (in the context of Problem 8.5.4) with a suffix tree – An illustration of Algorithm SMA$_2$.

The proof is straightforward and may be found in [23, 24].

Let us now analyse the complexity of the algorithms we have described. The naive approach to solve Problem 8.5.4 requires $nN^2\mathcal{V}(e,k)$ time to find single models that could correspond to either part of a structured model (and $nN\mathcal{V}(e,k)$ space to store all potential parts). If we denote by Δ the value $d_{max} - d_{min} + 1$, finding which pair of single models may be put together to produce a structured model could then be done in time proportional to:

$$\underbrace{\mathcal{V}(e,k)}_{(1)}\underbrace{\Delta\mathcal{V}(e,k)}_{(2)}\underbrace{nN}_{(3)}\underbrace{nN}_{(4)}$$

where (1) is the maximum number of single models to which a position may belong, (2) is the maximum number of models to which a position at a distance between $k+d_{min}$ and $k+d_{max}$ from the first may belong, (3) is the maximum number of comparisons that must be done to check whether two single models may form a structured one and, finally, (4) is the number of starting positions to consider.

To obtain the complexity of Algorithm SMA$_1$, we have to calculate the total number of visits we may do to nodes between level $2k+d_{max}$ (the deeper level we ever reach) and the root. To count this, we need to consider, for each node between levels $2k+d_{min}$ and $2k+d_{max}$ in $\mathcal{GT}$, how many times it could represent the node-occurrence of a model composed of two boxes, each one having length k and separated by a space of length d_{min} to d_{max}. This number is at most:

$$\begin{array}{rcl}\sum_{i=d_{min}}^{d_{max}} n_{2k+i}\mathcal{V}^2(e,k) & \leq & \min\{2,\Delta\}n_{2k+d_{max}}\mathcal{V}^2(e,k) \\ & \leq & \min\{2,\Delta\}n_{2k+d_{max}}k^{2e}|\Sigma|^{2e},\end{array}$$

where Δ denotes the value $d_{max} - d_{min} + 1$ and $n_{2k+d_{max}}$ is the number of tree nodes at depth $2k + d_{max}$. This last number is never more than nN. The $\min\{2,\Delta\}$ in the bound comes from the fact that the degree of any internal node of $\mathcal{GT}$ is at least 2. Since each visit to a node requires at most $O(N)$ operations, the time complexity of the Algorithm SMA$_1$ is $O(\min\{2,\Delta\}Nn_{2k+d_{max}}\mathcal{V}^2(e,k))$, that is, $O(Nn_{2k+d_{max}}\mathcal{V}^2(e,k))$. The space complexity is $O(N^2n)$, as for the extraction of single models.

In the case of Algorithm SMA$_2$, we have to consider the number of operations necessary for building the two parts of each model using $\mathcal{GT}$ or $\mathcal{GT}'$, as well as the number of operations needed to obtain $\mathcal{GT}'$ from $\mathcal{GT}$ and then to restore back $\mathcal{GT}$.

The single models composing either two parts of a structured model may be built in at most $Nn_k\mathcal{V}^2(e,k)$ operations. The reason for this is that, when spelling either part of a model, we are working with nodes between the root and level k only (there are at most $2n_k$ such nodes), and there are $\mathcal{V}^2(e,k)$ ways of spelling two paths from a node at level k to the root (each path corresponding to one part of a structured model) allowing for up to e substitutions in each.

The total number of operations needed to modify the first k levels of the suffix tree $\mathcal{GT}$ to obtain $\mathcal{GT}'$ before the identification of a second part at a right distance of the first is upper-bounded by

$$\sum_{i=d_{min}}^{d_{max}} Nn_{2k+i}\mathcal{V}(e,k) + Nn_k\mathcal{V}(e,k) \leq \min\{2,\Delta\}Nn_{2k+d_{max}}\mathcal{V}(e,k),$$

where the first summand corresponds to the visits to nodes z coming from w for all m_1 and the second one corresponds to the propagations from z to the root for all m_1.

Restoring $\mathcal{GT}$ to start the extraction of another structured model from a different first part takes $O(Nn_k\mathcal{V}(e,k))$ operations using $O(Nn_k)$ additional space (size of array L, each cell possibly pointing to a node at level k in $\mathcal{GT}$ or to nil). The total time complexity of the second algorithm is therefore $O(Nn_k\mathcal{V}^2(e,k)+\min\{2,\Delta\}Nn_{2k+d_{max}}\mathcal{V}(e,k)+Nn_k\mathcal{V}(e,k))$. This results in an $O(Nn_k\mathcal{V}^2(e,k) + Nn_{2k+d_{max}}\mathcal{V}(e,k))$ time complexity. Space complexity is slightly higher than for the previous algorithm: $O(N^2n+Nn_k)$ where $n_k \leq Nn$. The second term is for array L.

In either case, the complexity obtained is better both in terms of time and space than the one given by a naive approach to Problem 8.5.4 (see above).

Algorithms for Problem 8.5.5

We describe now how the algorithms seem for Problem 8.5.4 can be extended to extract structured models with $p > 2$ parts.

It is immediate how to extend Algorithm SMA$_1$ to extract structured models composed of $p > 2$ parts. After extracting the first i parts of a structured model $((m_1, \ldots, m_p), (d_{min}, d_{max}))$ for $1 \leq i < p-1$, one jumps down in the tree $\mathcal{GT}$ (following normal tree arcs) to get to the d_{min}- to d_{max}-descendants of every node-occurrence of $((m_1, \ldots, m_i), (d_{min}, d_{max}))$ then continues the extraction from there using Lemma 8.5.5.

A pseudo-code is given below.

```
**Algorithm** Extended-SMA$_1$
**procedure** ExtractModels(Model $m$, Block $i$)
1.  **for** each node-occurrence $v$ of $m$ **do**
2.      **if** $i > 1$ **then**
3.          put in *PotentialStarts* the children $w$ of $v$ at levels
            $(i-1)k + (i-1)d_{min}$ to $(i-1)k + (i-1)d_{max}$
4.      **else**
5.          put $v$ (the root) in *PotentialStarts*
6.  **for** each model $m_i$ (and its occurrences) obtained by doing a recursive
            depth-first traversal from the root of the virtual model tree $\mathcal{M}$
            while simultaneously traversing $\mathcal{GT}$ from the node-occurrences
            in *PotentialStarts* (Lemma 8.5.5 and quorum constraint) **do**
7.      **if** $i < p$ **then**
8.          ExtractModels($m = m_1 \cdots m_i$, $i+1$)
9.      **else**
10.         report the complete model $m = ((m_1, \cdots, m_p), (d_{min}, d_{max}))$
            as valid
```

Let us now describe how to extend Algorithm SMA$_2$ to solve Problem 8.5.5. This extension is slightly more complex then the previous one and thus calls for a few remarks. The operations done to modify the tree between building $m_{i \geq 1}$ and m_{i+1} are almost the same as those described for Algorithm SMA$_2$, except for two facts. One is that up to $(p-1)$ arrays L are now needed to restore the tree after each modification it undergoes. The second, more important difference is that we need to keep for each node v_k at level k reached from an ascent up $\mathcal{GT}$'s suffix links a list, noted $Lptr_{v_k}$, of pointers to the nodes at lower levels that affected the contents of v_k. The reason for this is that tree $\mathcal{GT}$ is modified up to level k only (resulting in tree $\mathcal{GT}'$) as these are the only levels concerned by the search for occurrences of each box of a structured model. Lower levels of $\mathcal{GT}$ remain unchanged, in particular the boolean arrays at each node below level k. To obtain the correct information concerning the potential end node-occurrences of boxes

i for $i > 2$ (*i.e.* to which strings such occurrences belong), we therefore cannot descend $\mathcal{GT}$ from the ends of node-occurrences in $\mathcal{GT}'$ of box $(i-1)$. If we did, we would not miss any occurrence but we could get more, *e.g.* ones that did not have an occurrence of a previous box in the model. We might thus overcount some strings and consider as valid a model which, in fact, no longer satisfied the quorum. We have to go down $\mathcal{GT}$ from the ends of node-occurrences *in* $\mathcal{GT}$, that is from the original ends of node-occurrences in $\mathcal{GT}$ of the boxes built so far. These are reached from the list of pointers $Lptr_{v_k}$ for the nodes v_k that are identified as occurrences of the box currently just treated. For models composed of p boxes, we need at most $(p-1)$ lists $Lptr_{v_k}$ for each node v_k at level k.

A pseudo-code for the algorithm is as follows:

```
Algorithm Extended-SMA$_2$
procedure ExtractModels(Model $m$, Block $i$)
1.  for each node-occurrence $v$ of $m$ do
2.      if $i > 2$ then
3.          put in $PotentialEnds$ the children $w$ at levels $ik + (i-1)d_{min}$
                to $ik + (i-1)d_{max}$
4.          for each node-occurrence $w$ in $PotentialEnds$ do
5.              follow fast suffix-link to node $z$ at level $k$
6.              put $z$ in $L(i)$
7.              if first time $z$ is reached then
8.                  initialize $colours_z$ with zero
9.                  put $z$ in $NextEnds$
10.             add $colours_w$ to $colours_z$
11.         do a depth-first traversal of $\mathcal{GT}$ to update the boolean arrays
                from the root to all $z$ in $NextEnds$ (let $\mathcal{GT}'$ be the $k$-deep
                tree obtained by such an operation)
12.     if $i = 1$ then
13.         $Tree = \mathcal{GT}$
14.     else
15.         $Tree = \mathcal{GT}'$
16. for each model $m_i$ (and its occurrences) obtained by doing a recursive
            depth-first traversal from the root of the virtual model tree
            $\mathcal{M}$ while simultaneously traversing $Tree$ from the root
            (Lemma 8.5.5 and quorum constraint) do
17.     if $i < p$ then
18.         ExtractModels($m = m_1 \cdots m_i$, $i + 1$)
19.     else
20.         report the complete model $m = ((m_1, \cdots, m_p), (d_{min}, d_{max}))$
            as valid
21.     if $i > 1$ then
22.         restore tree $\mathcal{GT}$ to its original state using $L(i)$
```

Using a same reasoning as before, it is not difficult to see that the first algorithm requires $O(Nn_{pk+(p-1)d_{max}}\ \mathcal{V}^p(e,k))$ time, where $\mathcal{V}^p(e,k)) \leq k^{pe}|\Sigma|^{pe}$. The space complexity remains the same as for solving Problem 8.5.4, that is $O(N^2n)$.

In the case of the second algorithm, the p single models composing a structured model may be built in a number of operations upper bounded by $O(Nn_k\ \mathcal{V}^p(e,k))$.

The total number of operations needed to modify the first k levels of the suffix tree $\mathcal{GT}$ to obtain $\mathcal{GT}'$ before the identification of a box $(i+1)$ for $i > 2$ at a right distance of box i is upper-bounded by:

$$\textstyle\sum_{j=d_{min}}^{d_{max}} Nn_{ik+(i-1)j}\mathcal{V}^{i-1}(e,k)) + (Nn_k\mathcal{V}(e,k),$$

which is at most

$$\min\{2,\Delta\}Nn_{ik+(i-1)d_{max}}\mathcal{V}^{i-1}(e,k).$$

Restoring $\mathcal{GT}'$ as we back off to the preceding box takes, as before, $O(Nn_k\ \mathcal{V}(e,k))$ operations using $O(N(p-1)n_k)$ additional space (size of arrays $L(1)$ to $L(p)$).

The total time complexity of the second algorithm is therefore of $O(Nn_k\mathcal{V}^p(e,k) + Nn_{pk+(p-1)d_{max}}\ \mathcal{V}^{p-1}(e,k))$. The space complexity is $O(N^2n + N(p-1)n_k)$.

Algorithms for Problem 8.5.6

Let us now extend the previous algorithms to handle restricted intervals of unknown limits. In the case where the distances between the two parts m_1 and m_2 of a single model vary inside a restricted interval whose limits are unknown, Algorithm SMA$_1$ can be extended in the following way. Once a first part m_1 of a structured model $((m_1,m_2),(d_{min},d_{max},\delta))$ has been extracted, we jump as before to nodes w in V_2. As we now must verify that:

- there exists d, with $d_{min}+\delta \leq d \leq d_{max}-\delta$, such that level($w$) - level($v$) is equal to $d \pm \delta$;
- (more particularly) d is *the same* for pairs of occurrences (one occurrence for each part of the structured model) present in at least q *distinct strings*;

we just need to keep at each node its distance from level k and to count the number of distinct strings for each restricted interval $d \pm \delta$ separately.

A second algorithm can be derived, now extending Algorithm SMA$_2$. In this case, in order to verify the same two points mentioned above, we have to keep an additional information at the nodes z situated at level k that are reached from w by jumping back up the tree (following suffix links). This information is required because a node at level k may be reached from nodes

w corresponding to different distances from occurrences of the previous box. We therefore need to have at each node z an array of dimension not N but $((d_{max} - d_{min} - (2 * \delta)) \times N)$. The node-occurrences at each extension step of the second part of a model are added for each cell $i \in (d_{max} - d_{min} - (2 * \delta))$ in turn. If for any i, this number is at least q, the model is valid and the second part may be further extended (if its length is still smaller than k).

We denote this boolean array $Colours_z$ with a capital C to stress that it is now multi-dimensional. If it is the first time a node z is reached from w, the l cells of $Colours_z$ for $l \in [\max\{d_{min} + \delta, \text{level}(w) - \text{level}(z) - \delta\}, \min\{d_{max} - \delta, \text{level}(w) - \text{level}(z) + \delta\}]$ are set equal to $colours_w$ and all the other cells are initialized to zero, otherwise $colours_w$ is added (boolean "or") to the l cells of $Colours_z$. Once all nodes v and w have been treated, the information contained in the nodes z that were reached during this operation are propagated up the tree from level k to the root (using normal tree arcs) in the following way: if $\bar{z}$ and $\hat{z}$ have same parent z, then, for all l such that $d_{min} + \delta_1 \leq l \leq d_{max} - \delta_1$, $Colours_z[l] = Colours_{\bar{z}}[l] \cup Colours_{\hat{z}}[l]$.

The time complexity of the first algorithm described above for solving Problem 8.5.6 remains $O(N n_{2k+d_{max}} \mathcal{V}^2(e,k))$ and the space complexity $O(nN^2)$.

The time complexity of the second algorithm for solving the same problem becomes $O(N\ \Delta'\ n_k \mathcal{V}^2(e,k) + N\Delta' n_{2k+d_{max}}\ \mathcal{V}(e,k))$ where $\Delta' = d_{max} - d_{min} - (2 * \delta)$. The space complexity is $O(N^2 n + N\Delta' n_k)$.

Algorithms for Problem 8.5.7

Few changes in the previous ideas are required when one wishes to consider structured models that are composed of more than two boxes separated by intervals of distances of the type $d \pm \delta$ for some d and a fixed δ. The main one concerns the second algorithm: the boolean arrays at each node in the suffix tree have now to be of dimension $N(p-1)\Delta'$. The Δ' comes from having to handle restricted intervals of unknown limits as we saw in the previous problem. The $(p-1)$ comes from the fact that d may be different for each pair of successive boxes in the structured model. The time and space complexity will therefore be further multiplied by a term of $(p-1)$.

The time complexity of the second algorithm (the only one for which there is a change) for solving Problem 8.5.7 is $O(N\Delta'(p-1)n_k\mathcal{V}^p(e,k) + N\Delta'(p-1)n_{pk+(p-1)d_{max}}\ \mathcal{V}^{p-1}(e,k))$. The space complexity is $O(N^2 n + N\Delta'(p-1)n_k)$.

8.5.5 Satellites

The satellite problem we propose to solve is the following:

Problem 8.5.8 *Given a string s and parameters min_repeat, min_range, max_range, max_jump, and e (possibly also g), find all consensus satellite models m that are valid for s.*

In fact, the original papers [34, 35] reported a set of disjoint “fittest” trains realizing each model m given a measure of “fitness”.

The algorithm presented below is the only combinatorial, non-heuristical developed so far for identifying tandem arrays. Other exact approaches either treated the case of tandem repeats only [22, 19], did not allow for errors [3, 5, 26, 46] or required generating all possible (not just valid) models of a given length [8, 10, 32].

We first treat the problem of building prefix satellite models.

As with all previous cases considered in this paper, satellite models are constructed by increasing lengths. In order to determine if a model is valid, we must have some representation of the train or wagons that make it so. There are two possibilities:

- we can keep track of each validating train and its associated wagons, or
- we can keep track of individual wagons, and, on the fly, determine if they can be combined into validating trains.

The first possibility is appealing because model extension is straightforward. We would just have to check, for each wagon of each train, whether it can be extended in relation to the extended model, and then count how many wagons remain to see whether the train it belonged to is still a valid one. However, there are generally many overlapping trains involving many of the same wagons for a given model. Common wagons may be present more than once in the list of occurrences of m if this is kept as a list of trains. This approach entails redundancies that lead to an inefficient algorithm. We therefore adopt the second approach, of keeping track of wagons and determining if they can be assembled into trains as needed.

The rules of prefix-model extension are given in Lemma 8.5.6 below. A wagon is identified by a triple (i, j, d) indicating that it is the substring $s_i s_{i+1} \dots s_j$ of s and is $d \leq e$ differences away from its model. Position i indicates the left-end of the wagon, and j its right-end. Contrary to the previous algorithms presented in this paper, models and their occurrences (the wagons) will be extended to the left. This is just to facilitate verifying Property 8.4.3. Right ends of occurrences are calculated but will be used only for checking Property 8.4.4.

Lemma 8.5.6 *The triple (i, j, d) encodes a wagon of $m' = \alpha m$ with $\alpha \in \Sigma$ and $m \in \Sigma^k$ if, and only if, $d \leq e$ and at least one of the following conditions is true:*

(match)	$(i+1, j, d)$ *is a wagon of* m *and* $s_i = \alpha$;
(substitution)	$(i+1, j, d-1)$ *is a wagon of* m *and* $s_i \neq \alpha$;
(deletion)	$(i, j, d-1)$ *is a wagon of* m;
(insertion)	$(i+1, j, d-1)$ *is a wagon of* αm.

For each prefix-model m, we keep a list of the wagons of m that are in at least one train validating m. We describe such wagons as being valid with respect to m. When we extend a model (to the left) to $m' = \alpha m$, we perform two tasks:

- First, determine which valid wagons of m can be extended as above to become wagons of m'.
- Second, of these newly determined wagons of m', keep only those that are valid with respect to m'. This requires effectively assemblying wagons into trains, something that is not needed in an approach that would keep track of trains directly.

Note that we need not actually enumerate the trains in the second step, we simply must determine if a wagon is part of one. This will allow us to perform an extension step in time linear with the length of the string.

As a final insight, consider the directed graph $G = (V, E)$ where V is the set of all valid wagons and there is an edge from wagon u to v if $\mathit{left}_v - \mathit{left}_u \in \mathit{JUMP}$. Then a wagon u is valid if it is part of a path of length *min_repeat* or more in G. Determining this is quite simple as the graph is clearly acyclic. In the computation that will follow, we will effectively compute the length of the longest path to u in Lcnt_u and the length of the longest path from u in Rcnt_u. If $\mathit{Lcnt}_u + \mathit{Rcnt}_u > \mathit{min\_repeat}$ then u is valid.

We now consider how to obtain the onsensus satellite models.

We encode the collection of all wagons of m in a set, $L_m \subseteq \{1\ldots, n\}$, and an $(n+1) \times (2e+1)$-element array D_m as follows:

1. $i \in L_m$ if and only if i is the left-end of at least one wagon valid with respect to m.
2. for each $i \in L_m$, the value $D_m[i, \delta]$ for $\delta \in [-e, e]$ is the edit distance of m from wagon $s_i s_{i+1} \ldots s_{i+|m|-1+\delta}$.

Intuitively, L_m gives the left-ends of all valid wagons which is all we need to check Properties 8.4.2 and 8.4.3. D_m gives us the distances we need for extending models, together with the right-ends needed for checking Property 8.4.4. Formally, $(i, i+|m|-1+\delta, d)$ is a valid wagon of m if and only if $i \in L_m$ and $d = D_m[i, \delta] \leq e$.

The complete algorithm is described in the sequel. When Extend(αm) is called, it is assumed that L_m is known along with the relevant D_m values. The routine computes these items for the extension αm and recursively for the extensions thereof. Lines 0-5 compute the set of left-ends of wagons for

αm derivable from wagons of m that are valid. While Lemma 8.5.6 gives us a way to do so, recall that we are using dynamic programming to compute all extensions simultaneously. This corresponds to adding the last row to the dynamic programming matrix of s versus αm. At first L_m gives all the positions in row $|m|$ that have value e or less (and are valid) and D_m gives their values. From these, we compute the positions in row $|m| + 1$ in the obvious sparse fashion to arrive at $L_{\alpha m}$ and the values $D_{\alpha m}$.

```
**int** $Lcnt[1..n]$, $Rcnt[1..n]$

**procedure** Extend($\alpha m$)
         $L_{\alpha m} \leftarrow \emptyset$
1.   **for** $i + 1 \in L_m$ (in decreasing order) **do**
2.       **for** $\delta \in [-e, e]$ **do**
3.           $D_{\alpha m}[i, \delta] \leftarrow \min \left\{ \begin{array}{l} D_m[i+1, \delta] + (\textbf{if } s_i = \alpha \textbf{ then } 0 \textbf{ else } 1), \\ \textbf{if } i \in L_m \textbf{ then } D_m[i, \delta+1] + 1, \\ \textbf{if } i{+}1 \in L_{\alpha m} \textbf{ then } D_{\alpha m}[i+1, \delta-1] + 1 \end{array} \right\}$
4.       **if** $min_\delta\{D_{\alpha m}[i, \delta]\} \leq e$ **then**
5.           $L_{\alpha m} \leftarrow L_{\alpha m} \cup \{i\}$
6.   **for** $i \in L_{\alpha m}$ (in decreasing order) **do**
7.       $Rcnt[i] \leftarrow \max_{k \in (i+JUMP) \cap L_{\alpha m}}\{Rcnt[k]\} + 1$
8.   **for** $i \in L_{\alpha m}$ (in increasing order) **do**
9.       $Lcnt[i] \leftarrow \max_{k \in (i-JUMP) \cap L_{\alpha m}}\{Lcnt[k]\} + 1$
10.  **for** $i \in L_{\alpha m}$ **do**
11.      **if** $Lcnt[i] + Rcnt[i] \leq min\_repeat$ **then** $L_{\alpha m} \leftarrow L_{\alpha m} - \{i\}$
12.  **if** $L_{\alpha m} \neq \emptyset$ **then**
13.          **if** $|\alpha m| \in [min\_range, max\_range]$ **then**
14.              Record($\alpha m$)
15.          **if** $|\alpha m| < max\_range$ **then**
16.              **for** $\beta \in \Sigma$ **do**
17.                  Extend($\beta \alpha m$)
```

Once wagons have been extended when possible, we have to eliminate those that are no longer valid. This is performed by Lines 6 to 11. We compute, for each position $i \in L_{\alpha m}$, the maximum number of wagons in a train starting with a wagon whose left-end is at i in $Rcnt[i]$ (including itself), and the maximum number of wagons in a train ending with a wagon whose left-end is at i in $Lcnt[i]$. The necessary recurrences are given in Lines 7 and 9 of the algorithm where we recall that $JUMP = \{y : y \in \bigcup_{x \in [1, max\_jump]} x \times [min\_range, max\_range]\}$ and $i + JUMP$ denotes adding i to each element of $JUMP$. Observe that $Rcnt[i] + Lcnt[i] - 1$ is the length of the longest train containing a wagon whose left-end is at position i.

Clearly Lines 6-9 take $O(|L_{\alpha m}||JUMP|)$ time. However, when $L_{\alpha m}$ is a very large fraction of n, one can maintain an $Rcnt(Lcnt)$-prioritized queue of the positions in $(i + JUMP) \cap L_{\alpha m}$, to obtain an $O(n \cdot max\_jump \cdot \log|JUMP|)$ bound.

Finally in the remaining steps, Lines 12-17, the algorithm calls Record to record potential models and then recursively tries to extend the model if possible. The routine Record confirms that the model is a consensus model by checking Property 4.3 and recording the intervals spanned by trains valid for the consensus model, if any.

We start by observing that $O(|JUMP| + e)$ time is spent on a given left-end position for each prefix model matching the string beginning at that position. The term e comes from Lines 2 and 3 while $|JUMP|$ is the number of back or forward elements that have to be examined in order to determine, for each wagon, the length of the longest train it may belong to (Lines 9 and 11). The number of prefix models that could match a given position with e or fewer errors is by definition $\Sigma_{k=1}^{max\_range}\mathcal{V}(e,k)$. The total time taken by the algorithm is therefore bounded above by $O(n\cdot(|JUMP|+e)\cdot max\_range\cdot \mathcal{V}(e, max\_range)) = O(n \cdot max\_range^2 \cdot max\_jump \cdot \mathcal{V}(e, max\_range))$ as $e < max\_range$.

The space requirement is that of keeping all the information concerning at most max_range models at a time (a model m and all its prefixes). It is therefore on the order of at most $O(n \cdot max\_range \cdot e)$ as only $O(ne)$ storage is required to record the left-end positions and edit-distance at each possible right-end.

One of the by-products of the approach for finding satellites described above is that the algorithm is also capable of identifying periodic, or approximately periodic repeats that may be non-contiguous. These correspond to prefix models, *i.e.* to models that verify properties 8.4.2 and 8.4.3 but not necessarily Property 8.4.4.

A more substantial modification of the algorithm allows us to treat the presence of inverted repeats amongst the direct ones when these occur in tandem. The exercise of doing so entails distinguishing between direct and inverted wagons, extending properties 8.4.2 and 8.4.3 to accommodate the inverted wagons, and then modifying the basic algorithm accordingly.

The extension of properties 8.4.3 and 8.4.4 gives raise to the following new versions:

Property (8.4.3)'

$$end_{u_{i+1}} - end_{u_i} \in \begin{cases} JUMP, & \text{if both } u_i \text{ and } u_{i+1} \text{ are direct} \\ & \text{or both are inverted;} \\ JUMP\_INV, & \text{otherwise;} \end{cases}$$

where

$$end_{u_{i+1}} - end_{u_i} \in \begin{cases} left_u, & \text{if } u \text{ is direct;} \\ right_u, & \text{otherwise;} \end{cases}$$

and

$$JUMP\_INV = \bigcup_{x \in [0, max\_jump-1]} [x \times min\_range, (x+2) \times max\_range].$$

Property (8.4.4)'

$$start_{u_{i+1}} - end_{u_i} \in GAP,$$

where

$$start_u = \begin{cases} right_u, & \text{if } u \text{ is direct;} \\ left_u, & \text{otherwise;} \end{cases}$$

and

$$end_u = \begin{cases} left_u, & \text{if } u \text{ is direct;} \\ right_u, & \text{otherwise.} \end{cases}$$

A wagon is now identified by a quadruple (i, j, d, f) where the additional variable f is a flag indicating whether we are dealing with a direct or an inverted occurrence. Further details may be found in [34].

References

[1] P. Bieganski, J. Riedl, J. V. Carlis, and E. Retzel.Generalized suffix trees for biological sequence data: applications and implementations.In *Proc. of the 27th Hawai Int. Conf. on Systems Sci.*, pages 35–44. IEEE Computer Society Press, 1994.

[2] B. Charlesworth, P. Sniegowski, and W. Stephan.The evolutionary dynamics of repetitive DNA in eukaryotes.*Nature*, 371:215–220, 1994.

[3] B. Clift, D. Haussler, R. McConnell, T. D. Schneider, and G. D. Stormo.Sequence landscapes.*Nucleic Acids Res.*, 14:141–158, 1986.

[4] T. E. Creighton.*Proteins: Structures and Molecular Properties.*W.H. Freeman, 1993.

[5] M. Crochemore.An optimal algorithm for computing the repetitions in a word.*Inf. Proc. Letters*, 12:244–250, 1981.

[6] M. Crochemore and W. Rytter.*Text algorithms.*Oxford University Press, 1994.

[7] M. Dayhoff, R. Schwartz, and B. Orcutt.A model of evolutionary change in proteins.In M. Dayhoff, editor, *Atlas of Protein Sequence an Structure*, volume 5 suppl.3, pages 345–352. Natl. Biomed. Res. Found., 1978.

[8] O. Delgrange.*Un algorithme rapide pour une compression modulaire optimale. Application à l'analyse de séquences génétiques.*Thèse de doctorat, Université de Lille I, 1997.

[9] V. Escalier, J. Pothier, H. Soldano, and A. Viari.Pairwise and multiple identification of three dimensional common substructures in proteins.*J. Computational Biology*, 1996.

