ANNALS OF DISCRETE MATHEMATICS

53

The Steiner Tree Problem

MONOGRAPH

F.K. HWANG
D.S. RICHARDS
P. WINTER

NORTH-HOLLAND

THE STEINER TREE PROBLEM

ANNALS OF DISCRETE MATHEMATICS 53

THE STEINER TREE PROBLEM

Frank K. HWANG

AT&T Bell Laboratories
Murray Hill, NJ, U.S.A.

Dana S. RICHARDS

University of Virginia
Thornton Hall, VA, U.S.A.

Pawel WINTER

University of Copenhagen
Copenhagen, Denmark

1992

NORTH-HOLLAND – AMSTERDAM • LONDON • NEW YORK • TOKYO

ELSEVIER SCIENCE PUBLISHERS B.V.
Sara Burgerhartstraat 25
P.O. Box 211, 1000 AE Amsterdam, The Netherlands

Library of Congress Cataloging-in-Publication Data

Hwang, Frank.
The Steiner tree problem / Frank K. Hwang, Dana S. Richards, Pawel Winter.
p. cm. -- (Annals of discrete mathematics ; 53)
Includes bibliographical references and indexes.
ISBN 0-444-89098-X (alk. paper)
1. Steiner systems. I. Richards, Dana Scott, 1955- II. Winter, Pawel, 1952- . III. Title. IV. Series.
QA166.3.H83 1992
511'.5--dc20 92-25345
CIP

ISBN: 0 444 89098 X

This book is printed on acid-free paper.

Printed in The Netherlands

Foreword

The *Steiner problem* asks for a shortest network which spans a given set of points. Minimum spanning networks have been well-studied when all connections are required to be between the given points. The novelty of the Steiner tree problem is that new auxiliary points can be introduced between the original points so that a spanning network of all the points will be shorter than otherwise possible. These new points are called *Steiner points.*

It was soon recognized that locating these Steiner points was a difficult problem. It has opened a rich field of intriguing analyses and challenging associated problems. A large literature has arisen trying many different avenues to understand these problems. As these avenues diverged and more applications were found, the literature became more fragmented, leading to duplicated results and wasted effort. In this book we hope to draw together the many threads.

Three major areas have emerged, which comprise the first three parts of this book. These are the *Euclidean Steiner problem*, the *rectilinear Steiner problem*, and the *Steiner problem in networks*. Since it can be shown that Steiner points in either the Euclidean or the rectilinear case belong to a finite set of points, these two problems can be regarded as special cases of the network problem. However, the study of these special geometries has led to the discovery of many interesting properties and better algorithms or heuristics, which would not have been found if they had been cast as network problems.

Historically, the Euclidean Steiner problem, discussed in Part I, was the original Steiner tree problem. It was proposed by Jarník and Kössler in 1934, generalizing with the special three-point case (going back to Fermat). It is often used as a problem to introduce computational geometry. The problem can be stated essentially in one line, and is comprehensible to people without any mathematical or computing background. Yet the problem has been extremely hard to solve.

The rectilinear case, found in Part III, was introduced by Hanan in 1965 in his study of wiring problems on circuits but is also known for its many important applications as in building designs and drainage networks. While historically it was introduced before the Steiner problem in networks, it appears later in this book due to the reliance on results in Part II. We devote an entire chapter to the specific application of wiring. It serves as a case study of the balance between what theory offers and applications need.

The Steiner problem in networks, discussed in Part II, was proposed independently by Hakimi and Levin a few years later. It has provided a fertile ground for graph theorists and computer scientists to design algorithms and heuristics. It has seen the most prolific research amongst the three areas. The Euclidean and the rectilinear literature straddles the fence by dealing with the problem both in geometric terms and with graph theoretic approaches (for the underlying topologies). Hence, the mathematical presentations in Parts I and III tend to be less detailed than in Part II.

Part IV contains two chapters discussing areas where the body of results is still emerging. The first chapter discusses *general metrics*. It contains clear generalizations of results in Parts I and III. Further, one often studies the network by transforming the original network into a *metric network*, (where the distance between two vertices in the transformed graph is defined as the length of a shortest path between the two vertices in the original network). This in turn makes the Steiner problem in networks a special case of the Steiner problem for general metrics.

The second chapter of Part IV is concerned with a biological problem and the recent attempts to find algorithms to solve it. The problem of constructing *phylogenetic trees* in the field of evolutionary biology is closely related to standard Steiner problems. An important aspect of this application area is that the problem statement often changes.

This book represents the collaboration of three authors with different styles and outlooks. Every effort was made to have the book give a cohesive view of the whole literature, while permitting individual differences. FKH wrote Part I and Chapter 1 of Part IV, PW wrote Part II, and DSR wrote Part III and Chapter 2 of Part IV. Of course we collectively take responsibility for the rise or fall of this book.

We would like to thank John Beasley, Marshall Bern, Jens Clausen, Dinzu Du, Joe Felsentein, Ed Gilbert, Jan Plesník, Jeff Salowe, Jim Smith, Stefan Voss and Jiafeng Weng for reading and commenting on parts of the book. FKH also acknowledges the excellent typing and drawing work done by Sue Pope, with some help from Susan Marko at the initial stage. PW acknowledges the drawing work done by Ruth Nielsen.

Contents

This Page Intentionally Left Blank

Part I

Euclidean Steiner Problem

This Page Intentionally Left Blank

Chapter 1

Introduction

The purpose of this chapter is to provide some general knowledge about the Euclidean Steiner problem to prepare the reader for subsequent chapters. The chapter begins with some historical background for this problem. Basic geometric and topological notions and characterizations are given in Section 1.2 and Section 1.3. Full Steiner trees, an important restricted case, are discussed in Section 1.4, and problem reductions are found in Section 1.5. The next section discusses the combinatorics of such trees, and Section 1.7 is concerned with the computational complexity. The final section discusses non-mathematical techniques, based on physical models, that have been proposed.

1.1 Historical Background

The historical background of the Euclidean Steiner problem (ESP) was well researched by Zacharis [19], with an English account given in Kuhn [12]. We essentially follow their historical findings here.

The origin of the ESP is often traced back to Fermat (1601-1665) who proposed the following problem: find in the plane a point, the sum of whose distances from three given points is minimal. This problem will be referred to as the *Fermat problem.* Torricelli had proposed a geometric solution to this problem before 1640. He asserted that the three circles circumscribing the equilateral triangles constructed on the sides of and outside the triangle intersect in the point that is sought. This point is called the *Torricelli point* (see Fig. 1.1).

Cavalieri in his 1647 book "Excercitationes Geometricae" showed that line segments from the three given points to the Torricelli point make 120° with each other. Simpson in his 1750 book "Doctrine and Application of Fluxions" asserted and proved that the three lines joining the outside vertices of the equilateral triangles defined above to the opposite vertices of the given triangle intersect in the Torricelli point. These three lines are called *Simpson lines* (see Fig. 1.1). Heinen in 1834 proved that the lengths of the three Simpson lines are the same and equal to the sum of distances from the Torricelli point to the three given points.

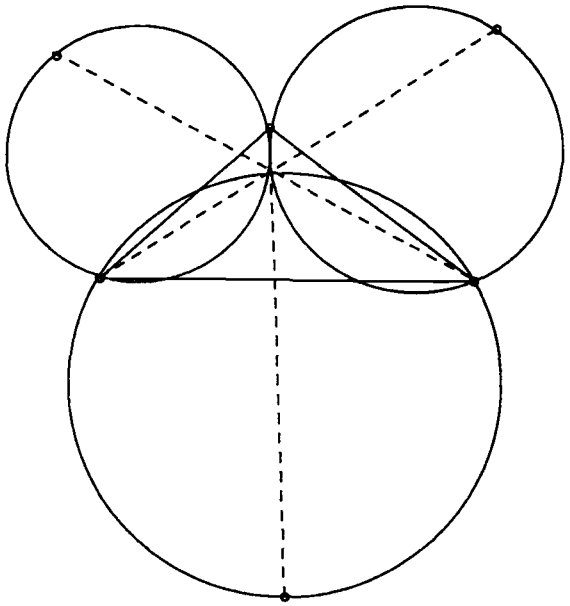


Figure 1.1: Torricelli point and Simpson lines

When one of the angles in the given triangle is at least 120°, the Torricelli point lies outside of the given triangle and is no longer the minimizing point. The minimizing point in this case is the vertex of the obtuse angle. This fact was probably first observed by Heinen in 1834, and also by Bertrand in 1853.

The *general Fermat problem* which seeks a point in plane (or d-space) the sum of whose distances (or weighted distances) from n given points is minimal was included in the book "Fluxions" by Simpson as an exercise and has attracted the attention of many well-known mathematicians including Steiner. Since this problem has nothing to do with the Steiner problem (except for the case $n = 3$) we are studying, we simply refer the reader to the excellent survey of Kuhn [12] on the problem.

Jarník and Kössler [11] in 1934 raised the following question in a Czeckoslovakia journal: Find a shortest network which interconnects n points in the plane. In particular, they studied the above problem when the n points are the corners of a regular n-gon. They found a shortest network for $n = 3, 4, 5$ and gave an elegant proof that any $n - 1$ sides of the regular n-gon constitute a shortest network for $n \geq 13$. Jarník and Kössler made no reference to the Fermat problem since their problem seemed to be quite different.

Courant and Robbins [3] in their famous 1941 book "What Is Mathematics?" first made the connection that the Fermat problem is the shortest interconnection network with $n = 3$. However, they neither credited Fermat for the $n = 3$ problem nor Jarník and Kössler for the general n problem. Instead, they referred to the

former as the Steiner problem and the latter either as the street network problem or simply, the Steiner problem. The popularity of their book has been the main reason that the misnomer "the Steiner problem" has stuck, but more importantly, that the interest on this problem has spread.

Melzak [13] first established many basic properties of a shortest interconnecting network and gave a finite solution to the Steiner problem. Gilbert and Pollak [7] gave a thorough treatment of the Steiner problem and christened the name *Steiner minimal trees* (SMT) for shortest interconnecting networks (note that whenever edges have positive lengths, a shortest network must be a tree), and *Steiner points* for vertices in an SMT which are not among the n original points. (Recently, an SMT is often called a *minimum Steiner tree*. We will stick to the traditional term to avoid the acronym MST which usually means something else.) They also extended the Steiner tree problem to d-dimensional spaces and studied a probabilistic version.

1.2 Some Basic Notions

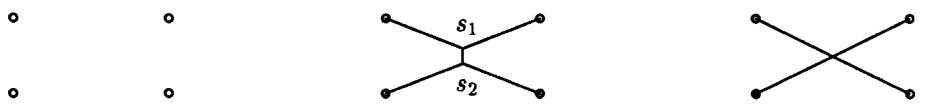
Consider a network T interconnecting a set N of n points, called *terminals*, in the Euclidean plane. Any vertex in T which is not a terminal is called a *Steiner point* . However, a Steiner point of degree one can clearly be deleted along with its edge to shorten the network. Furthermore, a Steiner point of degree two can also be deleted and its two edges replaced by a single edge connecting the two vertices adjacent to the Steiner point without increasing the length of the network. Therefore we adopt the convention that T will have no vertex of degree less than three except, possibly, some terminals. Consequently, a Steiner point is of degree at least three.

Let $G(T)$ denote the graph of T, where $G(T)$ represents the topological features of T but not the embedding in the plane and the edge lengths. Then $G(T)$ for a shortest network T must be a tree graph since if a cycle exists, then deleting any segment of T whose corresponding edge forming the said cycle in $G(T)$ decreases the length of the network. Therefore only networks with tree graphs need to be studied for the ESP. The traditional usage is to borrow the terminology of the graph for the network. Thus a network with a tree graph is called a *tree*, its segments are called *edges* and its nodes *vertices*. We will follow this tradition in Part I if no confusion arises. The graph of a network is referred to as a *topology*. Therefore the term "tree" is always used to designate a network (with a tree graph) whose edges have lengths. If there is no danger of confusion, we will abbreviate $G(T)$ to G.

By *shrinking* an edge is meant the operation of deleting an edge and collapsing its two endpoints . A reverse operation of that is called *splitting* a vertex which disconnects two edges $[a, v], [b, v]$ to a vertex v and connects a, b, v to a newly created Steiner point v' (shrinking the edge $[v, v']$ brings back the original graph). A topology is called a *degeneracy* of another if the former can be obtained from the latter through shrinking edges. The *set of degeneracies* of a topology G is denoted by $D(G)$. The use of the terms "splitting" and "degeneracy" is also extended to networks.

Gilbert and Pollak [7] defined a *Steiner tree* (ST) as a tree whose length cannot be shortened by a small perturbation, even when splitting is allowed. For a given

topology G they also defined a *relatively minimal tree* as a shortest tree with the topology G. Note that a relatively minimal tree does not have to exist for a given topology. Let N consist of the four corners of a 1×2 rectangle (Fig. 1.2a) and let G be the topology shown in Fig. 1.2b. Then the minimum occurs when $|s_1 - s_2| = 0$ (Fig. 1.2c) with a different topology which is a degeneracy of G.



(a) (b) (c)

Figure 1.2: The nonexistence of a relatively minimum tree

A *Steiner minimal tree* (SMT), as a global minimum, is clearly a shortest ST. On the other hand, an SMT must also be a relatively minimal tree for its topology. These relations underline the two general approaches to construct SMTs: find a shortest relatively minimal tree or find a shortest ST, which will be discussed in detail in Chapter 2.

1.3 Some Basic Properties

Suppose that two edges meet at a point v with an angle less than 120°. Then one can shorten the tree by, say, selecting points a and b on the two edges respectively with $|a - v| = |b - v|$, and locating the new Steiner point v' at the Torricelli point of Δabv. Therefore, a necessary condition for an ST is that no two edges can meet at a point with an angle less than 120°. We call this the *angle condition*. The angle condition implies that no vertex in an ST can have more than three edges. Since a Steiner point by convention has at least three edges, it must have exactly three edges. Another consequence is that no two edges can cross since the crossing will generate an angle of at most 90°. We state this as Theorem 1.1.

Theorem 1.1 *(a) No two edges of an ST can meet at an angle less than* 120°. *(b) An ST has no crossing edges. (c) Each Steiner point of an ST is of degree exactly three.*

A further consequence was first observed by Courant and Robbins [3] :

Theorem 1.2 *An ST for N contains at most $n - 2$ Steiner points.*

Proof: Suppose that an ST has k Steiner points. Then it has $n + k - 1$ edges. Since each Steiner point has three edges and each terminal at least one, the number of edges must be at least $(3k + n)/2$; the division by 2 accounts for the fact that each edge is counted at two vertices. It follows that

$$n + k - 1 \geq (3k + n)/2$$

or

$$n - 2 \geq k$$

□

An ST with the maximum $n-2$ Steiner points is called a *full Steiner tree* (FST).

Corollary 1.1 *Each terminal is of degree one in an FST.*

Corollary 1.1 can serve as a definition of FST in spaces where the maximum number of Steiner points is not necessarily $n-2$.

Corollary 1.2 *For $n \geq 4$ there exist at least two Steiner points in an FST which are each adjacent to two terminals.*

A topology is a *Steiner topology* (respectively *full Steiner topology*) if it meets the degree requirement of an ST (FST). To construct an SMT, we need only be concerned with trees whose topologies are Steiner. Clearly, for every Steiner topology G, there exists a full Steiner topology F (not necessarily unique) such that $G \in D(F)$. However, not every topology in $D(F)$ is Steiner. Denote by $D_S(F)$ the set of Steiner topologies in $D(F)$. The following theorem was given by Hwang and Weng [10].

Theorem 1.3 *An ST whose topology is in $D(F)$ for some full Steiner topology F is the unique minimum tree among trees whose topologies are in $D_S(F)$.*

Proof: Let T be a tree whose topology $G \in D_S(F)$. Define $f_{ij} = 1$ if $[v_i, v_j]$ is an edge in G and $f_{ij} = 0$ otherwise. Then the length of T is

$$|T| = \sum_{i<j} f_{ij} |v_i - v_j|$$

Consider $|T|$ as a function of the locations of the Steiner points. Since $|v_i - v_j|$ is a norm, $|T|$ is strictly convex except when all edges preserve their directions in a perturbation, then the convexity is not necessarily strict. However, even when the three edges incident to a Steiner point are parallel, moving the Steiner point towards the side where two incident edges lie, though preserving the directions of all three edges, decreases the total length. Hence the strict convexity is guaranteed. Since an ST is a local minimum, the strict convexity guarantees that it is also the unique minimum among trees with topologies in $D_S(F)$. □

We will call this unique ST the *ST of $D(F)$*.

It is easily seen that a Steiner topology can be uniquely partitioned into edge-disjoint subgraphs each of which is a full Steiner topology. Therefore we have

Corollary 1.3 *There exists at most one ST for a given Steiner topology.*

Since relatively minimal trees differ from STs only in angles but not in degrees, we also have

Corollary 1.4 *There exists at most one relatively minimal tree for a given topology.*

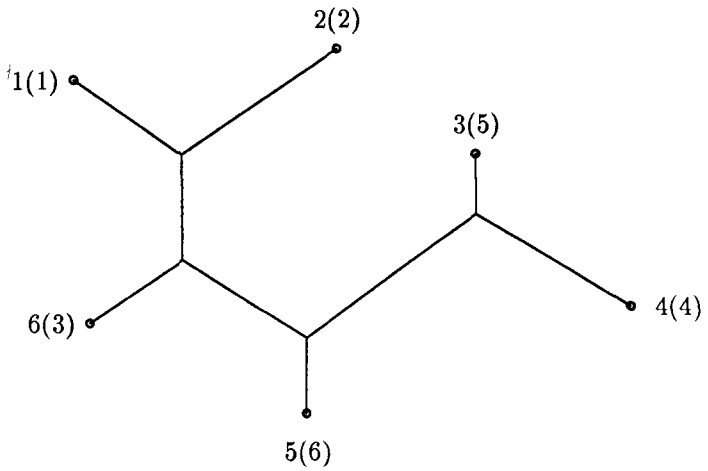


Figure 1.3: Two sequential orders (zigzag order shown in parentheses)

1.4 Full Steiner Trees

The unique decomposition of a Steiner topology into full Steiner topologies induces a unique decomposition of an ST into FSTs. As will be seen in the next chapter, FSTs are much easier to construct than STs. Therefore one way to construct an ST is to construct its FST components. This fact brings out the importance of studying FSTs.

Let T be an FST. Inflate the edges of T to have some width. Then the outside boundary of the inflated T, called the *circumference* of T, forms a cycle going through each terminal once and each edge twice, once in each direction. The cyclic order of the terminals on the circumference will be referred to as the *circumferential order*. Note that this is not the only way to order the terminals in an FST. Werner [18] introduced a *zigzag order* which goes from one terminal to another by making alternating left and right turns at Steiner points (see Fig. 1.3). Also note that both the circumferential and the zigzag order can be set in two ways, one reversing the order of the other, depending on which direction to turn first.