[10] V. Fischetti, G. Landau, J. Schmidt, and P. Sellers.Identifying periodic occurrences of a template with applications to protein structure.In Z. G. A. Apostolico, M. Crochemore and U. Manber, editors, *Combinatorial Pattern Matching*, volume 644 of *Lecture Notes in Computer Science*, pages 111–120. Springer-Verlag, 1992.

[11] Y. M. Fraenkel, Y. Mandel, D. Friedberg, and H. Margalit.Identification of common motifs in unaligned DNA sequences: application to *escherichia coli lrp* regulon.*Comput. Appl. Biosci.*, 11:379–387, 1995.

[12] D. J. Galas, M. Eggert, and M. S. Waterman.Rigorous pattern-recognition methods for DNA sequences. analysis of promoter sequences from escherichia coli.*J. Mol. Biol.*, 186:117–128, 1985.

[13] D. Gusfield.*Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology.*Cambridge University Press, 1997.

[14] S. Henikoff and J. Henikoff.Amino acid substitution matrices from protein blocks.*Proc. Natl. Acad. Sci. USA*, 89:10915–10919, 1992.

[15] L. C. K. Hui.Color set size problem with applications to string matching.In A. Apostolico, M. Crochemore, Z. Galil, and U. Manber, editors, *Combinatorial Pattern Matching*, volume 644 of *Lecture Notes in Computer Science*, pages 230–243. Springer-Verlag, 1992.

[16] I. Jonassen.Efficient discovery of conserved patterns using a pattern graph.*Comput. Appl. Biosci.*, 13:509–522, 1997.

[17] I. Jonassen, J. F. Collins, and D. G. Higgins.Finding flexible patterns in unaligned protein sequences.*Protein Science*, 4:1587–1595, 1995.

[18] I. Jonassen, I. Eidhammer, and W. R. Taylor.Discovery of local packing motifs in protein structures.*Proteins: Structure, Function, and Genetics*, 34:206–219, 1999.

[19] S. K. Kannan and E. W. Myers.An algorithm for locating non-overlapping regions of maximum alignment score.In Z. G. A. Apostolico, M. Crochemore and U. Manber, editors, *Combinatorial Pattern Matching*, volume 684 of *Lecture Notes in Computer Science*, page 7486. Springer-Verlag, 1993.

[20] R. Karp, R. Miller, and A. Rosenberg.Rapid identification of repeated patterns in strings, trees and arrays.In *Proc. 4th Annu. ACM Symp. Theory of Computing*, pages 125–136, 1972.

[21] A. Klingenhoff, K. Frech, K. Quandt, and T. Werner.Functional promoter modules can be detected by formal models independent of overall nucleotide sequence similarity.*Bioinformatics 1*, 15:180–186, 1999.

[22] G. Landau and J. Schmidt.An algorithm for approximate tandem repeats.In Z. G. A. Apostolico, M. Crochemore and U. Manber, editors, *Combinatorial Pattern Matching*, volume 684 of *Lecture Notes in Computer Science*, pages 120–133. Springer-Verlag, 1993.

[23] L. Marsan and M.-F. Sagot.Algorithms for extracting structured motifs using a suffix tree with an application to promoter and regulatory site consensus identification.*J. Computational Biology*, 7:345–362, 2000.

[24] L. Marsan and M.-F. Sagot.Extracting structured motifs using a suffix tree – algorithms and application to promoter consensus identification.In S. Istrail, P. Pevzner, and M. Waterman, editors, *RECOMB'00. Proceedings of Fourth Annual International Conference on Computational Molecular Biology.* ACM Press, 2000.

[25] E. M. McCreight.A space-economical suffix tree construction algorithm.*J. ACM*, 23:262–272, 1976.

[26] A. Milosavljevic and J. Jurka.Discovering simple DNA sequences by the algorithmic significance method.*Comput. Appl. Biosci.*, 9:407–411, 1993.

[27] L. Parida, I. Rigoutsos, A. Floratos, D. Platt, and Y. Gao.Pattern discovery on character sets and real-valued data: linear bound on irredundant motifs and polynomial time algorithms.In *Proc. of the eleventh ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 297–308. ACM Press, 2000.

[28] J. Pothier.1993.Personal communication.

[29] C. Queen, M. N. Wegman, and L. J. Korn.Improvements to a program for DNA analysis: a procedure to find homologies among many sequences.*Nucleic Acids Res.*, 10:449–456, 1982.

[30] G. N. Ramachandran, C. Ramakrishnan, and V. Sasisekharan.Stereochemistry of polypeptide chain configurations.*J. Mol. Biol.*, 87:95–99, 1963.

[31] J. Risler, M. Delorme, H. Delacroix, and A. Hénaut.Amino acid substitutions in srtucturally related proteins: a pattern recognition approach.*J. Mol. Biol.*, 204:1019–1029, 1988.

[32] E. Rivals and O. Delgrange.A first step toward chromosome analysis by compression algorithms.In N. G. Bourbakis, editor, *First International IEEE Symposium on Intelligence in Neural and Biological Systems*, pages 233–239. IEEE Computer Society Press, 1995.

[33] M.-F. Sagot, V. Escalier, A. Viari, and H. Soldano.Searching for repeated words in a text allowing for mismatches and gaps.In R. Baeza-Yates and U. Manber, editors, *Second South American Workshop on String Processing*, pages 87–100, Viñas del Mar, Chili, 1995. University of Chili.

[34] M.-F. Sagot and E. W. Myers.Identifying satellites and periodic repetitions in biological sequences.*J. of Computational Biology*, 10:10–20, 1998.

[35] M.-F. Sagot and E. W. Myers.Identifying satellites in nucleic acid sequences.In S. Istrail, P. Pevzner, and M. Waterman, editors, *RECOMB'98. Proceedings of Second Annual International Conference on Computational Molecular Biology*, pages 234–242. ACM Press, 1998.

[36] M.-F. Sagot and A. Viari.A double combinatorial approach to discovering patterns in biological sequences.In D. Hirschberg and G. Myers, editors, *Combinatorial Pattern Matching*, volume 1075 of *Lecture Notes in Computer Science*, pages 186–208. Springer-Verlag, 1996.

[37] M. F. Sagot, A. Viari, J. Pothier, and H. Soldano.Finding flexible patterns in a text - an application to 3D molecular matching.*Comput. Appl. Biosci.*, 11:59–70, 1995.

[38] M.-F. Sagot, A. Viari, and H. Soldano.A distance-based block searching algorithm.In C. Rawlings, D. Clark, R. Altman, L. Hunter, T. Lengauer, and S. Wodak, editors, *Third International Symposium on Intelligent Systems for Molecular Biology*, pages 322–331, Cambridge, England, 1995. AAAI Press.

[39] M.-F. Sagot, A. Viari, and H. Soldano.Multiple comparison: a peptide matching approach.*Theoret. Comput. Sci.*, 180:115–137, 1997.presented at *Combinatorial Pattern Matching 1995.*

[40] H. Soldano, A. Viari, and M. Champesme.Searching for flexible repeated patterns using a non transitive similarity relation.*Pattern Recognition Letters*, 16:233–246, 1995.

[41] R. Staden.Methods for discovering novel motifs in nucleic acid sequences.*Comput. Appl. Biosci.*, 5:293–298, 1989.

[42] E. Ukkonen.Constructing suffix trees on-line in linear time.In *IFIP'92*, pages 484–492, 1992.

[43] J. van Helden, A. F. Rios, and J. Collado-Vides.Discovering regulatory elements in non-coding sequences by analysis of spaced dyads.*Nucleic Acids Res.*, 28:1808–1818, 2000.

[44] A. Vanet, L. Marsan, A. Labigne, and M.-F. Sagot.Inferring regulatory elements from a whole genome. An analysis of the σ^{80} family of promoter signals.*J. Mol. Biol.*, 297:335–353, 2000.

[45] A. Vanet, L. Marsan, and M.-F. Sagot.Promoter sequences and algorithmical methods for identifying them.*Research in Microbiology*, 150:779–799, 1999.

[46] R. Verin and M. Crochemore.Direct construction of compact directed acyclic word graphs.In A. Apostolico and J. Hein, editors, *Combinatorial Pattern Matching*, volume 1264 of *Lecture Notes in Computer Science*, pages 116–129. Springer-Verlag, 1997.

[47] M. S. Waterman.General methods of sequence comparison.*Bull. Math. Biol.*, 46:473–500, 1984.

[48] M. S. Waterman.Multiple sequence alignments by consensus.*Nucleic Acids Res.*, 14:9095–9102, 1986.

[49] M. S. Waterman.Consensus patterns in sequences.In M. S. Waterman, editor, *Mathematical Methods for DNA Sequences*, pages 93–116. CRC Press, 1989.

Szemerédi's Regularity Lemma and Quasi-randomness

Y. Kohayakawa[1]
V. Rödl[2]

9.1 Introduction

A beautiful result of Szemerédi on the asymptotic structure of graphs is his regularity lemma. Roughly speaking, this result tells us that *any* large graph may be written as a union of induced, random looking bipartite graphs. There are many applications of this result—the reader is urged to consult the excellent survey of Komlós and Simonovits [48] for a thorough discussion on this fundamental result.

The original regularity lemma is best suited for attacking problems involving 'dense' graphs, that is, n-vertex graphs with $\geq cn^2$ edges for some constant $c > 0$. In the case of 'sparse graphs', that is, n-vertex graphs with $o(n^2)$ edges, one has to adapt the definitions to take into account the vanishing density of the graphs in question. It turns out that regularity lemmas for certain classes of such sparse graphs may be proved easily. More importantly, such results turned out to be quite important in dealing with certain extremal and Ramsey type problems involving subgraphs of random graphs. The interested reader is referred to [36].

One of our aims in this paper is to focus on a circle of ideas that concern 'local' characterizations of regularity, which we believe should be better known. One tool that will be required is the regularity lemma for sparse graphs. Since we would also like this paper to be useful as an introduction to the regularity lemma, we include some expository sections.

The contents of this paper fall naturally into four parts. We start by presenting the basic concepts and the statement of the regularity lemma

[1]Partially supported by MCT/CNPq through ProNEx Programme (Proc. CNPq 664107/1997–4), by CNPq (Proc. 300334/93–1 and 468516/2000–0), and by FAPESP (Proj. 96/04505–2)
[2]Partially supported by NSF Grant 0071261

in Section 9.2.1. In Sections 9.2.2 and 9.2.3, we state two variants of the regularity lemma for sparse graphs.

If the reader is not too familiar with the regularity lemma, we suggest skipping Sections 9.2.2 and 9.2.3 at first, and advancing directly to the second part of this paper, Section 9.3, where we discuss in detail an application of the regularity lemma in its original form. The result we prove in Section 9.3, which closely follows parts of [55], shows that if the edges of a graph are 'uniformly distributed', then the graph must have a rich subgraph structure. This result, Theorem 18, will be used to confirm a conjecture of Erdős and we shall also mention a classical result in Ramsey theory that may be deduced easily from this result. We believe that Theorem 18 also illustrates the importance of the notion of 'quasi-randomness', addressed later in Section 9.7. The proof of Theorem 18 also illustrates a typical application of the regularity lemma. We hope that the uninitiated readers who are interested in regularity will study this proof in detail.

In Section 9.4 we mention some other applications of the regularity lemma that have emerged more recently. Our choice of topics for Section 9.4 has to do in part with the ideas and techniques that appear in Section 9.3 and some natural questions that they suggest. One application we discuss has an algorithmic flavour (see Section 9.4.2). In the following section, Section 9.5, we prove the version of the regularity lemma for sparse graphs given in Section 9.2.2.

In the third part of this paper, Section 9.6, we discuss a key fact that states that a certain local property of bipartite graphs is, roughly speaking, equivalent to the property of being regular in the sense of Szemerédi. This fact was the key tool for the development of the algorithmic version of the regularity lemma.

In the final part of this paper, Section 9.7, we discuss a new quasi-random graph property, by which we mean, following Chung, Graham, and Wilson [15], a property that belongs to a certain rather large and disparate collection of equivalent graph properties, shared by almost all graphs. To prove that our property is a quasi-random property in the sense of [15], we shall make use of the sparse regularity lemma.

A few remarks are in order. To focus on the main point in Section 9.6, we carry out our discussion on the local condition for regularity restricting ourselves to the very basic case, namely, the case of n by n bipartite graphs with edge density 1/2. In fact, for the sake of convenience, instead of talking about bipartite graphs, we shall consider n by n matrices whose entries are are $+1$s and -1s (and whose density of $+1$s will turn out to be $\sim 1/2$). We shall see that if the rows of a $\{\pm 1\}$-matrix are pairwise orthogonal, then the matrix has small *discrepancy*, which may be thought of as an indication that our matrix is 'random looking'. The reader may find a fuller discussion of this in Frankl, Rödl, and Wilson [26].

The relevance of the ideas in Section 9.6 may be illustrated by the fact that several authors have made use of them, in some form, in different contexts; see [1, 2, 4, 5, 10, 15, 19, 62, 63] and the proof of the upper bound in Theorem 15.2 in [23], due to J. H. Lindsey. We believe that these ideas should be carried over to the sparse case in some way as well, since this may prove to be quite fruitful; the interested reader is referred to [38, 39] and to Alon, Capalbo, Kohayakawa, Rödl, Ruciński, and Szemerédi [3].

We hope that our discussion in Section 9.6 will naturally lead the reader to the results in the final part of the paper, namely, the results concerning our quasi-random graph property. Indeed, Sections 9.6.1 and 9.6.2, which capture the essence of our discussion in Section 9.6, are quite gentle and we hope that the reader will find them useful as a preparation for the technically more involved Section 9.7. Before we close the introduction, we mention that our quasi-random property allows one to check whether an n-vertex graph is quasi-random in time $O(n^2)$. The fastest algorithms so far had time complexity $O(M(n)) = O(n^{2.376})$, where $M(n)$ denotes the time needed to square a $\{0,1\}$-matrix over the integers [17]. Furthermore, in a forthcoming paper with Thoma [41], we shall present how this quasi-random property may be used to develop a deterministic $O(n^2)$ time algorithm for the regularity lemma, improving on the result of Alon, Duke, Lefmann, Rödl, and Yuster [4, 5]. The reader is referred to [37] for a discussion on the algorithmic aspects of regularity.

9.1.1 Remarks on notation and terminology

If $\delta > 0$, we write $A \sim_\delta B$ to mean that

$$\frac{1}{1+\delta}B \le A \le (1+\delta)B. \tag{9.1}$$

We shall use the following non-standard notation: we shall write $O_1(x)$ for any term y that satisfies $|y| \le x$. Clearly, if $A \sim_\delta B$, then $A = (1+O_1(\delta))B$.

Given an integer n, we write $[n]$ for the set $\{1, \dots, n\}$. If X is a set and k is an integer, we write $\binom{X}{k}$ for the set of all k-element subset of X. We write $X \triangle Y$ for the symmetric difference $(X \setminus Y) \cup (Y \setminus X)$ of the sets X and Y.

We usually write G^n for a graph on n vertices. We denote the complete graph on k vertices by K^k. We usually write $e(G)$ for the number of edges in the graph G. We denote the set of neighbours of a vertex x in a graph G by $\Gamma(x) = \Gamma_G(x)$. If G is a graph and $\{u, w\} \in E(G) \subset \binom{V(G)}{2}$ is an edge of G, we often write uw and wu for this edge $\{u, w\}$. Sometimes we write $B = (U, W; E)$ for a bipartite graph B with a fixed bipartition $V(B) = U \cup W$, where $E = E(B)$.

As customary, if $G = (V, E)$ and $H = (U, F)$ are graphs with $U \subset V$ and $F \subset E$, then we say that H is a *subgraph* of G, and we write $H \subset G$. Moreover, if $U = V$, then we say that H is a *spanning* subgraph of G.

If $W \subset V$, then the subgraph of G *induced* by W in G is the subgraph

$$\left(W, E \cap \binom{W}{2}\right), \tag{9.2}$$

usually denoted by $G[W]$. A subgraph H of G is an *induced subgraph* if $H = G[V(H)]$, that is, every edge of G that has both its endpoints in the vertex set $V(H)$ of H is necessarily an edge of H as well.

Acknowledgement

The authors are very grateful to the editors of this volume for their extreme patience.

9.2 The regularity lemma

Our aim in this section is to present the original regularity lemma of Szemerédi and two closely related versions of the regularity lemma for sparse graphs.

9.2.1 Preliminary definitions and the regularity lemma

Let a graph $G = G^n$ of order $|V(G)| = n$ be fixed. For U, $W \subset V = V(G)$, we write $E(U, W) = E_G(U, W)$ for the set of edges of G that have one endvertex in U and the other in W. We set $e(U, W) = e_G(U, W) = |E(U, W)|$. The rather natural concept of *density* $d(U, W) = d_G(U, W)$ of a pair (U, W) in G is defined as follows: for any two disjoint non-empty sets U, $W \subset V$, we let

$$d_G(U, W) = \frac{e_G(U, W)}{|U||W|}. \tag{9.3}$$

Szemerédi's regularity lemma asserts the existence of partitions of graphs into a bounded number of remarkably 'uniform' pieces, known as *ε-regular pairs.*

Definition 1 (ε-regular pair) *Let $0 < \varepsilon \le 1$ be a real number. Suppose G is a graph and U and $W \subset V = V(G)$ are two disjoint, non-empty sets of vertices of G. We say that the pair (U, W) is (ε, G)-regular, or simply ε-regular, if we have*

$$|d_G(U', W') - d_G(U, W)| \le \varepsilon \tag{9.4}$$

for all $U' \subset U$ and $W' \subset W$ with

$$|U'| \ge \varepsilon|U| \quad \textit{and} \quad |W'| \ge \varepsilon|W|. \tag{9.5}$$

If a pair (U, W) *fails* to be ε-regular, then a pair (U', W') that certifies this fact is called a *witness* for the ε-irregularity of (U, W). Thus, if (U', W') is such a witness, then (9.5) holds but (9.4) fails.

In the regularity lemma, the vertex set of the graphs will be partitioned into a bounded number of blocks, basically all of the same size.

Definition 2 ((ε, k)-equitable partition) *Given a graph G, a real number $0 < \varepsilon \leq 1$, and an integer $k \geq 1$, we say that a partition $Q = (C_i)_0^k$ of $V = V(G)$ is (ε, k)-equitable if we have*

(*i*) $|C_0| \leq \varepsilon n$,

(*ii*) $|C_1| = \ldots = |C_k|$.

The class C_0 is referred to as the exceptional class of Q.

When the value of ε is not relevant, we refer to an (ε, k)-equitable partition as a *k-equitable* partition. Similarly, Q is an *equitable* partition of V if it is a k-equitable partition for some k. We may now introduce the key notion of ε-regular *partitions* for the graph G.

Definition 3 (ε-regular partition) *Given a graph G, we say that an (ε, k)-equitable partition $Q = (C_i)_0^k$ of $V = V(G)$ is (ε, G)-regular, or simply ε-regular, if at most $\varepsilon\binom{k}{2}$ pairs (C_i, C_j) with $1 \leq i < j \leq k$ are not ε-regular.*

We may now state the celebrated lemma of Szemerédi [60].

Theorem 4 (The regularity lemma) *For any given $\varepsilon > 0$ and $k_0 \geq 1$, there are constants $K_0 = K_0(\varepsilon, k_0) \geq k_0$ and $N_0 = N_0(\varepsilon, k_0)$ such that any graph $G = G^n$ with $n \geq N_0$ vertices admits an (ε, G)-regular, (ε, k)-equitable partition of its vertex set with $k_0 \leq k \leq K_0$.*

We shall not prove Theorem 4 here. However, a proof of a generalization of this result will be presented in detail later (see Section 9.5).

Some remarks on Theorem 4

Before we proceed, we make a few quite simple remarks on the concept of regularity and on the formulation of Theorem 4. The remarks below are primarily intended for the readers with little familiarity with the regularity lemma.

Remark 5 *Let $B = (U, W; E)$ be a bipartite graph with vertex classes U and W and edge set E. Suppose $|U| = |W| = m$ and, say, $|E| = \lfloor m^2/2 \rfloor$. Is such a graph typically ε-regular? I.e., is the pair (U, W) typically ε-regular? It turns out that this is indeed the case.*

Fact 6 *Let $\mathcal{B}(U, W; m, M)$ be the collection of all bipartite graphs $B = (U, W; E)$ on a fixed pair of sets U and W with $|U| = |W| = m$ and $|E| = M$. For $0 < \varepsilon \leq 1$, let $\mathcal{R}(U, W; m, M; \varepsilon) \subset \mathcal{B}(U, W; m, M)$ be the set of all ε-regular bipartite graphs in $\mathcal{B}(U, W; m, M)$. If $0 < \varepsilon \leq 1$ is a fixed constant and $M(m)$ is such that, say,*

$$\lim_{m\to\infty} M(m)/m^2 = p, \tag{9.6}$$

where $0 < p < 1$, then

$$\lim_{m\to\infty} \frac{|\mathcal{R}(U, W; m, M(m); \varepsilon)|}{|\mathcal{B}(U, W; m, M(m))|} = 1. \tag{9.7}$$

The result above tells us that 'almost all' (dense) bipartite graphs are ε-regular. Fact 6 follows easily from standard large deviation inequalities. The reader is referred to, say, Chapter 7 of [12, 14] (the well-known monographs [13, 35] will also certainly do).

Remark 7 *Bipartite graphs that are very sparse are necessarily ε-regular. We may make this observation precise as follows. Suppose $B = (U, W; E) \in \mathcal{B}(U, W; m, M)$, where $d(U, W) = M/m^2 \leq \varepsilon^3$. Then B is automatically ε-regular. Indeed, a witness (U', W') to the ε-irregularity of (U, W) must be such that*

$$d(U', W') > d(U, W) + \varepsilon \geq \varepsilon. \tag{9.8}$$

Therefore $e(U, W) \geq e(U', W') \geq d(U', W')|U'||W'| > \varepsilon|U'||W'| \geq \varepsilon^3 m^2$. However, by assumption, $e(U, W) = M \leq \varepsilon^3 m^2$. This contradiction shows that such a witness cannot exist. Therefore B is indeed ε-regular.

It should be also clear that bipartite graphs that are very dense are also automatically ε-regular. The reader is invited to work out the details.

Remark 8 *Suppose we have a graph $G = G^n$. Trivially, any k-equitable partition of $V(G)$ with $k = 1$ is ε-regular. However, in an ε-regular partition $(C_i)_0^k$ for G, we do not have any information about the edges incident to the exceptional class C_0, nor do we have any information about the edges contained within the C_i $(1 \leq i \leq k)$. Therefore the 1-equitable partitions of G are of no interest. The lower bound k_0 in the statement of Theorem 4 may be used to rule out partitions into a small number of blocks.*

In fact, the number of edges within the C_i $(1 \leq i \leq k)$ in an (ε, k)-equitable partition is at most $k^{-1}\binom{n}{2} \leq k_0^{-1}\binom{n}{2}$, and the number of edges incident to C_0 is at most εn^2, since $|C_0| \leq \varepsilon n$. Therefore, one usually chooses k_0 and ε so that

$$\frac{1}{k_0}\binom{n}{2} + \varepsilon n^2 \tag{9.9}$$

is a negligible number of edges for the particular application in question.

Remark 9 *Let $G = G^n$ be a given graph. Sometimes it is a little more convenient to consider regular partitions for G in which no exceptional class is allowed. One may instead require that the partition $(C_i)_1^k$ of $V = V(G)$ should be such that*

$$\left\lfloor \frac{n}{k} \right\rfloor \le |C_1| \le \ldots \le |C_k| \le \left\lceil \frac{n}{k} \right\rceil, \tag{9.10}$$

and such that $\ge (1-\varepsilon)\binom{k}{2}$ of the pairs (C_i, C_j) with $1 \le i < j \le k$ are ε-regular. We leave it as an exercise to deduce this version of the regularity lemma from Theorem 4.

Remark 10 *Suppose we allow regular partitions as in Remark 9 above. Then, as a side effect, we may omit the condition that the graph $G = G^n$ should satisfy $n \ge N_0(\varepsilon, k_0)$. Indeed, it suffices to use the fact that the partition of the vertex set of a graph into singletons is ε-regular. Indeed, let $K_0 = K_0(\varepsilon, k_0)$ be the upper bound for the number of classes in the ε-regular partitions with at least k_0 parts, in the sense of Remark 9, whose existence may be ensured, and suppose $N_0 = N_0(\varepsilon, k_0)$ is such that any graph with $n \ge N_0$ vertices is guaranteed to admit such a partition. Now let $K_0' = \max\{K_0, N_0\}$, and observe that, then, any graph admits an ε-regular partition into k parts, where $k_0 \le k \le K_0'$. Indeed, if the given graph G has fewer than N_0 vertices, it suffices to consider the partition of $V(G)$ into singletons.*

For the sake of completeness, we explicitly state the conclusion of Remarks 9 and 10 as a theorem.

Theorem 11 *For any given $\varepsilon > 0$ and $k_0 \ge 1$, there is a constant $K_0 = K_0(\varepsilon, k_0) \ge k_0$ such that any graph G admits a partition $(C_i)_1^k$ of its vertex set such that*

(*i*) $k_0 \le k \le K_0$,

(*ii*) $\lfloor n/k \rfloor \le |C_1| \le \ldots \le |C_k| \le \lceil n/k \rceil$, *and*

(*iii*) *at least $(1-\varepsilon)\binom{k}{2}$ of the pairs (C_i, C_j) with $1 \le i < j \le k$ are ε-regular.*

Irregular pairs and the number of blocks in regular partitions

The notion of an ε-regular partition given in Definition 3 gives us a little breathing room in that it allows up to $\varepsilon\binom{k}{2}$ irregular pairs (C_i, C_j) in a k-equitable partition $\bigcup_{0 \le i \le k} C_i$. Whether this is required is a rather natural question (already raised by Szemerédi [60]): is there a strengthening of the regularity lemma that guarantees the existence of an (ε, k)-equitable partition with all the $\binom{k}{2}$ pairs ε-regular for any large enough graph?

As observed by several researchers, Lovász, Seymour, Trotter, and the authors of [5] among others (see [5, p. 82]), the irregular pairs *are required.*

A simple example that shows this is as follows: let $B = (U, W; E)$ be the bipartite graph with $U = W = [n]$, and $ij \in E$ if and only if $i \leq j$. The reader is invited to prove that, for small enough $\varepsilon > 0$, any (ε, k)-equitable, ε-regular partition of this graph requires at least ck ε-irregular pairs, where $c = c(\varepsilon) > 0$ is some constant that depends only on ε.

Let us now turn to the value of the constants $K_0 = K_0(\varepsilon, k_0)$ and $N_0 = N_0(\varepsilon, k_0)$ in the statement of the regularity lemma, Theorem 4. As we discussed in Remark 10, the requirement that we should only deal with graphs $G = G^n$ with $n \geq N_0$ is not important. However, $K_0 = K_0(\varepsilon, k_0)$ is much more interesting.

The original proof of Theorem 4 gave for K_0 a tower of 2s of height proportional to ε^{-5}, which is quite a large constant for any reasonable ε. (How such a number comes about may be seen very clearly in the proof of Theorem 13, given in Section 9.5.) As proved by Gowers [34], there are graphs for which such a huge number of classes are required in any ε-regular partition. We only give a weak form of the main result in [34] (see Theorem 15 in [34]).

Theorem 12 *There exist absolute constants $\varepsilon_0 > 0$ and $c_0 > 0$ for which the following holds. For any $0 < \varepsilon \leq \varepsilon_0$, there is a graph G for which the number of classes in any ε-regular partition of its vertex set must be at least as large as a tower of 2s of height at least $c_0\varepsilon^{-1/16}$.*

Roughly speaking, the strongest result in [34] states that one may weaken the requirements on the ε-regular partition in certain natural ways and still have the same lower bound as in Theorem 12. The interested reader should study the ingenious probabilistic constructions in [34].

Before we proceed, let us mention again that the readers who are not too familiar with the regularity lemma may at first prefer to skip the next two sections, namely, Sections 9.2.2 and 9.2.3, and proceed directly to Section 9.3, where a typical application of Theorem 4 is discussed in detail.

9.2.2 A regularity lemma for sparse graphs

We shall now state a version of the regularity lemma for sparse graphs. We in fact consider a slightly more general situation, including the case of ℓ-partite graphs G, where ℓ is some fixed integer.

Let a partition $P_0 = (V_i)_1^\ell$ $(\ell \geq 1)$ of $V = V(G)$ be fixed. For convenience, let us write $(U, W) \prec P_0$ if $U \cap W = \emptyset$ and either $\ell = 1$ or else $\ell \geq 2$ and for some $i \neq j$ $(1 \leq i, j \leq \ell)$ we have $U \subset V_i$, $W \subset V_j$.

Suppose $0 < \eta \leq 1$. We say that G is (P_0, η)-*uniform* if, for some $0 < p \leq 1$, we have that for all U, $W \subset V$ with $(U, W) \prec P_0$ and $|U|$, $|W| \geq \eta n$, we have

$$\big|e_G(U, W) - p|U||W|\big| \leq \eta p|U||W|. \qquad (9.11)$$

As mentioned above, the partition P_0 is introduced to handle the case of ℓ-partite graphs ($\ell \geq 2$). If $\ell = 1$, that is, if the partition P_0 is trivial, then we are thinking of the case of ordinary graphs. In this case, we shorten the term (P_0, η)-uniform to *η-uniform*.

The prime example of an η-uniform graph is of course a *random graph* $G_p = G_{n,p}$. For any $\eta > 0$ a random graph G_p with $p = p(n) = C/n$ is almost surely η-uniform provided $C \geq C_0 = C_0(\eta)$, where $C_0(\eta)$ depends only on η. Let $0 < p = p(n) \leq 1$ be given. The standard binomial random graph $G_p = G_{n,p}$ has as vertex set a fixed set $V(G_p)$ of cardinality n and two such vertices are adjacent in G_p with probability p, with all such adjacencies independent. For concepts and results concerning random graphs, see, e.g., Bollobás [13] or Janson, Łuczak, and Ruciński [35]. (A lighter introduction may be Chapter 7 of Bollobás [12, 14].)

We still need to introduce a few further definitions. Let a graph $G = G^n$ be fixed as before. Let $H \subset G$ be a spanning subgraph of G. For U, $W \subset V$, let

$$d_{H,G}(U,W) = \begin{cases} e_H(U,W)/e_G(U,W) & \text{if } e_G(U,W) > 0 \\ 0 & \text{if } e_G(U,W) = 0. \end{cases}$$

Suppose $\varepsilon > 0$, U, $W \subset V$, and $U \cap W = \emptyset$. We say that the pair (U, W) is *(ε, H, G)-regular*, or simply *ε-regular*, if for all $U' \subset U$, $W' \subset W$ with $|U'| \geq \varepsilon|U|$ and $|W'| \geq \varepsilon|W|$, we have

$$|d_{H,G}(U', W') - d_{H,G}(U, W)| \leq \varepsilon.$$

If P and Q are two equitable partitions of V (see Definition 2 in Section 9.2.1), we say that Q *refines* P if every non-exceptional class of Q is contained in some non-exceptional class of P. If P' is an arbitrary partition of V, then Q *refines* P' if every non-exceptional class of Q is contained in some block of P'. Finally, we say that an (ε, k)-equitable partition $Q = (C_i)_0^k$ of V is *(ε, H, G)-regular*, or simply *ε-regular*, if at most $\varepsilon\binom{k}{2}$ pairs (C_i, C_j) with $1 \leq i < j \leq k$ are not ε-regular. We may now state an extension of Szemerédi's lemma to subgraphs of (P_0, η)-uniform graphs.

Theorem 13 *Let $\varepsilon > 0$ and k_0, $\ell \geq 1$ be fixed. Then there are constants $\eta = \eta(\varepsilon, k_0, \ell) > 0$, $K_0 = K_0(\varepsilon, k_0, \ell) \geq k_0$, and $N_0 = N_0(\varepsilon, k_0, \ell)$ satisfying the following. For any (P_0, η)-uniform graph $G = G^n$ with $n \geq N_0$, where $P_0 = (V_i)_1^\ell$ is a partition of $V = V(G)$, if $H \subset G$ is a spanning subgraph of G, then there exists an (ε, H, G)-regular (ε, k)-equitable partition of V refining P_0 with $k_0 \leq k \leq K_0$.*

Remark 14 *To recover the original regularity lemma of Szemerédi from Theorem 13, simply take $G = K^n$, the complete graph on n vertices.*

9.2.3 A second regularity lemma for sparse graphs

In some situations, the sparse graph H to which one would like to apply the regularity lemma is not a subgraph of some fixed η-uniform graph G. A simple variant of Theorem 13 may be useful in this case. For simplicity, we shall not state this variant for 'P_0-partite' graphs as we did in Section 9.2.2.

Let a graph $H = H^n$ of order $|V(H)| = n$ be fixed. Suppose $0 < \eta \leq 1$, $D \geq 1$, and $0 < p \leq 1$ are given. We say that H is an *(η, D)-upper-uniform graph with respect to density* p if, for all U, $W \subset V$ with $U \cap W = \emptyset$ and $|U|$, $|W| \geq \eta n$, we have $e_H(U, W) \leq Dp|U||W|$. In what follows, for any two disjoint non-empty sets U, $W \subset V$, let the *normalized p-density* $d_{H,p}(U, W)$ of (U, W) be

$$d_{H,p}(U, W) = \frac{e_H(U, W)}{p|U||W|}. \tag{9.12}$$

Now suppose $\varepsilon > 0$, U, $W \subset V$, and $U \cap W = \emptyset$. We say that the pair (U, W) is *(ε, H, p)-regular*, or simply *(ε, p)-regular*, if for all $U' \subset U$, $W' \subset W$ with $|U'| \geq \varepsilon|U|$ and $|W'| \geq \varepsilon|W|$ we have

$$|d_{H,p}(U', W') - d_{H,p}(U, W)| \leq \varepsilon.$$

We say that an (ε, k)-equitable partition $P = (C_i)_0^k$ of V is *(ε, H, p)-regular*, or simply *(ε, p)-regular*, if at most $\varepsilon\binom{k}{2}$ pairs (C_i, C_j) with $1 \leq i < j \leq k$ are not (ε, p)-regular. We may now state a version of Szemerédi's regularity lemma for (η, D)-upper-uniform graphs.

Theorem 15 *For any given $\varepsilon > 0$, $k_0 \geq 1$, and $D \geq 1$, there are constants $\eta = \eta(\varepsilon, k_0, D) > 0$, $K_0 = K_0(\varepsilon, k_0, D) \geq k_0$, and $N_0 = N_0(\varepsilon, k_0, D)$ such that any graph $H = H^n$ with $n \geq N_0$ vertices that is (η, D)-upper-uniform with respect to density $0 < p \leq 1$ admits an (ε, H, p)-regular (ε, k)-equitable partition of its vertex set with $k_0 \leq k \leq K_0$.*

9.3 An application of the regularity lemma

Here we present an application of the regularity lemma. We believe that this is a fairly illustrative example and we also hope that it will introduce the notion of pseudorandomness in a natural way. We follow certain parts of [55] closely.

9.3.1 A simple fact about almost all graphs

We start with two definitions. We shall say that a graph G is *k-universal* if G contains all graphs with k vertices as induced subgraphs. As we shall see below, large graphs are typically k-universal for any small k. Our second definition captures another property of typical graphs, namely, the property that their edges are 'uniformly distributed'.

Definition 16 (Property $\mathcal{R}(\gamma,\delta,\sigma)$) *We say that a graph $G = G^n$ of order n has property $\mathcal{R}(\gamma,\delta,\sigma)$ if, for all $S \subset V = V(G)$ with $|S| \geq \gamma n$, the number of edges $e(S) = e(G[S])$ induced by S in G satisfies*

$$e(S) = (\sigma + O_1(\delta))\binom{|S|}{2}. \tag{9.13}$$

Let us write $\mathcal{G}(n,M)$ for the set of all graphs on the vertex set $[n] = \{1,\dots,n\}$ with M edges. Clearly, we have

$$|\mathcal{G}(n,M)| = \binom{\binom{n}{2}}{M} \tag{9.14}$$

for all integers $n \geq 0$ and $0 \leq M \leq \binom{n}{2}$. Let $\mathcal{U}(n,M;k)$ be the subset of $\mathcal{G}(n,M)$ of all the k-universal graphs, and let $\mathcal{R}(n,M;\gamma,\delta,\sigma)$ be the subset of $\mathcal{G}(n,M)$ of all the graphs $G \in \mathcal{G}(n,M)$ satisfying property $\mathcal{R}(\gamma,\delta,\sigma)$.

The following fact is easy to prove.

Fact 17 *Let $k \geq 1$ be an integer and let $0 < \gamma \leq 1$, $0 < \delta \leq 1$, and $0 < \sigma < 1$ be real numbers. Put $M = M(n) = \left\lfloor \sigma\binom{n}{2} \right\rfloor$. Then we have*

$$\lim_{n\to\infty} \frac{|\mathcal{U}(n,M;k)|}{|\mathcal{G}(n,M)|} = 1 \tag{9.15}$$

and

$$\lim_{n\to\infty} \frac{|\mathcal{R}(n,M;\gamma,\delta,\sigma)|}{|\mathcal{G}(n,M)|} = 1. \tag{9.16}$$

In the usual language of random graphs, one says that *almost all* $G \in \mathcal{G}(n,M)$ are k-universal to mean that (9.15) holds. Similarly, one says that almost all $G \in \mathcal{G}(n,M)$ satisfy $\mathcal{R}(\gamma,\delta,\sigma)$ because of (9.16). If γ and δ are small, the latter assertion may be interpreted to mean that the edges of a typical graph $G \in \mathcal{G}(n,M)$ are uniformly distributed.

The most direct way to verify Fact 17 is by proving (9.15) and (9.16) independently. However, it turns out that, for *any deterministic graph* $G = G^n$, having property $\mathcal{R}(\gamma,\delta,\sigma)$ for any fixed $0 < \sigma < 1$ implies the k-universality of G. (Of course, the constants γ and δ have to be suitably small with respect to k, and n has to be suitably large with respect to k.) Thus, roughly speaking, having uniformly distributed edges is a *stronger* property than being universal. (Quite surprisingly, if one strengthens the notion of k-universality to include information on the number of copies of all k-vertex graphs for fixed $k \geq 4$, these properties become *equivalent* in a certain precise sense; see Section 9.3.2 for a short discussion on this point.)

We shall prove that uniform distribution of edges implies universality by making use of the regularity lemma. We shall in fact prove a stronger statement, and we shall see that this statement, coupled with an auxiliary result, confirms a conjecture of Erdős.

9.3.2 The statement of the results

Let us state the first result we discuss in this section.

Theorem 18 *For all integers $k \geq 1$ and real numbers $0 < \sigma < 1$ and $0 < \delta < 1$ with $\delta < \sigma < 1 - \delta$, there exist $\gamma > 0$ and N_0 for which the following holds. If $G = G^n$ is a graph of order $n \geq N_0$ that satisfies property $\mathcal{R}(\gamma, \delta, \sigma)$, then G is k-universal.*

We shall prove Theorem 18 in Section 9.3.3. It may be worth mentioning that the constant δ, which controls the 'error' in (9.13), is quantified *universally* in Theorem 18 (under the obviously necessary condition that we should have $\delta < \sigma < 1 - \delta$). Thus, the result above tells us that, whatever the magnitude of the error, we may ensure k-universality by requiring control over small enough sets. Somewhat surprisingly, one may also prove a result in which it is the quantity γ that is quantified universally, that is, we are told that we have control over sets of some fixed cardinality, say $\lfloor n/2 \rfloor$, and we would like to guarantee k-universality by requiring a tight enough control over such sets. We make this precise in the following result, proved in [55].

Theorem 19 *For all integers $k \geq 1$ and real numbers $0 < \sigma < 1$ and $0 < \gamma < 1$, there exist $\delta > 0$ and N_1 for which the following holds. If $G = G^n$ is a graph of order $n \geq N_1$ that satisfies property $\mathcal{R}(\gamma, \delta, \sigma)$, then G is k-universal.*

We shall not prove the above result here. We only remark that the proof of Theorem 19 is based on the same tools that are used to prove Theorem 18, but it is a little more delicate. Theorem 19 is closely related to the following result, which was conjectured by Erdős (see [21] or [11, Chapter VI, p. 363]).

Theorem 20 *For every integer $k \geq 1$ and real number $0 < \sigma < 1$, there is an $\varepsilon > 0$ for which the following holds. Suppose a graph $G = G^n$ has $M = \lfloor \sigma \binom{n}{2} \rfloor$ edges, and for all $W \subset V = V(G)$ with $|W| = \lfloor n/2 \rfloor$ we have*

$$e(G[W]) \geq \sigma \binom{\lfloor n/2 \rfloor}{2} (1 - \varepsilon). \tag{9.17}$$

Then, if $n \geq n_0(k, \sigma)$, the graph G contains a K^k.

We shall deduce Theorem 20 from Theorem 18 below. Nikiforov [53] recently proved Theorem 20 by making use of different techniques.

Proof of Theorem 20

Theorem 20 follows from Theorem 18 and the auxiliary claim below.

Claim 21 *For all real numbers $0 < \gamma < 1$, $0 < \delta < 1$, and $0 < \sigma < 1$, there is an $\varepsilon > 0$ for which the following holds. Suppose a graph $G = G^n$ has $M = \lfloor \sigma \binom{n}{2} \rfloor$ edges, and for all $W \subset V = V(G)$ with $|W| = \lfloor n/2 \rfloor$ inequality (9.17) holds. Then, if $n \geq n_1(\gamma, \delta, \sigma)$, the graph G is such that for all $U \subset V = V(G)$ with $|U| \geq \gamma n$ we have*

$$e(G[U]) \geq (\sigma - \delta)\binom{|U|}{2}. \tag{9.18}$$

Observe that the conclusion about G in Claim 21 above is very close to property $\mathcal{R}(\gamma, \delta, \sigma)$. Clearly, the difference is that we do not have the upper bound in (9.13) in Definition 16, which is natural, given the one-sided hypothesis about G in Claim 21. Let us now prove Theorem 20 assuming Theorem 18 and Claim 21.

Proof. (Proof of Theorem 20) Let k and σ as in the statement of Theorem 20 be given. Put

$$\delta = \frac{1}{2}\sigma, \tag{9.19}$$

and let

$$\sigma' = \frac{1}{2}\left(\left(1 - \frac{1}{k}\right) + (\sigma - \delta)\right) \quad \text{and} \quad \delta' = \frac{1}{2}\left(\left(1 - \frac{1}{k}\right) - (\sigma - \delta)\right). \tag{9.20}$$

Clearly, we have

$$0 < \sigma' - \delta' = \sigma - \delta < \sigma' + \delta' = 1 - \frac{1}{k} < 1, \tag{9.21}$$

and, in particular, $\delta' < \sigma' < 1 - \delta'$. Hence, we may invoke Theorem 18 with k, σ', and δ'. Theorem 18 then gives us

$$\gamma = \gamma(k, \sigma', \delta') \quad \text{and} \quad N_0(k, \sigma', \delta'). \tag{9.22}$$

Let us now feed γ, δ, and σ into Claim 21. We obtain

$$\varepsilon = \varepsilon(\gamma, \delta, \sigma) \quad \text{and} \quad n_1(\gamma, \delta, \sigma). \tag{9.23}$$

Finally, let $n_0(k)$ be such that any graph with $n \geq n_0(k)$ vertices and $> (1 - 1/k)\binom{n}{2}$ edges must contain a K^k. Put

$$n_0 = n_0(k, \sigma) = \max\left\{N_0(k, \sigma', \delta'), n_1(\gamma, \delta, \sigma), \frac{1}{\gamma}n_0(k)\right\}. \tag{9.24}$$

We claim that ε given in (9.23) and n_0 given in (9.24) will do in Theorem 20.

To verify this claim, suppose a graph $G = G^n$ with $n \geq n_0$ vertices has $M = \lfloor \sigma \binom{n}{2} \rfloor$ edges, and for all $W \subset V = V(G)$ with $|W| = \lfloor n/2 \rfloor$

inequality (9.17) holds. Then, by the choice of ε and $n_0 \geq n_1(\gamma, \delta, \sigma)$ (see (9.23)), we may deduce from Claim 21 that

(‡) *for all $U \subset V = V(G)$ with $|U| \geq \gamma n$ inequality (9.18) holds.*

Now, since $n \geq n_0 \geq \gamma^{-1} n_0(k)$, we know that if $U \subset V = V(G)$ is such that $|U| \geq \gamma n$ and

$$e(G[U]) > \left(1 - \frac{1}{k}\right)\binom{|U|}{2}, \tag{9.25}$$

then $G[U] \supset K^k$. Therefore we may assume that

(‡‡) *inequality (9.25) fails for all $U \subset V = V(G)$ with $|U| \geq \gamma n$.*

Assertions (‡) and (‡‡) imply that property $\mathcal{R}(\gamma, \delta', \sigma')$ holds for G (see (9.21)). By the choice of γ and $n_0 \geq N_0(k, \sigma', \delta')$ (see (9.22)), we may now deduce from Theorem 18 that G is k-universal. This completes the proof of Theorem 20. □

We shall now turn to Claim 21, but before we proceed, we state the following basic fact. Given a set of vertices $W \subset V(G)$ with $|W| \geq 2$ in a graph G, the *edge density* $d(W)$ of W is defined to be $e(G[W])\binom{|W|}{2}^{-1}$.

Fact 22 *Let G be a graph and suppose we are given $W \subset V(G)$ with $|W| \geq 2$. Suppose also that $2 \leq u \leq |W|$ is fixed. Then*

$$d(W) = \operatorname*{Ave}_{U} d(U), \tag{9.26}$$

where the average is taken over all $U \subset W$ with $|U| = u$.

Proof. The one-line proof goes as follows:

$$\begin{aligned}\operatorname*{Ave}_{U} d(U) &= \binom{|W|}{u}^{-1} \sum_{U} d(U) = \binom{|W|}{u}^{-1} \sum_{U} e(G[U])\binom{|U|}{2}^{-1} \\ &= e(G[W])\binom{|W|}{u}^{-1}\binom{u}{2}^{-1}\binom{|W|-2}{u-2} = e(G[W])\binom{|W|}{2}^{-1},\end{aligned} \tag{9.27}$$

where, clearly, the average and the sums are over all $U \subset W$ with $|U| = u$. □

Let us now prove Claim 21.

Proof. Let $0 < \gamma < 1$, $0 < \delta < 1$, and $0 < \sigma < 1$ be fixed, and suppose that the graph $G = G^n$ is as in the statement of the Claim 21. We shall prove that if ε is small enough and n is large enough, then inequality (9.18) holds for all $U \subset V = V(G)$ with $|U| \geq \gamma n$.

Observe first that it suffices to consider sets $U \subset V$ with $|U| = \lceil \gamma n \rceil$, because of Fact 22. We may also suppose that $\lceil \gamma n \rceil < \lfloor n/2 \rfloor$ and, in fact, $0 < \gamma < 1/2$.

Let $U \subset V$ be such that $u = |U| = \lceil \gamma n \rceil$. Put $T = V \setminus U$. Let the number of edges between U and T be $\sigma_1 ut$, where $t = |T| = n - u$. Let also $\sigma_2 \binom{t}{2}$ be the number of edges induced by T in G. We have

$$e(G[U]) + \sigma_1 ut + \sigma_2 \binom{t}{2} = \left\lfloor \sigma \binom{n}{2} \right\rfloor. \tag{9.28}$$

Put $t' = \lfloor n/2 \rfloor - u > 0$. We now select a t'-element subset T' of T uniformly at random, and consider the edges that are induced by $U \cup T'$. Fix an edge xy of G, with $x \in U$ and $y \in T$. Then, xy will be induced by $U \cup T'$ if and only if $y \in T'$. However, this happens with probability $\binom{t-1}{t'-1}\binom{t}{t'}^{-1} = t'/t$. Given that there are $\sigma_1 ut$ such edges xy, the expected number of these edges that will be induced by $U \cup T'$ is

$$\sigma_1 ut \times \frac{t'}{t} = \sigma_1 ut'. \tag{9.29}$$

Now fix an edge xy of G with both x and y in T. Then, xy will be induced by $U \cup T'$ with probability

$$\binom{t-2}{t'-2}\binom{t}{t'}^{-1} = \frac{t'(t'-1)}{t(t-1)}. \tag{9.30}$$

Since there are $\sigma_2 \binom{t}{2}$ such edges xy, the expected number of these edges that will be induced by $U \cup T'$ is

$$\sigma_2 \binom{t}{2} \frac{t'(t'-1)}{t(t-1)} = \sigma_2 \binom{t'}{2}. \tag{9.31}$$

Therefore, by (9.29) and (9.31), the expected number of edges that are induced by $U \cup T'$ is

$$e(G[U]) + \sigma_1 ut' + \sigma_2 \binom{t'}{2}. \tag{9.32}$$

For the remainder of the proof, we fix a set T' such that this number of induced edges $e(G[U \cup T'])$ is at least as large as given in (9.32). Since $U \cup T'$ is a set with $\lfloor n/2 \rfloor$ vertices, by our hypothesis on G we have

$$e(G[U]) + \sigma_1 ut' + \sigma_2 \binom{t'}{2} \geq \sigma \binom{\lfloor n/2 \rfloor}{2}(1-\varepsilon). \tag{9.33}$$

Subtracting (9.33) from (9.28), we obtain

$$\sigma_1 u(t-t') + \sigma_2 \left(\binom{t}{2} - \binom{t'}{2}\right) \leq \sigma \left(\binom{n}{2} - (1-\varepsilon)\binom{\lfloor n/2 \rfloor}{2}\right). \tag{9.34}$$

Suppose now that U induces fewer than $(\sigma - \delta)\binom{u}{2}$ edges. Then (9.33) gives that

$$(\sigma - \delta)\binom{u}{2} + \sigma_1 ut' + \sigma_2 \binom{t'}{2} > \sigma \binom{\lfloor n/2 \rfloor}{2}(1-\varepsilon). \tag{9.35}$$

We deduce that

$$\sigma_1 u > \frac{1}{t'}\left(\sigma\binom{\lfloor n/2\rfloor}{2}(1-\varepsilon)-\sigma_2\binom{t'}{2}-(\sigma-\delta)\binom{u}{2}\right). \tag{9.36}$$

Plugging (9.36) into (9.34), we obtain

$$\begin{aligned}\left(\frac{t}{t'}-1\right)\left(\sigma\binom{\lfloor n/2\rfloor}{2}(1-\varepsilon)-\sigma_2\binom{t'}{2}-(\sigma-\delta)\binom{u}{2}\right)\\ +\sigma_2\left(\binom{t}{2}-\binom{t'}{2}\right)<\sigma\left(\binom{n}{2}-(1-\varepsilon)\binom{\lfloor n/2\rfloor}{2}\right).\end{aligned} \tag{9.37}$$

Observe that $t/t'-1\to 1/(1-2\gamma)$ as $n\to\infty$. Therefore, dividing (9.37) by n^2 and letting $n\to\infty$, we obtain

$$\begin{aligned}\frac{1}{1-2\gamma}\left(\frac{\sigma}{8}(1-\varepsilon)-\frac{1}{2}\sigma_2\left(\frac{1}{2}-\gamma\right)^2-\frac{1}{2}(\sigma-\delta)\gamma^2\right)\\ +\frac{1}{2}\sigma_2\left(\frac{3}{4}-\gamma\right)\le\frac{\sigma}{2}\left(\frac{1}{2}-\frac{1-\varepsilon}{8}\right),\end{aligned} \tag{9.38}$$

or, rearranging terms,

$$\frac{\sigma}{8}(1-\varepsilon)+\frac{1}{4}\sigma_2(1-2\gamma)(1-\gamma)-\frac{1}{2}(\sigma-\delta)\gamma^2\le\sigma\left(\frac{1}{2}-\frac{1-\varepsilon}{8}\right)(1-2\gamma). \tag{9.39}$$

We now observe that Fact 22 and our hypothesis on G implies that $\sigma_2\ge\sigma(1-\varepsilon)$. Therefore (9.39) implies that

$$\frac{\sigma}{8}(1-\varepsilon)+\frac{1}{4}\sigma(1-\varepsilon)(1-2\gamma)(1-\gamma)-\frac{1}{2}(\sigma-\delta)\gamma^2\le\sigma\left(\frac{1}{2}-\frac{1-\varepsilon}{8}\right)(1-2\gamma). \tag{9.40}$$

Letting $\varepsilon\to 0$ in (9.40), we obtain

$$\frac{\sigma}{8}+\frac{1}{4}\sigma(1-2\gamma)(1-\gamma)-\frac{1}{2}(\sigma-\delta)\gamma^2\le\frac{3}{8}\sigma(1-2\gamma). \tag{9.41}$$

However, inequality (9.41) reduces to

$$\frac{1}{2}\delta\gamma^2\le 0, \tag{9.42}$$

which does not hold. Therefore, there is an $\varepsilon_0=\varepsilon_0(\gamma,\delta,\sigma)>0$ such that (9.40) fails for all $0<\varepsilon\le\varepsilon_0$. Moreover, there is $n_0=n_0(\gamma,\delta,\sigma)\ge 1$ such that (9.37) fails for all $n\ge n_0$. However, this implies that if $0<\varepsilon\le\varepsilon_0$ and $n\ge n_0$, then U induces at least than $(\sigma-\delta)\binom{u}{2}$ edges. We have thus found $\varepsilon_0=\varepsilon_0(\gamma,\delta,\sigma)$ and $n_0=n_0(\gamma,\delta,\sigma)$ as required, and Claim 21 is proved. □

An application in Ramsey theory

Before we proceed to the proof of Theorem 18, we state a pleasant corollary to that result. Let G and $H_1, \ldots, H_r$ be graphs. We write

$$G \xrightarrow{\text{ind}} (H_1, \ldots, H_r) \tag{9.43}$$

to mean that, however we colour the edges of G with colours $c_1, \ldots, c_r$, there must be some i such that G contains an *induced* subgraph H' isomorphic to H_i and with all its edges coloured with colour c_i.

Theorem 23 *For any collection of graphs $H_1, \ldots, H_r$, there is a graph G for which (9.43) holds.*

Theorem 23 was independently proved by Deuber [18], Erdős, Hajnal, and Pósa [22], and Rödl [54]. We leave it as an exercise for the reader to deduce from Theorem 18 that, in fact, *almost all* graphs $G \in \mathcal{G}(n, M)$ satisfy (9.43) if $M = \lfloor \sigma \binom{n}{2} \rfloor$, where $0 < \sigma < 1$ is any fixed constant (see [52]).

Uniform edge distribution and subgraph frequency

The proof of Theorem 18 given below may be adapted to prove the following result: for any $\varepsilon > 0$ and $0 < \sigma < 1$, and any integer $k \geq 1$, there is a $\delta > 0$ such that if $G = G^n$ satisfies property $\mathcal{R}(\delta, \delta, \sigma)$, then, as long as $n \geq n_0(\varepsilon, \sigma, k)$,

(*) *for any graph $H = H^k$ on k vertices, the number of induced embeddings $f: V(H) \to V(G)$ of H in G is*

$$(1 + O_1(\varepsilon))(n)_k \sigma^{e(H)} (1 - \sigma)^{\binom{k}{2} - e(H)}. \tag{9.44}$$

As customary, above we write $(a)_b$ for $a(a-1) \ldots (a-b+1)$. It is straightforward that the expected number of embeddings f as above in the random graph $G \in \mathcal{G}(n, M)$ is given by (9.44), where $M = M(n) = \lfloor \sigma \binom{n}{2} \rfloor$, and in fact the number of such embeddings *is* this number for almost all $G \in \mathcal{G}(n, M)$. Thus, again, the deterministic property $\mathcal{R}(\delta, \delta, \sigma)$ captures a feature of random graphs. Surprisingly, this 'numerical' version of k-universality for $k = 4$, that is, property (*) for $k = 4$, implies property $\mathcal{R}(\delta, \delta, \sigma)$, as long as ε is small enough with respect to δ and σ.

The properties above, together with several others, are now known as *quasi-random* graph properties. The interested reader is referred to Thomason [62, 63], Frankl, Rödl, and Wilson [26], and Chung, Graham, and Wilson [15] (see also Alon and Spencer [8, Chapter 9]). The study of quasi-randomness is appealing in its own right, but one may perhaps argue that investigating quasi-randomness for graphs is especially important because of the intimate relation between quasi-randomness, ε-regularity, and the regularity lemma.

In Section 9.7, we shall introduce a new quasi-random property for graphs.

9.3.3 The proof of Theorem 18

The proof of Theorem 18 is based on the regularity lemma, Theorem 4, and on an *embedding lemma*, which asserts the existence of certain embeddings of graphs.

In this proof, γ, δ, σ, ε, β, and ε_k will always denote positive constants smaller than 1.

The embedding lemma

We start with a warm-up. Suppose we have a tripartite graph $G = G^{3\ell}$, with tripartition $V(G) = B_1 \cup B_2 \cup B_3$, where $|B_1| = |B_2| = |B_3| = \ell > 0$. Suppose also that all the 3 pairs (B_i, B_j), $1 \le i < j \le 3$, are ε-regular, with $d(B_i, B_j) = \sigma > 0$ for all $1 \le i < j \le 3$.