Theorem 1.3 shows that for a given set N of terminals and a full Steiner topology F, the ST in $D(F)$, if it exists, is unique. So, theoretically speaking, the locations of the Steiner points can be uniquely determined from N and F only. However, no such explicit functions are available and the locations can be computed only algorithmically. However, if the ST is known to have a full Steiner topology, then things are different. Hwang and Weng [9] used hexagonal coordinates and gave an algebraic solution of the (hexagonal) coordinates of the Steiner points. Since these coordinates are also coordinates of terminals, the ST and its length can be

explicitly formulated as functions of N and F (see Section 2.3).

Booth [1] used spherical coordinates and obtained a more pleasing formula for the length. By Corollary 1.2, there exist two terminals p_n, p_1 adjacent to a Steiner points s. Use the circumferential order to label the other $n-2$ terminals and define

$$\alpha_k = |p_{k+1} - p_k| \text{ for } k = 1, \ldots, n-1$$

Let $\overrightarrow{p_n p_1}$ be the vector defining the direction of the positive x-axis with s lying in the upper half-plane. Define β_k to be the angle of the vector $\overrightarrow{p_k p_{k+1}}$. Then

$$\overrightarrow{p_k p_{k+1}} = \alpha_k e^{i\beta_k}$$

Also define the angle λ, $0 < \lambda < \pi$, by

$$\overrightarrow{p_1 s} = |p_1 - s| e^{i\lambda}$$

Finally, set $w = e^{i\pi/3}$, and define the sequence $\{y_q\}$, $1 \leq q \leq n-1$, inductively by

(i) $y_1 = 1$, $y_{n-1} = w^{-1}$.

(ii) Given y_s and y_t, $1 \leq s < t \leq n-1$, then

$$w^{-1} y_s + w y_t = y_q$$

where q is the unique integer, $s < q < t$, for which the clockwise paths from P_s to P_{s+1}, P_t to P_{t+1}, and P_q to P_{q+1} have exactly one Steiner point in common.

Booth proved the following theorem using an inductive argument on n.

Theorem 1.4 *The length z of an FST for $N = \{p_1, \ldots, p_n\}$ is given by the formula*

$$z e^{i\lambda} = \sum_{k=1}^{n-1} \alpha_k y_k e^{i\beta_k}$$

Note that λ can also be solved from the above equation.

1.5 Steiner Hulls and Decompositions

A *Steiner hull* for a given set of points N is defined to be a region which is known to contain an SMT. A known Steiner hull allows an SMT algorithm to confine its computations within a given area, hence the smaller a Steiner hull is, the better. The following two lemmas give some useful tools to restrict SMTs.

Lemma 1.1 ([7]) (The Lune property) *Let uv be any line of an SMT. Let $L(u,v)$ be the region consisting of all points p satisfying*

$$|pu| < |uv| \text{ and } |pv| < |uv|$$

$L(u,v)$ is the lune-shaped intersection of circles of radius $|uv|$ centered on u and v. No other vertex of the SMT can lie in $L(u,v)$.

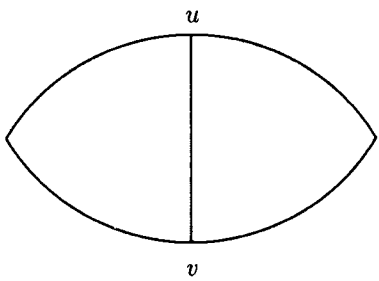


Figure 1.4: A lune

Proof: If q were such a vertex, the SMT would contain either a path from q to u not containing v, or vice versa. In the former case, for example, the SMT can be shortened by deleting $[u, v]$ and adding $[q, v]$, a contradiction. □

Lemma 1.2 ([7]) (The Wedge property) *Let W be any open wedge-shaped region having angle $120°$ or more and containing none of the terminals. Then W contains no Steiner points.*

Proof: Without loss of generality, let W cover the span of angles from $-60°$ to $60°$. Suppose to the contrary that W contains a Steiner point. Let s be the Steiner point in W with the largest x-coordinate. Of the three edges at s, one leaves s in a direction within $\pm 60°$ of the positive x-axis. This edge cannot leave W and so cannot end at a terminal. Furthermore, its endpoint has a larger x-coordinate than s, a contradiction. □

Corollary 1.5 ([7]) *The convex hull of N is a Steiner hull.*

Proof: Each supporting line of the convex hull defines a $180°$ wedge free of terminals. □

Theorem 1.5 *Let H be a Steiner hull of N. By sequentially removing wedges abc from the remaining region, where a, b, c are terminals but Δabc contains no other terminal, a and c are on the boundary and $\angle abc \geq 120°$, a Steiner hull H' invariant to the sequence of removal is obtained.*

Proof: It will first be shown that no SMT can intersect such a wedge abc, hence the wedge is removable. By Lemma 1.1 no edge of an SMT cuts across the wedge or b will be a vertex lying in the lune of that edge. By Lemma 1.2, no Steiner point lies inside of the wedge. Thus the wedge contains no part of an SMT.

To prove the invariance property, suppose that there exists a terminal b' such that $\angle ab'c$ also $\geq 120°$ (see Fig. 1.5).

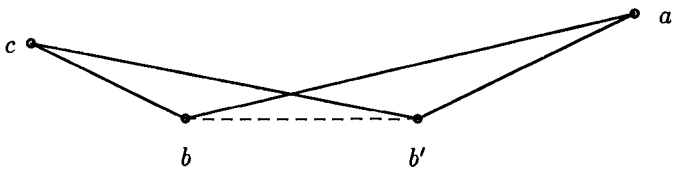


Figure 1.5: Two removal sequences

Then

$$\angle ab'b > \angle ab'c \geq 120°$$

and

$$\angle b'bc > \angle abc \geq 120°$$

Hence if Δabc is removed, then $\Delta ab'b$ can be removed in the next step. Whereas if $\Delta ab'c$ is removed first, then $\Delta b'bc$ can be removed next. In either case one obtains the same Steiner hull. By repeating this argument one can show the invariance of more involved sequences. □

Theorem 1.5 was first given by Cockayne [2] without the invariance property.

Properties of 4-terminal SMTs have been well studied (see Section 5.1). By critically using the results of [4], Hwang, Song, Ting and Du [8] were able to give a sufficient condition for removing a quadrilateral.

Theorem 1.6 *Let a, b, c, d be four points on the boundary of a Steiner hull H of N such that*

(i) $a, b \in N$,

(ii) *$abcd$ is a convex quadrilateral containing no terminals with $\angle a \geq 120°$, $\angle b \geq 120°$,*

(iii) *$\angle bxa \geq \angle a + \angle b - 150°$ where x is the intersection of the two diagonals $[a, c]$ and $[b, d]$.*

Then $H - abcd$ is a Steiner hull.

We omit the proof as it involves different analyses for several subcases. The analyses are based on a thorough knowledge about the 4-terminal SMT. The lack

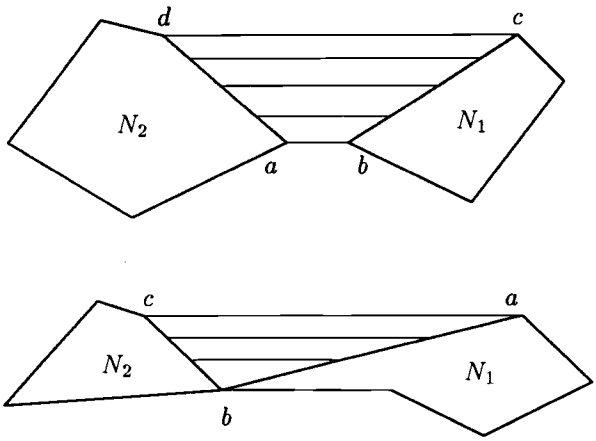


Figure 1.6: Two decompositions

of such knowledge on the k-terminal SMT for $k \geq 5$ explains why no sufficient conditions for removing general k-gons are known.

When a Steiner hull consists of two subregions meeting at one point x, then we can partition the original problem into two subproblems each covering the set of terminals in the respective subregion plus the point x (if it is not a terminal). Since the time complexities of all known SMT algorithms are exponential in n, partitioning a problem into several smaller problems greatly reduces the time required.

Note that the Steiner hull of Theorem 1.6 induces a decomposition of N into N_1 and N_2. The Steiner hull of Theorem 1.5 will induce a decomposition at the point b if b also lies on the boundary (see Fig. 1.6).

Theorem 1.7 ([2]) *The terminals on a Steiner hull have a cyclic order consistent with the circumferential order of any full Steiner topology.*

Proof: Take any three terminals a, b, c on the boundary of the Steiner hull. There exists a vertex d of the ST, d can be one of a, b, c, such that the three Steiner paths ad, bd, cd meet only at d. Since a, b, c are all on the Steiner hull while the three paths cannot go outside of the hull, the relative positions of the three paths are invariant and consistent with the circumferential order. □

Define a *crossing-free cycle* of n points as a closed path which visits each point at least once without two edges crossing each other, except that an edge between two points may be used twice, once in each direction.

Theorem 1.8 ([16]) *There exists a crossing-free cycle which is a Steiner hull.*

Proof: Define the circumference of a nonfull ST as the union of the circumferences of its full subtrees. By skipping all the Steiner points on the circumference of an

SMT and replacing each path between two adjacent terminals (after skipping) by a straight edge, a crossing-free cycle is obtained. □

1.6 The Number of Steiner Topologies

Let $f(n), n \geq 2$, denote the number of full Steiner topologies with $(n-2)$ Steiner points. Clearly, $f(2) = 1$; the unique Steiner topology is an edge connecting two terminals. Note that in a full Steiner topology every terminal is adjacent to a Steiner point. Let F be a full Steiner topology with $n+1$ terminals. If one removes the terminal p_{n+1} and also its adjacent Steiner point, one obtains a full Steiner topology with n terminals. This shows that every full Steiner topology with $n+1$ terminals can be obtained from a full Steiner topology with n terminals by adding a Steiner point s in the middle of one of the $(2n-3)$ edges and adding an edge connecting s to p_{n+1}. Hence

$$f(n+1) = (2n-3)f(n)$$

which has the solution

$$f(n) = 2^{-(n-2)}(2n-4)!/(n-2)!$$

Let $F(n,k)$ denote the number of Steiner topologies with $|N| = n$ and k Steiner points such that no terminal is of degree three. Then $F(n,k)$ can be obtained from $f(k)$ by first selecting $k+2$ terminals and a full Steiner topology on it, and then adding the remaining $n-k-2$ terminals one at a time at interior points of some edges. The first terminal can go to one of $2k+1$ edges, the second to one of $2k+2$ edges , . . ., and the $(n-k-2)nd$ to one of the $n+k-2$ edges. Thus

$$F(n,k) = \binom{n}{k+2} f(k) \frac{(n+k-2)!}{(2k)!}$$

Now consider Steiner topologies with terminals of degree three. Suppose that there are n_3 of them. Such topologies are obtainable from Steiner topologies with $n-n_3$ terminals and $k+n_3$ Steiner points by labeling n_3 of the Steiner points as terminals. Let $F(n)$ denote the number of Steiner topologies with $|N| = n$. Then

Theorem 1.9 ([7])

$$F(n) = \sum_{k=0}^{n-2} \sum_{n_3=0}^{\lfloor (n-k-2)/2 \rfloor} \binom{n}{n_3} F(n-n_3, k+n_3)(k+n_3)!/k!$$

Table 1.1 gives $f(n)$ and $F(n)$ for $n = 2, 3, \ldots, 8$.

Even though $f(n)$ is much smaller than $F(n)$, it is still a *superexponential* function, i.e., increasing faster than an exponential function. The number $f(n)$ can be greatly reduced by using the geometry of the set N. Cockayne proved

n	2	3	4	5	6	7	8
$f(n)$	1	1	3	15	105	945	10395
$F(n)$	1	4	31	360	5625	110880	2643795

Table 1.1: $f(n)$ and $F(n)$ for $n = 2, 3, ..., 8$

Theorem 1.10 ([2]) *If all terminals lie on a Steiner hull, then the number of full Steiner topologies is* $\binom{2n-4}{n-2}/(n-1)$, *the* $(n-2)nd$ *Catalan number.*

Proof: Let F denote a full Steiner topology. Let p be a terminal adjacent to a Steiner point s in F. View F as a graph which connects p to the root s of a rooted binary tree. Use the well-known parenthesis representation for the rooted binary tree with $n-1$ leaves. By Theorem 1.7, the sequence of leaves is independent of F if the same p is chosen. Thus there is a one-to-one correspondence between the set of full Steiner topologies with n terminals and the set of parenthesis representations with $n-1$ symbols, whose number is the $(n-2)nd$ Catalan number. □

Corollary 1.6 *If* i *terminals are on the Steiner hull, then the number of Steiner topologies is* $\frac{n!}{i!}\binom{2n-4}{n-2}/(n-1)$.

Note that

$$\binom{2n-4}{n-2}/(n-1) \cong \frac{4^n}{16n\sqrt{\pi n}}$$

Thus if there are only constant number of terminals not on the Steiner hull, then there are only exponentially many Steiner topologies, instead of superexponentially many. By Theorem 1.8 there exists a crossing-free cycle which is a Steiner hull and all terminals lie on the hull.

1.7 Computational Complexity

One of the primary tools used to understand the computational complexity of combinatorial problems has been the theory of NP-completeness [5] . The theory is normally applied to decision (yes/no) problems. The optimization problem:

- **GIVEN:** A set N of terminals in the Euclidean plane.
- **FIND:** A Steiner tree of shortest length spanning N.

can be recast as a decision problem:

- **GIVEN:** A set N of terminals in the Euclidean plane and an integer B.
- **DECIDE:** Is there a Steiner tree T that spans N such that $|T| \leq B$.

Unfortunately both of these problems suffer from numerical difficulties, involving computations of irrationals (on finite-precision machines) in addition to the combinatorial difficulties involved in exploring the many topologies. Hence another related problem has been proposed, that is perhaps simpler, known as the *discrete Euclidean Steiner problem* :

- **GIVEN:** A set N of terminals with integer coordinates in the Euclidean plane and an integer B.
- **DECIDE:** Is there a Steiner tree T that spans N, such that all Steiner points have integer coordinates, and the *discrete length* of T is less than or equal to B, where the discrete length of each edge of T is the smallest integer not less than the length of that edge.

The *class NP* is the class of decision problems that can possibly be solved in polynomial time, when the answer is "yes", provided a hint can be given as to the form of the solution. (See [5] for a technical definition of the "nondeterminism" used here.) For example, for the discrete problem above it would be easy to verify that a tree T has discrete length bounded by B, if such a T exists, when provided with the "hint" of the topology of T and the locations of its Steiner points. Currently the original Euclidean Steiner decision problem is not known to be in NP, and may not be, due to the difficulty of the numeric computations. There are many decision problems found in this book. However this problem is the only one not known to be in NP.

A problem is *NP-hard* if it is as "hard" as any problem in NP. In this sense a decision problem A is as hard as B if, with polynomial time overhead, an algorithmic solution to A can be used to solve B. (Again, [5] has formal definitions.) A problem is *NP-complete* if it is both NP-hard and in NP. The typical way to show a new problem A is NP-hard is to show it is as hard as B, where B is known to be NP-hard (or NP-complete).

The discrete Euclidean Steiner decision problem has been shown to be NP-hard by Garey, Graham and Johnson [6]. They used a mapping to the known NP-complete problem known as the *exact cover by 3-sets* (defined in Section 1.5 of Part II, where a simpler mapping is exhibited). The details of the mapping used are quite lengthy and are omitted here. Since this discrete problem is also in NP, as argued above, it follows that it is NP-complete.

Return to the original Euclidean Steiner decision problem. Suppose that coordinates are in symbolic expressions which can be evaluated in time bounded by a polynomial in m and $\log \Delta$, where m is the number of bits in an expression and $\pm\Delta$ is the approximation error range. Any polynomial time algorithm which solves the ESP can still solve the exact cover by 3-sets problem. Using such arguments it follows that the ESP is NP-hard [6]. However it is not known to be NP-complete, since its membership in NP is open.

Garey, Graham and Johnson in fact gave an even stronger result. Define a *fully polynomial approximation scheme* to be an algorithm which, for each instance I of the problem and each $\epsilon > 0$, finds a solution for I with value v satisfying

$$\frac{v}{OPT(I)} \leq 1 + \epsilon$$

and whose running time is bounded by a polynomial in $1/\epsilon$ and the length of the input for I. They showed that no such scheme exists for the ESP unless every problem in NP has a polynomial time solution.

In this book we adopt the convention ([5], p. 114) of referring to the optimization version of NP-hard decision problems as being "NP-hard", even when the

corresponding decision problem is in fact known to be NP-complete. We emphasize that the only decision problem in this book that is NP-hard but not NP-complete is the original ESP above.

1.8 Physical Models

Three physical devices have been proposed to model the ESP.

(i) *The string model* [14,15]. This model generates a relatively minimal tree. Let G be a given Steiner topology of a set N. A sheet has holes at locations corresponding to the terminals. Strings of equal lengths and with equal weights attached to them under the sheet run through the holes. A Steiner point is represented by a sliding ring connected above the sheet to the three strings corresponding to the three edges. The strings and rings settle into a relatively minimal tree with respect to G (see Fig. 1.7). A variant uses pegs at the holes and rolls elastic bands over the pegs for the strings.

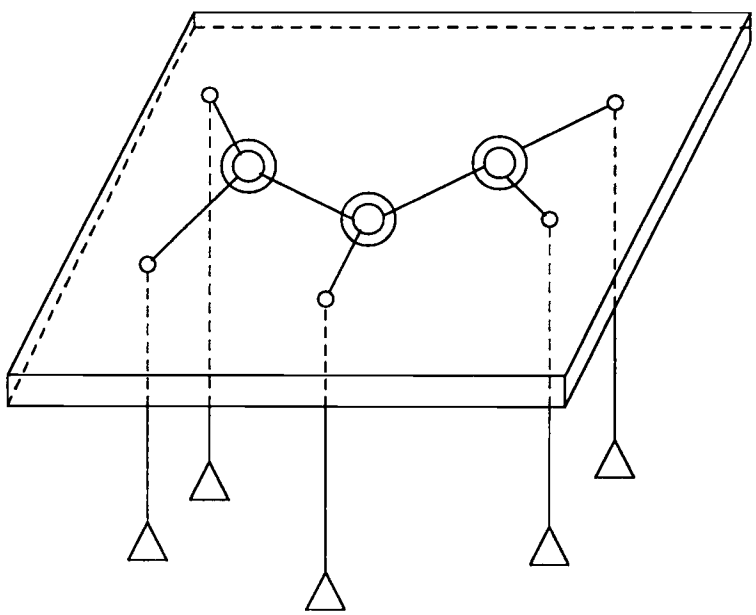


Figure 1.7: String model

(ii) *The soap film model* [14]. This model generates an ST with some Steiner topology not determined in advance. Two parallel transparent plates are separated by posts located at the terminals. The whole thing is then dipped into soapy water. When lifted, a layer of thin film interconnecting the posts into an ST (with height) is formed (see Fig. 1.8). Repeating the experiment may generate various STs with different topologies.

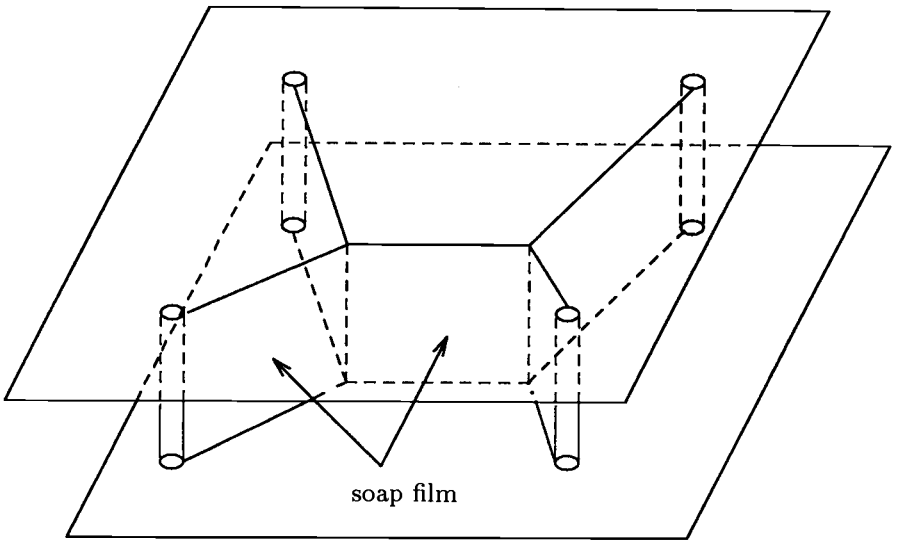


Figure 1.8: Soap film model

(iii) *The membrane model* [17]. This model generates a heuristic SMT, but not much is known about the solution. A rubber membrane circumscribed by a ring is pinned to a sheet at locations of terminals. The lifting of the ring creates a canonical funnel with little hills and valleys at the bottom (see Fig. 1.9). The points at which forces of the membrane are most balanced are thought to be natural candidates for Steiner points.

The main disadvantages of these physical models are:

1. They do not produce SMTs.
2. It could be time-consuming to construct a large model.
3. Mechanical errors can build up in large models.

Therefore, in the rest of the text we will concentrate on mathematical solutions to the ESP.

References

[1] R. S. Booth, Analytic formulas for full Steiner trees, *Discrete Comput. Geom.* **6** (1991) 69-82.

[2] E. J. Cockayne, On the Steiner problem, *Canad. Math. Bull.* **10** (1967) 431–450.

[3] R. Courant and H. Robbins, *What is Mathematics?* Oxford Univ. Press, New York (1941).

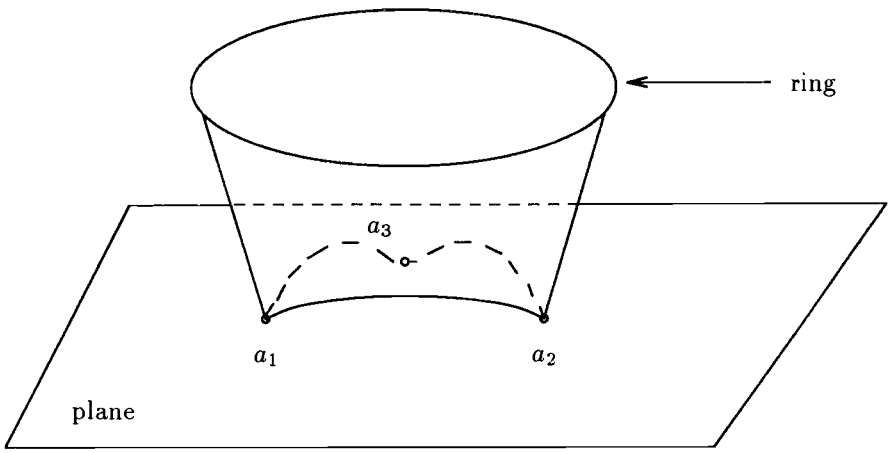


Figure 1.9: Membrane model

[4] D. Z. Du, F. K. Hwang, G. D. Song and G. Y. Ting, Steiner minimal trees on sets of four points, *Discrete Comput. Geom.* **2** (1987) 401–414.

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

[6] M. R. Garey, R. L. Graham and D. S. Johnson, The complexity of computing Steiner minimal trees, *SIAM J. Appl. Math.* **32** (1977) 835–859.

[7] E. N. Gilbert and H. O. Pollak, Steiner minimal trees, *SIAM J. Appl. Math.* **16** (1968) 1–29.

[8] F. K. Hwang, G. D. Song, G. Y. Ting and D. Z. Du, A decomposition theorem on Euclidean Steiner minimal trees, *Discrete Comput. Geom.* **3** (1988) 367–382.

[9] F. K. Hwang and J. F. Weng, Hexagonal coordinate systems and Steiner minimal trees, *Discrete Math.* **62** (1986) 49–57.

[10] F. K. Hwang and J. F. Weng, The shortest network under a given topology, *J. Algorithms*, to appear.

[11] V. Jarník and O. Kössler, O minimálních grafech obsahujících n daných bodu, *Čas. Pěstování Mat.* **63** (1934) 223–235.

[12] H. W. Kuhn, Steiner's problem revisited, in G. B. Dantzig and B. C. Eaves (eds.) *Studies in Optimization, Studies in Math.* **10** Math. Assoc. Amer. (1975) 53–70.

[13] Z. A. Melzak, On the problem of Steiner, *Canad. Math. Bull.* **4** (1961) 143–148.

[14] W. Miehle, Link-length minimization in networks, *Oper. Res.* **6** (1958) 232–243.

[15] G. Polya, *Induction and Analogy in Mathematics*, Princeton Univ. Press, New Jersey (1954).

[16] W. D. Smith, How to find Steiner minimal trees in Euclidean d-space, *Algorithmica* **7** (1992) 137-177.

[17] J. Soukup, Minimum Steiner trees, roots of a polynomial and other magic, *ACM/SIGMAP Newsletter* **22** (1977) 37–51.

[18] C. Werner, Networks of minimal length, *The Canad. Geographer* **13** (1969) 47–69.

[19] M. Zacharis, Encyklopädie der Mathematischen Wissenschaften, III AB9.

This Page Intentionally Left Blank

Chapter 2

Exact Algorithms

We briefly survey what Torricelli, Cavalieri, Simpson, Heinen and Bertrand found about the ST for three terminals a, b and c. If one of the angles of Δabc is at least 120°, then by Heinen's or Bertrand's result, the ST consists of simply the two edges subtending the obtuse angle. So assume that all internal angles of Δabc are less than 120°. Suppose one replaces a and b by a point d which forms an equilateral triangle with a and b such that d is on the opposite side to the Steiner point s (in reference to line ab). The position of s for the 3-terminal case is known to be on the same side as c. Thus the location of d is determined. Let C be the circle circumscribing the three points a, b and d. By Torricelli's and Cavalieri's results s is the intersection of $[c, d]$ with C. Hence the location of s is determined (see Fig. 2.1). Heinen showed that the length of line $[c, d]$ is equal to the length of the Steiner tree.

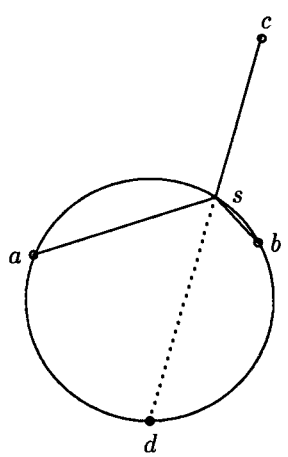


Figure 2.1: A 3-point SMT algorithm

The above 3-terminal algorithm provides a base for generalizing to n terminals. The step that substitutes d for a and b will be called the *merging stage*, and the step that locates s and adds the connections $[a, s]$ and $[b, s]$ is the *reconstruction stage.*

In the next three sections we present a fundamental exact algorithm based on the two stages mentioned above, and improvements on that algorithm. Section 2.4 presents a purely numerical approach. The next section presents a technique for pruning the search. The last three sections present recent exact algorithms.

2.1 The Melzak Algorithm

Melzak [13] proposed the first finite algorithm for ESP. His approach is to find a relatively minimal tree for every topology on N, and select a shortest one among these trees as an SMT. Furthermore, only those relatively minimal trees which are STs need be constructed, since an SMT must be an ST. Cockayne [4] streamlined the Melzak algorithm and emphasized that this approach requires only a subroutine to find FST, since if a topology G is not full, then it can be broken down into components $G_1, \ldots, G_m$ such that G_i is a full topology of a subset N_i of N with $\cup_i N_i = N$ and two subsets intersect at at most one terminal. Thus the core of the Melzak algorithm is an algorithm to construct FST, which will be called the *Melzak FST algorithm.* The Melzak FST algorithm is an iterative version of the 3-point algorithm given at the beginning of this chapter.

To find an SMT, the Melzak algorithm considers all topologies, using the Melzak FST algorithm to construct its components, and stores the tree if all full components exist. At the end all stored trees are compared and a shortest one is selected as an SMT. The Melzak algorithm has been coded by Cockayne and Schiller [7], and later improved by Boyce and Seery [2].

The Melzak FST algorithm constructs an FST for a given full topology. (If an FST does not exist for that topology, then the algorithm comes up empty.) Let F denote the given full topology. The Melzak FST algorithm consists of a merging stage and a reconstruction stage. During the merging stage one reduces the number of terminal points from n to 2, one at a time. At the beginning, the n terminals are the only terminal points and the associated topology is F. Call two terminal points *siblings* if they are adjacent to the same Steiner point in the associated topology. At each merging step two arbitrary siblings a and b are replaced by a new terminal point, called an *equilateral point* or an *E-point*, which forms an equilateral triangle with a and b. Denote this E-point by (ab). (There are two choices of (ab), one on each side of line ab. This point will be discussed later.) A corresponding change is made on the associated topology by deleting a and b with their edges to the Steiner point s, and treating s now as a terminal point. Note that the new associated topology has one fewer terminal point and one fewer Steiner point (the point s), but remains a full topology. The merging stage ends when the associated topology consists of only two terminal points and no Steiner point. Note that every terminal except at most one is replaced during the merging process.

The reconstruction stage starts by connecting the two terminal points left at the end of the reduction stage by a straight edge, again, called a *Simpson line*. At each reconstruction step the tree T already constructed is modified into a new tree T' by replacing an edge of T which has an E-point by a Steiner point and its three edges. Again refer to Fig. 2.1 except that c is not necessarily a terminal. Let $[c, d]$ be such an edge in T where d is generated in the merging stage by replacing two terminal points a and b. Let C denote the circle circumscribing the three points a, b and c. If the 120° arc ab does not intersect the edge $[c, d]$ at an internal point, then the reconstruction stage ends and no FST exists for the topology F. If it intersects at an internal point s, then T' is obtained from T by deleting $[c, d]$ and inserting $[a, s]$, $[b, s]$ and $[c, s]$. The reconstruction step continues until either at some step no internal intersection exists or the new tree contains all terminals.

Consider the modification of T into T'. Without loss of generality, assume that bad is the counterclockwise order of the three points on the circle C. Since both $\angle dsb$ and $\angle asd$ face a 120° arc of C, they are 60° each. This proves that the three edges $[a, s]$, $[b, s]$ and $[c, s]$ meet at 120° and the output of the reconstruction stage, if it exists, is an FST which, by Theorem 1.3, is the unique FST. On the other hand, suppose that T is an FST for F. Consider the merging sequence which always chooses the third equilateral point on the opposite side of the Steiner point adjacent to the two replaced sibling terminal points. Clearly, for such a merging sequence the reconstruction can always proceed until T is reconstructed. Hence the existence of an FST implies the existence of a merging sequence for which the Melzak FST algorithm finds the FST.

Assume that the geometric constructions, like finding a third equilateral point, finding a circumscribing circle, and finding the intersection of an arc with a line, take constant time. Then the reconstruction stage is a one-pass linear time algorithm. Unfortunately, the merging stage needs exponential time since at each step we have two choices of the E-point, and backtracking is required to determine the correct choice. The reason that there are two choices is, unlike the 3-point case, that one does not know in general which side the Steiner point in question lies. Although many ad hoc procedures have been offered [8] to eliminate some choices, they do not affect the exponential order of magnitude.

The next section reviews an $O(n)$ time implementation of the Melzak FST algorithm due to Hwang [10]. The crux of the implementation is a set of criteria which correctly determines the E-point without backtracking. Before leaving this section, observe that by repeatedly using Heinen's result on a Simpson line for 3-point set, the following result holds.

Theorem 2.1 *The length of the FST equals that of a Simpson line.*

2.2 A Linear-Time FST Algorithm

The following is an account of the Hwang linear variant of the Melzak FST algorithm with some details provided by Smith [14].

Let F be a given full topology on n points. Construction for the $n = 3$ case

was given before and construction for the $n = 4$ case is also straightforward. So assume $n \geq 5$.

Let r be an arbitrary terminal in F. Treat F as a rooted binary tree with root r. Compute the distance of other vertices from r, where "distance" here means the number of edges between the two vertices. This can be done in $O(n)$ time by a breadth-first search. Let a be a terminal farthest away from r. Since $n \geq 5$, the distance from r to a is at least three. Let s_1 and s_2 be two Steiner points on the ra path such that a is adjacent to s_1 and s_1 is adjacent to s_2. Since each Steiner point is of degree three, s_1 must be adjacent to a third vertex b, and s_2 to v, where b and v are not on the ra path. Furthermore, b must be a terminal or a cannot be a terminal farthest away from r (see Fig. 2.2).

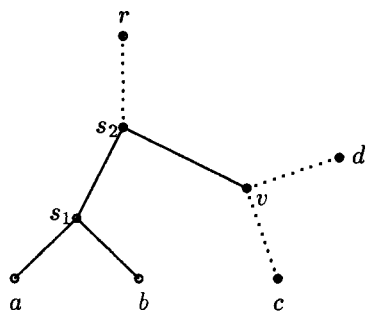


Figure 2.2: A specified topology

If v is a terminal, then delete $\{a, b, s_1\}$ and their incident edges from F and add the E-point (ab) and edge $[s_2, (ab)]$, where (ab) lies on the opposite side of line ab from v. If v is a Steiner point, let c and d be its two children nodes. Then c and d must be terminals by the definition of a. If c and d lie on the same side of line ab, then do the same as the previous case except (ab) lies on the opposite side of line ab from c. If a and b lie on the same side of line cd, then delete $\{c, d, v\}$ and their incident edges and add the E-point (cd) and edge $[s_2, (cd)]$, where (cd) lies on the opposite side of line cd from a. If lines ab and cd cross, then the algorithm stops since no FST exists for F by Theorem 1.1.

The above procedure is reiterated (with the same r but different a) until only four terminal points are left. The FST is then found by brute force.

The correctness of the algorithm and its $O(n)$ time requirement were verified by Hwang [10]. Therefore

Theorem 2.2 *There exists an $O(n)$ time implementation of the Melzak FST algorithm.*

2.3 Two Ideas on the Melzak Algorithm

The major inefficiency of the Melzak algorithm comes from the superexponential number of Steiner topologies needed to be processed. The following method cuts down the superexponentiality to exponentiality.

From Theorem 1.8 there exists a crossing-free cycle which is a Steiner hull. Since it is not known which crossing-free cycle is a Steiner hull in advance, one has to process all crossing-free cycles. Ajtai et al. [1] showed that there are at most C^n crossing-free cycles with $C < 10^{13}$. Smith [14] and Hayward [9] strengthened this upper bound considerably. Smith also showed that all the crossing-free cycles can be generated in exponential time.

Consider a given crossing-free cycle C. Suppose that C consists of d edge-disjoint subcycles. Then the corresponding SMT must also be a union of d FSTs each contained in a subcycle. The number of full Steiner topologies in the i^{th} subcycle with n_i terminals cyclically ordered is

$$2^{-(n_i-2)}(2n_i-4)!/(n_i-2)!$$

which was shown in Section 1.6 to be an exponential number. The total number of Steiner topologies needed to be processed by the Melzak FST algorithm for one crossing-free cycle is the product of the above numbers over i. Finally, there are exponentially many crossing-free cycles to be examined. Since a product of two exponential numbers is still exponential, the Melzak FST algorithm needs to be run only exponentially many times. Although this algorithm is of great theoretical interest, the large value of C somewhat circumvents its practicality.

Clark [3], and also Hwang and Weng [11], proposed an algebraic version of the Melzak FST algorithm which replaces the geometric constructions like finding intersections, constructing triangles and circles by solving a system of linear equations. Besides the possible advantage in easier coding, it also does not have to go 2-pass, one for the merging stage and the other for reconstruction, as in the original geometric version. Hwang and Weng found it advantageous to use a hexagonal coordinate system.

Let U, V, W be the three axes going through the origin and cutting the plane into six 60° cones. Represent each point in space by a vector (u, v, w) (there is more than one way to choose a representation system). Hwang and Weng [11] proved,

Lemma 2.1 $u + v + w = 0$ *for any point* p.

Let T denote an FST and G its topology. First assume that the lines in T are all parallel to the three axes, U, V and W. Such a tree will be called a *hexagonal tree*. An edge in a hexagonal tree can consist of more than one segment. Note that if a line is parallel to an axis, then any two points on the line share a same coordinate. This observation allows one to sequentially write down the coordinates of all Steiner points from the coordinates of the terminals, starting from Steiner points adjacent to two terminals. Since G is a bipartite graph, the vertices in G can be partitioned into two disjoint subsets N_1 and N_2 such that edges exist only

between N_1 and N_2. By repeatedly applying Lemma 2.1 to each Steiner point in G, the *characteristic equation* of T,

$$\sum_{p \in N_1} [u(p) + v(p) + w(p)] - \sum_{p \in N_2} [u(p) + v(p) + w(p)] = 0$$

is obtained. Note that all the $u(p)$, $v(p)$, $w(p)$ will cancel out except those derived from terminals.

Now consider the general case that the lines of an FST T are parallel to the axes only after a counter-clockwise rotation of angle θ. Define $l = \cos\theta$ and $k = \sin\theta/\sqrt{3}$. Then the new coordinates can be obtained from the original coordinates through the transformation

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{bmatrix} l & k & -k \\ -k & l & k \\ k & -k & l \end{bmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Note that the three transformations θ, $\theta + 120°$, $\theta + 240°$ all lead to the same hexagonal coordinate system except differing in calling which one U. Once the choice is made, the characteristic equation can then be expressed as a linear (homogeneous) function in l and k. Therefore l and k, hence θ, can be solved by noting that $l^2 + 3k^2 = 1$. Since the two solutions (l, k) and $(-l, -k)$ will lead to the same coordinates except reversing all signs, one may assume without loss of generality that $k \leq 0$. In these senses a characteristic equation leads to a unique transformation.