We claim that, then, the graph G contains a triangle provided ε is small with respect to σ. To prove this claim, first observe that, from the ε-regularity of (B_1, B_2) and of (B_1, B_3), one may deduce that there are at least $(1-4\varepsilon)\ell > 0$ vertices b_1 in B_1 such that their degrees into B_2 and B_3 are both at least $(\sigma - \varepsilon)\ell$ and at most $(\sigma + \varepsilon)\ell$ (see Claim 27 below). However, by the ε-regularity of (B_2, B_3), at least

$$(\sigma - \varepsilon)|\Gamma(b_1) \cap B_2||\Gamma(b_1) \cap B_3| \ge (\sigma - \varepsilon)^3 \ell^2 > 0 \tag{9.45}$$

edges are induced by the pair $(\Gamma(b_1) \cap B_2, \Gamma(b_1) \cap B_3)$ as long as $\sigma - \varepsilon \ge \varepsilon$, that is, $\varepsilon < \sigma/2$. Thus the claim is proved. Note that, in fact, we have proved that if $\varepsilon < \sigma/2$, then the number of triangles in G is at least

$$c\ell^3 = c(\sigma, \varepsilon)\ell^3, \tag{9.46}$$

where $c(\sigma, \varepsilon) = (1 - 4\varepsilon)(\sigma - \varepsilon)^3$. Clearly, $c(\sigma, \varepsilon) \to \sigma^3$ as $\varepsilon \to 0$. For comparison, let us observe that the number of triangles is $\sim \sigma^3 \ell^3$ as $\ell \to \infty$ if G is drawn at random from all the tripartite graphs on (B_1, B_2, B_3) with $\lfloor \sigma \ell^2 \rfloor$ edges within all the pairs (B_i, B_j).

Let us now turn to the embedding lemma that we shall use to prove Theorem 18. We have already seen the essence of the proof of this lemma in the warm-up above. In order to state the lemma concisely, we introduce the following definition.

Definition 24 (Property $\mathcal{P}(k, \ell, \beta, \varepsilon)$) *A graph G has property $\mathcal{P}(k, \ell, \beta, \varepsilon)$ if it admits a partition $V = V(G) = \bigcup_{1 \le i \le k} B_i$ of its vertex set such that*

(i) $|B_i| = \ell$ *for all* $1 \le i \le k$,

(ii) all the $\binom{k}{2}$ *pairs* (B_i, B_j), *where* $1 \le i < j \le k$, *are* ε*-regular, and*

(iii) $\beta < d(B_i, B_j) < 1 - \beta$ *for all* $1 \le i < j \le k$.

The embedding lemma is as follows.

Lemma 25 *For all $0 < \beta < 1/2$ and $k \geq 1$, there exist $\varepsilon_k = \varepsilon_k(k, \beta) > 0$ and $\ell_k = \ell_k(k, \beta)$ so that every graph with property $\mathcal{P}(k, \ell, \beta, \varepsilon_k)$ with $\ell \geq \ell_k$ is k-universal.*

Remark 26 *If H is some graph on k vertices and G is a graph satisfying property $\mathcal{P}(k, \ell, \beta, \varepsilon_k)$, then one may in fact estimate the number of copies of H in G (cf. (9.46)). Variants of Lemma 25 that give such numerical information are sometimes referred to as counting lemmas.*

Before we start the proof of Lemma 25, we state and prove a simple claim on regular pairs. If u is a vertex in a graph G and $W \subset V(G)$, then we write $d_W(u)$ for the degree $|\Gamma(u) \cap W|$ of u 'into' W.

Claim 27 *Let (U, W) be an ε-regular pair in a graph G, and suppose $d(U, W) = \varrho$. Then the number of vertices $u \in U$ satisfying*

$$(\varrho - \varepsilon)|W| \leq d_W(u) = |\Gamma(u) \cap W| \leq (\varrho + \varepsilon)|W| \tag{9.47}$$

is more than $(1 - 2\varepsilon)|U|$.

Proof. Suppose for a contradiction that Claim 27 is false. Let $U^- \subset U$ be the set of $u \in U$ for which the *first* inequality in (9.47) fails, and let $U^+ \subset U$ be the set of $u \in U$ for which the *second* inequality in (9.47) fails. We are assuming that $|U^+ \cup U^-| \geq 2\varepsilon|U|$. Therefore, say, $|U^+| \geq \varepsilon|U|$. However, we then have

$$d(U^+, W) > \varrho + \varepsilon. \tag{9.48}$$

Since (U, W) is ε-regular, such a witness of ε-irregularity cannot exist. The case in which $|U^-| \geq \varepsilon|U|$ is similar. This proves Claim 27. □

We now give the proof of the embedding lemma, Lemma 25.

Proof. (Proof of Lemma 25) The proof will be by induction on k. For $k = 1$ the statement of the lemma is trivial. For $k = 2$, it suffices to take $\varepsilon_2 = \varepsilon_2(2, \beta) = \beta$ and $\ell_2(2, \beta) = 1$. Indeed, observe that the fact that $0 < d(B_1, B_2) < 1$ implies that there must be b_i and $b_i' \in B_i$ ($i \in \{1, 2\}$) such that $b_1 b_2$ is an edge and $b_1' b_2'$ is not an edge. For the induction step, suppose that $k \geq 3$ and that the assertion of the lemma is true for smaller values of k and for all $0 < \beta < 1/2$.

Suppose we are given some β, with $0 < \beta < 1/2$. Let

$$\varepsilon_k = \varepsilon_k(k, \beta) = \min\left\{\frac{1}{2k}, \frac{1}{2}\beta\varepsilon_{k-1}\right\}, \tag{9.49}$$

and

$$\ell_k = \ell_k(k, \beta) = \max\left\{2\left\lceil\frac{\ell_{k-1}}{\beta}\right\rceil, k\right\}, \tag{9.50}$$

where

$$\varepsilon_{k-1} = \varepsilon_{k-1}(k-1, \beta/2) \quad \text{and} \quad \ell_{k-1} = \ell_{k-1}(k-1, \beta/2). \tag{9.51}$$

We claim that the choices for ε_k and ℓ_k in (9.49) and (9.50) will do. Thus, let G be a graph satisfying property $\mathcal{P}(k, \ell, \beta, \varepsilon_k)$, where $\ell \geq \ell_k$. Let $B_1, \ldots, B_k$ be the blocks of the partition of $V = V(G)$ ensured by Definition 24. Suppose H is a graph on the vertices $x_1, \ldots, x_k$. We shall show that there exist $b_1, \ldots, b_k$, with $b_i \in B_i$, such that the map $\phi\colon x_i \mapsto b_i$ is an embedding of H into G (that is, ϕ is an isomorphism between H and $G[b_1, \ldots, b_k]$, the graph induced by the b_i in G).

Pick a vertex $b_k \in B_k$ for which

$$(d(B_k, B_j) - \varepsilon_k)\ell < d_{B_j}(b_k) = |\Gamma(b_k) \cap B_j| < (d(B_k, B_j) + \varepsilon_k)\ell \tag{9.52}$$

for all $1 \leq j < k$. The existence of such a vertex b_k follows from Claim 27. Indeed, the claim tells us that the number of vertices that fail (9.52) for some $1 \leq j < k$ is at most

$$2(k-1)\varepsilon_k \ell < \ell = |B_k|, \tag{9.53}$$

since $\varepsilon_k \leq 1/2k$ (see (9.49)). For all $1 \leq j < k$, we now choose sets $\widetilde{B}_j \subset B_j$ satisfying the following properties:

(*i*) $|\widetilde{B}_j| = \lceil \beta\ell/2 \rceil \geq \ell_{k-1}$,

(*ii*) if $x_j x_k \in E(H)$, then $bb_k \in E(G)$ for all $b \in \widetilde{B}_j$, and if $x_j x_k \notin E(H)$, then $bb_k \notin E(G)$ for all $b \in \widetilde{B}_j$.

The existence of the sets $\widetilde{B}_j$ ($1 \leq j < k$) follows from our choice of b_k. Indeed, (9.52) tells us that b_k has more than

$$(d(B_k, B_j) - \varepsilon_k)\ell > (\beta - \varepsilon_k)\ell \geq \left(\beta - \frac{1}{2}\beta\varepsilon_{k-1}\right)\ell \geq \frac{1}{2}\beta\ell \tag{9.54}$$

neighbours in B_j. Similarly, (9.52) tells us that b_k has more than

$$(1 - d(B_k, B_j) - \varepsilon_k)\ell > (\beta - \varepsilon_k)\ell \geq \frac{1}{2}\beta\ell \tag{9.55}$$

non-neighbours in B_j.

Now fix a pair $1 \leq i < j < k$, and let $X_i \subset \widetilde{B}_i$ and $X_j \subset \widetilde{B}_j$ be such that $|X_i| \geq \varepsilon_{k-1}|\widetilde{B}_i|$ and $|X_j| \geq \varepsilon_{k-1}|\widetilde{B}_j|$. Then

$$\min\{|X_i|, |X_j|\} \geq \varepsilon_{k-1}|\widetilde{B}_i| = \varepsilon_{k-1}|\widetilde{B}_j| \geq \frac{2\varepsilon_k}{\beta}\left\lceil \frac{\beta\ell}{2} \right\rceil \geq \varepsilon_k \ell. \tag{9.56}$$

From the ε_k-regularity of the pair (B_i, B_j), we deduce that

$$\begin{aligned} |d(X_i, X_j) - d(\widetilde{B}_i, \widetilde{B}_j)| &\leq |d(X_i, X_j) - d(B_i, B_j)| \\ &\quad + |d(B_i, B_j) - d(\widetilde{B}_i, \widetilde{B}_j)| \leq 2\varepsilon_k \leq \varepsilon_{k-1}. \end{aligned} \tag{9.57}$$

Therefore all the pairs $(\widetilde{B}_i, \widetilde{B}_j)$ with $1 \le i < j < k$ are ε_{k-1}-regular. Our induction hypothesis then tells us that there exist $b_j \in B_j$ $(1 \le j < k)$ for which the map $x_j \mapsto b_j$ $(1 \le j < k)$ is an isomorphism between $H - x_k$ and $G[b_1, \dots, b_{k-1}]$. Clearly, $\phi\colon x_j \mapsto b_j$ $(1 \le j \le k)$ is an isomorphism between H and $G[b_1, \dots, b_k]$. □

Proof of Theorem 18

We are now able to prove Theorem 18. We shall make use of two well known results from graph theory: Ramsey's theorem and Turán's theorem.

Proof. (Proof of Theorem 18) Let $\delta_1 = \max\{\sigma + \delta - 1/2, 1/2 - \sigma + \delta\}$. We clearly have $0 < \delta_1 < 1/2$ and in fact

$$\begin{aligned} 0 < \frac{1}{2} - \delta_1 \quad &\le \quad \frac{1}{2} - \left(\frac{1}{2} - \sigma + \delta\right) = \sigma - \delta \\ &\le \quad \sigma + \delta = \frac{1}{2} + \left(\sigma + \delta - \frac{1}{2}\right) \le \frac{1}{2} + \delta_1 < 1. \end{aligned} \tag{9.58}$$

The inequalities in (9.58) imply that property $\mathcal{R}(\gamma, \delta, \sigma)$ implies property $\mathcal{R}(\gamma, \delta_1, 1/2)$. Therefore we may assume in Theorem 18 that $\sigma = 1/2$ and $0 < \delta < 1/2$. We may further assume that

$$k \ge \frac{3}{\beta}, \quad \text{where} \quad \beta = \frac{1}{2} - \delta > 0. \tag{9.59}$$

We now define the constants γ and N_0 promised in Theorem 18. Put

$$\varepsilon = \min\left\{\frac{1}{R(k,k,k)}, \varepsilon_k\right\}, \tag{9.60}$$

where $\varepsilon_k = \varepsilon_k(k, \beta/2)$ is the number whose existence is guaranteed by Lemma 25, and $R(k,k,k)$ is the usual Ramsey number for K^k and three colours: $R(k,k,k)$ is the minimal integer R such that, in any colouring of the edges of K^R with three colours, we must have a K^k all of whose edges are coloured with the same colour.

Put $k_0 = R(k,k,k)$, and invoke Theorem 4 with this k_0 and ε given in (9.60). We obtain constants $K_0(\varepsilon, k_0) \ge k_0$ and $N_0(\varepsilon, k_0)$. Now let

$$N_0 = \max\left\{N_0(\varepsilon, k_0), \frac{1}{1-\varepsilon} K_0(\varepsilon, k_0)\ell_k\right\}, \tag{9.61}$$

where $\ell_k = \ell_k(k, \beta/2)$ is given by Lemma 25. Furthermore, we let

$$\gamma = \frac{k(1-\varepsilon)}{K_0(\varepsilon, k_0)}. \tag{9.62}$$

Our aim is to show that the choices for N_0 and γ given in (9.61) and (9.62) will do.

Suppose a graph $G = G^n$ with $n \geq N_0$ vertices satisfies property $\mathcal{R}(\gamma, \delta, 1/2)$. We shall use the regularity lemma to find an induced subgraph G' of G that satisfies property $\mathcal{P}(k, \ell, \beta/2, \varepsilon_k)$, where $\ell \geq \ell_k$. An application of the embedding lemma, Lemma 25, will then complete the proof.

Let $V = V(G) = \bigcup_{0 \leq i \leq t} C_i$ be an ε-regular, (ε, t)-equitable partition for G with $k_0 \leq t \leq K_0(\varepsilon, k_0)$. The existence of such a partition is ensured by Theorem 4. Let $\ell = |C_i|$ $(1 \leq i \leq t)$.

Consider the graph F on the vertex set $[t] = \{1, \dots, t\}$, where $ij \in E(F)$ if and only if (C_i, C_j) is an ε-regular pair in G. We know that F has at least $(1-\varepsilon)\binom{t}{2}$ edges. By the well-known theorem of Turán [64], it follows that F has a clique with $R = R(k, k, k)$ vertices. Adjust the notation so that this clique is induced by the vertices $1, \dots, R$. Then the blocks C_i $(1 \leq i \leq R)$ are such that all the pairs (C_i, C_j) with $1 \leq i < j \leq R$ are ε-regular.

We now define a partition $T_1 \cup T_2 \cup T_3$ of the set $\binom{[R]}{2}$ of the pairs ij $(1 \leq i < j \leq R)$ as follows: the pair ij belongs to T_1 if and only if $d(C_i, C_j) \leq \beta/2$; the pair ij belongs to T_2 if and only if $\beta/2 < d(C_i, C_j) < 1 - \beta/2$; and, finally, the pair ij belongs to T_3 if and only if $d(C_i, C_j) \geq 1 - \beta/2$.

By the definition of $R = R(k, k, k)$, we know that there is a set $J \subset [R]$ with $|J| = k$ such that $F[J]$ is monochromatic, that is, $\binom{J}{2} \subset T_\alpha$ for some $\alpha \in \{1, 2, 3\}$. We consider the graph

$$G' = G\left[\bigcup_{j \in J} C_j\right] \tag{9.63}$$

induced by $\bigcup_{j \in J} C_j$ in G. Suppose $\alpha = 1$. Then the number of edges $e(G')$ in G' satisfies

$$e(G') \leq \binom{k}{2} \frac{\beta}{2} \ell^2 + k \binom{\ell}{2} \leq \frac{\beta k^2 \ell^2}{4} + \frac{k \ell^2}{2} < \left(\frac{1}{2} - \delta\right) \binom{k\ell}{2}, \tag{9.64}$$

where we have used (9.59) and the fact that $k\ell \geq k > 6$. Since $|V(G')| = k\ell \geq (1-\varepsilon)kn/K_0(\varepsilon, k_0) = \gamma n$ (see (9.62)), inequality (9.64) contradicts property $\mathcal{P}(\gamma, \delta, 1/2)$. This contradiction shows that $\alpha \neq 1$. If $\alpha = 3$, then we obtain a similar contradiction. In this case, as a little calculation using (9.59) shows, the graph G' satisfies

$$e(G') \geq \binom{k}{2} \left(1 - \frac{\beta}{2}\right) \ell^2 > \left(\frac{1}{2} + \delta\right) \binom{k\ell}{2}. \tag{9.65}$$

Thus $\alpha \neq 3$ and we conclude that $\alpha = 2$. We finally observe that, by (9.61), we have

$$\ell \geq \frac{(1-\varepsilon)n}{K_0(\varepsilon, k_0)} \geq \frac{(1-\varepsilon)N_0}{K_0(\varepsilon, k_0)} \geq \ell_k. \tag{9.66}$$

Therefore, as promised, the graph G' satisfies property $\mathcal{P}(k, \ell, \beta/2, \varepsilon_k)$ for $\ell \geq \ell_k$. To complete the induction step, it suffices to invoke Lemma 25.

The proof of Theorem 18 is complete. □

9.4 Further applications

In this section, we mention a few more applications of the regularity lemma to illustrate some further aspects of its uses.

9.4.1 Embedding large bounded degree graphs

Lemma 25, the embedding lemma, deals with induced embedding, that is, there we are concerned with embedding certain graphs as *induced* subgraphs in a given graph. In several applications, one is interested in finding embeddings as subgraphs that need not be necessarily induced. In this section, we shall briefly discuss some variants of Lemma 25 for 'non-induced' embeddings.

Let us say that a graph G has property $\mathcal{P}_{\rm w}(k,\ell,\beta,\varepsilon)$ if it satisfies the conditions in Definition 24, except that, instead of (*iii*) in that definition, we only require the following weaker property:

(*iv*) $d(B_i,B_j)>\beta$ for all $1\le i<j\le k$.

We now state a variant of the embedding lemma for subgraphs; at the expense of requiring that the graph to be embedded should have bounded degree, we gain on the size of the graph that we are able to embed. For convenience, let us say that a graph H is of *type* (m,k) if H admits a proper vertex colouring with k colours in such a way that every colour occurs at most m times.

Lemma 28 *For all $k\ge 1$, $\beta>0$ and $\Delta\ge 1$, there exist $\varepsilon=\varepsilon(k,\beta,\Delta)>0$, $\nu=\nu(k,\beta,\Delta)>0$, and $\ell_0=\ell_0(k,\beta,\Delta)$ so that every graph with property $\mathcal{P}_{\rm w}(k,\ell,\beta,\varepsilon)$ with $\ell\ge\ell_0$ contains all graphs of type $(\nu\ell,k)$ that have maximum degree at most Δ.*

Let us stress that the lemma above allows us to embed bounded degree graphs $H=H^n$ in certain graphs $G=G^N$ with N only linearly larger than n. The regularity lemma and Lemma 28 were the key tools in Chvátal, Rödl, Szemerédi, and Trotter [16], where it is proved that the Ramsey number of a bounded degree graph $H=H^n$ is linear in n.

The proof of Lemma 28 in [16] gives for ν an exponentially small quantity in Δ. Thus, one has to have 'a lot of extra room' for the embedding. A recent, beautiful result of Komlós, Sárközy, and Szemerédi [44] (see [45] for an algorithmic version), known as the *blow-up lemma*, shows that one need not waste so much room; in fact, one does not have to waste any room at all if a small extra hypothesis is imposed on the graph where the embedding is to take place.

Let (U,W) be an ε-regular pair in a graph G. We say that (U,W) is (ε,δ)*-super-regular* if

(†) *for all $u \in U$, we have $d(u) \geq \delta|W|$, and for all $w \in W$, we have $d(w) \geq \delta|U|$.*

Observe that (†) implies that $d(U,W) \geq \delta$. Let us say that a graph G satisfies property $\mathcal{P}_{\mathrm{w}}(k,\ell,\beta,\varepsilon,\delta)$ if it satisfies $\mathcal{P}_{\mathrm{w}}(k,\ell,\beta,\varepsilon)$, with (*ii*) in the definition of property $\mathcal{P}(k,\ell,\beta,\varepsilon)$, Definition 24, strengthened to

(*v*) all the $\binom{k}{2}$ pairs (B_i,B_j), where $1 \leq i < j \leq k$, are (ε,δ)-super-regular.

We may now state the blow-up lemma.

Theorem 29 *For all $k \geq 1$, $\beta > 0$, $\delta > 0$, and $\Delta \geq 1$, there exist $\varepsilon = \varepsilon(k,\beta,\delta,\Delta) > 0$ and $\ell_0 = \ell_0(k,\beta,\delta,\Delta)$ so that every graph that satisfies property $\mathcal{P}_{\mathrm{w}}(k,\ell,\beta,\varepsilon,\delta)$ with $\ell \geq \ell_0$ contains all graphs of type (ℓ,k) that have maximum degree at most Δ.*

The striking difference between Lemma 28 and Theorem 29 is that, with the rather weak additional condition (†), we are able to embed *spanning subgraphs* (that is, we may take $\nu = 1$).

Theorem 29 is one of the key ingredients in the recent successes of Komlós, Sárközy, and Szemerédi in tackling well-known, hard conjectures such as Seymour's conjecture and Pósa's conjecture on powers of Hamiltonian cycles [46, 47], a conjecture of Bollobás on graph packings [43], and Alon and Yuster's conjecture [9] (see [42, p. 175]).

We shall not discuss the proof of Theorem 29, which is indeed quite difficult (see [56, 57] for alternative proofs). The reader should consult Komlós [42] for a survey on the blow-up lemma.

9.4.2 Property testing

We shall now discuss a recent application of regularity to a complexity problem. We shall see how the regularity lemma may be used to prove the correctness of certain algorithms. This section is based on results due to Alon, Fischer, Krivelevich, and Szegedy [6, 7]. These authors develop a new variant of the regularity lemma and use it to prove a far reaching result concerning the *testability* of certain graph properties.

As a starting point, we state the following result, which the reader should first try to prove with bare hands.

Theorem 30 *For any $\varepsilon > 0$ there is a $\delta > 0$ for which the following holds. Suppose a graph $G = G^n = (V,E)$ is such that $G - F = (V, E \setminus F)$ contains a triangle for any set $F \subset \binom{V}{2}$ with $|F| \leq \varepsilon\binom{n}{2}$. Then G contains at least δn^3 triangles.*

The theorem above follows easily from the warm-up result in Section 9.3.3 and the regularity lemma. A proof of Theorem 30 that does not use the regularity lemma (in any form!) would be of considerable interest.

Theorem 30 implies that we may efficiently distinguish triangle-free graphs from graphs that contain triangles in a robust way, that is, graphs G as in the statement of this theorem. Indeed, one may simply randomly pick a number of vertices, say N, from the input graph $G = G^n$ and then check whether a triangle is induced. If we catch no triangle, we return the answer 'yes, the graph G is triangle-free'. If we do catch a triangle, we return the answer 'no, the graph G is "ε-far" from being triangle-free'.

The striking fact about the algorithm above is that it will return the correct answer with high probability if N is a large enough constant with respect to ε. Here, N need not grow with n, the number of vertices in the input graph $G = G^n$, and hence this is a *constant time* algorithm. In this section, we shall briefly discuss some far reaching generalizations of this result.

Definitions and the testability result

The general notion of property testing was introduced by Rubinfeld and Sudan [58], but in the context of combinatorial testing it is the work of Goldreich and his co-authors [29, 30, 31, 32, 33] that are most relevant to us.

Let $\mathcal{G}^n$ be the collection of all graphs on a fixed n-vertex set, say $[n] = \{1, \dots, n\}$. Put $\mathcal{G} = \bigcup_{n \geq 1} \mathcal{G}^n$. A *property* of graphs is simply a subset $\mathcal{P} \subset \mathcal{G}$ that is closed under isomorphisms. There is a natural notion of distance in each $\mathcal{G}^n$, the *normalized Hamming distance*: the distance $d(G, H) = d_n(G, H)$ between two graphs G and $H \in \mathcal{G}^n$ is $|E(G) \triangle E(H)| \binom{n}{2}^{-1}$, where $E(G) \triangle E(H)$ denotes the symmetric difference of the edge sets of G and H.

We say that a graph G is *ε-far* from having property $\mathcal{P}$ if

$$d(G, \mathcal{P}) = \min_{H \in \mathcal{P}} d(G, H) \geq \varepsilon, \tag{9.67}$$

that is, a total of $\geq \varepsilon \binom{n}{2}$ edges have to be added to or removed from G to turn it into a graph that satisfies $\mathcal{P}$.

An *ε-test* for a graph property $\mathcal{P}$ is a randomized algorithm $\mathcal{A}$ that receives as input a graph G and behaves as follows: if G has $\mathcal{P}$ then with probability $\geq 2/3$ we have $\mathcal{A}(G) = 1$, and if G is ε-far from having $\mathcal{P}$ then with probability $\geq 2/3$ we have $\mathcal{A}(G) = 0$. The graph G is given to $\mathcal{A}$ through an oracle; we assume that $\mathcal{A}$ is able to generate random vertices from G and it may *query* the oracle whether two vertices that have been generated are adjacent.

We say that a graph property $\mathcal{P}$ is *testable* if, for all $\varepsilon > 0$, it admits an ε-test that makes at most Q queries to the oracle, where $Q = Q(\varepsilon)$ is a constant that depends only on ε. Note that, in particular, we require the number of queries to be independent of the order of the input graph.

Goldreich, Goldwasser, and Ron [30, 31], besides showing that there exist NP graph properties that are not testable, proved that a large class of interesting graph properties *are* testable, including the property of being k-colourable, of having a clique with $\geq \varrho n$ vertices, and of having a cut with $\geq \varrho n^2$ edges, where n is the order of the input graph. The regularity lemma is not used in [30, 31]. The fact that k-colourability is testable had in fact been proved implicitly in [20], where regularity is used.

We are now ready to turn to the result of Alon, Fischer, Krivelevich, and Szegedy [6, 7]. Let us consider properties from the first order theory of graphs. Thus, we are concerned with properties that may be expressed through quantification of vertices, Boolean connectives, equality, and adjacency. Of particular interest are the properties that may be expressed in the form

$$\exists x_1, \dots, x_r \, \forall y_1, \dots, y_s \, A(x_1, \dots, x_r, y_1, \dots, y_s), \tag{9.68}$$

where A is a quantifier-free first order expression. Let us call such properties *of type* $\exists\forall$. Similarly, we define properties *of type* $\forall\exists$. The main result of [6, 7] is as follows.

Theorem 31 *All first order properties of graphs that may be expressed with at most one quantifier as well as all properties that are of type* $\exists\forall$ *are testable. Furthermore, there exist first order properties of type* $\forall\exists$ *that are not testable.*

The first part of the proof of the positive result in Theorem 31 involves the reduction, up to testability, of properties of type $\exists\forall$ to a certain generalized colourability property. A new variant of the regularity lemma is then used to handle this generalized colouring problem.

A variant of the regularity lemma

In this section we shall state a variant of the regularity lemma proved in [6, 7].

Let us say that a partition $P = (C_i)_{i=1}^k$ of a set V is an *equipartition* of V if all the sets C_i $(1 \leq i \leq k)$ differ by at most 1 in size. In this section, we shall be interested in partitions as in Remark 9 and Theorem 11. Below, we shall have an equipartition of V

$$P' = \{C_{i,j} : 1 \leq i \leq k,\ 1 \leq j \leq \ell\}$$

that is a refinement of a given partition $P = (C_i)_{i=1}^k$. In this notation, we understand that, for all i, all the $C_{i,j}$ $(1 \leq j \leq \ell)$ are contained in C_i.

Theorem 32 *For every integer* k_0 *and every function* $0 < \varepsilon(r) < 1$ *defined on the positive integers, there are constants* $K = K(k_0, \varepsilon)$ *and* $N = N(k_0, \varepsilon)$ *with the following property. If* G *is any graph with at least* N *vertices, then there exist equipartitions* $P = (C_i)_{1 \leq i \leq k}$ *and* $P' = (C_{i,j})_{1 \leq i \leq k,\, 1 \leq j \leq \ell}$ *of* $V = V(G)$ *such that the following hold:*

(*i*) *$|P| = k \geq k_0$ and $|P'| = k\ell \leq K$;*

(*ii*) *at least $(1-\varepsilon(0))\binom{k}{2}$ of the pairs $(C_i, C_{i'})$ with $1 \leq i < i' \leq k$ are $\varepsilon(0)$-regular;*

(*iii*) *for all $1 \leq i < i' \leq k$, at least $(1-\varepsilon(k))\ell^2$ of the pairs $(C_{i,j}, C_{i',j'})$ with j, $j' \in [\ell]$ are $\varepsilon(k)$-regular;*

(*iv*) *for at least $(1-\varepsilon(0))\binom{k}{2}$ of the pairs $1 \leq i < i' \leq k$, we have that for at least $(1-\varepsilon(0))\ell^2$ of the pairs j, $j' \in [\ell]$ we have*

$$|d_G(C_i, C_{i'}) - d_G(C_{i,j}, C_{i',j'})| \leq \varepsilon(0).$$

Suppose we have partitions P and P' as in Theorem 32 above and that $\varepsilon(k) \ll 1/k$. It is not difficult to see that then, for many 'choice' functions $j\colon [k] \to [\ell]$, we have that $\widetilde{P} = (C_{i,j(i)})_{1\leq i \leq k}$ is an equipartition of an induced subgraph of G such that the following hold:

(*a*) *all* the pairs $(C_{i,j(i)}, C_{i',j(i')})$ are $\varepsilon(k)$-regular,

(*b*) for at least $(1-\varepsilon(0))\binom{k}{2}$ of the pairs $1 \leq i < i' \leq k$, we have

$$|d_G(C_i, C_{i'}) - d_G(C_{i,j(i)}, C_{i',j(i')})| \leq \varepsilon(0).$$

Roughly speaking, this consequence of Theorem 32 lets us have some grip on the irregular pairs. Even if $(C_i, C_{i'})$ is irregular, the pair $(C_{i,j(i)}, C_{i',j(i')})$ is regular and hence we have some control over the induced bipartite graph $G[C_i, C_{i'}]$. For instance, if in some application we have to construct some bipartite graph within $G[C_i, C_{i'}]$, we may do so by working on the subgraph $G[C_{i,j(i)}, C_{i',j(i')}]$.