Weng [16] observed that any edge in an SMT can be replaced by two segments of a hexagonal tree (for a given axis-system) connecting the same two end points so that its length is increased by a factor at most $2/\sqrt{3}$ (the maximum occurs when the two segments are of equal length). It follows

Theorem 2.3 *The length of an SMT is lower bounded by $\sqrt{3}/2$ times the length of a minimum hexagonal tree for any given axis-system.*

This result will be used in the proof of the Steiner ratio conjecture in Chapter 3.

2.4 A Numerical Algorithm

As shown in the proof of Theorem 1.3, the length of a tree in $D(F)$ is a strict convex function of the locations of Steiner points. Hence the local optimum is the global optimum and one may attempt to solve for these locations by setting the first partial derivatives to zero. However, this yields a system of nonlinear equations and a numerical method is required. Smith [14] commented that the length function has nonsmooth, complicated behavior at and near the optimum, so that the usual numerical methods are ineffective. He proposed an iterative procedure where, at each step, all of the Steiner points are updated simultaneously and compatibly as follows:

Let (x_k, y_k) denote the Euclidean coordinate of the Steiner point k. At the i^{th} step, $i = 0, 1, \ldots$, solve the system of $2n-4$ linear equations

$$x_k^{i+1} = \frac{\sum_j \frac{x_j^{i+1}}{[(x_j^i - x_k^i)^2 + (y_j^i - y_k^i)^2]^{1/2}}}{\sum_j \frac{1}{[(x_j^i - x_k^i)^2 + (y_j^i - y_k^i)^2]^{1/2}}}$$

$$y_k^{i+1} = \frac{\sum_j \frac{y_j^{i+1}}{[(x_j^i - x_k^i)^2 + (y_j^i - y_k^i)^2]^{1/2}}}{\sum_j \frac{1}{[(x_j^i - x_k^i)^2 + (y_j^i - y_k^i)^2]^{1/2}}}$$

where each summation is over all j such that $[j, k]$ is an edge in F. Smith proved

Theorem 2.4 *From all initial choices of Steiner point coordinates (x_k^0, y_k^0), $k = 1, \ldots, n-2$, except for a set of measure zero in R^{2n-4}, the iteration converges to the unique optimum Steiner point coordinates. This convergence happens in such a way that the sequence of the lengths is decreasing over the iteration.*

Smith also noted that the system of linear equations in each iteration can be solved in $O(n)$ time (using floating point operations) by a straightforward application of the Gaussian elimination method, and that the rate of convergence is geometric in general. He warned that it is risky to stop the iteration when (x_k^{i+1}, y_k^{i+1}) is close enough to (x_k^i, y_k^i) for all $k = 1, \ldots, n-2$. A better idea is to check whether the angles at Steiner points are close enough to 120°.

While the numerical algorithm presented here may not offer any significant advantage over the Melzak algorithm in the Euclidean plane, it can be easily generalized to high-dimension while the Melzak algorithm cannot. We will study the high-dimensional ESP in Section 6.1.

2.5 Pruning

Pruning means to cut short a running of the Melzak FST algorithm before its completion by recognizing that the tree partially constructed cannot be a subtree of an SMT. Furthermore, any other topology with the same partial construction can be scratched from being considered to produce an SMT.

Although many pruning techniques have been scattered over the literature (see [8] for a good collection), Winter [17] proposed a systematic way to use them to recognize the nonoptimality of an E-point. This is done by associating with each E-point an arc $p_i p_j$ on which the Steiner point corresponding to the E-point must lie. Let v_i and v_j be the two terminal points yielding the E-point. Then arc $p_i p_j$ = arc $v_i v_j$ originally. The pruning techniques keep shrinking the arc $p_i p_j$ until it disappears (p_i crosses p_j). The nonexistence of the Steiner point implies that any FST requiring the generation of this E-point at the reduction stage does not exist.

Recall that the lune property (Lemma 1.1) forbids any vertex to lie in the lune $L(u,v)$ where $[u,v]$ is an edge of an SMT. Since $[v_i,s]$ is an edge, p_j can be pushed towards v_i until $L(v_i,p_j)$ contains no terminals. Similarly, we can push p_i.

The *deciding region* for an E-point (ab) is the region where no points of any SMT containing (ab) can lie within.

Corollary 2.1 ([8]) *$I = \cap_s\{L(s,a) \cup L(s,b)\}$, where s ranging over the* 120° arc ab, *is a subset of the deciding region.*

It was also shown that I can be approximated by the region bounded by the 120° arc ab and two 60° curves ao and bo centered at $o-b$ and $o-a$ respectively, where o is the center of the circle circumscribing a, b and (ab). Furthermore, the deciding region cannot be much different from I as any point p for which $\angle apb < 120°$ cannot be in the deciding region.

Winter [17] extended the wedge property (Lemma 1.2) to the following case:

Corollary 2.2 *Let* arc p_ip_j *be the arc associated with the E-point e. Draw lines ep_i and ep_j, and let C_e denote the open region enclosed by these two lines and* arc p_ip_j. *Also draw* 60° *cones C_i and C_j at p_i and p_j such that $C_i \cap C_e = ep_i$ and $C_j \cap C_e = ep_j$. Then there must exist two terminals, one in $C_e \cup C_i$ and the other in $C_e \cup C_j$. If the two terminals are not there, then all constructions containing that E-point can be scratched.*

He also proved (see Fig. 2.3)

Theorem 2.5 *Let $e = (e_iv_j)$ and $e_i = (v_{i1}v_{i2})$ be two E-points. Let* arc p_ip_j *and* arc $p_{i1}p_{i2}$ *be associated with e and e_i respectively. Let R denote the open region enclosed by lines e_ip_{i1}, e_ip_{i2} and* arc $p_{i1}p_{i2}$. *Then* arc p_ip_j *can be shrunk to* arc $p_i'p_j' = R \cap$ arc p_ip_j.

Proof: Let s and s_i be the Steiner points on arc p_ip_j and arc $p_{i1}p_{i2}$ respectively. Then s lies on the extension of $[e_i,s_i]$. Theorem 2.5 follows immediately. □

Note that if $R \cap \text{arc}\, p_ip_j = \emptyset$. Then the E-point can be eliminated.

Simulation results given by Winter showed these pruning techniques to be very effective.

2.6 The GEOSTEINER Algorithm

Winter [17] proposed an approach to find an SMT which can be considered as reverse of the Melzak approach. The Melzak approach considers all topologies on N and partitions each such topology into full subtopologies. In the reverse version, full subtopologies are first constructed and then combined into a topology on N. More specifically, an enumeration scheme is developed to generate full topologies for each subset of N. Pruning is heavily used to screen out subsets for which no ST

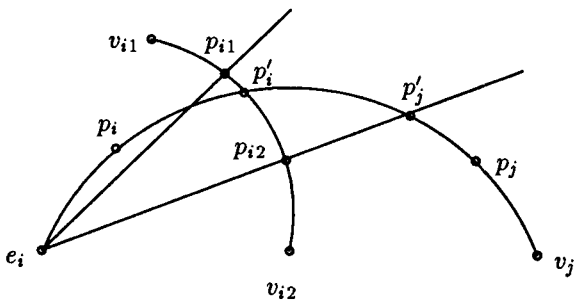


Figure 2.3: Arc shrinking

exists. If a full topology cannot be eliminated on a subset, then the Melzak FST algorithm is used to construct an FST on this subset. In fact, only the merging stage needs be done here and the length of the Simpson line is stored along with the merging process. For any set of terminals for which a Simpson line is stored, the reconstruction stage is done only for the topology with the shortest Simpson line. All possible unions of existing FSTs are examined through length tests, degree tests and cycle tests to see if the union is an ST for N. A shortest such union is an SMT for N.

Note that the only time that the nonexistence of an FST is discovered during the merging stage is when lines ab and cd cross in the Hwang linear variant as discussed in Section 2.2. Otherwise, one can always replace two terminal points by an E-point and proceed. Thus much time can be wasted in the merging stage only to discover at the reconstruction stage that the corresponding FST does not exist. The GEOSTEINER algorithm improves on this by doing the screening while constructing FSTs for subsets of terminals. Furthermore, this screening is done in a dynamic programming sense to rule out all FSTs which contain the forbidden FST as a subtree. For example, if $R \cap \text{arc } p_i p_j = \emptyset$ in Fig. 2.3, then one needs not look into any FST whose construction will encounter a stage at which the topology has v_{i1} and v_{i2} as siblings (adjacent to a Steiner point s_i) and s_i and v_j as siblings.

This reverse approach is practical only if the number of existing FSTs is small so that no excessive storage is required and the number of unions of these FSTs is manageable. Winter's simulation results showed that this is indeed the case. In fact, he stated that the number of FSTs never exceeded 100 in his experiments of up to 20 terminals. Computation time was less than 30 seconds (a UNIVAC-1100 computer was used) for all randomly generated sets of up to 15 terminals. For $n \geq 15$, the computation time needed to extract the STs from stored FSTs dominated that for the FST construction.

Cockayne and Hewgill [5] coded the GEOSTEINER algorithm into a version which they called EDSTEINER86. They used the decomposition theorems to reduce the size of the problem, and introduced an incompatibility matrix for stored FSTs to speed up the union-forming stage. This brought up the solvable range to $n \leq 17$, and also the time spent in FST construction now dominated. More recently Cockayne and Hewgill [6] developed a new version called EDSTEINER89. In this work they claimed that it is the prohibitive time spent in SMT extraction for $n > 17$ which prevents EDSTEINER86 to go further. They devised a better incompatibility matrix to overcome this problem and reported a solvable range of $n \leq 32$. Again, the time spent in FST construction dominated. However, for larger problems up to 100-point sets and a 20-hour cutoff time, they found all unfinished problems stuck at the SMT-extraction stage. Finally, they ran a hundred 100-terminal problems and found that 77 of them completed in a 12-hour cutoff time. Again, the FST construction dominated in time on these completed problems.

2.7 The Negative Edge Algorithm

Trietsch and Hwang [15] observed that every topology on N belongs to a $D_S(F)$ for some full topology F. Furthermore there exists a unique ST of $D_S(F)$. Therefore, one approach to find an SMT is to find the ST of $D_S(F)$ for every F and select a shortest ST. The core of such an approach is an algorithm to find the ST of $D_S(F)$ for a given F. Trietsch and Hwang proposed the *negative edge algorithm* along this line.

Referring back to Section 2.1, let $[c, d]$ be an edge of a partially constructed tree T such that d is generated in the merging stage by replacing two terminal points a and b. Suppose that $[c, d]$ does not intersect the 120° arc ab internally. Then the Melzak FST algorithm ends with no ST constructed. This can happen in two ways. The first case is that c lies inside the circle C circumscribing the three points a, b and d. The second case is that $[c, d]$ intersects C at a point s outside of the 120° arc ab, where s can be a or b or d. In the first case we extend $[c, d]$ to meet the 120° arc ab at s. (By the Hwang linear variant, c and d must lie on opposite sides of ab; hence s must lie on the 120° arc ab.) In either case s will be called a *complementary point.* The three edges at s from vertices a, b, c form two 60° angles (see Fig. 2.4). The middle edge is referred to as a *negative edge* and its length is treated as negative.

By substituting a complementary point for a nonexistent Steiner point, the Melzak FST algorithm always yields an interconnecting tree, which will be called a

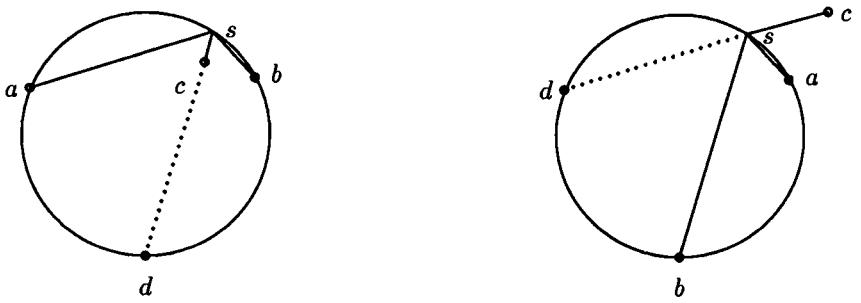


Figure 2.4: Complementary point and negative edge

complementary tree. The total length of a complementary tree, with some lengths negative, is known as its *complementary target value.*

Whenever the Melzak FST algorithm yields a complementary tree, then an FST does not exist for the given full topology F. However, the ST of $D_S(F)$ may exist. The following procedure catches this ST by shrinking negative edges.

Consider a generated complementary tree. A first thought might be to get rid of all negative edges altogether. However, this operation will not yield an ST as there are examples that shrinking negative edges can turn some positive edges negative. There is no proof that repeating the operation will arrive at an ST.

Trietsch and Hwang proposed an algorithm, called the *negative edge algorithm*, which shrinks one negative edge at a time, and always a longest one. The algorithm starts by applying the Melzak FST algorithm to yield a complementary tree. If the tree contains no negative edge, the algorithm stops and the ST is obtained. If the longest negative edge is between two complementary points, then this topology can be discarded since shrinking this negative edge will create a point of degree four. Therefore it suffices to consider the case that there exists a longest negative edge between a terminal v and a Steiner point s. The algorithm shrinks this longest negative edge. This implies a partition of the original full Steiner topology F into two full subtopologies at v. The negative edge algorithm now applies to these two subtopologies. Suppose that two STs are generated for these two subtopologies. The algorithm then inspects whether the two edges at v meet at an angle at least 120°. If it does, then the ST of $D_S(F)$ is found. If not, then a Steiner point has to be reintroduced to v, i.e., v is split. The Melzak FST algorithm now applies to the full component containing v, and if any negative edges emerge, the algorithm runs another iteration again.

It seems that the negative edge algorithm may run in looping. However, Trietsch and Hwang proved that the complementary target value is increasing at every iteration and hence looping cannot occur. Since the number of complementary trees is finite, the algorithm has to end and the output tree contains no negative edge (otherwise, the target value can be further increased). Thus

Theorem 2.6 *The negative edge algorithm is finite.*

Unfortunately, there is no proof that the algorithm stops in polynomial time.

2.8 The Luminary Algorithm

The algorithms reported in Section 2.7 and in this section have the same goal of constructing the unique ST of $D_S(F)$. However, they achieve that goal with very different philosophy. The negative edge algorithm uses the Melzak FST algorithm to generate an interconnecting tree, either an FST or a complementary tree, and then tries to fix the problem of negative edges if they appear. The *luminary algorithm*, proposed by Hwang and Weng [12], takes full care during the construction process to make sure that the output is the unique ST of $D_S(F)$, with no post-processing. While constructing a tree T, a *track* denotes a line which may contain an edge of T. The Melzak FST algorithm retains every track of the FST of F until proven wrong during the merging stage. Similarly, the luminary algorithm retains every track of the ST of $D_S(F)$ until proven wrong during its merging stage. However, since that ST can assume any topology in $D_S(F)$ at the beginning, the luminary algorithm has to carry many more tracks. Such a track is called a *ray* since it has an orientation during construction. An object which radiates rays is called a *luminary*.

The luminary algorithm has merging and reconstruction stages similar to the Melzak FST algorithm. Except that in the Melzak FST algorithm the basic units of the merging operations are terminal points; in the luminary algorithm, they are luminaries. While a luminary is much more complicated than a terminal point, the luminaries appearing in the algorithm are of special types which form a closed system.

The luminary algorithm takes a full Steiner topology F as input and yields the unique Steiner tree T of $D_S(F)$, if it exists, as output. Since the construction of T for $n = 2$ is trivial and for $n = 3$ is the same as the Melzak algorithm, assume $n \geq 4$ from now on.

The merging stage consists of $n-2$ steps. At each step two adjacent luminaries are merged into a new luminary which consists of all rays consistent (with respect to the angle condition) with a pair of rays from the two merging luminaries (if no new luminary is generated, then T does not exist). Typically two rays meet at a 120° angle generates a third ray rooted at the intersection point and moving away at a 120° angle from each of the two rays (Fig. 2.5a). The intersection point and the three tracks of the three rays of course correspond to a potential Steiner point with its three edges. A ray can also hit a terminal and generate a set of rays spanning a 120° angle all rooted at the terminals and at least 120° apart from the generating ray (see Fig. 2.5b). This type of merging takes care of the nonfull topologies where a terminal is of degree two, and occasionally, of degree three. In the latter case, the three edges of the degree-3 terminal are the generating ray and the two boundary rays of the generated set. There are also other types of merges to take care of degree-3 terminals.

Each luminary is identified with a vertex along with one of its edges in F and two luminaries are adjacent at s if the edges they are identified with are incident

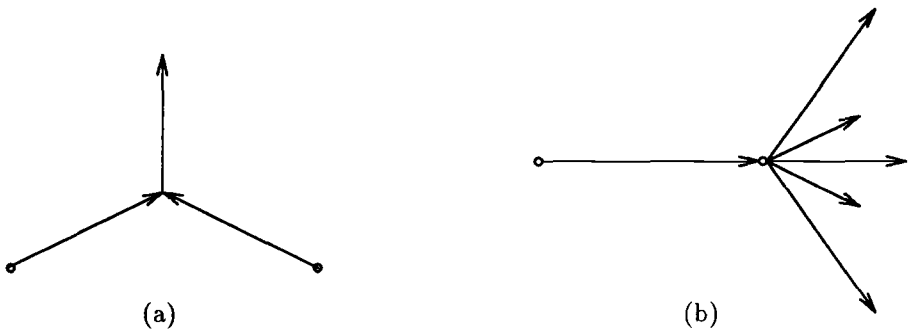


Figure 2.5: Two types of merge

to the same Steiner point. At the beginning, the set of luminaries consists of the terminals (with their edges) which radiate rays in 360° (meaning every direction is a track). Whenever two luminaries are merged, the new luminary generated is identified with the third edge at s. Figure 2.6 shows the merging of two terminals. The new luminary again has rays in 360°, including two 120° sets of rays each rooted at one of the two terminals.

The reconstruction stage starts when only two luminaries L and L' remain. Since there are $2n-1$ edges to start with and each merging step retires two edges, the two remaining luminaries must be identified with the same edge, but from different ends. Therefore T exists if and only if L and L' have an opposing pair of rays $r \in L$ and $r' \in L'$ running into each other from opposite directions. If such a pair exists, the reconstruction proceeds by tracing the rays which generate r and r', then tracing the rays which generate these rays, and doing this recursively until all terminal points are traced. All the ray segments traced during this process, including r and r', constitute the edge-set of T.

It can be shown that merging two luminaries of the special type takes $O(n)$ time. Since there are $n-2$ merging steps, the merging stage takes $O(n^2)$ time. It can also be shown that it takes at most $O(n^2)$ time to find an opposing pair, if it exists, for two given luminaries. Thus the first step of the reconstruction stage takes $O(n^2)$ time. Finally, the remaining reconstruction stage operates exactly in the same way as in the Melzak FST algorithm and takes $O(n)$ time. Thus it is shown:

Theorem 2.7 ([12]) *The luminary algorithm requires $O(n^2)$ time to output the unique ST of $D_S(F)$, if it exists.*

References

[1] M. Ajtai, V. Chvatal, M. M. Newborn, E. Szemeredi, Crossing free subgraphs, *Ann. Discrete Math.* **12** (1982) 9–12.

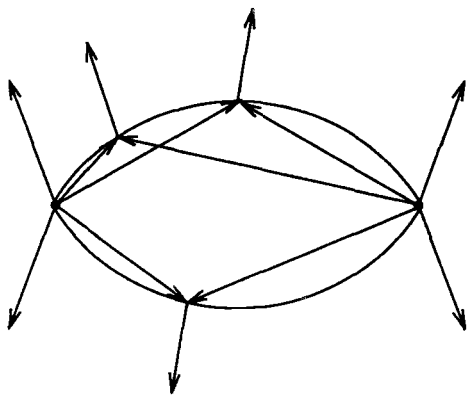


Figure 2.6: Merging of two terminals

[2] W. M. Boyce and J. B. Seery, STEINER 72: An improved version of the minimal network problem, Technical Report, Bell Labs (1975).

[3] R. C. Clark, Communication networks, soap films and vectors, *Phys. Ed.* **16** (1981) 32–37.

[4] E. J. Cockayne, On the Steiner problem, *Canad. Math. Bull.* **10** (1967) 431–450.

[5] E. J. Cockayne and D. E. Hewgill, Exact computation of Steiner minimal trees in the plane, *Inf. Process. Lett.* **22** (1986) 151–156.

[6] E. J. Cockayne and D. E. Hewgill, Leaf paths of full Steiner trees and computation of planar Steiner minimal trees, *Algorithmica* **7** (1992) 219-229.

[7] E. J. Cockayne and D. G. Schiller, Computation of a Steiner minimal tree, in *Combinatorics*, D. J. A. Welsh and D. R. Woodall (eds.), *Inst. Math. Appl.* (1972) 53–71.

[8] E. N. Gilbert and H. O. Pollak, Steiner minimal trees, *SIAM J. Appl. Math.* **16** (1968) 1–29.

[9] R. B. Hayward, A lower bound for the optimal crossing-free Hamiltonian cycle problem, *Discrete Comput. Geom.* **2** (1987) 327–343.

[10] F. K. Hwang, A linear time algorithm for full Steiner trees, *Oper. Res. Lett.* **5** (1986) 235–237.

[11] F. K. Hwang and J. F. Weng, Hexagonal coordinate systems and Steiner minimal trees, *Discrete Math.* **62** (1986) 49–57.

[12] F. K. Hwang and J. F. Weng, The shortest network under a given topology, *J. Algorithms*, to appear.

[13] Z. A. Melzak, On the problem of Steiner, *Canad. Math. Bull.* **4** (1961) 143–148.

[14] W. D. Smith, How to find Steiner minimal trees in Euclidean d-space? *Algorithmica* **7** (1992) 137-177.

[15] D. Trietsch and F. K. Hwang, An improved algorithm for Steiner trees, *SIAM J. Appl. Math.* **50** (1990) 244–263.

[16] J. F. Weng, Steiner problem in hexagonal metrics, unpublished manuscript.

[17] P. Winter, An algorithm for the Steiner problem in the Euclidean plane, *Networks* **15** (1985) 323–345.

This Page Intentionally Left Blank

Chapter 3

The Steiner Ratio

A *spanning tree* for a given set N of n terminals is a tree interconnecting N using only edges $[p_i, p_j]$ where p_i and p_j are in N. A shortest spanning tree is called a *minimum spanning tree* (MST). Figure 3.1 illustrates an MST and an SMT for $N = \{$the three corners of an equilateral triangle$\}$.

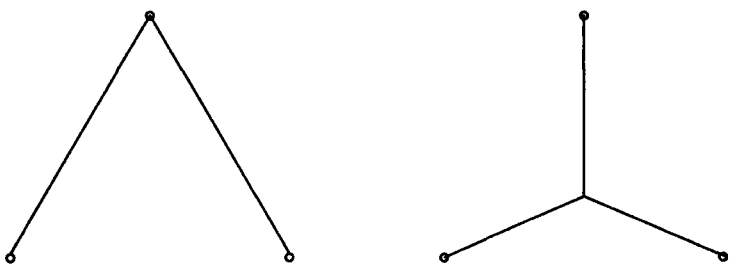


Figure 3.1: An MST and an SMT for three corners of an equilateral triangle

Let $|E|$ denote the total length of a set E of edges. Clearly,

$$|\text{SMT}(N)| \leq |\text{MST}(N)|$$

for all N since the former is chosen from a family of trees which includes the subfamily of spanning trees.

While constructing an SMT(N) for general N is an NP-hard problem as shown in Chapter 1, there exists an $O(n \log n)$ time algorithm [18] to construct MST(N). Therefore, an MST can be used as a heuristic of SMT. The question is how good is this heuristic. If N is the set of three corners in Fig. 3.1, then it is easily verified that

$$\rho(N) = \frac{|\text{SMT}(N)|}{|\text{MST}(N)|} = \frac{\sqrt{3}}{2} \cong .866$$

Is this a typical ratio? In fact Gilbert and Pollak [11] did many simulations and never found a ratio smaller than $\sqrt{3}/2$. Define the *Steiner ratio*

$$\rho = \inf_N \rho(N)$$

They conjectured in 1968 that $\rho = \sqrt{3}/2$.

Since many heuristics of SMT are based on improving an MST, their performances are also closely related to the Steiner ratio. Thus the determination of the Steiner ratio has some practical interests. This problem has also turned out to be a real intellectual challenge with a \$500 cash award placed on it by R. L. Graham. Clever techniques have been developed to prove the conjecture for special classes of N, or to obtain lower bounds for ρ. However, regardless of how clever the techniques are in reducing the computation load, the computation load seems to overwhelm the techniques whenever a larger class of N is attempted. Thus it was a real surprise when a proof for the general conjecture finally emerged which required essentially no computation. Because the proof does not depend on computations specific to the Euclidean plane, it is felt that the proof technique will be useful to other Steiner problems, and possibly to general minimax problems (see Section 3.4 for elaboration).

It is clear that it suffices to prove $\rho \geq \sqrt{3}/2$ as the example in Fig. 3.1 has proved the other direction of the inequality. Furthermore, to prove any lower bound B of ρ, it suffices to prove it for FSTs since if $\rho \geq B$ holds for all full component subtrees, $\rho \geq B$ holds for their union.

3.1 Lower Bounds of ρ

The first lower bound $\rho > 1/2$ was given by Beardwood, Halton and Hammersley [1]. Let S denote an FST and T an MST. The polygon obtained by connecting every pair of consecutive terminals on the circumference of S by a straight line is called the *characteristic polygon* of S. Each terminal appears once on the characteristic polygon. The deletion of any edge yields a spanning tree whose length is strictly less than the perimeter of the polygon, which is at most twice $|S|$. If S is not full, the union of the characteristic polygons on the full subtrees of S is called a *characteristic area*. Deleting one edge from each characteristic polygon yields a spanning tree that is wanted.

One popular approach to prove a lower bound B is to inspect a local structure (a subgraph) of a full SMT, and to show that terminals in this subgraph can be directly connected by edges whose total length does not increase by a factor more than B. The induction hypothesis that B holds for smaller sets of points is applied to the remaining tree, and a proper union of the two trees draws the desired conclusion. Graham and Hwang [12] first applied this line of argument to the following local structure (see Fig. 3.2):

Without loss of generality, assume that $|a,s| \geq |b,s|$ (where we use $|a,b|$ to represent the edge length $|[a,b]|$). Then it is easily verified that $|a,s|/|a,b| \geq 1/\sqrt{3}$.

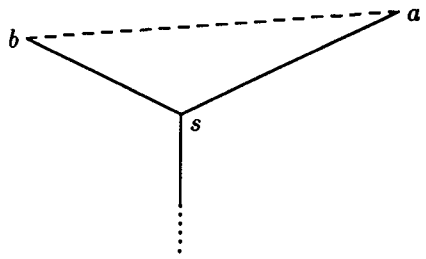


Figure 3.2: A local structure

Let S' be an SMT and T' an MST for the point set $N \setminus \{a\}$. Then

$$\begin{aligned}\frac{|S|}{|T|} &\geq \frac{|a,s| + |S \setminus [a,s]|}{|a,b| + |T'|} \\ &\geq \frac{|a,s| + |S'|}{|a,b| + |T'|} \\ &\geq \min\left\{\frac{|a,s|}{|a,b|}, \frac{|S'|}{|T'|}\right\} = \frac{1}{\sqrt{3}}\end{aligned}$$

by induction on n (the $n = 3$ case follows from the first result reported in Section 3.2).

Chung and Hwang [4] looked into a 5-edge local structure and improved the bound to .743. Du and Hwang [5] did a more careful analysis to obtain a bound of .8. Finally, Chung and Graham [3] did a very detailed analysis and enlisted help from the symbolic computation system MAXIMA to wrestle a bound of .824. Further improvements on the bound are perhaps possible by considering larger local structures and doing more thorough analyses. However, the marginal return is decreasing and it is likely that $\sqrt{3}/2$ target can be approached, but not reached after all.

Although the Steiner ratio conjecture was not proved by this line of attack, it may still be worthwhile to learn since it could be useful for a similar problem in a different metric or a higher dimension.

3.2 The Small n Case

Consider the conjecture for $n \leq k$, where $k \geq 3$ is a given integer. For $k = 3$ Gilbert and Pollak [11] gave the following simple proof:

Let a, b, c be three terminals with an SMT $S = [a,s] \cup [b,s] \cup [c,s]$. Without loss of generality, assume that $[a,s]$ is not longer than the other two edges. If $s = a$, then $S = T$ (an MST) and the conjecture is trivially true. So assume that $s \neq a$. Let b' (c') be a point on $[b,s]$ ($[c,s]$) such that $|b',s| = |a,s|$ ($|c',s| = |a,s|$); see Fig. 3.3.

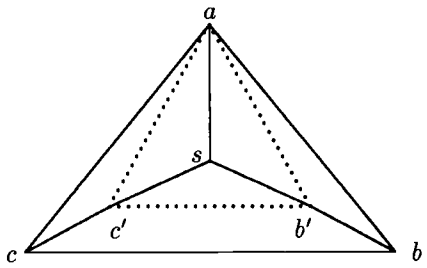


Figure 3.3: A 3-point SMT

By noting that (a, b', c') forms an equilateral triangle, we have

$$\begin{aligned} |S| &= |b,b'| + |c,c'| + |a,s| + |b',s| + |c',s| \\ &= |b,b'| + |c,c'| + \frac{\sqrt{3}}{2}(|a,b'| + |a,c'|) \\ &\geq \frac{\sqrt{3}}{2}(|a,b| + |a,c|) \\ &\geq \frac{\sqrt{3}}{2}|T| \end{aligned}$$

Pollak [13] proved the conjecture for $k = 4$. He considered various patterns of an MST on four terminals and showed that for each pattern there exists an SMT satisfying the $\sqrt{3}/2$ ratio. The difficulty of applying this approach to larger k is that it is difficult to identify an SMT for a given pattern. Therefore, it is often necessary to consider several subcases and prove the $\sqrt{3}/2$ ratio for each. This necessity to branch makes the total number of cases needed to be studied grows very fast.

Du, Yao and Hwang [9] used a different approach which classifies an SMT into patterns according to some angle conditions. For each pattern they showed a spanning tree satisfying the $\sqrt{3}/2$ ratio (thus an MST will also satisfy the ratio). For $k = 4$ there are only four patterns without subcases, the proof was simple and short. They [8] were able to extend this approach to prove the conjecture for $k = 5$. However, there are about 25 patterns to verify and the proof was published with many details left out.

Three other proofs for the case $k = 5$ have subsequently been given. Booth [2] gave a proof similar to the proof in [8] except taking advantage of the length formula of Theorem 1.4. Friedel and Widmayer [10] used an idea similar to the variational approach independently derived by Rubinstein and Thomas (who gave the third proof) which is the theme of the next section.

3.3 The Variational Approach

The material of this section is drawn from Rubinstein and Thomas [14]. Let F be a full Steiner topology for N, so that F has $2n-3$ edges. A tree whose topology is in $D_S(F)$ can be specified by a set $Y = \{y_1, \ldots, y_{2n-3}\}$ where y_i is the length of edge i. Since the variational approach is used to prove results which do not depend on scaling, it is assumed that

$$\sum_{i=1}^{2n-3} y_i = 1$$

The $(2n-4)$-dimensional simplex

$$\Delta(F) = \{Y = \{y_1, \ldots, y_{2n-3}\} : \sum_{i=1}^{2n-3} y_i = 1 \ , \ y_i \geq 0 \ \text{ for all } \ i\}$$

is called a *configuration space.* An element $Y \in \Delta(F)$ is called a *configuration.*

Consider a configuration Y in $\Delta(F)$. Let V be a vector at Y. Define T to be an MST for Y such that $T(h)$, which has the same topology as T, is also an MST for $Y + hV$ for h sufficiently small. Let S be an SMT for Y with topology in $D_S(F)$. Then $\rho(V) = L_S(V)/L_T(V)$ is continuous and differentiable in any direction of V at Y, where L_S and L_T are the lengths of the respective trees. For a function f let f' denote the first derivative of f.

$$\begin{aligned} \rho'(V) &= \frac{L_T(V)L_S'(V) - L_S(V)L_T'(V)}{L_T^2(V)} \\ &= \frac{L_T'(V)}{L_T(V)} \left(\frac{L_S'(V)}{L_T'(V)} - \rho \right) \end{aligned}$$

It follows that

Theorem 3.1 *Suppose that* $L_T'(V) < 0$. *Then*

$$\rho'(V) \left\{ \begin{array}{c} > \\ = \\ < \end{array} \right\} 0 \Longleftrightarrow \frac{L_S'(V)}{L_T'(V)} \left\{ \begin{array}{c} < \\ = \\ > \end{array} \right\} \rho$$

Suppose that $L_T'(V) > 0$. *Then*

$$\rho'(V) \left\{ \begin{array}{c} > \\ = \\ < \end{array} \right\} 0 \Longleftrightarrow \frac{L_S'(V)}{L_T'(V)} \left\{ \begin{array}{c} > \\ = \\ < \end{array} \right\} \rho$$

Theorem 3.1 can be used to attack the Steiner ratio conjecture in the following way. Suppose that a configuration Y achieves the minimum ρ at a value less than $\sqrt{3}/2$. To establish a contradiction, it suffices to seek a vector V at Y such that $L_T'(V) < 0$ but $L_S'(V)/L_T'(V) \geq \sqrt{3}/2 > \rho$, since from Theorem 3.1, $\rho'(V) < 0$ and ρ cannot be a minimum.

Therefore the crux of a proof is to find for every topology a vector which shrinks S at a rate at least $\sqrt{3}/2$ times the shrinking rate of T. Note that whether the shrunk S is an SMT is not of concern, since if it is not, then ρ would be even smaller (strengthening the contradiction). Also, note that if two topologies differ only in the labeling of terminals, then they can be represented by configurations in the same $\Delta(F)$. Thus there is only one F to be concerned for $n = 4$ or 5, and three for $n = 6$. Of course, the choice of V may depend on the geometry of the terminals and subcases have to be treated. Rubinstein and Thomas demonstrated the effectiveness of their approach by giving a fairly simple proof of the Steiner ratio conjecture for $k = 5$ [14], and by then doing $k = 6$, which had not been previously solved [15]. They [16] also proved that the Steiner ratio conjecture holds for cocircular points. Later, they [17] sharpened the technique by considering the second derivative of ρ, and used that to prove a long-standing conjecture of Graham (see Section 5.1).

3.4 The Steiner Ratio Conjecture as a Maximin Problem

The results in this and the next two sections were obtained by Du and Hwang [6,7].

Let $Y \in \Delta(F)$ be defined as in the last section and let $P(F, Y)$ denote the set of terminals corresponding to F and Y. Let $G(F, Y)$ denote a spanning tree on $P(F, Y)$ with topology G. Define

$$L(F, Y) = \min_G |G(F, Y)|$$

Since $|G(F, Y)|$ is continuous in Y, so is $L(F, Y)$. Define

$$L(F) = \max_{Y \in \Delta(F)} L(F, Y)$$

Since there are only finitely many F, there exists a F^* such that

$$L(F^*) = \max_F L(F) = \max_{P(F,Y)} \min_G |G(F, Y)|$$

A point set $P(F, Y)$ achieving the maximum will be called a *maximum point.* When F is understood, then Y will be referred to as a maximum point. By a previous comment, it suffices to prove the Steiner ratio conjecture for full SMTs. A maximum point is interior if all its edge lengths are positive. Since the length of an SMT is unity by definition, it follows:

Lemma 3.1 *Suppose that a full Steiner topology F^* exists for $P(F^*, Y)$. Then proving the Steiner ratio conjecture is equivalent to proving*

$$\max_{P(F,Y)} \min_G |G(F, Y)| \leq 2/\sqrt{3}$$

An important property of the above maximin problem is that

Lemma 3.2 *$|G(F,Y)|$ is convex in Y.*

Proof: Suppose that two terminals a and b are connected in F through a path p. Let I denote the index set of edges in p. Then

$$|a,b| = \left|\sum_{i\in I} e_i y_i\right|$$

where e_i is the unit vector in the direction of the edge i. Hence the length of an edge is a convex function of y_i, and $|G(F,Y)|$, a sum of convex functions, is also convex. □

Let $M(F,Y)$ denote the set of spanning tree topologies G such that $G(F,Y)$ is an MST for $P(F,Y)$. Using the finiteness of the number of G, a continuity argument shows

Lemma 3.3 *For every Y there is a neighborhood of Y such that $M(F,X) \subseteq M(F,Y)$ for every X in that neighborhood.*

Lemmas 3.2 and 3.3 lead to the following crucial result.

Lemma 3.4 *Suppose that $P(F^*,Y)$ is an interior maximum point and that X is a point in $\Delta(F^*)$ satisfying $M(F^*,Y) \subseteq M(F^*,X)$. Then $P(F^*,X)$ is also a maximum point.*

Proof: For any G in $M(F^*,Y)$, define $A(G) = \{Z \in \Delta(F^*) : |G(F^*,Z)| \leq L(F^*,Y)\}$. By Lemma 3.2, $A(G)$ is a convex region. Note that the union of all $A(G)$ for $G \in M(F^*,Y)$ covers a neighborhood of Y; for otherwise, configuration Z can be found in every neighborhood of Y such that for every $G \in M(F^*,Y)$,

$$|G(F^*,Z)| > L(F^*,Y)$$

In particular,

$$L(F^*,Z) = \min\{|G(F^*,Z)| : G \in M(F^*,Z)\} > L(F^*)$$

since by Lemma 3.3,

$$G \in M(F^*,Z) \Rightarrow G \in M(F^*,Y)$$

for Z sufficiently close to Y. Thus there exists a Z in $\Delta(F^*)$ such that $L(F^*,Z) > L(F^*,Y)$, contradicting the assumption that $P(F^*,Y)$ is a maximum point.