We have already observed that we must allow irregular pairs in Theorem 4 (see Section 9.2.1). In a way, Theorem 32 presents a way around this difficulty.

Theorem 32 and its corollary mentioned above are the main ingredients in the proof of the following result (see [6, 7] for details).

Theorem 33 *For every $\varepsilon > 0$ and $h \geq 1$, there is $\delta = \delta(\varepsilon, h) > 0$ for which the following holds. Let H be an arbitrary graph on h vertices and let $\mathcal{P} = \mathrm{Forb}_{\mathrm{ind}}(H)$ be the property of not containing H as an induced subgraph. If an n-vertex graph G is ε-far from $\mathcal{P}$, then G contains δn^h induced copies of H.*

The case in which H is a complete graph follows from the original regularity lemma (the warm-up observation of Section 9.3.3 proved this for $H = K^3$), but the general case requires the corollary to Theorem 32 discussed above. Note that Theorem 33 immediately implies that the property of membership in $\mathrm{Forb}_{\mathrm{ind}}(H)$ (in order words, the property of not containing an induced copy of H) is a testable property for any graph H.

The proof of Theorem 31 requires a generalization of Theorem 33 related to the colouring problem alluded to at the end of the previous section. We refer the reader to [6, 7]. We close by remarking that Theorem 32 has an algorithmic version, although we stress that this is not required in the proof of Theorem 31.

9.5 Proof of the regularity lemma

We now prove the regularity lemma for sparse graphs. We shall prove Theorem 13. The proof of Theorem 15 is similar. We observe that the proof below follows very closely the proof of the original regularity lemma, Theorem 4. Indeed, to recover a proof of Theorem 4 from the proof below, it suffices to set $G = K^n$.

9.5.1 The refining procedure

Fix $G = G^n$ and put $V = V(G)$. Also, assume that $P_0 = (V_i)_1^\ell$ is a fixed partition of V, and that G is (P_0, η)-uniform for some $0 < \eta \le 1$. Moreover, let $p = p(G)$ be as in (9.11).

We start with a 'continuity' result. Let $H \subset G$ be a spanning subgraph of G.

Lemma 34 *Let $0 < \delta \le 10^{-2}$ be fixed. Let U, $W \subset V(G)$ be such that $(U, W) \prec P_0$, and $\delta|U|$, $\delta|W| \ge \eta n$. If $U^* \subset U$, $W^* \subset W$, $|U^*| \ge (1-\delta)|U|$, and $|W^*| \ge (1-\delta)|W|$, then*

(*i*) $|d_{H,G}(U^*, W^*) - d_{H,G}(U, W)| \le 5\delta$,

(*ii*) $|d_{H,G}(U^*, W^*)^2 - d_{H,G}(U, W)^2| \le 9\delta$.

Proof. Note first that we have $\eta \le \delta$, as $\eta n \le \delta|U|$, $\delta|W| \le \delta n$. Let U^*, W^* be as given in the lemma. We first check (*i*).

(*i*) We start by noticing that

$$\begin{aligned} d_{H,G}(U^*, W^*) &\ge \frac{e_H(U,W) - 2(1+\eta)p\delta|U||W|}{e_G(U,W)} \\ &\ge d_{H,G}(U,W) - 2\delta\frac{1+\eta}{1-\eta} \ge d_{H,G}(U,W) - 3\delta. \end{aligned}$$

Moreover,

$$\begin{aligned} d_{H,G}(U^*, W^*) &\le \frac{e_H(U,W)}{e_G(U^*,W^*)} \\ &\le \frac{e_H(U,W)}{(1-\eta)p|U^*||W^*|} \end{aligned}$$

$$\begin{aligned}
&\leq \frac{e_H(U,W)}{(1-\eta)p(1-\delta)^2|U||W|} \\
&\leq \frac{1+\eta}{(1-\eta)(1-\delta)^2} d_{H,G}(U,W) \\
&\leq d_{H,G}(U,W) + 5\delta.
\end{aligned}$$

Thus (i) follows.

(ii) The argument here is similar. First

$$\begin{aligned}
d_{H,G}(U^*,W^*) &\geq \frac{\big(e_H(U,W) - 2(1+\eta)p\delta|U||W|\big)^2}{e_G(U,W)^2} \\
&\geq d_{H,G}(U,W)^2 - \frac{4(1+\eta)p\delta|U||W|e_H(U,W)}{e_G(U,W)(1-\eta)p|U||W|} \\
&\geq d_{H,G}(U,W)^2 - 4\delta\frac{1+\delta}{1-\delta} \\
&\geq d_{H,G}(U,W)^2 - 5\delta.
\end{aligned}$$

Secondly,

$$\begin{aligned}
d_{H,G}(U^*,W^*)^2 &\leq \frac{e_H(U,W)^2}{e_G(U^*,W^*)^2} \\
&\leq \frac{e_H(U,W)^2}{(1-\eta)^2p^2|U^*|^2|W^*|^2} \\
&\leq \frac{e_H(U,W)^2}{(1-\eta)^2(1-\delta)^4p^2|U||W|} \\
&\leq \left(\frac{1+\eta}{(1-\eta)(1-\delta)^2}\right)^2 d_{H,G}(U,W)^2 \\
&\leq d_{H,G}(U,W)^2 + 9\delta.
\end{aligned}$$

Thus (ii) follows. □

In what follows, a constant $0 < \varepsilon \leq 1/2$ and a spanning subgraph $H \subset G$ of G is fixed. Also, we let $P = (C_i)_0^k$ be an (ε, k)-equitable partition of $V = V(G)$ refining P_0, where $4^k \geq \varepsilon^{-5}$. Moreover, we assume that $\eta \leq \eta_0 = \eta_0(k) = 1/k4^{k+1}$ and that $n = |G| \geq n_0 = n_0(k) = k4^{1+2k}$.

We now define an equitable partition $Q = Q(P)$ of $V = V(G)$ from P as follows. First, for each (ε, H, G)-irregular pair (C_s, C_t) of P with $1 \leq s < t \leq k$, we choose $X = X(s,t) \subset C_s$, $Y = Y(s,t) \subset C_t$ such that (i) $|X|$, $|Y| \geq \varepsilon|C_s| = \varepsilon|C_t|$, and (ii) $|d_{H,G}(X,Y) - d_{H,G}(C_s,C_t)| \geq \varepsilon$. For fixed $1 \leq s \leq k$, the sets $X(s,t)$ in

$$\{X = X(s,t) \subset C_s : 1 \leq t \leq k \text{ and } (C_s, C_t) \text{ is not } (\varepsilon, H, G)\text{-regular}\}$$

define a natural partition of C_s into at most 2^{k-1} blocks. Let us call such blocks the *atoms* of C_s. Now let $q = 4^k$ and set $m = \lfloor |C_s|/q \rfloor$ $(1 \leq s \leq k)$. Note that $\lfloor |C_s|/m \rfloor = q$ as $|C_s| \geq n/2k \geq 2q^2$. Moreover, for later use, note that $m \geq \eta n$. We now let Q' be a partition of $V = V(G)$ refining P such that (i) C_0 is a block of Q', (ii) all other blocks of Q' have cardinality m, except for possibly one, which has cardinality at most $m-1$, (iii) for all $1 \leq s \leq k$, every atom $A \subset C_s$ contains exactly $\lfloor |A|/m \rfloor$ blocks of Q', (iv) for all $1 \leq s \leq k$, the set C_s contains exactly $q = \lfloor |C_s|/m \rfloor$ blocks of Q'.

Let C'_0 be the union of the blocks of Q' that are not contained in any class C_s $(1 \leq s \leq k)$, and let C'_i $(1 \leq i \leq k')$ be the remaining blocks of Q'. We are finally ready to define our equitable partition $Q = Q(P)$: we let $Q = (C'_i)_1^{k'}$.

Lemma 35 *The partition $Q = Q(P) = (C'_i)_0^{k'}$ defined from P as above is a k'-equitable partition of $V = V(G)$ refining P, where $k' = kq = k4^k$, and $|C'_0| \leq |C_0| + n4^{-k}$.*

Proof. Clearly Q refines P. Moreover, clearly $m = |C'_1| = \ldots = |C'_{k'}|$ and, for all $1 \leq s \leq k$, we have $|C'_0| \leq |C_0| + k(m-1) \leq |C_0| + k|C_s|/q \leq |C_0| + n4^{-k}$. □

In what follows, for $1 \leq s \leq k$, we let $C_s(i)$ $(1 \leq i \leq q)$ be the classes of Q' that are contained in the class C_s of P. Also, for all $1 \leq s \leq k$, we set $C_s^* = \bigcup_{1 \leq i \leq q} C_s(i)$. Now let $1 \leq s \leq k$ be fixed. Note that $|C_s^*| \geq |C_s| - (m-1) \geq |C_s| - q^{-1}|C_s| \geq |C_s|(1-q^{-1})$. As $q^{-1} \leq 10^{-2}$ and $q^{-1}|C_s| \geq m \geq \eta n$, by Lemma 34 we have, for all $1 \leq s < t \leq k$,

$$|d_{H,G}(C_s^*, C_t^*) - d_{H,G}(C_s, C_t)| \leq 5q^{-1} \tag{9.69}$$

and

$$|d_{H,G}(C_s^*, C_t^*)^2 - d_{H,G}(C_s, C_t)^2| \leq 9q^{-1}. \tag{9.70}$$

9.5.2 Defect form of the Cauchy–Schwarz inequality

As in [60], the following ‘defect’ form of the Cauchy–Schwarz inequality will be used in the proof of Theorem 13.

Lemma 36 *Let $y_1, \ldots, y_v \geq 0$ be given. Suppose $0 \leq \varrho = u/v < 1$, and $\sum_{1 \leq i \leq u} y_i = \alpha\varrho \sum_{1 \leq i \leq v} y_i$. Then*

$$\sum_{1 \leq i \leq v} y_i^2 \geq \frac{1}{v}\left(1 + (\alpha-1)^2 \frac{\varrho}{1-\varrho}\right)\left\{\sum_{1 \leq i \leq v} y_i\right\}^2. \tag{9.71}$$

Since it is for the same price, we prove a weighted version of Lemma 36. The statement and proof of Lemma 37 below are from [25] (see also [24, 60]).

Lemma 37 *Let σ_i and d_i $(i \in I)$ be non-negative reals with $\sum_{i\in I} \sigma_i = 1$. Set $d = \sum_{i\in I} \sigma_i d_i$. Let $J \subset I$ be a proper subset of I such that $\sum_{j\in J} \sigma_j = \sigma < 1$ and*

$$\sum_{j\in J} \sigma_j d_j = \sigma(d + \mu).$$

Then

$$\sum_{i\in I} \sigma_i d_i^2 \geq d^2 + \frac{\mu^2\sigma}{1-\sigma}. \tag{9.72}$$

Proof. Let $\mathbf{u}_J = \left(\sqrt{\sigma_j}\right)_{j\in J}$, $\mathbf{v}_J = \left(\sqrt{\sigma_j} d_j\right)_{j\in J}$, $\mathbf{u}_{I\setminus J} = \left(\sqrt{\sigma_i}\right)_{i\in I\setminus J}$, and $\mathbf{v}_{I\setminus J} = \left(\sqrt{\sigma_i} d_i\right)_{i\in I\setminus J}$.

We use the Cauchy–Schwarz inequality in the form $|\langle \mathbf{x}, \mathbf{y}\rangle|^2 \leq \|\mathbf{x}\|^2 \|\mathbf{y}\|^2$. Taking $\mathbf{x} = \mathbf{u}_J$ and $\mathbf{y} = \mathbf{v}_J$ and $\mathbf{x} = \mathbf{u}_{I\setminus J}$ and $\mathbf{y} = \mathbf{v}_{I\setminus J}$, respectively, we infer that

$$\left(\sum_{j\in J} \sigma_j d_j\right)^2 \leq \sum_{j\in J} \sigma_j \sum_{j\in J} \sigma_j d_j^2,$$

and

$$\left(\sum_{i\in I\setminus J} \sigma_i d_i\right)^2 \leq \sum_{i\in I\setminus J} \sigma_i \sum_{i\in I\setminus J} \sigma_i d_i^2.$$

Therefore

$$\sum_{i\in I} \sigma_i d_i^2 \geq \frac{1}{\sigma}\left(\sum_{j\in J} \sigma_j d_j\right)^2 + \frac{1}{1-\sigma}\left(\sum_{i\in I-J} \sigma_i d_i\right)^2$$
$$= \sigma(d+\mu)^2 + (1-\sigma)\left(d - \frac{\sigma\mu}{1-\sigma}\right)^2 = d^2 + \frac{\mu^2\sigma}{1-\sigma},$$

as required. □

Proof. (Proof of Lemma 36) To prove Lemma 36, simply take $\sigma_i = 1/v$ and $d_i = y_i$ $(1 \leq i \leq v)$ in Lemma 37. Then $d = v^{-1}\sum_{1\leq i\leq v} y_i$, $\sigma = \varrho$, and $\mu = (\alpha - 1)d$. Inequality (9.72) then reduces to (9.71). □

9.5.3 The index of a partition

Similarly to [60], we define the *index* $\operatorname{ind}(R)$ of an equitable partition $R = (C_i)_0^r$ of $V = V(G)$ to be

$$\operatorname{ind}(R) = \frac{2}{r^2} \sum_{1\leq i<j\leq \ell} d_{H,G}(C_i, C_j)^2.$$

Note that trivially $0 \leq \operatorname{ind}(R) < 1$.

9.5.4 The index of subpartitions

Our aim now is to show that, for $Q = Q(P)$ defined as above, we have $\operatorname{ind}(Q) \geq \operatorname{ind}(P) + \varepsilon^5/100$.

The draw case

We start with the following lemma.

Lemma 38 *Suppose* $1 \leq s < t \leq k$. *Then*

$$\frac{1}{q^2}\sum_{i,\,j=1}^{q} d_{H,G}(C_s(i), C_t(j))^2 \geq d_{H,G}(C_s, C_t)^2 - \frac{\varepsilon^5}{100}.$$

Proof. By the (P_0, η)-uniformity of G and the fact that $(C_s, C_t) \prec P_0$, we have

$$\begin{aligned}
\frac{1}{q^2}\sum_{1\leq i\leq q}\sum_{1\leq j\leq q} d_{H,G}(C_s(i), C_t(j)) &= \frac{1}{q^2}\sum_{i,\,j}\frac{e_H(C_s(i), C_t(j))}{e_G(C_s(i), C_t(j))} \\
&\geq \sum_{i,\,j}\frac{e_H(C_s(i), C_t(j))}{(1+\eta)q^2p|C_s(i)||C_t(j)|} \\
&= \frac{e_H(C_s^*, C_t^*)}{(1+\eta)p|C_s^*||C_t^*|} \\
&\geq \frac{1-\eta}{1+\eta}d_{H,G}(C_s^*, C_t^*) \\
&\geq d_{H,G}(C_s^*, C_t^*) - 2\eta.
\end{aligned}$$

Thus, by the Cauchy–Schwarz inequality, we have

$$\frac{1}{q^2}\sum_{1\leq i\leq q}\sum_{1\leq j\leq q} d_{H,G}(C_s(i), C_t(j))^2 \geq d_{H,G}(C_s^*, C_t^*)^2 - 4\eta.$$

Furthermore, by (9.70), we have $d_{H,G}(C_s^*, C_t^*)^2 \geq d_{H,G}(C_s, C_t)^2 - 9q^{-1}$. Since we have $9q^{-1} + 4\eta \leq \varepsilon^5/100$, the lemma follows. □

The winning case

The inequality in Lemma 38 may be improved if (C_s, C_t) is an (ε, H, G)-irregular pair, as shows the following result.

Lemma 39 *Let* $1 \leq s < t \leq k$ *be such that* (C_s, C_t) *is not* (ε, H, G)-*regular. Then*

$$\frac{1}{q^2}\sum_{i,\,j=1}^{q} d_{H,G}(C_s(i), C_t(j))^2 \geq d_{H,G}(C_s, C_t)^2 + \frac{\varepsilon^4}{40} - \frac{\varepsilon^5}{100}.$$

Proof. Let $X = X(s,t) \subset C_s$, $Y = Y(s,t) \subset C_t$ be as in the definition of Q. Let $X^* \subset X$ be the maximal subset of X that is the union of blocks of Q, and similarly for $Y^* \subset Y$. Without loss of generality, we may assume that $X^* = \bigcup_{1\le i\le q_s} C_s(i)$, and $Y^* = \bigcup_{1\le j\le q_t} C_t(j)$. Note that $|X^*| \ge |X| - 2^{k-1}(m-1) \ge |X|(1 - 2^{k-1}m/|X|) \ge |X|(1 - 2^{k-1}/q\varepsilon) = |X|(1 - 1/\varepsilon 2^{k+1})$, and similarly $|Y^*| \ge |Y|(1 - 1/\varepsilon 2^{k+1})$. However, we have $1/\varepsilon 2^{k+1} \le 10^{-2}$ and $|X|/\varepsilon 2^{k+1}$, $|Y|/\varepsilon 2^{k+1} \ge \eta n$. Thus, by Lemma 34, we have $|d_{H,G}(X^*, Y^*) - d_{H,G}(X,Y)| \le 5/\varepsilon 2^{k+1}$. Moreover, by (9.69), we have $|d_{H,G}(C_s^*, C_t^*) - d_{H,G}(C_s, C_t)| \le 5q^{-1}$. Since $|d_{H,G}(X,Y) - d_{H,G}(C_s,C_t)| \ge \varepsilon$ and $5q^{-1} + 5/\varepsilon 2^{k+1} \le \varepsilon/2$, we have

$$|d_{H,G}(X^*,Y^*) - d_{H,G}(C_s^*, C_t^*)| \ge \varepsilon/2. \tag{9.73}$$

For later reference, let us note that $q_s m = |X^*| \ge |X| - 2^{k-1}m \ge \varepsilon|C_s| - 2^{k-1}m \ge \varepsilon q m - 2^{k-1}m$, and hence $q_s \ge \varepsilon q - 2^{k-1} \ge \varepsilon q/2$. Similarly, we have $q_t \ge \varepsilon q/2$. Let us now set $y_{ij} = d_{H,G}(C_s(i), C_t(j))$ for i, $j = 1, \dots, q$. In the proof of Lemma 38 we checked that

$$\sum_{1\le i\le q}\sum_{1\le j\le q} y_{ij} \ge \frac{1-\eta}{1+\eta} q^2 d_{H,G}(C_s^*, C_t^*) \ge (1-2\eta)q^2 d_{H,G}(C_s^*, C_t^*).$$

Similarly, one has

$$\sum_{1\le i\le q}\sum_{1\le j\le q} y_{ij} \le (1+3\eta)q^2 d_{H,G}(C_s^*, C_t^*),$$

$$\sum_{1\le i\le q_s}\sum_{1\le j\le q_t} y_{ij} \ge (1-2\eta)q_s q_t d_{H,G}(X^*, Y^*),$$

and

$$\sum_{1\le i\le q_s}\sum_{1\le j\le q_t} y_{ij} \le (1+3\eta)q_s q_t d_{H,G}(X^*, Y^*).$$

Let us set $\varrho = q_s q_t/q^2 \ge \varepsilon^2/4$, and $d_{s,t}^* = d_{H,G}(C_s^*, C_t^*)$. We now note that by (9.73) we either have

$$\begin{aligned}\sum_{1\le i\le q_s}\sum_{1\le j\le q_t} y_{ij} &\ge \frac{1-2\eta}{1+3\eta}\cdot\frac{q_s q_t}{q^2}\left(1 + \frac{\varepsilon}{2(d_{s,t}^*)^2}\right)\sum_{1\le i\le q}\sum_{1\le j\le q} y_{ij}\\ &\ge \varrho\left(1 + \frac{\varepsilon}{3(d_{s,t}^*)^2}\right)\sum_{1\le i\le q}\sum_{1\le j\le q} y_{ij},\end{aligned}$$

or else

$$\begin{aligned}\sum_{1\le i\le q_s}\sum_{1\le j\le q_t} y_{ij} &\le \frac{1+3\eta}{1-2\eta}\cdot\frac{q_s q_t}{q^2}\left(1 - \frac{\varepsilon}{2(d_{s,t}^*)^2}\right)\sum_{1\le i\le q}\sum_{1\le j\le q} y_{ij}\\ &\le \varrho\left(1 - \frac{\varepsilon}{3(d_{s,t}^*)^2}\right)\sum_{1\le i\le q}\sum_{1\le j\le q} y_{ij}.\end{aligned}$$

We may now apply Lemma 36 to conclude that

$$\begin{aligned}
\sum_{1\le i\le q}\sum_{1\le j\le q} y_{ij}^2 &\ge \frac{1}{q^2}\left(1+\frac{\varepsilon^2}{9(d_{s,t}^*)^2}\cdot\frac{\varrho}{1-\varrho}\right)\left\{\sum_{1\le i\le q}\sum_{1\le j\le q} y_{ij}\right\}^2\\
&\ge \frac{1}{q^2}\left(1+\frac{\varepsilon^2\varrho}{9(d_{s,t}^*)^2}\right)\left\{q^2(1-2\eta)d_{s,t}^*\right\}^2\\
&\ge q^2(1-4\eta)\left((d_{s,t}^*)^2+\frac{\varepsilon^2\varrho}{9}\right)\\
&\ge q^2\left((d_{s,t}^*)^2+\frac{\varepsilon^2\varrho}{10}-4\eta\right).
\end{aligned}$$

Therefore

$$\begin{aligned}
\frac{1}{q^2}\sum_{1\le i\le q}\sum_{1\le j\le q} d_{H,G}(C_s(i),C_t(j))^2 &\ge d_{H,G}(C_s^*,C_t^*)^2+\frac{\varepsilon^2\varrho}{10}-4\eta\\
&\ge d_{H,G}(C_s,C_t)^2+\frac{\varepsilon^4}{40}-(9\eta^{-1}+4\eta)\\
&\ge d_{H,G}(C_s,C_t)^2+\frac{\varepsilon^4}{40}-\frac{\varepsilon^5}{100},
\end{aligned}$$

as required. □

9.5.5 Proof of Theorem 13

We are now ready to prove the main lemma needed in the proof of Theorem 13.

Lemma 40 *Suppose $k\ge 1$ and $0<\varepsilon\le 1/2$ are such that $4^k\ge 1800\varepsilon^{-5}$. Let $G=G^n$ be a (P_0,η)-uniform graph of order $n\ge n_0=n_0(k)=k4^{2k+1}$, where $P_0=(V_i)_1^\ell$ is a partition of $V=V(G)$, and assume that $\eta\le\eta_0=\eta_0(k)=1/k4^{k+1}$. Let $H\subset G$ be a spanning subgraph of G. If $P=(C_i)_0^k$ is an (ε,H,G)-irregular (ε,k)-equitable partition of $V=V(G)$ refining P_0, then there is a k'-equitable partition $Q=(C_i')_0^{k'}$ of V such that (i) Q refines P, (ii) $k'=k4^k$, (iii) $|C_0'|\le|C_0|+n4^{-k}$, and (iv) $\mathrm{ind}(Q)\ge\mathrm{ind}(P)+\varepsilon^5/100$.*

Proof. Let P be as in the lemma. We show that the k'-equitable partition $Q=(C_i')_0^{k'}$ defined from P as above satisfies (i)–(iv). In view of Lemma 35, it only remains to check (iv). By Lemmas 38 and 39, we have

$$\begin{aligned}
\mathrm{ind}(Q) &= \frac{2}{(kq)^2}\sum_{1\le i\le q}\sum_{1\le j\le q} d_{H,G}(C_i',C_j')^2\\
&\ge \frac{2}{k^2}\sum_{1\le s<t\le k}\frac{1}{q^2}\sum_{1\le i\le q}\sum_{1\le j\le q} d_{H,G}(C_s(i),C_t(j))^2
\end{aligned}$$

$$\begin{aligned}
&\geq \frac{2}{k^2}\left\{\sum_{1\leq s<t\leq k}\left(d_{H,G}(C_s,C_t)^2-\frac{\varepsilon^5}{100}\right)+\varepsilon\binom{k}{2}\frac{\varepsilon^4}{40}\right\}\\
&\geq \operatorname{ind}(P)-\frac{\varepsilon^5}{100}+\frac{\varepsilon^5}{50}\\
&\geq \operatorname{ind}(P)+\frac{\varepsilon^5}{100}.
\end{aligned}$$

This completes the proof of the lemma. □

We now deduce Theorem 13 from Lemma 40.

Proof. (Proof of Theorem 13) Let $\varepsilon > 0$, $k_0 \geq 1$, and $\ell \geq 1$ be given. We may assume that $\varepsilon \leq 1/2$. Pick $s \geq 1$ such that $4^{s/4\ell} \geq 1800\varepsilon^{-5}$, $s \geq \max\{2k_0, 3\ell/\varepsilon\}$, and $\varepsilon 4^{s-1} \geq 1$. Let $f(0) = s$, and put inductively

$$f(t) = f(t-1)4^{f(t-1)} \qquad (t \geq 1).$$

Let $t_0 = \lfloor 100\varepsilon^{-5}\rfloor$ and set

$$N = \max\{n_0(f(t))\colon 0 \leq t \leq t_0\} = f(t_0)4^{2f(t_0)+1},$$

$$K_0 = \max\{6\ell/\varepsilon, N\},$$

and

$$\eta = \eta(\varepsilon, k_0, \ell) = \min\{\eta_0(f(t))\colon 0 \leq t \leq t_0\} = 1/4f(t_0+1) > 0.$$

Finally, we take $N_0 = N_0(\varepsilon, k_0, \ell) = K_0$. We claim that η, K_0, and N_0 as defined above will do.

To prove our claim, let $G = G^n$ be a fixed (P_0, η)-uniform graph with $n \geq N_0$, where $P_0 = (V_i)_1^\ell$ is a partition of $V = V(G)$. Furthermore, let $H \subset G$ be a spanning subgraph of G. We have $n \geq N_0 = K_0$. Suppose $t \geq 0$. Let us say that an equitable partition $P^{(t)} = (C_i)_0^k$ of V is *t-valid* if (i) $P^{(t)}$ refines P_0, (ii) $s/4\ell \leq k \leq f(t)$, (iii) $\operatorname{ind}\{P^{(t)}\} \geq t\varepsilon^5/100$, and (iv) $|C_0| \leq \varepsilon n(1-2^{-(t+1)})$. We now verify that a 0-valid partition $P^{(0)}$ of V does exist. Let $m = \lceil n/s\rceil$, and let Q be a partition of V with all blocks of cardinality m, except for possibly one, which has cardinality at most $m-1$, and moreover such that each V_i $(1 \leq i \leq \ell)$ contains $\lfloor |V_i|/m\rfloor$ blocks of Q. Grouping at most ℓ blocks of Q into a single block C_0, we arrive at an equitable partition $P^{(0)} = (C_i)_0^k$ of V that is 0-valid. Indeed, (i) is clear, and to check (ii) note that $k \leq n/m \leq s = f(0)$, and that there is $1 \leq i \leq \ell$ such that $|V_i| \geq n/\ell$, and so $k \geq \lfloor |V_i|/m\rfloor \geq \lfloor (n/\ell)/\lceil n/s\rceil\rfloor \geq (1/2)\{(n/\ell)/(2n/s)\} = s/4\ell$. Also, (iii) is trivial and (iv) does follow, since $|C_0| < \ell m \leq \ell\lceil n\varepsilon/3\ell\rceil \leq n\varepsilon/2$ as $n \geq K_0 \geq 6\ell/\varepsilon$.

Now note that if there is a t-valid partition $P^{(t)}$ of V, then $t \leq t_0 = \lfloor 100\varepsilon^{-5}\rfloor$, since $\operatorname{ind}\{P^{(t)}\} \leq 1$. Suppose t is the maximal integer for which there is a t-valid partition $P^{(t)}$ of V. We claim that $P^{(t)}$ is (ε, H, G)-regular. Suppose to the contrary that $P^{(t)}$ is not (ε, H, G)-regular. Then simply note that Lemma 40 gives a $(t+1)$-valid equitable partition $P^{(t+1)} = Q = Q(P^{(t)})$, contradicting the maximality of t. This completes the proof of Theorem 13. □

9.6 Local conditions for regularity

As briefly discussed in the introduction, our aim in this section is to discuss a well-known 'local' condition for regularity. It should be stressed that in Sections 9.6 and 9.7, we are concerned with dense graphs, that is, we are in the context of the original regularity lemma. (See Section 9.6.5 for a very brief discussion on extensions of the results in Section 9.6 to the sparse case.)

9.6.1 The basic argument

In this section, we give a result of Lindsey (see the proof of the upper bound in Theorem 15.2 in [23]) because it contains one of the key ideas used in developing local conditions for regularity.

Let $H = (h_{ij})$ be an n by n Hadamard matrix. Thus H is a $\{\pm 1\}$-matrix whose rows are pairwise orthogonal. Let

$$\operatorname{disc}(H; a, b) = \max_{I,J} \left| \sum_{i\in I,\, j\in J} h_{ij} \right|, \tag{9.74}$$

where the maximum is taken over all sets of rows I and all sets of columns J with $|I| = a$ and $|J| = b$. We also let the *discrepancy* of H be

$$\operatorname{disc}(H) = \max_{a,b} \operatorname{disc}(H; a, b), \tag{9.75}$$

where the maximum is taken over all $1 \le a \le n$ and $1 \le b \le n$.

Theorem 41 *For any n by n Hadamard matrix H, and any $1 \le a \le n$ and $1 \le b \le n$, we have*

$$\operatorname{disc}(H; a, b) \le \sqrt{abn}. \tag{9.76}$$

Proof. Let the rows of H be $\mathbf{v}_1, \dots, \mathbf{v}_n$, and fix a and b. Suppose, without loss of generality, that $I = \{1, \dots, a\}$ and $J = \{1, \dots, b\}$. Let also $\mathbf{1}_J = (1, \dots, 1, 0, \dots)^T \in \mathbb{R}^n$ be the characteristic vector of J. By the Cauchy–Schwarz inequality, we have

$$\left| \sum_{I,J} h_{ij} \right| = \left| \left\langle \sum_{i\in I} \mathbf{v}_i, \mathbf{1}_J \right\rangle \right| \le \left\| \sum_{i\in I} \mathbf{v}_i \right\| \sqrt{|J|} = \left\| \sum_{i\in I} \mathbf{v}_i \right\| \sqrt{b}. \tag{9.77}$$

From the pairwise orthogonality of the vectors $\mathbf{v}_i$, we have

$$\left\|\sum_{i\in I}\mathbf{v}_i\right\| = \sqrt{\sum_{i\in I}\|\mathbf{v}_i\|^2} = \sqrt{n|I|} = \sqrt{na}. \tag{9.78}$$

Plugging (9.78) into (9.77), we have

$$\left|\sum_{I,J} h_{ij}\right| \leq \sqrt{abn},$$

and the result follows. □

Corollary 42 *The discrepancy* disc(H) *of an n by n Hadamard matrix H satisfies* disc$(H) \leq n^{3/2}$.

An easy generalization of Theorem 41 above concerns the case in which we weaken the condition that the rows of H should be precisely orthogonal. Let us say that two vectors u, $v \in \mathbb{R}^n$ are *ε-quasi-orthogonal* if $|\langle u, v\rangle| \leq \varepsilon n$. Our next result roughly states that if the rows of an n by n matrix H are $o(1)$-quasi-orthogonal, then the discrepancy of H is $o(n^2)$.

Theorem 43 *Let $\delta > 0$ be fixed and let H be an n by n matrix whose rows $\mathbf{v}_i$ $(1 \leq i \leq n)$ are δ-quasi-orthogonal and $\|\mathbf{v}_i\| \leq \sqrt{n}$ for all $1 \leq i \leq n$. Then*

$$\mathrm{disc}(H; a, b) \leq n^2\sqrt{2\delta} \tag{9.79}$$

for all $a \geq 1/\delta$ and all $b \geq 1$.

Proof. We proceed exactly in the same manner as in the proof of Theorem 41. However, instead of (9.78), we observe that

$$\left\|\sum_{i\in I}\mathbf{v}_i\right\|^2 = \left\langle \sum_{i\in I}\mathbf{v}_i, \sum_{i\in I}\mathbf{v}_i\right\rangle = \sum_{i\in I}\langle\mathbf{v}_i,\mathbf{v}_i\rangle + \sum_{i\neq j}\langle\mathbf{v}_i,\mathbf{v}_j\rangle, \tag{9.80}$$

where the last sum is over all $i \neq j$ with i, $j \in I$. The result now follows from the hypotheses on the $\|\mathbf{v}_i\|$ and on the $\langle\mathbf{v}_i,\mathbf{v}_j\rangle$. Indeed, the right-hand side of (9.80) is at most

$$an + 2\binom{a}{2}\delta n \leq a^2\delta n\left(1 + \frac{1}{a\delta}\right) \leq 2a^2\delta n. \tag{9.81}$$

Therefore, the right-hand side of (9.77) is at most $\sqrt{2a^2b\delta n} \leq n^2\sqrt{2\delta}$. Theorem 43 follows. □

Before we proceed, let us observe that, in fact, the hypothesis of δ-quasi-orthogonality of the $\mathbf{v}_i$ $(1 \leq i \leq n)$ in Theorem 43 may be further weakened to

$$\langle\mathbf{v}_i,\mathbf{v}_j\rangle \leq \delta n \text{ for all } i \neq j \text{ with } 1 \leq i,\ j \leq n. \tag{9.82}$$

Indeed, hypothesis (9.82) above suffices for us to estimate the last sum in (9.80). The reader may also observe that, in fact, it suffices to require that the inequality in (9.82) should hold for *most* pairs $\{i, j\}$ with $i \neq j$, with little loss in the conclusion (9.79). We omit the details.

Finally, let us observe that Theorem 41 concerns matrices in which the number of $+1$s is about the same as the number of -1s, and hence the average entry is about 0. In general, we shall be interested in the case in which the rows of our matrix are pairwise quasi-orthogonal (or, more generally, the rows satisfy (9.82)), the average entry is about 0, but the entries are not necessarily ± 1. Adapting carefully the argument in the proof of Theorem 41 to this more general case gives Lemma 45, to be discussed in the Section 9.6.3.

9.6.2 The converse

In the previous section, we proved that the pairwise orthogonality of the rows of a $\{\pm 1\}$-matrix has as a somewhat unexpected consequence the fact that the matrix must have small discrepancy. In this section, we prove that $o(n^2)$ discrepancy for an n by n matrix implies the existence of only $o(n^2)$ pairs of rows that are 'substantially' non-orthogonal (in fact, we prove that the number of pairs $\{i, j\}$ violating the condition in (9.82) is $o(n^2)$). Thus, roughly speaking, we shall prove the converse of the results in Section 9.6.1.

Theorem 44 *Let $\delta > 0$ be a real number and let $H = (h_{ij})$ be an n by n matrix with entries in $\{\pm 1\}$. Let the rows of H be $\mathbf{v}_i$ $(1 \leq i \leq n)$. Let D be the graph on $V = V(D) = [n] = \{1, \ldots, n\}$ whose edges are the pairs $\{i, j\}$ $(i \neq j)$ for which we have*

$$\langle \mathbf{v}_i, \mathbf{v}_j \rangle > \delta n. \tag{9.83}$$

If D is such that $e(D) = |E(D)| \geq \delta n^2$, then

$$\operatorname{disc}(H) > \frac{1}{2}\delta^2 n^2. \tag{9.84}$$

Before we give the proof of Theorem 44, let us give the underlying argument in its simplest form. Let us suppose that we have the following convenient set-up:

$$\mathbf{v}_1 = \mathbf{1} = (1, \ldots, 1) \in \mathbb{R}^n \tag{9.85}$$

and

$$\langle \mathbf{v}_1, \mathbf{v}_i \rangle = \Omega(n). \tag{9.86}$$

for all $i \in I$, where $|I| = \Omega(n)$. Clearly, we may restate (9.86) by saying that all the vectors $\mathbf{v}_i$ $(i \in I)$ have a 'surplus' of $+1$s of order $\Omega(n)$. Since we have $|I| = \Omega(n)$ such vectors $\mathbf{v}_i$, if we sum all the entries of these $\mathbf{v}_i$ we obtain a discrepancy of $\Omega(n^2)$.

To prove Theorem 44, we concentrate our attention on a vertex of high degree in D, and we consider a subset of the columns of H so that the simplifying hypothesis (9.85) holds.

Proof. (Proof of Theorem 44) We start by noticing that the average degree of D is $\geq 2\delta n$. Let $i_0 \in [n] = V(D)$ be a vertex of D with degree $\geq 2\delta n$. We let I be the neighbourhood $\Gamma(i_0)$ of i_0 in D. Therefore,

$$|I| \geq 2\delta n. \tag{9.87}$$

For $\alpha \in \{+,-\}$, let

$$J_\alpha = \{j \in [n] \colon h_{i_0 j} = \alpha\}. \tag{9.88}$$

Clearly, we have $\mathbf{v}_{i_0} = \mathbf{1}_{J_+} - \mathbf{1}_{J_-}$, where we write $\mathbf{1}_S$ for the characteristic vector of a set S. For any $i \in [n]$ and $\alpha \in \{+,-\}$, let $\mathbf{v}_i^\alpha$ be the restriction of $\mathbf{v}_i = (h_{ij})_{1\leq j\leq n}$ to J_α, that is,

$$\mathbf{v}_i^\alpha = (h_{ij})_{j\in J_\alpha}. \tag{9.89}$$

For any $i \in I = \Gamma(i_0)$, we have

$$\left\langle \mathbf{v}_{i_0}^+, \mathbf{v}_i^+ \right\rangle + \left\langle \mathbf{v}_{i_0}^-, \mathbf{v}_i^- \right\rangle = \langle \mathbf{v}_{i_0}, \mathbf{v}_i \rangle > \delta n.$$

Therefore, either

$$\left\langle \mathbf{1}_{J_+}, \mathbf{v}_i^+ \right\rangle = \left\langle \mathbf{v}_{i_0}^+, \mathbf{v}_i^+ \right\rangle > \frac{1}{2}\delta n, \tag{9.90}$$

or else

$$\left\langle -\mathbf{1}_{J_-}, \mathbf{v}_i^- \right\rangle = \left\langle \mathbf{v}_{i_0}^-, \mathbf{v}_i^- \right\rangle > \frac{1}{2}\delta n. \tag{9.91}$$

Let

$$I_+ = \{i \in I = \Gamma(i_0) \colon (9.90) \text{ holds}\}, \tag{9.92}$$

and let

$$I_- = \{i \in I = \Gamma(i_0) \colon (9.91) \text{ holds}\}. \tag{9.93}$$

Clearly, $I = \Gamma(i_0) = I_+ \cup I_-$ and hence

$$\max\{|I_+|, |I_-|\} \geq \frac{1}{2}|I| \geq \delta n, \tag{9.94}$$

where we used (9.87). Let us now put $S_\alpha = \sum h_{ij}$ for $\alpha \in \{+,-\}$, where the sum runs over all $i \in I_\alpha$ and $j \in J_\alpha$. Observe that then

$$S_+ = \sum \left\{ h_{ij} \colon i \in I_+,\, j \in J_+ \right\} = \left\langle \sum_{i\in I_+} \mathbf{v}_i^+, \mathbf{1}_{J_+} \right\rangle > \frac{1}{2}\delta n |I_+|, \tag{9.95}$$

and

$$S_- = \sum \left\{ h_{ij} \colon i \in I_-,\, j \in J_- \right\} = \left\langle \sum_{i \in I_-} \mathbf{v}_i^-, \mathbf{1}_{J_-} \right\rangle < -\frac{1}{2}\delta n |I_-|, \tag{9.96}$$

where the inequalities follow from (9.90) and (9.91). We now observe that (9.94) gives that

$$\mathrm{disc}(H) \geq \max\{|S_+|, |S_-|\} > \max\{|I_+|, |I_-|\}\frac{1}{2}\delta n \geq \frac{1}{2}\delta^2 n^2, \tag{9.97}$$

which completes the proof. □

We shall discuss the 'full' version of Theorem 44 above in Section 9.6.3.

9.6.3 The general results

We now state the 'general versions' of the results in Sections 9.6.1 and 9.6.2. We follow [19]. The results in this section are stated for graphs instead of matrices.

The sufficiency of the condition

Recall that we write $\Gamma(x) = \Gamma_G(x)$ for the neighbourhood of a vertex x in a graph G. Moreover, if $B \subset V(G)$ is a subset of vertices of our graph G, we write $d_B(x)$ for the degree $|\Gamma(x) \cap B|$ of x into B, and, similarly, we write $d_B(x, x')$ for the 'joint degree' $|\Gamma(x) \cap \Gamma(x') \cap B|$ of x and x' into B.

We now state the 'full' version of Theorem 43 (see also the comment concerning the weaker hypothesis (9.82)).

Theorem 45 *Let ε be a constant with $0 < \varepsilon < 1$. Let $G = (V, E)$ be a graph with (A, B) a pair of disjoint, nonempty subsets of V with $|A| \geq 2/\varepsilon$. Set $\varrho = d(A, B) = e(A, B)/|A||B|$. Let D be the collection of all pairs $\{x, x'\}$ of vertices of A for which*

(i) $d_B(x)$, $d_B(x') > (\varrho - \varepsilon)|B|$,

(ii) $d_B(x, x') < (\varrho + \varepsilon)^2|B|$.

Then if $|D| > (1/2)(1 - 5\varepsilon)|A|^2$, the pair (A, B) is $(16\varepsilon)^{1/5}$-regular.

We only give a brief sketch for the proof of Theorem 45 here. The first step is to construct an A by B 'adjacency' matrix M, whose entries are -1 and $\lambda = (1 - \varrho)/\varrho$. A -1 entry indicates the absence of the edge and the entry λ indicates the presence of the edge. It is not difficult to check that the discrepancy of this matrix M is tightly connected with the regularity of the pair (A, B). Indeed, we have

$$\mathrm{disc}(M; a', b') = \frac{1}{\varrho} \max_{A', B'} \left| e(A', B') - \varrho |A'||B'| \right|, \tag{9.98}$$

where the maximum is taken over all $A' \subset A$ and $B' \subset B$ with $|A'| = a'$ and $|B'| = b'$. On the other hand, by making use of the hypothesis on D, a careful application of Lindsey's argument gives that

$$\mathrm{disc}(M; a', b') \leq \frac{1}{\varrho}(16\varepsilon)^{1/5} a'b' \tag{9.99}$$

for all $a' \geq \varepsilon|A|$ and $b' \geq \varepsilon|B|$. Theorem 45 follows from (9.98) and (9.99). See [19] for the details.

The necessity of the condition

We now turn to the converse of Theorem 45. The 'full' version of Theorem 44 is as follows.

Theorem 46 *Let $G = (V, E)$ be a graph with (A, B) an ε-regular pair of disjoint, nonempty subsets of V, having density $d(A, B) = e(A, B)/|A||B| = \varrho$, where $\varrho|B| \geq 1$ and $0 < \varepsilon < 1$. Then*

(i) all but at most $2\varepsilon|A|$ vertices $x \in A$ satisfy

$$(\varrho - \varepsilon)|B| < d_B(x),\ d_B(x') < (\varrho + \varepsilon)|B|,$$

(ii) all but at most $2\varepsilon|A|^2$ pairs $\{x, x'\}$ of vertices of A satisfy

$$d_B(x, x') < (\varrho + \varepsilon)^2|B|.$$

Theorem 46 may be proved by adapting the proof of Theorem 44. See [19] for the details.

9.6.4 Algorithmic versions

Let us briefly discuss some algorithmic aspects. The reader is referred to [37] for a survey.

In algorithmic applications of regularity, once an ε-regular partition is obtained, one typically makes use of constructive versions of results such as the embedding lemma, Lemma 25. The reader will have no difficulty in observing that an efficient algorithm is implied in the proof of Lemma 25.

The question is, then, whether ε-regular partitions may be constructed efficiently. It turns out that this is indeed the case [4, 5]. The main tool to prove this is the local characterization of regularity that we have been discussing in this section. In fact, Theorems 45 and 46 imply Lemma 47 below (see [4, 5, 19]), which is the key ingredient of the constructive version of the regularity lemma given in [4, 5].

Recall that a bipartite graph $B = (U, W; E)$ with vertex classes U and W and edge set E is said to be ε-regular if (U, W) is an ε-regular pair with respect to B. Thus, a witness to the ε-irregularity of B is a pair (U', W') with $U' \subset U$, $W' \subset W$, $|U'|$, $|W'| \geq \varepsilon n$, and $|d_B(U', W') - d_B(U, W)| > \varepsilon$. Below, we write $M(n)$ for the time required to square an $n \times n$ matrix with entries in $\{0, 1\}$ over the integers. By a result of Coppersmith and Winograd [17], we have $M(n) = O(n^{2.376})$. (We leave it as an easy exercise for the reader to see how matrix multiplication comes into play here; without fast matrix multiplication, we would have an algorithm with running time $O(n^3)$ in Lemma 47 below.)

Lemma 47 *There exists an algorithm $\mathcal{A}$ for which the following holds. When $\mathcal{A}$ receives as input an $\varepsilon > 0$ and a bipartite graph $B = (U, W; E)$ with $|U| = |W| = n \geq (2/\varepsilon)^5$, it either correctly asserts that B is ε-regular, or else it returns a witness for the ε'-irregularity of B, where $\varepsilon' = \varepsilon'_{\mathcal{A}}(\varepsilon) = \varepsilon^5/16$. The running time of $\mathcal{A}$ is $O(M(n))$.*

Note that Lemma 47 leaves open what the behaviour of $\mathcal{A}$ should be when B is ε-regular but is not ε'-regular. Despite this fact, Lemma 47 does indeed imply the existence of a polynomial-time algorithm for finding ε-regular partitions of graphs. A moment's thought should make it clear that what is required is an algorithmic version of Lemma 40. Lemma 47 readily provides such a result. We leave the proof of this assertion as an exercise for the reader.

Summing up the results discussed so far, we have the following theorem, which is an algorithmic version of Szemerédi's regularity lemma [4, 5].

Theorem 48 *There is a deterministic algorithm $\mathcal{B}$ and functions $K_0(\varepsilon, k_0)$ and $N_0(\varepsilon, k_0)$ for which the following holds. On input $G = G^n$, $0 < \varepsilon \leq 1$, and $k_0 \geq 1$, where $n \geq N_0(\varepsilon, k_0)$, algorithm $\mathcal{B}$ returns an ε-regular, (ε, k)-equitable partition for G in time $O(M(n))$, where $k_0 \leq k \leq K_0(\varepsilon, k_0)$.*

Let us observe that the constant implied in the big O notation in Theorem 48 depends on ε and k_0.

In [41], we shall show how to improve on the running time given in Lemma 47 (at the cost of decreasing the value of $\varepsilon' = \varepsilon'(\varepsilon)$ substantially). The key idea is to make use of the quasi-random property to be discussed in Section 9.7. The algorithm for constructing ε-regular partitions given in [41] has running time $O(n^2)$ for graphs of order n, where, again, the implicit constant depends on ε and k_0.

The algorithms we have discussed so far are all deterministic. If one allows randomization, one may develop algorithms that run in $O(n)$ time, as shown by Frieze and Kannan [27, 28].

A coNP-completeness result

The reader may find it unsatisfactory that, strictly speaking, we did not solve the problem of characterizing precisely the ε-regular pairs. Indeed, Lemma 47 can only tell the difference between $\varepsilon'_{\mathcal{A}}(\varepsilon)$-regular pairs and ε-irregular pairs, and $\varepsilon'_{\mathcal{A}}(\varepsilon) \ll \varepsilon$. This is, by no means, an accident. Consider the decision problem below.

Problem 49 *Given a graph G, a pair (U, W) of non-empty, pairwise disjoint sets of vertices of G, and a positive ε, decide whether this pair is ε-regular with respect to G.*

It should be clear that, in the case in which the answer to Problem 49 is negative for a given instance, we would like to have a witness for the ε-irregularity of the given pair. Indeed, an algorithm that is able to solve Problem 49 and is also able to provide such a witness in the case in which the answer is negative would prove Lemma 47 with $\varepsilon' = \varepsilon$. Unfortunately, such an algorithm does not exist, unless P = NP, as shows the following result of Alon, Duke, Lefmann, Rödl, and Yuster [4, 5].

Theorem 50 *Problem 49 is coNP-complete.*

Let us remark in passing that Theorem 50 is proved in [4, 5] for the case in which $\varepsilon = 1/2$; for a proof for arbitrary $0 < \varepsilon \le 1/2$, see Taraz [61].

9.6.5 The sparse case

As proved in [38], Theorems 45 and 46 do not generalize to graphs of vanishing density. However, in view of the applicability of those results, it seems worth pursuing the sparse case. In [38], we prove that natural generalizations of Theorems 45 and 46 *do hold* for subgraphs of sparse random graphs. Examples of applications of these generalizations appear in [3] (cf. Theorem 1.5) and [40]. We do not go into the details here.

9.7 A new quasi-random property

In this section, we present a new quasi-random graph property, in the sense of Chung, Graham, and Wilson [15]. In the introduction and in Section 9.3.2, we very briefly discussed the basics of quasi-randomness, and mentioned the close relationship between quasi-randomness, ε-regularity, and the regularity lemma as a strong motivation for studying quasi-random graph properties.

In Section 9.6, we discussed 'local' conditions for regularity, and observed that these conditions were the key for developing a $O(n^{2.376})$-time algorithm that checks whether a given bipartite graph is regular (see Lemma 47). In turn, this led to a $O(n^{2.376})$-time algorithm for finding regular partitions of graphs. The quasi-random property that we present in this section allows one to check regularity, somewhat surprisingly, in time $O(n^2)$. Since we deal with dense input graphs, this running time is proportional to the input size, and hence we have a *linear time algorithm.* (The corresponding linear time algorithm for finding regular partitions of graphs, which is based on some additional ideas, will be presented in [41].)

The proof of the fact that our property is indeed a quasi-random property will make use of the sparse regularity lemma, Theorem 15. To simplify the notation, we restrict our discussion to the case of graphs with density $\sim 1/2$. Moreover, we deal with quasi-randomness and arbitrary graphs, instead of regularity and bipartite graphs. We hope that the reader finds the correspondence between these two contexts clear.

9.7.1 Basic definitions

We start with the definition of a standard quasi-random graph property.

Definition 51 ($(1/2,\varepsilon,\delta)$-quasi-randomness) *Let reals $0<\varepsilon\leq 1$ and $0<\delta\leq 1$ be given. We shall say that a graph G is $(1/2,\varepsilon,\delta)$-quasi-random if, for all U, $W\subset V(G)$ with $U\cap W=\emptyset$ and $|U|$, $|W|\geq\delta n$, we have*

$$\left|e_G(U,W)-\frac{1}{2}|U||W|\right|\leq\frac{1}{2}\varepsilon|U||W|. \tag{9.100}$$

Before we proceed, we need to introduce a technical definition concerning graphs with uniformly distributed edges.

Definition 52 ((ϱ,A)-uniformity) *If $0<\varrho\leq 1$ and A are reals, we say that an n-vertex graph $J=J^n$ is (ϱ,A)-uniform if, for all U, $W\subset V(J)$ with $U\cap W=\emptyset$, we have*

$$\big|e_J(U,W)-\varrho|U||W|\big|\leq A\sqrt{r|U||W|}, \tag{9.101}$$

where $r=\varrho n$.

As it will become clear later, we shall be mainly concerned with (ϱ,A)-uniform graphs J with *constant* average degree, that is, graphs $J=J^n$ with $O(n)$ edges. The construction of such (ϱ,A)-uniform graphs $J=J^n$ with linearly many edges will be briefly discussed in Section 9.7.3.

In the sequel, when dealing with a (ϱ,A)-uniform graph $J=J^n$, we usually write r for ϱn. Let us remark for later reference that the following fact, whose simple proof will be given in Section 9.7.3, holds.

Fact 53 *If J is a (ϱ,A)-uniform graph, then, for any $U\subset V(J)$, we have*

$$\left|e_J(U)-\varrho\binom{|U|}{2}\right|\leq A\sqrt{r}|U|. \tag{9.102}$$

We shall now define a property for n-vertex graphs $G=G^n$, based on a fixed (ϱ,A)-uniform graph $J=J^n$ with the same vertex set as G. Below, we write $ij\in J$ to mean that ij is an edge of the graph J. We recall that we denote the neighbourhood of a vertex x in a graph G by $\Gamma(x)=\Gamma_G(x)$, and we write $X\bigtriangleup Y$ for the symmetric difference of X and Y.

Definition 54 (Property $P_{J,\triangle}(\varepsilon)$) *Let $G = G^n$ and $J = J^n$ be n-vertex graphs on the same vertex set. Let $0 < \varepsilon \leq 1$ be a real number. We say that G satisfies property $P_{J,\triangle}(\varepsilon)$ if we have*

$$\sum_{ij \in J} \left| |\Gamma_G(i) \triangle \Gamma_G(j)| - \frac{1}{2}n \right| \leq \frac{1}{2}\varepsilon n e(J). \tag{9.103}$$

Our new quasi-random property is $P_{J,\triangle}(\varepsilon)$ above. It should be now clear why it is interesting for us to have (ϱ, A)-uniform graphs J with as few edges as possible: the number of terms in the sum in (9.103) is $e(J)$. Since each term of that sum may be computed in $O(n)$ time if, say, we have access to the adjacency matrix of G, it follows that the time required to verify property $P_{J,\triangle}(\varepsilon)$ is $O(ne(J))$, which is $O(n^2)$ if we have linear-sized (ϱ, A)-uniform graphs J.

For technical reasons, we need to introduce a variant of property $P_{J,\triangle}(\varepsilon)$.

Definition 55 (Property $P'_{J,\triangle}(\gamma, \varepsilon)$) *Let $G = G^n$ and $J = J^n$ be n-vertex graphs on the same vertex set. Let $0 < \gamma \leq 1$ and $0 < \varepsilon \leq 1$ be two real numbers. We shall say that G satisfies property $P'_{J,\triangle}(\gamma, \varepsilon)$ if the inequality*

$$\left| |\Gamma_G(i) \triangle \Gamma_G(j)| - \frac{1}{2}n \right| \leq \frac{1}{2}\varepsilon n \tag{9.104}$$

fails for at most $\gamma e(J)$ edges $ij \in J$ of J.

As a quick argument will show, properties $P_{J,\triangle}(\varepsilon)$ and $P'_{J,\triangle}(\gamma, \varepsilon)$ are equivalent under suitable assumptions on the parameters; see Lemma 60.

Our main result in Section 9.7 is that, roughly speaking, properties $P_{J,\triangle}(o(1))$ and $P'_{J,\triangle}(o(1), o(1))$ are equivalent to $(1/2, o(1), o(1))$-quasi-randomness. We make the form of this equivalence precise in the next section.

9.7.2 The equivalence result

Theorems 56 and 57 below are the main results of Section 9.7. Intuitively, Theorem 56 states that property $P_{J,\triangle}(o(1))$ is a *sufficient* condition for $(1/2, o(1), o(1))$-quasi-randomness, whereas Theorem 57 states that $P'_{J,\triangle}(o(1), o(1))$ is a *necessary* condition. Lemma 60 tells us that $P_{J,\triangle}(o(1))$ and $P'_{J,\triangle}(o(1), o(1))$ are equivalent.