Suppose to the contrary that $P(F^*,X)$ is not a maximum point, i.e., $L(F^*,X) < L(F^*)$. Since $M(F^*,Y) \subseteq M(F^*,X)$,

$$|G(F^*,X)| = L(F^*,X) < L(F^*,Y)$$

or

$$X \in A(G)$$

for every $G \in M(F^*, Y)$. Suppose that the configuration $Z_c \equiv Y + c(Y - X)$ for some positive c is in $A(G)$. Then $Y = \lambda X + (1-\lambda)Z_c$, where $0 < \lambda = c/(1+c) < 1$. Hence Y is an interior point of the segment $[X, Z_c]$. By Lemma 3.2,

$$|G(F^*, Y)| \leq \lambda |G(F^*, X)| + (1-\lambda)|G(F^*, Z_c)| < L(F^*, Y)$$

contradicting the definition of $L(F^*, Y)$. Therefore for all $c > 0$, Z_c is not in $A(G)$ for every $G \in M(F^*, Y)$, contradicting what was shown earlier, that the union of these $A(G)$ covers a neighborhood of Y. □

3.5 Critical Structures

In Section 3.4 the Steiner ratio conjecture is transformed into a maximin problem and some important properties of the maximum point are derived. In this section these properties are translated back to the original geometric problem, and effectively used to restrict the geometry of the set of terminals.

Let $\Gamma(F, Y)$ denote the union of all MSTs for $P(F, Y)$. Lemma 3.5 can be found in [10,14], and Lemma 3.6 in [14].

Lemma 3.5 *$\Gamma(F, Y)$ has no crossing edges.*

Let $C(F, Y)$ denote the convex hull of $P(F, Y)$. From Lemma 3.5 $\Gamma(F, Y)$ divides $C(F, Y)$ into disjoint areas each of which is bounded by a polygon whose vertices are terminals. Such a polygon is called *a polygon of* $\Gamma(F, Y)$, if it is a subgraph of $\Gamma(F, Y)$.

Lemma 3.6 *Every polygon of $\Gamma(F, Y)$ has at least two equal longest edges.*

$\Gamma(F, Y)$ is said to be *critical* if $M(F, Y)$ is maximal, i.e., there does not exist $X \in \Delta(F)$ such that $M(F, Y) \subset M(F, X)$. Since the number of MST topologies is finite, the set of critical Γ cannot be empty for any F. Furthermore, if all maximum points are interior, then by Lemma 3.4 there must exist a maximum point which has a critical Γ. Therefore it suffices to prove the conjecture for $P(F, Y)$ such that $\Gamma(F, Y)$ is critical. Call an edge length *independent* if its only dependence on other edge lengths are through the triangle inequalities.

Lemma 3.7 *Suppose that all edges in any triangulation of $C(F^*, Y)$ are independent and all maximum points are interior. Then $\Gamma(F^*, Y)$ is critical only if it partitions $C(F^*, Y)$ into exactly $n - 2$ equilateral triangles.*

Proof: It suffices to prove that $\Gamma(F^*, Y)$ cannot be critical if one of the following occurs:

(i) $\Gamma(F^*, Y)$ has a *free edge*, an edge not on any polygon of Γ.

(ii) $\Gamma(F^*, Y)$ has a polygon of more than three edges.

(iii) $\Gamma(F^*, Y)$ has a nonequilateral triangle.

First, assume that $\Gamma(F^*, Y)$ has a free edge e. Embedding $\Gamma(F^*, Y)$ into a triangulation R of $C(F^*, Y)$ and let e be a side of a triangle t. Consider an MST T containing e. Since e is free, at least one of the two other sides of t is not in R. Let e' be such a side so that replacing e by e' in T results in another spanning tree. Clearly, $|e| < |e'|$. By the assumption of independent edges, it is possible to decrease $|e'|$ while holding all other edge lengths constant in R. Let l denote the length of the shrinking e' from $l(e')$ to $l(e)$. Let $P(l)$ denote the set of terminals corresponding to l. Then $P(l(e')) = P(F^*, Y)$. Consider the set L of all $l \in [l(e), l(e')]$ satisfying the condition that $P(l) = P(F^*, X)$ for a maximum point X. Since the set of maximum points is a closed set and contains the point Y, the set L is nonempty and closed. Therefore, there exists a minimal element l^* in L. Let $P(l^*) \equiv P(F^*, X)$. Since both X and Y are maximum points, the length of an MST for $P(F^*, X)$ equals that for $P(F^*, Y)$. Therefore any MST at $P(F^*, Y)$ is an MST at $P(F^*, X)$. It follows $M(F^*, Y) \subseteq M(F^*, X)$. If the equality holds, then $l^* > l(e)$ since when $l^* = l(e)$, dropping e and adding e' would yield one more MST. By the assumption that all maximum points are internal, there exists a neighborhood of X such that if Z is a configuration in that neighborhood, F^* still exists for $P(F^*, Z)$. Z can always be chosen close enough to X such that Lemma 3.3 applies and $M(F^*, Z) \subseteq M(F^*, X)$. Note that for $G, G' \in M(F^*, X)$, $|G(F^*, Z)| = |G'(F^*, Z)|$ since all edge lengths in R except $|e'|$ remain unchanged. Therefore if any $G \in M(F^*, X)$ is in $M(F^*, Z)$, then every $G \in M(F^*, X)$ is in $M(F^*, Z)$. Since $M(F^*, Z)$ is nonempty, there exists a G belonging to both $M(F^*, Z)$ and $M(F^*, X)$. Hence $M(F^*, X) = M(F^*, Z)$. Furthermore, $M(F^*, Z) = M(F^*, X) = M(F^*, Y)$ implies that e' is not in any MST in $M(F^*, Z)$. Hence $L(F^*, Z) = L(F^*, X) = L(F^*, Y)$. Note that although F^* exists on Z, $P(F^*, Z)$ is not necessarily in $\Delta(F^*)$ as $\sum_{i=1}^{2n-3} y_i$ does not have to be one. But there always exists a positive number h such that $hZ \in \Delta(F^*)$. Since $M(F^*, hZ) = M(F^*, Z) = M(F^*, X)$, hZ is a maximum point by Lemma 3.4. Hence $L(F^*, X) = L(F^*, hZ) = hL(F^*, Z) = hL(F^*, X)$. It follows that $h = 1$ and $P(F^*, Z) \in \Delta(F^*)$, i.e., Z is also a maximum point, contradicting the assumption that X is the minimal element in L.

The two other cases are proved by analogous arguments. For case (ii), the operation is to decrease the length of an edge not in $\Gamma(F^*, Y)$. For case (iii), excluding cases (i) and (ii), $\Gamma(F^*, Y)$ is a triangulation consisting of isosceles triangles. The operation is to decrease the length of all shortest edges in $\Gamma(F^*, Y)$ uniformly. □

3.6 A Proof of the Steiner Ratio Conjecture

Lemma 3.7 says that the validity of the Steiner ratio conjecture depends only on whether it holds for n points consisting of $n - 2$ equilateral triangles. This is a tremendous reduction of the original problem and indeed, an elegant solution exists for the reduced problem. However, this simplification is possible only if the assumptions of independent edges and interior maximal points are valid. Fig. 3.4

gives an example that edges in a triangulation are not independent (the lengths of any five edges determine the sixth length).

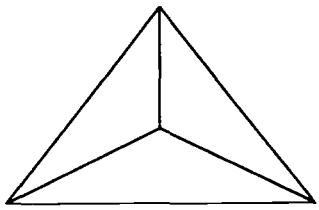


Figure 3.4: Dependent edges

Note that if all terminals lie on the boundary of a triangulation, then the edges are independent. To see this, order the triangles into a sequence and construct the triangles in order of that sequence. Since every triangle to be added to the sequence in construction has only one of its side fixed, the other two sides can take any length as long as the triangle inequality is satisfied. This shows that any set of independent edges can be realized.

The concept of "inner spanning tree" is to be introduced to assure that any triangulation has all terminals on its boundary. Let S denote an FST. A characteristic polygon is viewed as drawn on a spiral surface so it is always simple with no crossing lines (see Fig. 3.5). Note that all terminals lie on the boundary of a characteristic polygon. A spanning tree is called *inner* if it lies inside of the characteristic polygon of S.

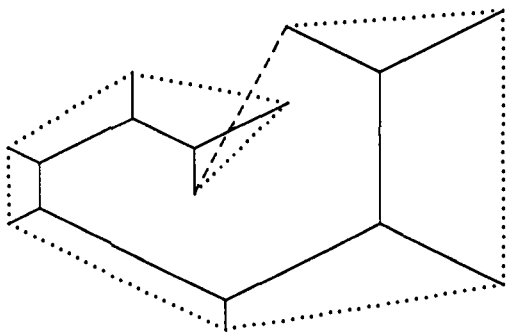


Figure 3.5: A spiral characteristic polygon

The case that the ST on $P(F, Y)$ itself has crossing edges is not considered here. That case was dealt with in [6] by deleting these points corresponding to crossing STs from $\Delta(F)$. A possible alternative treatment is to define the characteristic polygon on a more sophisticate surface than the spiral one.

Lemmas 3.2–3.6 remain true if only inner spanning trees are considered, except proofs are messier in general (see [6] for details). Also note that for the inner spanning case, Lemma 3.3 implies that there is a neighborhood of X such that for Y in the neighborhood,

$$L(F,Y) = \min_{G \in M(F,X)} |G(F,Y)|$$

Therefore $L(F,X)$ is continuous at X. We now prove the Steiner ratio conjecture with MST replaced by minimum inner spanning tree. Note that this is a stronger result than the original conjecture.

Theorem 3.2 *The ratio of the lengths of an SMT and a minimum inner spanning tree is at least* $\sqrt{3}/2$.

Proof: Suppose that Theorem 3.2 is not true. Then there exists a smallest n (≥ 4) such that Theorem 3.2 is not true for a set of n terminals. First, it will be shown that all maximum points are interior.

For an arbitrary point set $P(F,X)$, let $S(F,X)$ denote the ST, if it exists, for $P(F,X)$. Suppose to the contrary there exists a maximum point X^* on the boundary of $\Delta(F)$, that is $S(F,X^*)$ have some shrunk edges. Then $L(X^*) = L(F,X^*) > 2/\sqrt{3}$. If there exists a shrunk edge incident to a terminal, then $S(F,X^*)$ can be decomposed into smaller STs each having fewer than n points. By the inductive assumption Theorem 3.2 holds for these STs. Since the union of the inner spanning trees for these smaller STs is an inner spanning tree for $P(F,X^*)$, $L(F,X^*) \leq 2/\sqrt{3}$, a contradiction to the assumption that X^* is a maximum point.

Therefore every shrunk edge is between two Steiner points. In this case it is easy to find a full topology F', called a *companion* of F, which satisfies the following two conditions (Fig. 3.6):

(i) F and F' have the same circumferential order.

(ii) There is a tree T interconnecting the n terminals in $P(F,X^*)$ with topology F' and $|T| < |S(F,X^*)|$.

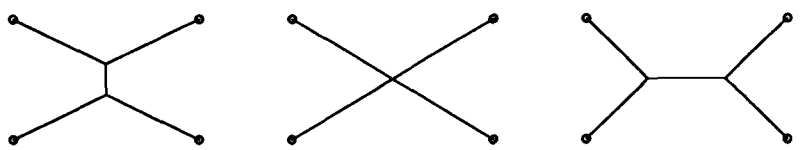


Figure 3.6: A companion topology

Suppose that the ST of topology F' for $P(F,X^*)$ exists. Then there exists a point Y in the $2n-4$ dimensional simplex such that $P(F',Y) = P(F,X)$. However, $Y \notin \Delta(F')$ since

$$|S(F',Y)| \leq |T| < |S(F,X^*)|$$

Define

$$h = \frac{1}{|S(F', Y)|}$$

and note $h > 1$. Since $S(F', hY)$ is similar to $S(F', Y)$, $L(F', hY) = hL(F', Y)$. Furthermore, since F and F' have the same circumferential order, they have the same characteristic area and the same set of inner spanning trees. Thus

$$L(F', hY) = hL(F', Y) = hL(F, X^*) > L(F, X^*)$$

contradicting the assumption that X^* is a maximum point as $hY \in \Delta(F')$.

Next consider the case that the ST of topology F' for $P(F, X^*)$ does not exist. Then $S(F', Y)$ is not defined and the above argument needs modification. Expand the definition of $\Delta(F)$ to $\bar{\Delta}(F)$ such that a point in $\bar{\Delta}(F)$ not only specifies the edge lengths, but also the angles at the Steiner points (thus a point in $\bar{\Delta}(F)$ does not necessarily correspond to an ST). The L functions will be replaced by the $\bar{L}$ functions whenever $\bar{\Delta}$ is the underlying field. Then by an argument similar to the L case, it can be shown that $\bar{L}(F') > \bar{L}(F)$.

Let $T(F', Y)$ denote a tree with length $\bar{L}(F')$. Then $T(F', Y)$ is not an FST. Thus either Y has a shrunk edge incident to a terminal, or there exists an angle A less than 120°. In the former case, a smaller counterexample can be obtained by decomposing $T(F', Y)$ to smaller FSTs, contradicting the definition of n. In the latter case, at least one side of A must shrink, or $T(F', Y)$ can be shorter without changing the topology. Since one may assume that the former case does not occur, the shrunk side must be an edge between two Steiner points. Thus there exists a companion F'' of F' such that $\bar{L}(F'') > \bar{L}(F)$.

Repeating this argument, one obtains infinitely many companions of F, contradicting the fact that the number should be finite. Therefore, all maximum points are interior.

Finally, consider a set N of n terminals forming $n - 2$ equilateral triangles. We quote the following lemma from [6] which is of some interest itself. Recall the definition of hexagonal tree from Section 2.3.

Lemma 3.8 *Let P be a set of lattice points on a triangular lattice (with equilateral triangles) which defines an system of axes. There exists a minimum hexagonal tree whose junctions are all lattice points.*

Clearly, the length of a minimum inner spanning tree for P equals to $L_H(P)$, the length of a minimum hexagonal tree. By Theorem 2.3, $L_S(P) \geq (\sqrt{3}/2)L_H(P)$. Theorem 3.2 is proved. □

Corollary 3.1 $\rho = \sqrt{3}/2$.

References

[1] J. Beardwood, J. H. Halton and J. M. Hammersley, The shortest path through many points, *Proc. Cambridge Phil. Soc.* **55** (1959) 299–327.

[2] R. S. Booth, The Steiner ratio for five points, *Ann. Oper. Res.* **33** (1991) 419–436.

[3] F. R. K. Chung and R. L. Graham, A new bound for the Steiner minimal trees, *Ann. N.Y. Acad. Sci.* **440** (1985) 325-346.

[4] F. R. K. Chung and F. K. Hwang, A lower bound for the Steiner tree problem, *SIAM J. Appl. Math.* **34** (1978) 27-36.

[5] D. Z. Du and F. K. Hwang, A new bound for the Steiner ratio, *Trans. Am. Math. Soc.* **278** (1983) 137-148.

[6] D. Z. Du and F. K. Hwang, A proof of Gilbert and Pollak's conjecture on the Steiner ratio, *Algorithmica* **7** (1992) 121-135.

[7] D. Z. Du and F. K. Hwang, The Steiner ratio conjecture of Gilbert and Pollak is true, *Proc. Natl. Acad. Sci.* **8** (1990) 9464-9466.

[8] D. Z. Du, F. K. Hwang and E. N. Yao, The Steiner ratio conjecture is true for five points, *J. Comb. Theory, Ser. A* **38** (1985) 230-240.

[9] D. Z. Du, E. N. Yao and F. K. Hwang, A short proof of a result of Pollak on Steiner minimal trees, *J. Comb. Theory, Ser. A* **32** (1982) 356-400.

[10] J. Friedel and P. Widmayer, A simple proof of the Steiner ratio conjecture for five points, *SIAM J. Appl. Math.* **49** (1989) 960-967.

[11] E. N. Gilbert and H. O. Pollak, Steiner minimal trees, *SIAM J. Appl. Math.* **16** (1968) 323-345.

[12] R. L. Graham and F. K. Hwang, Remarks on Steiner minimal trees I, *Bull. Inst. Math. Acad. Sinica* **4** (1976) 177-182.

[13] H. O. Pollak, Some remarks on the Steiner problem, *J. Comb. Theory, Ser. A* **24** (1978) 278-295.

[14] J. H. Rubinstein and D. A. Thomas, The calculus of variations and the Steiner problem, *Ann. Oper. Res.* **33** (1991) 481-499.

[15] J. H. Rubinstein and D. A. Thomas, The Steiner ratio conjecture for six points, *J. Comb. Theory, Ser. A* **58** (1991) 54–77.

[16] J. H. Rubinstein and D. A. Thomas, The Steiner ratio conjecture for cocircular points, *Discrete Comput. Geom.* **7** (1992) 77-86.

[17] J. H. Rubinstein and D. A. Thomas, Graham's conjecture on shortest networks for points on a circle, *Algorithmica* **7** (1992) 193-218.

[18] M. I. Shamos and D. Hoey, Closest-point problems, *Proc. of the 16-th Ann. Symp. on Foundations of Computer Science* (1975) 151-162.

This Page Intentionally Left Blank

Chapter 4

Heuristics

Since the ESP is known to be NP-hard, it is important to derive effective heuristics to solve for general cases. By an effective heuristic we mean an algorithm which can run in time proportional to a low-degree polynomial of n, and whose output tree length does not exceed that of an SMT by a great deal.

Such a heuristic tree is readily available in the form of an MST, where an $O(n \log n)$ algorithm exists and whose length does not exceed that of an SMT by at most a factor of $2/\sqrt{3} - 1 \cong 15.5\%$ (the average excessive length is, of course, smaller). Therefore, the MST naturally becomes the standard, against which other heuristics are compared.

In this chapter we will report several heuristics proposed in the literature. In the first three sections the heuristics either improve on a given MST, or emulate a given MST algorithm. A simulated annealing algorithm is given in Section 4.4 and a probabilistic algorithm is given in the following section. Section 4.6 gives a simple but easily analyzed heuristic. An effective reduction to the Steiner problem in networks, discussed in Part II, is found in Section 4.7. Section 4.8 briefly introduces an important heuristic that is treated in Part IV.

For a heuristic H define the performance ratio

$$\rho(H) = \inf_N \frac{|\mathrm{SMT}(N)|}{|H(N)|}$$

Although most of the proposed heuristics have better average performance than the MST, no rigorous proof has existed to show any of them has a guaranteed performance ratio better than the Steiner ratio until very recently, a heuristic by Du, Zhang and Feng [6] broke the barrier. A task that needs to be the subject of future research is a thorough comparison of the performance of these heuristics, if not analytically, then at least experimentally.