Theorem 56 *For any $0 < \varepsilon \leq 1$, $0 < \delta \leq 1$, and $A \geq 1$, there exist $\varepsilon_0 = \varepsilon_0(\varepsilon, \delta, A) > 0$ and $r_0 = r_0(\varepsilon, \delta, A) \geq 1$ for which the following holds. Suppose $G = G^n$ and $J = J^n$ are two graphs on the same vertex set. Suppose further that $J = J^n$ is a (ϱ, A)-uniform graph with $r = \varrho n \geq r_0$. Then, if G satisfies property $P_{J,\triangle}(\varepsilon')$ for some $0 < \varepsilon' \leq \varepsilon_0$, then G is $(1/2, \varepsilon, \delta)$-quasi-random.*

Theorem 57 *For any $0 < \gamma \le 1$, $0 < \varepsilon \le 1$, and $A \ge 1$, there exist $\varepsilon_0 = \varepsilon_0(\gamma, \varepsilon, A) > 0$, $\delta_0 = \delta_0(\gamma, \varepsilon, A) > 0$, $r_1 = r_1(\gamma, \varepsilon, A) \ge 1$, and $N_1 = N_1(\gamma, \varepsilon, A) \ge 1$ for which the following holds. Suppose $G = G^n$ and $J = J^n$ are two graphs on the same vertex set, with $n \ge N_1$. Suppose further that $J = J^n$ is a (ϱ, A)-uniform graph with $r = \varrho n \ge r_1$. Then, if G is $(1/2, \varepsilon', \delta')$-quasi-random for some $0 < \varepsilon' \le \varepsilon_0$ and $0 < \delta' \le \delta_0$, then property $P'_{J,\triangle}(\gamma, \varepsilon)$ holds for G.*

Remark 58 *As our previous discussion suggests, it is of special relevance to us that in Theorems 56 and 57 the quantity $r = \varrho n$ is not required to grow with n.*

Remark 59 *We remark that Theorems 56 and 57 basically reduce to the results in Sections 9.6.1–9.6.3 if we take $J = J^n$ to be the complete graph K^n.*

Lemma 60 *Let a (ϱ, A)-uniform graph $J = J^n$ be given, and suppose $G = G^n$ is a graph on the same vertex set as J. Then the following assertions hold.*

(*i*) *If G satisfies property $P'_{J,\triangle}(\gamma, \varepsilon)$, then G satisfies property $P_{J,\triangle}(\varepsilon + \gamma)$.*

(*ii*) *If G satisfies property $P_{J,\triangle}(\varepsilon)$ and $0 < \varepsilon \le \varepsilon' \le 1$, then G satisfies property $P'_{J,\triangle}(\varepsilon/\varepsilon', \varepsilon')$.*

We shall prove Theorems 56 and 57 in two separate sections below. Here, we give the simple proof of Lemma 60.

Proof. (Proof of Lemma 60) Let $J = J^n$ and $G = G^n$ be as in the statement of Lemma 60. Suppose first that G has property $P'_{J,\triangle}(\gamma, \varepsilon)$. Then

$$\sum_{ij \in J} \left| |\Gamma_G(i) \triangle \Gamma_G(j)| - \frac{1}{2} n \right| \le \frac{1}{2} \varepsilon n e(J) + \frac{1}{2} n \gamma e(J) = \frac{1}{2} (\varepsilon + \gamma) n e(J). \tag{9.105}$$

Therefore property $P_{J,\triangle}(\varepsilon + \gamma)$ holds and (*i*) is proved. To prove (*ii*), suppose that G satisfies $P_{J,\triangle}(\varepsilon)$ and $0 < \varepsilon \le \varepsilon' \le 1$. If $P'_{J,\triangle}(\varepsilon/\varepsilon', \varepsilon')$ were to fail, then we would have $> (\varepsilon/\varepsilon') e(J)$ edges ij of J with

$$\left| |\Gamma_G(i) \triangle \Gamma_G(j)| - \frac{1}{2} n \right| > \frac{1}{2} \varepsilon' n. \tag{9.106}$$

But then

$$\sum_{ij \in J} \left| |\Gamma_G(i) \triangle \Gamma_G(j)| - \frac{1}{2} n \right| > \frac{1}{2} \varepsilon' n \times \frac{\varepsilon}{\varepsilon'} e(J) = \frac{1}{2} \varepsilon n e(J), \tag{9.107}$$

which contradicts $P_{J,\triangle}(\varepsilon)$. Thus $P'_{J,\triangle}(\varepsilon/\varepsilon', \varepsilon')$ must hold, and (*ii*) is proved. □

9.7.3 The existence of (ϱ, A)-uniform graphs

As promised before, in this section we discuss the construction of suitable (ϱ, A)-uniform graphs $J = J^n$ with linearly many edges. We state the following result without proof.

Lemma 61 *There exist absolute constant r_0 and n_0 for which the following holds. Let $r \geq r_0$ be a constant and let $n \geq n_0$ be given. Then we may explicitly construct an adjacency list representation of a particular $(\varrho, 5)$-uniform graph $J = J^n$ on $V(J) = [n]$ with $r \leq \varrho n \leq 2r$ in time $O(n(\log n)^{O(1)})$.*

Lemma 61 may be deduced in a straightforward manner from the celebrated construction of the Ramanujan graphs $X^{p,q}$ of Lubotzky, Phillips, and Sarnak [51] (see also [49, 50, 59]). We mention in passing that, for proving the existence of suitable parameters p and q in the proof of Lemma 61, it suffices to use Dirichlet's theorem on the density of primes in arithmetic progressions. We omit the details (see [41]).

We also promised to prove Fact 53 in this section.

Proof. (Proof of Fact 53) We may clearly assume that $u = |U| \geq 2$. Note that, for any $1 \leq s < u$, we have $2e(U)\binom{u-2}{s-1} = \sum_S e(S, U \setminus S)$, where the sum is extended over all $S \subset U$ with $|S| = s$. Thus

$$e(U) = \frac{1}{2}\binom{u}{s}\binom{u-2}{s-1}^{-1}\left\{\varrho|S||U \setminus S| + O_1\left(A\sqrt{rs(u-s)}\right)\right\} \quad (9.108)$$

for any $1 \leq s < u$. We use (9.108) with $s = \lfloor u/2 \rfloor$. Note that

$$\binom{u}{\lfloor u/2 \rfloor}\binom{u-2}{\lfloor u/2 \rfloor - 1}^{-1} = \frac{u(u-1)}{\lfloor u/2 \rfloor \lceil u/2 \rceil} \leq 4, \quad (9.109)$$

and so

$$e(U) = \varrho\binom{u}{2} + O_1\left(2A\sqrt{r\lfloor u/2 \rfloor \lceil u/2 \rceil}\right) = \varrho\binom{u}{2} + O_1\left(Au\sqrt{r}\right), \quad (9.110)$$

as required. □

In the next two sections, we prove Theorems 56 and 57.

9.7.4 Proof of Theorem 56

Let constants $0 < \varepsilon \leq 1$, $0 < \delta \leq 1$, and $A \geq 1$ be given. We then put

$$\varepsilon_0 = \varepsilon_0(\varepsilon, \delta, A) = \frac{1}{4}\varepsilon^2\delta^3 \quad \text{and} \quad r_0 = r_0(\varepsilon, \delta, A) = 2^6 A^2 \varepsilon^{-4}\delta^{-4}. \quad (9.111)$$

For later reference, let us observe that

$$\frac{A}{2\sqrt{r_0}} \leq \frac{1}{16} < \frac{1}{4}, \quad (9.112)$$

and that

$$\frac{A}{\sqrt{r_0}} = \frac{1}{8}\varepsilon^2\delta^2. \tag{9.113}$$

Our aim is to show that the values of ε_0 and r_0 given in (9.111) will do in Theorem 56. Thus, suppose we are given a graph $G = G^n$ and a (ϱ, A)-uniform graph $J = J^n$ on the same vertex set, say V, and suppose further that G satisfies property $P_{J,\triangle}(\varepsilon')$, where $0 < \varepsilon' \le \varepsilon_0$, and $r = \varrho n \ge r_0$. We have to show that G is $(1/2, \varepsilon, \delta)$-quasi-random.

In what follows, we assume that two disjoint sets U, $W \subset V$ with $|U|$, $|W| \ge \delta n$ are given. We wish to show that inequality (9.100) holds. The approach we take is similar in spirit to the one used in the proof of Theorem 41.

Let $\mathbf{A} = (a_{ij})_{i,j\in V}$ be the adjacency matrix of G with entries in $\{\pm 1\}$, with $a_{ij} = 1$ if $ij \in G$ and $a_{ij} = -1$ if $ij \notin G$. Let us write $\mathbf{v}_i = (a_{ij})_{j\in V}$ $(i \in V)$ for the ith row of $\mathbf{A}$. We start by observing that property $P_{J,\triangle}(\varepsilon')$ implies that $\sum_{ij\in J} |\langle \mathbf{v}_i, \mathbf{v}_j\rangle|$ is small.

Lemma 62 *We have*

$$\sum_{ij\in J} |\langle \mathbf{v}_i, \mathbf{v}_j\rangle| \le \varepsilon' n e(J). \tag{9.114}$$

Proof. By the definition of the $\mathbf{v}_i$, we have

$$\langle \mathbf{v}_i, \mathbf{v}_j\rangle = n - 2|\Gamma_G(i) \triangle \Gamma_G(j)|,$$

and the result follows from the definition of property $P_{J,\triangle}(\varepsilon')$. □

Our aim now is to estimate the left-hand side of (9.114) from below. It turns out that one may give a good lower bound for this quantity in terms of the number of G-edges $e_G(U, W)$ between U and $W \subset V$ for any pair (U, W) as long as both U and W are large enough.

Recall that sets U, $W \subset V$ with $u = |U|$, $w = |W| \ge \delta n$ are fixed, and put $\mathbf{w}_i = (a_{ij})_{j\in W}$ for all $i \in U$. Thus, $\mathbf{w}_i$ is the restriction of $\mathbf{v}_i$ to the coordinates in W. For convenience, we shall write $\sum_{ij\in J}^U$ to indicate sum over all edges $ij \in J$ with both i and j in U.

Let us compare $\sum_{ij\in J}^U \langle \mathbf{v}_i, \mathbf{v}_j\rangle$ and $\sum_{ij\in J}^U \langle \mathbf{w}_i, \mathbf{w}_j\rangle$. Clearly,

$$\sum\nolimits_{ij\in J}^U \langle \mathbf{v}_i, \mathbf{v}_j\rangle = \sum\nolimits_{ij\in J}^U \langle \mathbf{w}_i, \mathbf{w}_j\rangle + \sum_{k\in V\setminus W} \sum\nolimits_{ij\in J}^U a_{ik}a_{jk}. \tag{9.115}$$

In the lemma below, we estimate $S_k^U = \sum_{ij\in J}^U a_{ik}a_{jk}$ for all $k \in V$. Recall that we write $O_1(x)$ for any term y satisfying $|y| \le x$.

Lemma 63 *Fix a vertex $k \in V$, and let $u = |U|$, $u^+ = u_k^+ = |\Gamma_G(k) \cap U|$, and $u^- = u_k^- = |U \setminus \Gamma_G(k)|$. Then*

$$S_k^U = \sum\nolimits_{ij\in J}^U a_{ik}a_{jk} = \frac{1}{2}\varrho\left((u^+ - u^-)^2 - u\right) + O_1\left(\frac{3}{2}Au\sqrt{r}\right). \tag{9.116}$$

In particular, we have

$$S_k^U \geq \frac{1}{2}\varrho(u^+ - u^-)^2 - 2Au\sqrt{r} \geq -2Au\sqrt{r}. \tag{9.117}$$

Proof. Note that an edge $ij \in J$ contributes $+1$ to the sum in (9.116) if $i, j \in \Gamma_G(k) \cap U$ or else $i, j \in U \setminus \Gamma_G(k)$. Similarly, the edge $ij \in J$ contributes -1 to that sum if $ij \in E\big(\Gamma_G(k) \cap U, U \setminus \Gamma_G(k)\big)$.

By the (ϱ, A)-uniformity of J (see also (9.102) in Fact 53), we have

$$\begin{aligned} S_k^U &= \sum\nolimits_{ij\in J}^{U} a_{ik}a_{jk} \\ &= \varrho\binom{u^+}{2} + O_1\left(Au^+\sqrt{r}\right) + \varrho\binom{u^-}{2} + O_1\left(Au^-\sqrt{r}\right) \\ &\quad - \varrho u^+u^- + O_1\left(A\sqrt{ru^+u^-}\right) \\ &= \varrho\left(\frac{1}{2}(u^+)^2 + \frac{1}{2}(u^-)^2 - u^+u^- - \frac{1}{2}(u^+ + u^-)\right) \\ &\quad + O_1\left(A\sqrt{r}(u^+ + u^- + \sqrt{u^+u^-})\right), \end{aligned}$$

from which (9.116) follows.

Since $A \geq 1$ and $r < n$, we have $\varrho u/2 \leq (1/2)Au\sqrt{r}$. Therefore, the right-hand side of (9.116) is at least

$$\frac{1}{2}\varrho(u^+ - u^-)^2 - \frac{1}{2}\varrho u - \frac{3}{2}Au\sqrt{r} > \frac{1}{2}\varrho(u^+ - u^-)^2 - 2Au\sqrt{r}. \tag{9.118}$$

Inequality (9.117) follows from (9.118) and Lemma 63 is proved. □

An immediate corollary to (9.115) and (9.117) is that

$$\sum\nolimits_{ij\in J}^{U} \langle \mathbf{v}_i, \mathbf{v}_j\rangle \geq \sum\nolimits_{ij\in J}^{U} \langle \mathbf{w}_i, \mathbf{w}_j\rangle - 2A(n-w)u\sqrt{r}, \tag{9.119}$$

where, as before, $w = |W|$. We now estimate $\sum\nolimits_{ij\in J}^{U} \langle \mathbf{w}_i, \mathbf{w}_j\rangle$ from below using Lemma 63. Put

$$u_*^+ = \operatorname*{Ave}_{k\in W} u_k^+ = \operatorname*{Ave}_{k\in W} |\Gamma_G(k) \cap U| = \frac{1}{w}e_G(U, W), \tag{9.120}$$

where $\mathrm{Ave}_{k\in W}$ denotes average over all $k \in W$.

Lemma 64 *We have*

$$\sum\nolimits_{ij\in J}^{U} \langle \mathbf{w}_i, \mathbf{w}_j\rangle \geq \frac{1}{2}\varrho w(2u_*^+ - u)^2 - 2Auw\sqrt{r}. \tag{9.121}$$

Proof. We make use of Lemma 63. We have $u_k^+ - u_k^- = 2u_k^+ - u$ for all k. Therefore, inequality (9.117) in Lemma 63 tells us that

$$\sum\nolimits_{ij\in J}^{U} \langle \mathbf{w}_i, \mathbf{w}_j\rangle = \sum_{k\in W}\sum\nolimits_{ij\in J}^{U} a_{ik}a_{jk}$$

$$\begin{aligned} &\geq \frac{1}{2}\varrho\sum_{k\in W}(u_k^+ - u_k^-)^2 - 2Auw\sqrt{r} \\ &= \frac{1}{2}\varrho\sum_{k\in W}(2u_k^+ - u)^2 - 2Auw\sqrt{r}, \end{aligned}$$

which, by convexity (or Cauchy–Schwarz), is at least as large as the right-hand side of (9.121). The proof of this lemma is complete. □

We now put Lemmas 62 and 64 and inequality (9.119) together to obtain

$$\begin{aligned} &\frac{1}{2}\varrho w(2u_*^+ - u)^2 - 2Aun\sqrt{r} \leq \sum\nolimits_{ij\in J}^{U}\langle \mathbf{w}_i, \mathbf{w}_j\rangle - 2Au(n-w)\sqrt{r} \\ &\leq \sum\nolimits_{ij\in J}^{U}\langle \mathbf{v}_i, \mathbf{v}_j\rangle \leq \sum\nolimits_{ij\in J}^{U}|\langle \mathbf{v}_i, \mathbf{v}_j\rangle| \leq \sum_{ij\in J}|\langle \mathbf{v}_i, \mathbf{v}_j\rangle| \leq \varepsilon' n e(J). \quad (9.122) \end{aligned}$$

We now make use of (9.102) in Fact 53 to deduce that

$$e(J) \leq \varrho\binom{n}{2} + An\sqrt{r} \leq \frac{1}{2}rn + An\sqrt{r}. \qquad (9.123)$$

Therefore

$$\varepsilon' n e(J) \leq \frac{1}{2}\varepsilon' r n^2 + \varepsilon' A n^2\sqrt{r}, \qquad (9.124)$$

and hence (9.122) gives that

$$\frac{1}{2}\varrho w(2u_*^+ - u)^2 \leq \frac{1}{2}\varepsilon' r n^2 + \varepsilon' A n^2\sqrt{r} + 2Aun\sqrt{r}. \qquad (9.125)$$

However, we have

$$\frac{1}{2}\varrho w(2u_*^+ - u)^2 = \frac{1}{2}\varrho w\left(\frac{2}{w}e_G(U,W) - u\right)^2 = 2\frac{\varrho}{w}\left(e_G(U,W) - \frac{1}{2}uw\right)^2. \qquad (9.126)$$

From (9.125) and (9.126), we obtain

$$\begin{aligned} \left|e_G(U,W) - \frac{1}{2}uw\right|^2 &\leq \frac{1}{4\varrho}\varepsilon' r n^2 w + \frac{1}{2\varrho}\varepsilon' A n^2 w\sqrt{r} + \frac{1}{\varrho}Auwn\sqrt{r} \\ &= \frac{1}{4}\varepsilon' n^3 w + \frac{1}{2\sqrt{r}}\varepsilon' A n^3 w + \frac{1}{\sqrt{r}}Auwn^2 \\ &= \varepsilon' n^3 w\left(\frac{1}{4} + \frac{A}{2\sqrt{r}}\right) + \frac{A}{\sqrt{r}}n^2 uw. \qquad (9.127) \end{aligned}$$

Using (9.111), (9.112), and (9.113) and the fact that $\varepsilon' \leq \varepsilon_0$ and $r \geq r_0$, we deduce that the last expression in (9.127) is at most

$$\begin{aligned} &\frac{1}{2}\varepsilon' n^3 w + \frac{1}{8}\varepsilon^2\delta^2 n^2 uw \leq \frac{1}{8}\varepsilon^2\delta^3 n^3 w + \frac{1}{8}\varepsilon^2\delta^2 n^2 uw \\ &\qquad \leq \frac{1}{8}\varepsilon^2 u^2 w^2 + \frac{1}{8}\varepsilon^2 u^2 w^2 = \left(\frac{1}{2}\varepsilon uw\right)^2. \qquad (9.128) \end{aligned}$$

Putting together (9.127) and (9.128), we deduce inequality (9.100).

The proof of Theorem 56 is complete.

9.7.5 Proof of Theorem 57

Let constants $0 < \gamma \leq 1$, $0 < \varepsilon \leq 1$, and $A \geq 1$ be given. Let us define the constants $\varepsilon_0 = \varepsilon_0(\gamma, \varepsilon, A)$, $\delta_0 = \delta_0(\gamma, \varepsilon, A)$, $r_1 = r_1(\gamma, \varepsilon, A)$, and $N_1 = N_1(\gamma, \varepsilon, A)$ as follows.

We start by putting

$$\varepsilon_0 = \varepsilon_0(\gamma, \varepsilon, A) = \frac{1}{2^6}\gamma\varepsilon. \tag{9.129}$$

The definitions of δ_0 and r_1 are a little more elaborate. Let

$$\varepsilon'' = \frac{1}{2^6}\gamma\varepsilon \leq \frac{1}{2^6} \tag{9.130}$$

and

$$k_0 = \left\lceil \frac{2^6}{\gamma\varepsilon} \right\rceil, \tag{9.131}$$

and put $D = 2$. Let

$$\eta = \eta(\varepsilon'', k_0, D) > 0 \quad \text{and} \quad K_0 = K_0(\varepsilon'', k_0, D) \geq k_0, \tag{9.132}$$

and $N_0 = N_0(\varepsilon'', k_0, D)$ be the constants whose existence is guaranteed by Theorem 15 for ε'', k_0, and $D = 2$. We may clearly assume that

$$K_0 \geq \frac{1}{2\varepsilon''}. \tag{9.133}$$

We now let

$$\delta_0 = \delta_0(\gamma, \varepsilon, A) = \min\left\{ \frac{1}{2^7}\gamma\varepsilon, \frac{1}{2K_0} \right\}, \tag{9.134}$$

and let

$$r_1 = r_1(\gamma, \varepsilon, A) = \max\left\{ (2AK_0)^2, \left(\frac{A}{\eta}\right)^2 \right\} \tag{9.135}$$

and

$$N_1 = N_1(\gamma, \varepsilon, A) = N_0(\varepsilon'', k_0, D). \tag{9.136}$$

We claim that these choices for ε_0, δ_0, r_1, and N_1 will do in Theorem 57. However, before we start the proof of this claim, let us observe that the constants above obey the following 'hierarchy':

$$\delta_0 \ll \frac{1}{K_0} \leq \frac{1}{k_0} \ll \gamma\varepsilon \tag{9.137}$$

and

$$\varepsilon_0,\ \varepsilon'' \leq \gamma\varepsilon. \tag{9.138}$$

Moreover,

$$r_1 \gg A,\ K_0,\ \frac{1}{\eta} \tag{9.139}$$

so that, in a (ϱ, A)-uniform graph $J = J^n$, the number of edges between two disjoint sets of vertices U and $W \subset V(J)$ is roughly equal to the expected quantity $\varrho|U||W|$, as long as

$$|U|,\ |W| \geq n \min\left\{\frac{1}{2K_0}, \eta\right\} \tag{9.140}$$

(see the proof of (9.143) below for details). The reader may find it useful to keep in mind the above relationship among our constants.

We now start with the proof that the above choices for ε_0, δ_0, r_1, and N_1 work. Let a (ϱ, A)-uniform graph $J = J^n$ with $n \geq N_1$ vertices be fixed and let G be a $(1/2, \varepsilon', \delta')$-quasi-random graph on $V = V(J)$, where $0 < \varepsilon' \leq \varepsilon_0$, $0 < \delta' \leq \delta_0$, and $r \geq r_1$. We shall prove that G has property $P'_{J,\triangle}(\gamma, \varepsilon)$.

Assume for a contradiction that $P'_{J,\triangle}(\gamma, \varepsilon)$ fails for G. Therefore we know that the number of edges $ij \in J$ in J that violate inequality (9.104) is greater than $\gamma e(J)$. Let us assume that the number of edges $ij \in J$ for which we have

$$|\Gamma_G(i) \triangle \Gamma_G(j)| - \frac{1}{2}n < -\frac{1}{2}\varepsilon n \tag{9.141}$$

is larger than $(\gamma/2)e(J)$. The case in which

$$|\Gamma_G(i) \triangle \Gamma_G(j)| - \frac{1}{2}n > \frac{1}{2}\varepsilon n \tag{9.142}$$

occurs for more than $(\gamma/2)e(J)$ edges ij of J is analogous. We let H be the graph on $V = V(J)$ whose edges are the edges $ij \in J$ that satisfy (9.141).

The regularity lemma for sparse graphs implies Lemma 65 below. We shall use the second form of the lemma, Theorem 15, although the first version, Theorem 13, would equally do (with the first version the calculations involved would be slightly longer).

Lemma 65 *The graph H contains an $(\varepsilon'', H, \varrho)$-regular pair (U, W) of ϱ-density $d_{H,\varrho}(U, W)$ at least $\gamma/4$ and with $|U| = |W| = m \geq n/2K_0$.*

Proof. Let $\eta_0 = \min\{1/2K_0, \eta\}$, where η and K_0 are as defined in (9.132). We claim that $H = H^n$ is an $(\eta_0, 2)$-upper-uniform graph with respect to density ϱ, that is, if U, $W \subset V = V(H)$ are disjoint and $|U|$, $|W| \geq \eta_0 n$, then

$$e_H(U, W) \leq 2\varrho|U||W|.$$

Because of the (ϱ, A)-uniformity of $J \supset H$, it suffices to check that

$$A\sqrt{r|U||W|} \leq \varrho|U||W| \tag{9.143}$$

(see (9.101)). However, this follows easily from (9.135) and the fact that $r = \varrho n \geq r_1$.

Having verified that H is $(\eta_0, 2)$-upper-uniform with respect to density ϱ, we may invoke Theorem 15 to obtain an $(\varepsilon'', H, \varrho)$-regular (ε'', k)-equitable partition $(C_i)_0^k$ of the vertex set of H with $k_0 \leq k \leq K_0$. Observe that

$$|C_i| \geq \frac{n}{2K_0} \text{ for all } 1 \leq i \leq k, \tag{9.144}$$

since $|C_0| \leq \varepsilon'' n < n/2$ (see (9.130)). We shall now apply a standard argument to show that we may take for (U, W) some pair (C_i, C_j). We already know from (9.144) that the C_i $(1 \leq i \leq k)$ have large enough cardinality. Put $m = |C_i|$ $(1 \leq i \leq k)$ and observe that

$$\frac{n}{2K_0} \leq m \leq \frac{n}{k}. \tag{9.145}$$

It suffices to prove the following claim to complete the proof of Lemma 65.

Claim 66 *There exist* $1 \leq i < j \leq k$ *for which the pair* (C_i, C_j) *is* $(\varepsilon'', H, \varrho)$*-regular and* $d_{H,\varrho}(C_i, C_j) \geq \gamma/4$.

Proof. Suppose for a contradiction that no pair (C_i, C_j) with $1 \leq i < j \leq k$ is good. Working under this hypothesis, we shall deduce that the number of edges in H is at most $(\gamma/2)e(J)$, which will contradict the definition of the graph H.

Let us turn to the estimation of $e(H)$. There are four types of edges in H: (i) edges that are induced by $(\varepsilon'', H, \varrho)$-regular pairs (C_i, C_j) with $1 \leq i < j \leq k$, (ii) edges that are induced by $(\varepsilon'', H, \varrho)$-irregular pairs (C_i, C_j) with $1 \leq i < j \leq k$, (iii) edges that are induced within the classes C_i $(1 \leq i \leq k)$, that is, edges in $\bigcup_{1 \leq i \leq k} H[C_i]$, and (iv) edges that are incident to the exceptional class C_0. We now estimate the number of edges of each type in turn.

Because of our assumption that no pair (C_i, C_j) will do for our claim, all the $(\varepsilon'', H, \varrho)$-regular pairs (C_i, C_j) with $1 \leq i < j \leq k$ are such that

$$d_{H,\varrho}(U, W) = \frac{e_H(U, W)}{\varrho |U||W|} < \frac{\gamma}{4}. \tag{9.146}$$

Thus, the number of edges of type (i) is

$$< \frac{\gamma}{4} \varrho m^2 \binom{k}{2} \leq \frac{\gamma}{4} \varrho \left(\frac{n}{k}\right)^2 \frac{k^2}{2} = \frac{\gamma}{4} \left(\varrho \frac{n^2}{2}\right). \tag{9.147}$$

We know that H is a $(\eta_0, 2)$-upper-uniform graph with respect to density ϱ, and that the C_i $(1 \leq i \leq k)$ have cardinality $m \geq (1/2K_0)n \geq \eta_0 n$. Therefore the number of edges induced by a pair (C_i, C_j) with $1 \leq i < j \leq k$ is at most $2\varrho m^2$. We also know that the number of $(\varepsilon'', H, \varrho)$-irregular pairs is at most $\varepsilon'' \binom{k}{2}$, and hence we deduce that the number of edges of type (ii)

is, by (9.130),

$$\leq 2\varrho m^2\varepsilon''\binom{k}{2} \leq 2\varepsilon''\varrho\left(\frac{n}{k}\right)^2\frac{k^2}{2} \leq \frac{\gamma}{2^5}\left(\frac{1}{2}\varrho n^2\right). \tag{9.148}$$

Fact 53 together with the fact that $Am\sqrt{r} \leq \varrho m^2$ (cf. (9.143)) imply that $e(H[C_i]) \leq (3/2)\varrho m^2$. Therefore, the number of edges of type (*iii*) is, by (9.131),

$$\leq \frac{3}{2}\varrho m^2 k \leq \frac{3}{2}\varrho\left(\frac{n}{k}\right)^2 k = \frac{3}{k}\left(\varrho\frac{n^2}{2}\right) \leq \frac{3}{2^6}\gamma\left(\frac{1}{2}\varrho n^2\right). \tag{9.149}$$

We now observe that, because of (9.133), we have $\varepsilon'' \geq 1/2K_0 \geq \eta_0$. Therefore, the number of edges of type (*iv*), that is, incident to C_0, is, by (9.130),

$$\leq \frac{3}{2}\varrho(\varepsilon''n)^2 + 2\varrho\varepsilon''n^2 = \left(3(\varepsilon'')^2 + 4\varepsilon''\right)\varrho\frac{n^2}{2} \leq \frac{5}{2^6}\gamma\left(\frac{1}{2}\varrho n^2\right). \tag{9.150}$$

We conclude from (9.147)–(9.150) that the number of edges in H satisfies

$$e(H) \leq \left(\frac{1}{2^2} + \frac{1}{2^5} + \frac{1}{2^3}\right)\gamma\left(\frac{1}{2}\varrho n^2\right) < \frac{7}{16}\gamma\left(\frac{1}{2}\varrho n^2\right). \tag{9.151}$$

We shall now estimate $e(J)$ from below. Fact 53 tells us that

$$e(J) \geq \varrho\binom{n}{2} - An\sqrt{r} = \frac{1}{2}\varrho n^2 - \frac{1}{2}\varrho n - An\sqrt{r} = \frac{1}{2}\varrho n^2 - \frac{r}{2} - An\sqrt{r}. \tag{9.152}$$

Using that $n \geq N_1 \geq 16$, we obtain $r/2 \leq (1/16)\varrho n^2/2$, and using that $r \geq r_1 \geq (2AK_0)^2 \geq (2Ak_0)^2 > (2^5A)^2$, we obtain that $An\sqrt{r} \leq (1/16)\varrho n^2/2$. We therefore conclude from (9.152) that

$$e(J) \geq \frac{7}{8}\left(\frac{1}{2}\varrho n^2\right). \tag{9.153}$$

Finally, (9.151) and (9.153) imply that $e(H) < (\gamma/2)e(J)$, which is a contradiction. Therefore some pair (C_i, C_j) must be as required, and the proof of Claim 66 is complete. □

We now fix a pair (C_i, C_j) as in Claim 66, and let $U = C_i$ and $W = C_j$. Recalling (9.144), we see that the pair (U, W) is as required in Lemma 65, and hence we are done. □

We now restrict our attention to the pair (U, W) given by Lemma 65. We shall in fact obtain a contradiction by estimating from above and from below the quantity

$$\left|\sum\nolimits_{ij\in H}^{(U,W)} \langle \mathbf{v}_i, \mathbf{v}_j\rangle\right|, \tag{9.154}$$

where $\sum_{ij\in H}^{(U,W)}$ denotes sum over all edges $ij \in H$ with $i \in U$ and $j \in W$. (The number of summands in (9.154) is, therefore, $e_H(U, W)$.)