4.1 Minimal Spanning Trees

Graham and Hell [9] gave a complete accounting of the history of the MST problem on which we base our discussion. The MST was first proposed by Borůvka [3] for the layout of a power-line network. His algorithm can be described as follows: Consider a graph with n terminals and e weighted edges. Connect each terminal with its nearest neighbor. Suppose that c connected components are formed through such connections. Define the distance between two components as the length of the shortest edge between two terminals, one in each component. If $c > 1$ treat each component as a terminal and repeat the procedure. Kruskal [15] gave the following variation: Sequentially choose the shortest edge which does not form a loop with edges already chosen until $n - 1$ edges are chosen. A third variation was first proposed by Jarník [11], but typically attributed to Prim [17]: Grow a component, which initially consists of a single terminal, by sequentially adding the shortest edge connecting a terminal not in the component with a terminal in the component. It has been shown (see [9]) that Kruskal's and Prim's algorithms can be implemented in $O(e \log n)$ time, while Borůvka's algorithm requires $O(e \log \log n)$ time. The weights of edges are assumed to be different in the last algorithm to avoid cycles.

For a complete graph, it seems that one cannot avoid to inspect the $e = O(n^2)$ edges to find an MST. However, when the terminals are embedded in the Euclidean plane and the weight of an edge is the Euclidean distance, one can use the geometric properties to devise an $O(n \log n)$ algorithm. These geometric properties are characterized by the concept of a Voronoi diagram. A *Voronoi diagram* for a given set $N = \{p_1, \ldots, p_n\}$ of terminals is a partition of the plane into closed or open areas $V_1, \ldots, V_n$ such that $p_i \in V_i$ for each p_i, and any point $p \in V_i$ is closer to p_i than to any other p_j (points on the boundary of V_i and V_j are equidistant to p_i and p_j). The graph on N with an edge (p_i, p_j) if V_i and V_j share a common side is called a *Delaunay triangulation.* Shamos and Hoey [19] gave an $O(n \log n)$ algorithm for constructing a Voronoi diagram and also showed that an MST is a subgraph of the Delaunay triangulation. Since the Delaunay triangulation is planar, it has at most $O(n)$ edges. Thus an MST can be constructed from the Voronoi diagram in $O(n)$ time by the Cheriton and Tarjan algorithm [5].

Theorem 4.1 *There exists an $O(n \log n)$ algorithm to construct an MST.*

Gilbert [8] studied the average performance of an MST. He proved

Theorem 4.2 *For n terminals randomly scattered in a region of area A, the expected length of an MST is less than $\sqrt{2An}$.*

From experimental results, Gilbert observed that the expected length is more likely to be $.68\sqrt{An}$.

4.2 Improving the MST

Chang [4] and Thompson [23] independently first came up with the idea of adding Steiner points to an MST to obtain a better heuristic SMT. Thompson suggested

a simple modification of an MST by inserting sequentially a Steiner point between any two edges meeting at an angle less than 120°. (A similar suggestion was also made by Korhonen [14] later.)

Chang suggested the following more elaborate scheme: Let T be the tree under construction. At the beginning T is the MST and at the end it is the output heuristic tree. A *full component* of T is a subtree with a full topology. At each step an ST T' on three vertices $\{x, y, z\}$ is constructed and the longest edge of T in any cycle of $T \cup T'$ is deleted to form a new tree T^*. Such a replacement is called a *Steinerization*, and takes place only when it reduces the length of T. The triple vertices $\{x, y, z\}$ is chosen from the following set P to maximize $|T| - |T^*|$, where P includes triples satisfying one of the following conditions:

(i) x, y, z are all terminals and belong to distinct full components.

(ii) Terminals x and y belong to the same full component and z is a terminal in a different full component.

(iii) x and y are Steiner points in different full components and z is a terminal adjacent to both x and y.

Fig. 4.1 illustrates the three types of triples where a replaced edge is marked by two short bars.

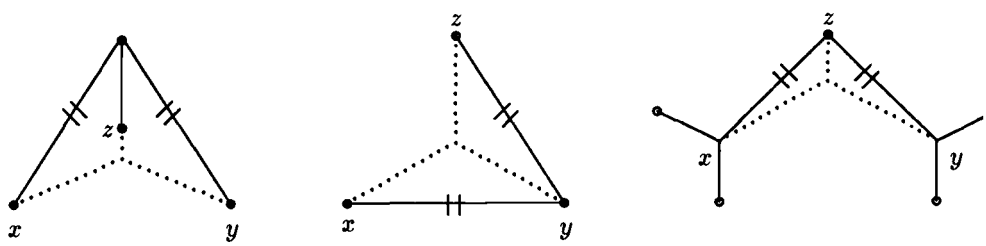


Figure 4.1: Three types of Steinerization

These recursive steps end when no choice of $\{x, y, z\}$ can lead to a positive $|T| - |T^*|$. Note that after at most $n - 2$ recursive steps all vertices are in one component and no $\{x, y, z\} \in P$ remains. The output tree of the recursive steps yields a topology on which we can then use the Melzak FST algorithm to construct an ST.

At each recursive step three vertices are chosen from $O(n)$ vertices. Hence each recursive step requires $O(n^3)$ time and there are $n - 2$ recursive steps, Chang's heuristic requires $O(n^4)$ time. Chang also gave experimental results which showed a typical saving of 3% over the MSTs.

Smith, Lee and Liebman [20] gave a much faster heuristic by constructing 3-point and 4-point STs. The 3-point and 4-point subsets are chosen from the vertices of triangles or two adjacent triangles of the Delaunay triangulation using the following rules:

1. Construct the Delaunay triangulation and an MST on it. Call two triangles adjacent if they share an edge.
2. Place all triangles with two edges in the MST in a priority queue Q.

3. Define the distance between two adjacent triangles as the distance between the two circumcenters (the line segment connecting the two circumcenters is a Voronoi edge). For each $t \in Q$, check its adjacent triangle t' to see whether t' has an MST edge and $t \cup t'$ is a convex quadrilateral. If there are more than one such t', take the closest one. Construct a 4-point SMT on the vertices of $t \cup t'$.
4. If no t' formed a convex quadrilateral with t in the above step then construct a 3-point SMT on the vertices of t.
5. After Q is exhausted, concatenate the 3-point and 4-point SMTs into a heuristic SMT for N by a disjoint-set-union procedure (to avoid cycles). The concatenation procedure considers those 3-point and 4-point SMTs with small Steiner ratios first (to achieve a large reduction of length).

Smith, Lee and Liebman showed that their heuristic can be implemented in $O(n \log n)$ time. The experimental results showed a slight drop in saving as compared to Chang's heuristic. Beasley [2] proposed a heuristic which iteratively selects four connected vertices from the current MST which can be replaced by an ST with the largest improvement. The newly created Steiner points then join the vertex-set V and a new MST is constructed. The MST is Steinerized and a new iteration starts. Although convergence of the procedure was not proved, experimental results showed a reasonable convergence time. Beasley also gave the following comparisons on the improvement of various heuristics over the MST (# denotes the number of test problems).

n	Beasley	Chang	Korh.	Sm-Lee-Lieb.
10	3.138	2.200	1.50	3.173
20	3.015	3.012	2.25	2.333
30	2.868	3.087	2.51	2.769
40	3.031		2.54	2.663
50	2.845		2.39	2.568
#	15	20	5	15

Table 4.1: Improvements over the MST

4.3 Greedy Trees

Smith and Shor [21] recently suggested a variation of MST which they called *greedy trees*. A greedy tree can be constructed by regarding the n terminals as a forest of n 1-vertex trees and sequentially add the shortest edge to merge two trees until the forest consists of a single tree (an MST would require each added edge to connect two terminals in the two merged trees). Fig. 4.2 illustrates a greedy tree with three terminals.

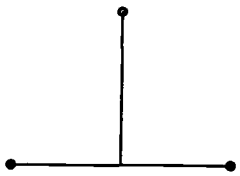


Figure 4.2: A 3-terminal greedy tree

A Kruskal-type algorithm can construct such a tree in $O(n^2)$ time. Smith and Shor showed that greedy trees have the following properties:

(i) A greedy tree is an MST of its vertices.

(ii) No angle in a greedy tree is less than 90°.

(iii) An edge connecting two terminals in a greedy tree is also an edge of an MST.

(iv) For given N a greedy tree is not longer than an MST.

While proofs of (i), (ii) and (iii) are straightforward, (iv) follows from the following lemma.

Lemma 4.1 *Given a forest of $k-1$ edges connecting terminals, the shortest segment which could be added without forming a cycle cannot be longer than the k^{th}-shortest edge of an MST.*

Smith and Shor also observed that another kind of greedy tree could be obtained by generalizing Prim's algorithm instead of Kruskal's algorithm: Grow a tree, which initially consists of the shortest edge connecting two terminals, by adding the shortest segment connecting the tree with an external terminal until all terminals are internal. Properties (i) – (iv) also hold for Prim-type greedy trees, although (iv) requires an alternate proof.

Let $GT(N)$ denote a greedy tree for N. Smith and Shor conjectured:

$$\inf_N \frac{|SMT(N)|}{|GT(N)|} = \frac{2\sqrt{3}}{2+\sqrt{3}} \cong .9282$$

where the infimum is achieved again at the three corners of an equilateral triangle.

4.4 An Annealing Algorithm

An annealing algorithm consists of an initial solution x_0, a perturbation scheme to produce a neighbor solution x_s, and a sequence of values for a control parameter which, starting at some initially high value c_0, decrease at each step until they reach a final value c_f.

Lundy [16] proposed an annealing algorithm as an SMT heuristic. In Lundy's algorithm a solution consists of a full topology and a "repositioning" scheme to reorient the Steiner points for an ST. Lundy considered five different repositioning schemes and determined by experimentation that the best one is to reorient the $n-2$ Steiner points sequentially. Since Hwang and Weng [10] have now given an $O(n^2)$ algorithm to find the ST of $D(G)$ for a given full topology G (see Section 2.6), one need not use the repositioning schemes to find the relatively minimum tree when it exists.

A perturbation scheme in Lundy's algorithm is a scheme to switch full topologies. Let F denote the current full topology. Select randomly a terminal a. Suppose that a is adjacent to a Steiner point s which is also adjacent to two other vertices b and c. Select randomly an edge $[d, e]$ which is not incident to s. Then a new full topology G is obtained from F by substituting edges $[b, c]$, $[s, d]$ and $[s, e]$ for edges $[d, e]$, $[s, b]$ and $[s, c]$ (see Fig. 4.3).

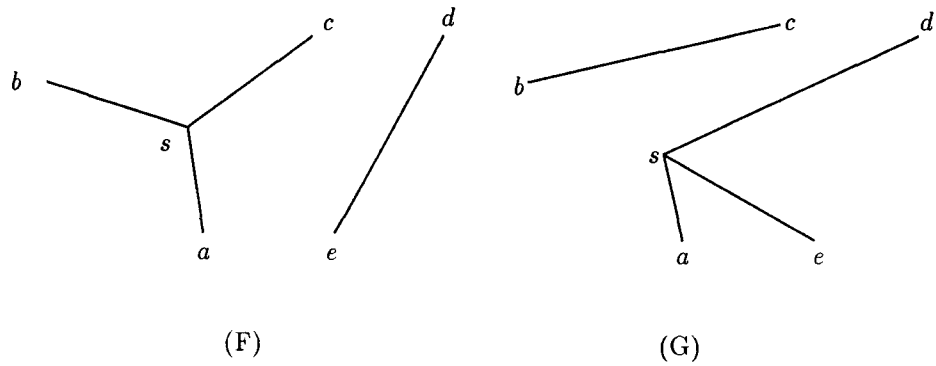


Figure 4.3: A switch of full topology

Since d and e must be connected to either b or c with a path not including $[s, b]$ and $[s, c]$ in F, d and c remain connected to b and c in G and G is a full topology.

After the perturbation, reposition the new tree. If the new tree is not longer than the old tree, accept it as the new solution. Otherwise, accept the new tree with probability e^{-l/c_i}, where l is the increase in length and c_i is the control parameter at step i (each topology switch constitutes a step). Lundy set $c_0 = 10\sqrt{n-1}$ and $c_1 = c_2 = \cdots = c_f = 1/(n \log n)$. He explained the choices by pointing out that $\sqrt{n-1}$ is proportional to the expected length of an edge for points randomly distributed in a unit square, and that the number of full topologies is roughly $c^{n \log n}$.

Lundy tested his algorithm with experimental sets of 20 and 50 points and found that it compared favorably with the Edward algorithm (see Section 6.1), a standard method used in finding evolution trees.

4.5 A Partitioning Algorithm

Karp [13] gave a partitioning algorithm for the traveling salesman problem. The set N of n points is assumed to be scattered randomly in a unit rectangular region R in the Euclidean plane. The algorithm partitions R into smaller regions R_i, each contains about t points. An optimum tour is constructed for the subset of points in each R_i, and these tours are then combined to yield a tour for N. Let $g(t)$ denote the time required to construct an optimum tour for t points. Karp showed that the time required by the partitioning algorithm is bounded by $2g(t)(n-1)/(t-1) + O(n \log n)$, where the first term represents the total time required to run the optimum subtour routine, and the second term time required to initially sort the n terminals as well as subsequent processing. He also proved that the difference between the length of the tour yielded by the partitioning algorithm and the length of an optimum tour T^* is bounded by $O(\sqrt{n/t})$.

When N is randomly generated by a Poisson distribution with mean n, Beardwood, Halton and Hammersley [1] proved that $|T^*| \to \beta\sqrt{n}$ with probability one, where β is a constant. Using this result and by properly choosing t, Karp showed that the partitioning algorithm is a *fully polynomial approximation scheme*, i.e., an algorithm which, for each set N and each $\epsilon > 0$, finds a tour T such that $|T|/|T^*| \leq 1 + \epsilon$ and that the time complexity is bounded by a polynomial in $1/\epsilon$ and in the length of the input.

Finally, Karp indicated the partitioning algorithm and its analysis are applicable to the Steiner tree problem, Euclidean or rectilinear distance. The actual description for such an application was carried out by Komlos and Shing for the rectilinear case and will be reported in Part III, Subsection 2.3.3.

4.6 Few's Algorithms

Few [7] gave a method to construct a tree interconnecting a set of n terminals lying in a unit square bounded by the lines $x = 0$, $x = 1$, $y = 0$ and $y = 1$. Partition the unit square into $2q$ congruent rectangles of length $1/2q$ and width 1. Label the $2q + 1$ lines $y = i/2q$, $i = 0, 1, \ldots, 2q$. Let T be the tree consisting of (i) the $q + 1$ lines $y = 2j/2q$, $j = 0, 1, \ldots, q$; (ii) the n shortest distance from each of the n terminals to the nearest such line; (iii) suitable portions of the lines $x = 0$ and $x = 1$. Similarly, let T' be the tree consisting of (i) the q lines $y = (2j + 1)/2q$, $j = 0, 1, \ldots, q - 1$; (ii) the n shortest distance from each of the n terminals to the nearest such line; (iii) suitable portions of the lines $y = 0$ and $y = 1$ (see Fig. 4.4, T in solid lines, T' in dotted lines with one overlapping segment). Note that the length of (iii) is never more than one for both T and T'. Therefore

$$|T| + |T'| \leq 2q + 1 + \frac{n}{2q} + 2 = 3 + 2q + \frac{n}{2q}$$

Minimizing the length over q, q should be the integer nearest to $\sqrt{n}/2$. With this choice, it can be shown that

$$|T| + |T'| \leq 2\sqrt{n} + 3.5$$

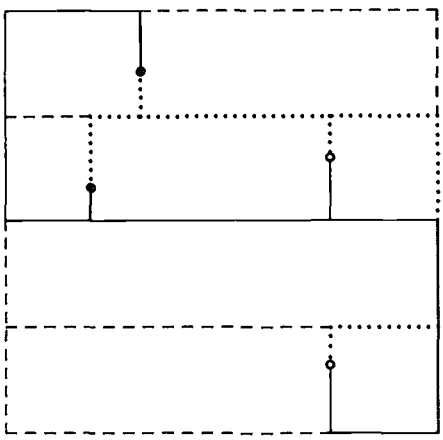


Figure 4.4: Few's two trees

Select the shorter tree between T and T'. Its length cannot exceed $\sqrt{n} + 1.75$. Therefore

Theorem 4.3 *For a set of n terminals lying in a unit square, Few's algorithm yields a tree whose length is at most $\sqrt{n} + 1.75$.*

Note that Few's algorithm actually constructs a rectilinear tree. So his algorithm is also a heuristic for rectilinear SMT. Furthermore, it can be generalized to a d-space.

4.7 A Graph Approximation Algorithm

Provan [18] gave the following approximation algorithm which transforms a Euclidean Steiner problem into a Steiner problem on a graph such that the difference of the lengths of the two SMTs in the two respective problems is bounded by ϵ for any predetermined $\epsilon > 0$.

Let N be the given set of terminals. Find the region R contained in the convex hull of N, and also find the values $x_{\min}, x_{\max}, y_{\min}$ and $y_{\max}$ of the minimum and maximum x-coordinate and y-coordinate, respectively, of points in N. Let $m = \lceil (8n - 12)/\epsilon \rceil$ and let $x_{\min} = x_0 < x_1 < \cdots < x_m = x_{\max}$ and $y_{\min} = y_0 < y_1 < \cdots < y_m = y_{\max}$ be values spaced equally between their endpoints. Let $V_0 = \{(x_i, y_j) : i, z = 0, \ldots, m\}$ be the lattice of points with these values. Define the weighted graph G_ϵ to be the complete graph with a vertex set $V = N \cup (V_0 \cap R)$ and edge weights $w(u, v)$ equal to the Euclidean distance between u and v. Then G_ϵ is the graph for the transformed Steiner problem.

Theorem 4.4 *Let T be an SMT for N and let T' be an SMT for N on G_ϵ. Then*

$$\frac{|T'|}{|T|} \leq 1 + \epsilon$$

Proof: Let $D = \max\{x_{\max} - x_{\min}, y_{\max} - y_{\min}\}$. Then $|T| \geq D$, since T must connect the points with coordinates determining D. Note that the lattice coordinates chosen satisfy

$$y_i - y_{i-1} \leq \frac{D\epsilon}{8n-12}$$

and

$$x_i - x_{i-1} \leq \frac{D\epsilon}{8n-12}$$

Also note that T contains at most $2n-3$ edges. Consider the subgraph T_0 of G_ϵ whose vertices are obtained by choosing, for each vertex v of T, the nearest vertex v' to v in G_ϵ, and whose edges are obtained by choosing for each edge (u,v) in T the edge (u',v') in G_ϵ. Then T_0 is a spanning graph for N. Furthermore, for each edge (u,v) in T we have

$$|u'-v'| \leq |u-v| + |u-u'| + |v-v'| < |u-v| + 4D\epsilon/(8n-12)$$

Consequently,

$$|T'| \leq |T_0| \leq |T| + (2n-3)[4D\epsilon/(8n-12)] \leq |T| + \epsilon|T|$$

□

Theorem 4.4 shows that any algorithm for the Steiner problem on a graph can be used to find an approximate solution to the Euclidean Steiner problem, using this transformation.