We start by noticing that we have the following lower bound for (9.154) from the definition of the edge set of H and the fact that (U, W) is a 'dense' pair for H.

Lemma 67 *We have*

$$\left|\sum\nolimits_{ij\in H}^{(U,W)} \langle \mathbf{v}_i, \mathbf{v}_j\rangle\right| > \frac{1}{4}\varepsilon\gamma n\varrho m^2. \tag{9.155}$$

Proof. For any $ij \in H$, by (9.141), we have

$$\langle \mathbf{v}_i, \mathbf{v}_j\rangle = n - 2|\Gamma_G(i) \triangle \Gamma_G(j)| > n - 2\left(\frac{1}{2}n - \frac{1}{2}\varepsilon n\right) = \varepsilon n.$$

Therefore, we have

$$\sum\nolimits_{ij\in H}^{(U,W)} \langle \mathbf{v}_i, \mathbf{v}_j\rangle > \varepsilon n e_H(U, W) \geq \frac{1}{4}\varrho\varepsilon\gamma n m^2,$$

since $d_{H,\varrho}(U, W) \geq \gamma/4$ and hence $e_H(U, W) \geq (1/4)\gamma\varrho m^2$. Inequality (9.155) is proved. □

Remark 68 *In the case in which H is the graph with edges ij for which (9.142) holds instead of (9.141), we have*

$$\langle \mathbf{v}_i, \mathbf{v}_j\rangle = n - 2|\Gamma_G(i) \triangle \Gamma_G(j)| < n - 2\left(\frac{1}{2}n + \frac{1}{2}\varepsilon n\right) = -\varepsilon n.$$

Therefore, we would have

$$\sum\nolimits_{ij\in H}^{(U,W)} \langle \mathbf{v}_i, \mathbf{v}_j\rangle < -\varepsilon n e_H(U, W) \leq -\frac{1}{4}\varrho\varepsilon\gamma n m^2,$$

and (9.155) would follow as well. For the remainder of the proof, it will not matter whether the edges of H satisfy (9.141) or (9.142). We shall only make use of (9.155).

Our upper bound for (9.154) will come from the $(1/2, \varepsilon', \delta')$-quasi-randomness of G and the $(\varepsilon'', H, \varrho)$-regularity of the pair (U, W). More specifically, we let

$$S_k^{(U,W)} = \sum\nolimits_{ij\in H}^{(U,W)} a_{ik}a_{jk} \tag{9.156}$$

for all $k \in V$, and show that this sum is essentially always small, which will tell us that $\sum_{ij\in H}^{(U,W)} \langle \mathbf{v}_i, \mathbf{v}_j\rangle = \sum_{k\in V} S_k^{(U,W)}$ is quite small.

Let a vertex $k \in V$ be given. We then let

$$U^+ = U_k^+ = \Gamma_G(k) \cap U \qquad U^- = U_k^- = U \setminus \Gamma_G(k) \tag{9.157}$$

$$W^+ = W_k^+ = \Gamma_G(k) \cap W \qquad W^- = W_k^- = W \setminus \Gamma_G(k). \tag{9.158}$$

Then, clearly,

$$S_k^{(U,W)} = e_H(U^+,W^+) + e_H(U^-,W^-) - e_H(U^+,W^-) - e_H(U^-,W^+). \tag{9.159}$$

Moreover, for most $k \in V$, we may estimate the four terms on the right-hand side of (9.159) by $\sim d_{H,\varrho}(U,W)m^2/4$.

Indeed, let us say that a vertex $k \in V \setminus (U \cup W)$ is (U,W)-*typical*, or simply *typical*, if

$$|U^+|,\ |U^-|,\ |W^+|,\ |W^-| = \frac{1}{2}(1+O_1(\varepsilon'))m \ge \varepsilon'' m. \tag{9.160}$$

Then, by the $(\varepsilon'', H, \varrho)$-regularity of the pair (U,W), we have

$$e_H(U^+,W^+),\ e_H(U^-,W^-),\ e_H(U^+,W^-),\ e_H(U^-,W^+) \sim \frac{1}{4}d_{H,\varrho}(U,W)m^2 \tag{9.161}$$

for any typical k. Let us make this remark more precise. For simplicity, let us write $\sigma = d_{H,\varrho}(U,W)$, and $u^+ = u_k^+ = |U^+|$, $u^- = u_k^- = |U^-|$ and similarly for w^+ and w^-.

Because $r \ge r_1 \ge (2AK_0)^2$, the graph $H = H^n$ is a $(1/2K_0, 2)$-upper-uniform graph with respect to density ϱ (cf. the proof of Lemma 65). Therefore, we have

$$\sigma = d_{H,\varrho}(U,W) \le 2, \tag{9.162}$$

since $|U|$, $|W| \ge (1/2K_0)n$.

From (9.160) and the $(\varepsilon'', H, \varrho)$-regularity of (U,W), we have

$$e_H(U^\alpha, W^\beta) = (\sigma + O_1(\varepsilon''))\,\varrho u^\alpha w^\beta, \tag{9.163}$$

for all α, $\beta \in \{+,-\}$. In particular, if we know that k is typical, we have

$$e_H(U^+,W^+),\ e_H(U^-,W^-) \le (\sigma+\varepsilon'')\varrho\left\{\frac{1}{2}(1+\varepsilon')m\right\}^2 \tag{9.164}$$

and

$$e_H(U^+,W^-),\ e_H(U^-,W^+) \ge (\sigma-\varepsilon'')\varrho\left\{\frac{1}{2}(1-\varepsilon')m\right\}^2. \tag{9.165}$$

A little computation now gives the first statement in the following lemma. The second statement is immediate.

Lemma 69 (*i*) *For any* (U,W)*-typical vertex* $k \in V \setminus (U \cup W)$, *we have*

$$\left|S_k^{(U,W)}\right| = \left|\sum\nolimits_{ij\in H}^{(U,W)} a_{ik}a_{jk}\right| \le 2\varrho m^2(\varepsilon'\sigma + \varepsilon''). \tag{9.166}$$

(*ii*) *For any vertex* $k \in V$, *we have*

$$\left|S_k^{(U,W)}\right| = \left|\sum\nolimits_{ij\in H}^{(U,W)} a_{ik}a_{jk}\right| \le \varrho m^2 + Am\sqrt{r}. \tag{9.167}$$

Proof. Let us prove (i). Let a (U,W)-typical vertex k be fixed. Using (9.159), (9.164), and (9.165), we obtain

$$\begin{aligned}
S_k^{(U,W)} &= \sum\nolimits_{ij\in H}^{(U,W)} a_{ik}a_{jk} \\
&\le (\sigma+\varepsilon'')\varrho u^+w^+ + (\sigma+\varepsilon'')\varrho u^-w^- \\
&\quad - (\sigma-\varepsilon'')\varrho u^+w^- - (\sigma-\varepsilon'')\varrho u^-w^+ \\
&\le 2(\sigma+\varepsilon'')\varrho\left\{\frac{1}{2}(1+\varepsilon')m\right\}^2 - 2(\sigma-\varepsilon'')\varrho\left\{\frac{1}{2}(1-\varepsilon')m\right\}^2 \\
&= \frac{1}{2}(\sigma+\varepsilon'')\varrho\left(1+2\varepsilon'+(\varepsilon')^2\right)m^2 \\
&\quad - \frac{1}{2}(\sigma-\varepsilon'')\varrho\left(1-2\varepsilon'+(\varepsilon')^2\right)m^2 \\
&= \frac{1}{2}\sigma\varrho m^2(4\varepsilon') + \frac{1}{2}\varepsilon''\varrho m^2(2+2(\varepsilon')^2) \\
&= \frac{1}{2}\varrho m^2(4\varepsilon'\sigma + \varepsilon''(2+2(\varepsilon')^2)) \\
&\le 2\varrho m^2(\varepsilon'\sigma+\varepsilon''),
\end{aligned}$$

and (i) is proved. To prove (ii) it suffices to recall that $H\subset J$ and that J is a (ϱ, A)-uniform graph, and hence

$$\left|S_k^{(U,W)}\right| = \left|\sum\nolimits_{ij\in H}^{(U,W)} a_{ik}a_{jk}\right| \le e_H(U,W) \le e_J(U,W) \le \varrho m^2 + Am\sqrt{r}, \tag{9.168}$$

as required. □

Our next lemma gives an upper bound for the quantity in (9.154). The reader will immediately see that this upper bound is a consequence of Lemma 69 and the fact that there are only very few atypical vertices k, because of the $(1/2,\varepsilon',\delta')$-quasi-randomness of G.

Lemma 70 *We have*

$$\left|\sum\nolimits_{ij\in H}^{(U,W)} \langle \mathbf{v}_i, \mathbf{v}_j\rangle\right| \le 2(2\delta'n+m)\left(\varrho m^2 + Am\sqrt{r}\right) + 2\varrho m^2 n\left(\varepsilon'\sigma+\varepsilon''\right). \tag{9.169}$$

Proof. We claim that the number of vertices $k\in V\setminus(U\cup W)$ that are not (U,W)-typical is, by the $(1/2,\varepsilon',\delta')$-quasi-randomness of G, less than $4\delta'n$. Indeed, if we had $\ge 4\delta'n$ vertices that are not (U,W)-typical, then we would have $\ge 2\delta'n$ vertices that are not 'typical' for either U alone or else for W alone. In other words, we would have $\ge 2\delta'n$ vertices $k\in V\setminus(U\cup W)$ for which, say,

$$|\Gamma_G(k)\cap U| = |U^+| > \frac{1}{2}(1+\varepsilon')m \tag{9.170}$$

and hence $|U \backslash \Gamma_G(k)| = |U^-| < (1/2)(1-\varepsilon')m$, or else we would have $\geq 2\delta' n$ vertices $k \in V \setminus (U \cup W)$ for which we have

$$|\Gamma_G(k) \cap U| = |U^+| < \frac{1}{2}(1-\varepsilon')m \tag{9.171}$$

and hence $|U \setminus \Gamma_G(k)| = |U^-| > (1/2)(1+\varepsilon')m$. Therefore there would be $\geq \delta' n$ vertices $k \in V \setminus (U \cup W)$ for which, say, (9.170) holds. Let $T \subset V \setminus (U \cup W)$ be the set of such vertices k. Then

$$|T| \geq \delta' n \tag{9.172}$$

and

$$e(T, U) > \frac{1}{2}(1+\varepsilon')|T|m = \frac{1}{2}(1+\varepsilon')|T||U|. \tag{9.173}$$

We also have

$$|U| = m \geq \frac{n}{2K_0} \geq \delta_0 n \geq \delta' n \tag{9.174}$$

(see (9.134)). Inequalities (9.172)–(9.174) say that the pair (T, U) is a witness against the $(1/2, \varepsilon', \delta')$-quasi-randomness of G. This contradiction confirms that, indeed, the number of vertices $k \in V \setminus (U \cup W)$ that are not (U, W)-typical is less than $4\delta' n$.

Using (9.166) for the (U, W)-typical vertices $k \in V \setminus (U \cup W)$, and using (9.167) for the vertices $k \in V \setminus (U \cup W)$ that are not (U, W)-typical and for all the vertices $k \in U \cup W$, we have

$$\begin{aligned} \left|\sum\nolimits_{ij\in H}^{(U,W)} \langle \mathbf{v}_i, \mathbf{v}_j \rangle\right| &= \left|\sum_{k\in V} \sum\nolimits_{ij\in H}^{(U,W)} a_{ik}a_{jk}\right| = \left|\sum_{k\in V} S_k^{(U,W)}\right| \\ &\leq (4\delta' n + 2m)\left(\varrho m^2 + Am\sqrt{r}\right) + 2\varrho m^2 n\left(\varepsilon'\sigma + \varepsilon''\right), \end{aligned}$$

as required. □

We finish the proof by deriving a contradiction comparing Lemmas 67 and 70. To that end, we first claim that

$$\frac{1}{8}\gamma\varepsilon > 2\left(2\delta_0 + \frac{1}{k_0}\right) + \varepsilon_0\sigma + \varepsilon'' \geq 2\left(2\delta' + \frac{1}{k_0}\right) + \varepsilon'\sigma + \varepsilon''. \tag{9.175}$$

To prove our claim, we first observe that, because $\delta' \leq \delta_0$ and $\varepsilon' \leq \varepsilon_0$, the second inequality in (9.175) is obvious. As to the first inequality in (9.175), observe that, because of (9.134), we have

$$4\delta_0 \leq \frac{1}{2^5}\gamma\varepsilon. \tag{9.176}$$

Moreover, because of (9.131), we have

$$\frac{2}{k_0} \leq \frac{1}{2^5}\gamma\varepsilon. \tag{9.177}$$

Since $\sigma \le 2$ (see (9.162)), we have from (9.129) that

$$\varepsilon_0\sigma \le \frac{1}{2^5}\gamma\varepsilon. \tag{9.178}$$

Inequalities (9.176)–(9.178) and (9.130) imply the first inequality in (9.175).

We now recall inequalities (9.155) and (9.169) to obtain that

$$\begin{aligned}\frac{1}{4}\varepsilon\gamma n\varrho m^2 &< \left|\sum\nolimits_{ij\in H}^{(U,W)} \langle \mathbf{v}_i, \mathbf{v}_j\rangle\right| \\ &\le 2(2\delta' n + m)\left(\varrho m^2 + Am\sqrt{r}\right) + 2\varrho m^2 n\left(\varepsilon'\sigma + \varepsilon''\right).\end{aligned} \tag{9.179}$$

Let us also recall that

$$Am\sqrt{r} \le \varrho m^2, \tag{9.180}$$

because $r = \varrho n \ge r_1 \ge (2AK_0)^2$ and $m \ge (1/2K_0)n$. Moreover, the fact that $m \le n/k$ gives us that

$$2\delta' n + m \le \left(2\delta' + \frac{1}{k}\right)n \le \left(2\delta' + \frac{1}{k_0}\right)n. \tag{9.181}$$

Inequalities (9.179), (9.180), and (9.144) give that

$$\begin{aligned}\frac{1}{4}\varepsilon\gamma n\varrho m^2 &\le 4(2\delta' n + m)\varrho m^2 + 2\varrho m^2 n\left(\varepsilon'\sigma + \varepsilon''\right) \\ &\le 4\left(2\delta' + \frac{1}{k_0}\right)\varrho n m^2 + 2\varrho m^2 n\left(\varepsilon'\sigma + \varepsilon''\right).\end{aligned} \tag{9.182}$$

Dividing (9.182) by $2\varrho m^2 n$, we obtain

$$\frac{1}{8}\varepsilon\gamma \le 2\left(2\delta' + \frac{1}{k_0}\right) + \varepsilon'\sigma + \varepsilon'', \tag{9.183}$$

which contradicts (9.175).

The proof of Theorem 57 is complete.

References

[1] N. Alon.Expanders, sorting in rounds and superconcentrators of limited depth.In *Proceedings of the 17th Annual ACM Symposium on the Theory of Computing (STOC 85)*, pages 98–102, 1985.

[2] N. Alon.Eigenvalues, geometric expanders, sorting in rounds, and Ramsey theory.*Combinatorica*, 6(3):207–219, 1986.

[3] N. Alon, M. Capalbo, Y. Kohayakawa, V. Rödl, A. Ruciński, and E. Szemerédi.Universality and tolerance (extended abstract).In *Proceedings of the 41st IEEE Annual Symposium on Foundations of Computer Science (FOCS 2000)*, pages 14–21, 2000.

[4] N. Alon, R. A. Duke, H. Lefmann, V. Rödl, and R. Yuster.The algorithmic aspects of the regularity lemma (extended abstract).In *33rd Annual Symposium on Foundations of Computer Science*, pages 473–481, Pittsburgh, Pennsylvania, 1992. IEEE Comput. Soc. Press.

[5] N. Alon, R. A. Duke, H. Lefmann, V. Rödl, and R. Yuster.The algorithmic aspects of the regularity lemma.*Journal of Algorithms*, 16(1):80–109, 1994.

[6] N. Alon, E. Fischer, M. Krivelevich, and M. Szegedy.Efficient testing of large graphs.submitted, 22pp., 1999.

[7] N. Alon, E. Fischer, M. Krivelevich, and M. Szegedy.Efficient testing of large graphs (extended abstract).In *40th Annual Symposium on Foundations of Computer Science*, pages 656–666, New York City, NY, 1999. IEEE Comput. Soc. Press.

[8] N. Alon and J. Spencer.*The Probabilistic Method.*Wiley-Interscience Series in Discrete Mathematics. John Wiley & Sons, New York, 1992.

[9] N. Alon and R. Yuster.H-factors in dense graphs.*Journal of Combinatorial Theory, Series B*, 66(2):269–282, 1996.

[10] L. Babai, P. Frankl, and J. Simon.Complexity classes in communication complexity theory (preliminary version).In *27th Annual Symposium on Foundations of Computer Science*, pages 337–347, Toronto, Ontario, Canada, 1986. IEEE.

[11] B. Bollobás.*Extremal graph theory.*Academic Press Inc. [Harcourt Brace Jovanovich Publishers], London, 1978.

[12] B. Bollobás.*Graph theory.*Springer-Verlag, New York, 1979.An introductory course.

[13] B. Bollobás.*Random graphs.*Academic Press Inc. [Harcourt Brace Jovanovich Publishers], London, 1985.

[14] B. Bollobás.*Modern graph theory.*Springer-Verlag, New York, 1998.

[15] F. R. K. Chung, R. L. Graham, and R. M. Wilson.Quasi-random graphs.*Combinatorica*, 9(4):345–362, 1989.

[16] V. Chvátal, V. Rödl, E. Szemerédi, and W. T. Trotter.The Ramsey number of a graph with bounded maximum degree.*Journal of Combinatorial Theory, Series B*, 34(3):239–243, 1983.

[17] D. Coppersmith and S. Winograd.Matrix multiplication via arithmetic progressions.*J. Symbolic Comput.*, 9(3):251–280, 1990.

[18] W. Deuber.A generalization of Ramsey's theorem.In A. Hajnal, R. Rado, and V. T. Sós, editors, *Infinite and Finite Sets*, volume 10 of *Colloquia Mathematica Societatis János Bolyai*, pages 323–332, Keszthely, 1973, 1975. North-Holland.

[19] R. A. Duke, H. Lefmann, and V. Rödl.A fast approximation algorithm for computing the frequencies of subgraphs in a given graph.*SIAM Journal on Computing*, 24(3):598–620, 1995.

[20] R. A. Duke and V. Rödl.On graphs with small subgraphs of large chromatic number.*Graphs and Combinatorics*, 1(1):91–96, 1985.

[21] P. Erdős.Some old and new problems in various branches of combinatorics.In *Proceedings of the Tenth Southeastern Conference on Combinatorics, Graph*

Theory and Computing (Florida Atlantic Univ., Boca Raton, Fla., 1979), pages 19–37, Winnipeg, Man., 1979. Utilitas Math.

[22] P. Erdős, A. Hajnal, and L. Pósa.Strong embeddings of graphs into colored graphs.In A. Hajnal, R. Rado, and V. T. Sós, editors, *Infinite and Finite Sets*, volume 10 of *Colloquia Mathematica Societatis János Bolyai*, pages 585–595, Keszthely, 1973, 1975. North-Holland.

[23] P. Erdős and J. Spencer.*Probabilistic methods in combinatorics.*Akademiai Kiado, Budapest, 1974.106pp.

[24] P. Frankl and V. Rödl.The uniformity lemma for hypergraphs.*Graphs and Combinatorics*, 8(4):309–312, 1992.

[25] P. Frankl and V. Rödl.Extremal problems on set systems.*Random Structures and Algorithms*, 2002.to appear.

[26] P. Frankl, V. Rödl, and R. M. Wilson.The number of submatrices of a given type in a Hadamard matrix and related results.*J. Combin. Theory Ser. B*, 44(3):317–328, 1988.

[27] A. Frieze and R. Kannan.The regularity lemma and approximation schemes for dense problems.In *37th Annual Symposium on Foundations of Computer Science (Burlington, VT, 1996)*, pages 12–20. IEEE Comput. Soc. Press, Los Alamitos, CA, 1996.

[28] A. Frieze and R. Kannan.Quick approximation to matrices and applications.*Combinatorica*, 19(2):175–220, 1999.

[29] O. Goldreich.Combinatorial property testing (a survey).In *Randomization methods in algorithm design (Princeton, NJ, 1997)*, pages 45–59. Amer. Math. Soc., Providence, RI, 1999.

[30] O. Goldreich, S. Goldwasser, and D. Ron.Property testing and its connection to learning and approximation.In *37th Annual Symposium on Foundations of Computer Science (Burlington, VT, 1996)*, pages 339–348. IEEE Comput. Soc. Press, Los Alamitos, CA, 1996.

[31] O. Goldreich, S. Goldwasser, and D. Ron.Property testing and its connection to learning and approximation.*Journal of the Association for Computing Machinery*, 45(4):653–750, 1998.

[32] O. Goldreich and D. Ron.Property testing in bounded degree graphs.In *29th ACM Symposium on Theory of Computing*, pages 406–419, El Paso, Texas, 1997.

[33] O. Goldreich and D. Ron.A sublinear bipartiteness tester for bounded degree graphs.*Combinatorica*, 19(3):335–373, 1999.

[34] W. T. Gowers.Lower bounds of tower type for Szemerédi's uniformity lemma.*Geometric and Functional Analysis*, 7(2):322–337, 1997.

[35] S. Janson, T. Łuczak, and A. Rucinski.*Random graphs.*Wiley-Interscience, New York, 2000.

[36] Y. Kohayakawa.Szemerédi's regularity lemma for sparse graphs.In F. Cucker and M. Shub, editors, *Foundations of Computational Mathematics*, pages 216–230, Berlin, Heidelberg, January 1997. Springer-Verlag.

[37] Y. Kohayakawa and V. Rödl.Algorithmic aspects of regularity.In G. Gonnet, D. Panario, and A. Viola, editors, *LATIN'2000: Theoretical Informatics*

(*Punta del Este, 2000*), Lecture Notes in Computer Science, pages 1–17. Springer, Berlin, 2000.

[38] Y. Kohayakawa and V. Rödl.Regular pairs in sparse random graphs I.submitted, 2001.

[39] Y. Kohayakawa and V. Rödl.Regular pairs in sparse random graphs II.in preparation, 2001.

[40] Y. Kohayakawa, V. Rödl, and E. Szemerédi.The size-Ramsey number of graphs of bounded degree.in preparation, 2001.

[41] Y. Kohayakawa, V. Rödl, and L. Thoma.An optimal algorithm for checking regularity (extended abstract).In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2002)*, 2002.to appear.

[42] J. Komlós.The blow-up lemma.*Combinatorics, Probability and Computing*, 8(1-2):161–176, 1999.Recent trends in combinatorics (Mátraháza, 1995).

[43] J. Komlós, G. N. Sárközy, and E. Szemerédi.Proof of a packing conjecture of Bollobás.*Combin. Probab. Comput.*, 4(3):241–255, 1995.

[44] J. Komlós, G. N. Sárközy, and E. Szemerédi.Blow-up lemma.*Combinatorica*, 17(1):109–123, 1997.

[45] J. Komlós, G. N. Sárközy, and E. Szemerédi.An algorithmic version of the blow-up lemma.*Random Structures and Algorithms*, 12(3):297–312, 1998.

[46] J. Komlós, G. N. Sárközy, and E. Szemerédi.On the Pósa-Seymour conjecture.*J. Graph Theory*, 29(3):167–176, 1998.

[47] J. Komlós, G. N. Sárközy, and E. Szemerédi.Proof of the Seymour conjecture for large graphs.*Ann. Comb.*, 2(1):43–60, 1998.

[48] J. Komlós and M. Simonovits.Szemerédi's regularity lemma and its applications in graph theory.In D. Miklós, V. T. Sós, and T. Szőnyi, editors, *Combinatorics—Paul Erdős is eighty,* vol. 2 (Keszthely, 1993), volume 2 of *Bolyai Society Mathematical Studies*, pages 295–352. János Bolyai Mathematical Society, Budapest, 1996.

[49] A. Lubotzky.*Discrete groups, expanding graphs and invariant measures.*Birkhäuser Verlag, Basel, 1994.With an appendix by Jonathan D. Rogawski.

[50] A. Lubotzky, R. Phillips, and P. Sarnak.Explicit expanders and the Ramanujan conjectures.In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing (STOC '86)*, pages 240–246, Berkeley, California, 1986. ACM.

[51] A. Lubotzky, R. Phillips, and P. Sarnak.Ramanujan graphs.*Combinatorica*, 8:261–277, 1988.

[52] J. Nešetřil and V. Rödl.Partition theory and its application.In *Surveys in combinatorics (Proc. Seventh British Combinatorial Conf., Cambridge, 1979)*, pages 96–156. Cambridge Univ. Press, Cambridge, 1979.

[53] V. Nikiforov.On a problem of Erdős about the local density of K_p-free graphs.submitted, 1999.

[54] V. Rödl.The dimension of a graph and generalized Ramsey theorems.Master's thesis, Charles University, 1973.

[55] V. Rödl.On universality of graphs with uniformly distributed edges.*Discrete Mathematics*, 59(1-2):125–134, 1986.

[56] V. Rödl and A. Ruciński.Perfect matchings in ε-regular graphs and the blow-up lemma.*Combinatorica*, 19(3):437–452, 1999.

[57] V. Rödl, A. Ruciński, and M. Wagner.An algorithmic embedding of graphs via perfect matchings.In *Randomization and approximation techniques in computer science (Barcelona, 1998)*, pages 25–34. Springer, Berlin, 1998.

[58] R. Rubinfeld and M. Sudan.Robust characterizations of polynomials with applications to program testing.*SIAM Journal on Computing*, 25(2):252–271, Apr. 1996.

[59] P. Sarnak.*Some applications of modular forms.*Cambridge University Press, Cambridge, 1990.

[60] E. Szemerédi.Regular partitions of graphs.In *Problèmes Combinatoires et Théorie des Graphes (Colloq. Internat. CNRS, Univ. Orsay, Orsay, 1976)*, pages 399–401, Paris, 1978. Colloques Internationaux CNRS n. 260.

[61] A. R. Taraz.Szemerédis Regularitätslemma, Apr. 1995.Diplomarbeit, Universität Bonn, 83pp.

[62] A. G. Thomason.Pseudorandom graphs.In *Random graphs '85 (Poznań, 1985)*, volume 144 of *North-Holland Math. Stud.*, pages 307–331. North-Holland, Amsterdam–New York, 1987.

[63] A. G. Thomason.Random graphs, strongly regular graphs and pseudorandom graphs.In C. Whitehead, editor, *Surveys in Combinatorics 1987*, volume 123 of *London Mathematical Society Lecture Note Series*, pages 173–195. Cambridge University Press, Cambridge–New York, 1987.

[64] P. Turán.Eine Extremalaufgabe aus der Graphentheorie.*Mat. Fiz. Lapok*, 48:436–452, 1941.in Hungarian, with German summary.