4.8 k-Size Quasi-Steiner Trees

A tree with a Steiner topology but not satisfying the angle conditions of an ST is called a *quasi-ST*. If every full component contains at most k terminals, then it is called a *k-size quasi-ST* (*k-size ST* if the angle conditions are satisfied). A shortest such tree is a *minimum k-size quasi-ST*, in which the subtree on each full component must be an SMT. Note that an MST is a minimum 2-size (quasi) ST.

A minimum k-size quasi-ST can be considered as a heuristic for SMT. Unfortunately, efficient algorithms are also unknown for such trees with $k \geq 3$. Du, Zhang and Feng [6], extending an idea of Zelikovsky, gave a heuristic for minimum k-size quasi-ST in an arbitrary metric space (see Part IV, Section 1.4). Since the heuristic will be described in details later, here we only give a brief sketch of its application to the Euclidean plane.

Construct full SMTs over all subsets of N of size j, $2 \leq j \leq k$. Use a greedy algorithm to link the SMTs of subsets into a k-size tree on N. Namely, start with a short SMT and then grow it by sequentially adding an SMT which will not form a cycle with the growing graph and which also satisfies some minimality conditions. The construction ends when no more SMT can be added.

Since all edges are SMTs, it is easily seen that the above algorithm outputs a tree interconnecting N. The time required to construct all the SMTs on subsets of size j, $2 \leq j \leq k$, is exponential in j (see Chapter 2). For each fixed k the

time complexity of the algorithm is $O(kn^k(n^2 \log n + f(k)))$ as shown later for an arbitrary metric space where $f(k)$ is the time required to construct an SMT for a set of k terminals with a given topology. The following result is a specialization of Theorem 1.14 of Part IV.

Theorem 4.5 *The heuristic of Du, Zhang and Feng has a performance ratio better than the Steiner ratio for* $k \geq 128$.

4.9 Other Heuristics

Soukup [22] commented that the function

$$f(z) = (z - p_1)(z - p_2) \cdots (z - p_n)$$

where $p_1, \ldots, p_n$ are the n terminals, can be regarded as a surface in a three dimensional space; thus resembling the membrane physical model of Section 1.8. Let $s_1, \ldots, s_{n-2}$ be the $n-2$ roots of $f''(z) = 0$. Soukup proposed to use an MST on $\{p_i\} \cup \{s_i\}$ with some postprocessing as a heuristic. One problem of this heuristic is the inherent difficulty of solving an $(n-2)$-degree polynomial. Obviously, numerical procedures are required for general n.

A k-SMT is a shortest interconnecting tree with at most k Steiner points. Clearly, k-SMTs can be used as heuristic SMTs. While a k-SMT can be used as a heuristic SMT, the time required for construction is a large polynomial of n unless k is very small, but then the performance cannot be good. Thus the k-SMT will be discussed in Section 6.4 as a generalization of the SMT rather than a heuristic. Here we just mention that Kallman [12] proved that the Steiner ratio for the 1-SMT is $\sqrt{3}/2$.

References

[1] J. Beardwood, J. H. Halton and J. M. Hammersley, The shortest path through many points, *Proc. Cambridge Phil. Soc.* **55** (1959) 299–327.

[2] J. E. Beasley, A heuristic for the Euclidean and rectilinear Steiner tree problem, Working paper, Management School, Imperial College, London (1989).

[3] O. Borůvka, O jistém problému minimálním, *Acta Societ. Scient. Natur. Moravicae* **3** (1926) 37–58.

[4] S. K. Chang, The generation of minimal trees with a Steiner topology, *J. Assoc. Comput. Mach.* **19** (1972) 699–711.

[5] D. Cheriton and R. E. Tarjan, Finding minimum spanning trees, *SIAM J. Comput.* **5** (1976) 724–742.

[6] D. Z. Du, Y. J. Zhang and Q. Feng, On better heuristic for Euclidean Steiner minimum trees, *Proc. of the 32-nd Ann. Symp. on Foundations of Computer Science* (1991) 431–439.

[7] L. Few, The shortest path and shortest road through n points, *Mathematika* **2** (1955) 141–144.

[8] E. N. Gilbert, Random minimal trees, *SIAM J. Appl. Math.* **13** (1965) 376–387.

[9] R. L. Graham and P. Hell, On the history of the minimum spanning tree problem, *Ann. Hist. Comput.* **7** (1985) 43–57.

[10] F. K. Hwang and F. J. Weng, The shortest network under a given topology, *J. Algorithms*, to appear.

[11] V. Jarník, O jistém problému minimálním, *Acta Societ. Scient. Natur. Moravicae* **6** (1930) 57–63.

[12] R. R. Kallman, A conjecture of Gilbert and Pollak on minimal trees, *Stud. Appl. Math.* **22** (1973) 141–151.

[13] R. M. Karp, Probabilistic analysis of partitioning algorithms for the traveling salesman problem in the plane, *Math. Oper. Res.* **2** (1977) 209–224.

[14] P. Korhonen, An algorithm for transforming a spanning tree into a Steiner tree, in *Survey of Math. Programming, Proc. of the 9-th Int. Math. Program. Symp.* **2** North-Holland (1974) 349–357.

[15] J. B. Kruskal, On the shortest spanning tree of a graph and the traveling salesman problem, *Proc. Am. Math. Soc.* **7** (1956) 48–50.

[16] M. Lundy, Applications of the annealing algorithm to combinatorial problems in statistics, *Biometrika* **72** (1985) 191–198.

[17] R. C. Prim, Shortest connection networks and some generalizations, *Bell System Tech. J.* **36** (1957) 1389–1401.

[18] J. S. Provan, Convexity and the Steiner tree problem, *Networks* **18** (1988) 55–72.

[19] M. J. Shamos and D. Hoey, Closest point problems, *Proc. of the 16-th Ann. Symp. on Foundations of Computer Science* (1975) 151–162.

[20] J. M. Smith, D. T. Lee and J. S. Liebman, An $O(n \log n)$ heuristic for Steiner minimal tree problems on the Euclidean metric, *Networks* **11** (1981) 23–29.

[21] W. D. Smith and P. W. Shor, Steiner tree problems, *Algorithmica* **7** (1992) 329-332.

[22] J. Soukup, Minimum Steiner trees, roots of a polynomial and other magic, *ACM/SIGMAP Newsletter* **22** (1977) 37–51.

[23] E. A. Thompson, The method of minimum evolution, *Ann. Hum. Genet.* **36** (1973) 333–340.

This Page Intentionally Left Blank

Chapter 5

Special Terminal-Sets

There are three motivations to study terminal-sets with special geometric configurations whose SMTs can be obtained through analysis. The apparent one is that these special configurations may turn up in real applications, or the solutions for those special configurations may provide good approximations for similar configurations. A more practical motivation is that examples of large terminal-sets for which SMTs are known are needed for testing the performances of heuristic algorithms. A third motivation is an academic one. One would like to draw a sharper boundary between Steiner problems solvable in polynomial time and those which cannot to achieve a better understanding of the source of intractability.

Most of the special sets discussed in this chapter can be solved in polynomial time. We also include a few cases which we have learned some important properties, though polynomial time algorithms for them are not available yet. At present the most important open problem on this topic seems to be the development of an efficient SMT algorithm for a convex set of terminals, or a proof of its nonexistence.

5.1 Four Terminals

SMTs for three terminals have been well understood since Fermat's time. However, a complete understanding of SMTs for four terminals is a relatively recent thing. Pollak [15] and Ollerenshaw [14] started the task which culminated in a paper by Du, Hwang, Song and Ting [6].

Let N denote a set $\{a, b, c, d\}$ of four terminals. The first question is whether a full SMT exists. Pollak answered this question with the following result.

Lemma 5.1 *The existence of a full SMT(N) implies that the four terminals a, b, c and d form a convex quadrilateral.*

Without loss of generality, assume that $[a, c]$ and $[b, d]$ are the two diagonals of the convex quadrilateral, and they intersect at the point o. There are two possible FSTs, one with a and b adjacent to the same Steiner point, and the other with a and d (see Fig. 5.1). Call them the *ab-tree* and the *ad-tree*, respectively.

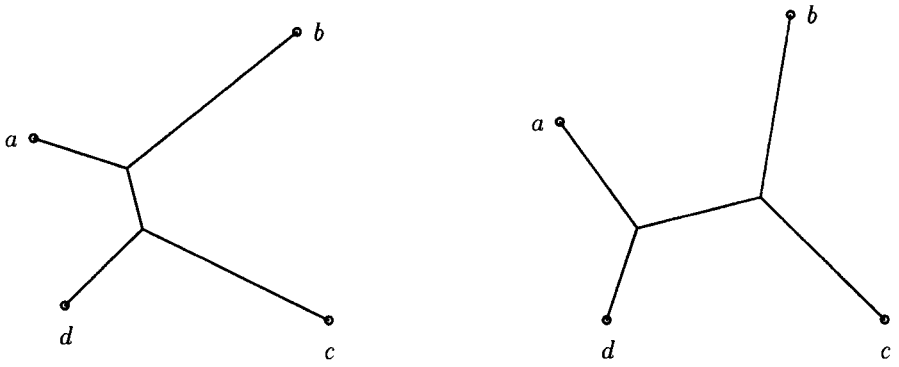


Figure 5.1: Two FSTs

Du, Hwang, Song and Ting gave necessary and sufficient conditions for the existence of the *ad*-tree, analogous conditions can be stated for the *ab*-tree.

Lemma 5.2 *Necessary and sufficient conditions for the existence of the ad-tree are (see Fig. 5.2):*

(i) *The quadrilateral abcd is convex.*

(ii) $\angle da(bc)$, $\angle(bc)da$, $\angle(da)bc$ *and* $\angle bc(da)$ *are all less than* $120°$.

(iii) $\angle aod < 120°$.

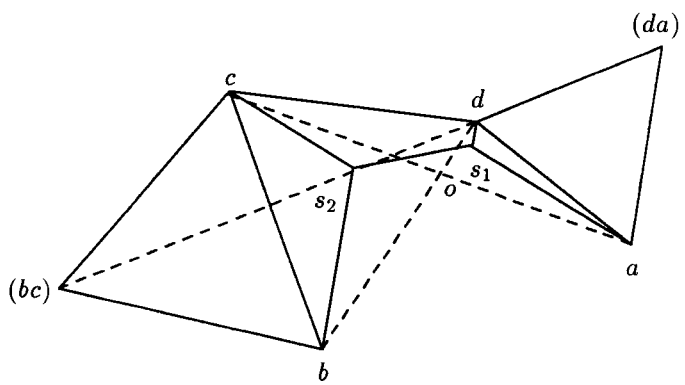


Figure 5.2: The *ad*-tree

Weng [18] showed that condition (i) is implied by conditions (ii) and (iii), and hence can be deleted from Lemma 5.2.

Given that an FST exists, how does one know whether it is an SMT? In particular, how can one compare the lengths of the two possible FSTs? Ollerenshaw gave an answer to the second question.

Lemma 5.3 *Suppose that both FSTs exist. The one with a longer edge connecting the two Steiner points is the shorter tree.*

Call the ad-tree *acute* and the ab-tree *obtuse* if $\angle aod < \angle boa$ and vice versa. Call both trees acute if $\angle aod = \angle boa = 90°$. Pollak gave an answer to the first question in terms of acute tree.

Theorem 5.1 *Let N denote a set of four terminals. If an acute FST exists for N, then it is an SMT.*

Proof: (sketch) The proof relies crucially on proving that the Simpson line of the acute FST is shorter than that of the obtuse tree regardless of whether the latter tree exists or not. Then the proof shows that every nonfull ST (including spanning trees) is at least as long as one of the Simpson line. □

Note that Pollak originally stated the theorem under the condition that both full SMTs exist. Du, Hwang, Song and Ting gave the current version and also gave a simpler proof for the comparison of the two Simpson lines. Furthermore, they showed that if the acute FST does not exist, then the obtuse FST can sometime be the SMT. This was argued by first constructing an example for which $\angle aod = \angle boa$ but the ad-tree violates conditions in Lemma 5.2 and does not exist. By Theorem 5.1 the ab-tree is the unique SMT. A continuity argument now extends the minimality to the case that $\angle boa$ is slightly above 90°. They also gave sufficient conditions under which various nonfull STs or spanning trees are SMTs.

There are some good reasons to study SMTs for four terminals. Pollak used his results on them to obtain a proof that the Steiner ratio is true for all 4-terminal sets.

Du, Hwang, Song and Ting were motivated by the need to prove a decomposition theorem involving the removal of a quadrilateral from a convex hull (see Theorem 1.6). The following result obtained by them was crucially used in that effort.

Theorem 5.2 *Let $abcd$ be a convex quadrilateral with $\angle a \geq 120°$, $\angle b \geq 120°$ and $\angle boa \geq \angle a + \angle b - 150°$. Let $a'b'c'd'$ be a quadrilateral embedded in $abcd$ with a', d' on $[a, d]$ and c', b' on $[c, b]$. Then an SMT for $\{a', b', c', d'\}$ cannot be full.*

Theorem 5.1 would be even better if no assumption of the existence of any FST is required. This was accomplished by Weng [18] by considering the ST in the degenerate class. Define the *$\overline{ab}$-tree* as the unique ST, if it exists, in the degenerate class of the ab-topology, and define the $\overline{ad}$-tree similarly. Then the shorter of the $\overline{ab}$-tree and the $\overline{ad}$-tree is an SMT (both are if they are of equal length). Weng gave conditions to compare the lengths of these two trees in the following theorem.

A quadrilateral is called *skew* if it contains two opposite sides each longer than or equal to a distinct remaining side, e.g., $|ab| \geq |ac|$, $|cd| \geq |bd|$. In the above example $[a, b]$ and $[c, d]$ will be referred to as the *dominant* sides.

Theorem 5.3 *Suppose that abcd is a skew quadrilateral. Then the $\overline{ab}$-tree ($\overline{ad}$-tree) is an SMT if and only if $[a,d]$ ($[a,b]$) is a dominant side.*

Weng also gave an embedding result parallel to Theorem 5.2.

Theorem 5.4 *Let $a'b'c'd'$ be a quadrilateral embedded in a convex quadrilateral abcd where $[a',d']$ is on $[a,d]$ and $[b',c']$ on $[b,c]$. If the $\overline{ad}$-tree is an SMT for $\{a,b,c,d\}$, then the $\overline{a'd'}$-tree is an SMT for $\{a',b',c',d'\}$.*

This is proved by showing that $|\overline{ab}\text{-tree}| - |\overline{ad}\text{-tree}|$ is increasing as a moves to a' (or b moves to b', ...).

5.2 Cocircular Terminals

The first modern paper on the Steiner problem was by Jarnik and Kössler [13] who studied the case where the terminals are the n corners of a regular n-gon. They constructed SMTs for $n = 3, 4, 5, 6$ (an SMT for the last case is an MST) and also showed by one sweeping elegant argument that an MST for the corners of a regular n-gon, $n \geq 13$, is an SMT.

Graham [10] considered the more general configurations of cocircular terminals and conjectured that if the distances between all pairs of adjacent terminals do not exceed the radius, then an MST is an SMT. Later, he updated the conjecture to allow one pair of consecutive terminals to have a distance greater than the radius. Du, Hwang and Chao [5] gave the following result which establishes a relation between the Steiner ratio and an SMT for cocircular terminals.

Lemma 5.4 *Suppose that N is a set of cocircular terminals on a unit circle such that at most one pair of adjacent terminals has a distance exceeding m with*

$$m = min\left\{ \left[\alpha\beta + \sqrt{\alpha^2 + (1-\beta^2)/4}\right] /(\alpha^2 + 1/4), \gamma \right\}$$

where

$$\begin{aligned} \alpha &= \sqrt{3} + 1 - 1/2\bar{\rho} \\ \beta &= 1 - (1-\bar{\rho})\pi/\bar{\rho} \end{aligned}$$

($\bar{\rho}$ is any lower bound of the Steiner ratio), and

$$\gamma = 2(\sqrt{3}+1)/[(\sqrt{3}+1)^2 + 1/4] \cong .708$$

Then an MST for N is also an SMT.

By substituting $\bar{\rho} = \sqrt{3}/2$, it follows $m \cong .708$. Let l_n denote the length of a side of a regular n-gon circumscribed in a unit circle. Clearly, $l_n = 2\sin(\pi/n)$ is decreasing in n. Since

$$l_9 = 2\sin 20^\circ \cong .6840 < .708$$

Du, Hwang and Weng [8] used Lemma 5.4 to show that an MST is an SMT for l_n for $n \geq 9$. They also analyzed all possible cases for $n = 7$ and 8 to complete the proof for the following result.

Theorem 5.5 *An MST for the corners of a regular n-gon, $n \geq 6$, is an SMT.*

Back to the cocircular case. Recall from Section 3.3 that Rubinstein and Thomas developed the variational approach and for a direction V at the configuration $Y \in \Delta(F)$, computed

$$\rho'(V) = \frac{L'_T(V)}{L_T(V)} \left[\frac{L'_S(V)}{L'_T(V)} - \rho \right]$$

where S is an SMT and T an MST for Y such that $T(h)$, which has the same topology as T, is an MST for $Y + hV$ with h sufficiently small. They called V a *reversible* direction if reversing the direction of variation at ρ yields a change of signs of $L'_T(V)$ and $L'_S(V)$. This implies that $\rho'(V) = 0$, and consequently,

$$L'_S(V) = \rho L'_T(V)$$

if V is reversible. Furthermore, the second derivative of ρ also has a simple expression for reversible variations, i.e.,

$$\rho''(V) = \frac{L''_S(V) - \rho L''_T(V)}{L_T}$$

The strategy is to show that for every configuration at which ρ is minimal, a reversible variation can be constructed for which

$$L''_S(V) - \rho L''_T(V) < 0$$

Therefore $\rho''(V) < 0$, contradicting the minimality of ρ.

In general, $L''_S(V)$ and $L''_T(V)$ are much harder to compute than $L'_S(V)$ and $L'_T(V)$. Rubinstein and Thomas [16] introduced some methods to compute these second derivatives at reversible variations. Using the conditions on $\rho'(V)$ and $\rho''(V)$, they were able to show that if ρ achieves a minimum at the set $N = \{p_1, \ldots, p_n\}$, ordered around the circle, such that there exist an MST and an SMT, not identical, then the following facts hold:

Fact 1. If $|p_i, p_{i+1}| > 1$, then $|p_{i-1}, p_i| = |p_{i+1}, p_{i+2}| = 1$.
Fact 2. Either $|p_{i-1}, p_i| \geq 1$ or $|p_i, p_{i+1}| \geq 1$ for $1 \leq i \leq n$.
Fact 3. There exists an i, $0 \leq i \leq n-1$, such that $|p_i, p_{i+1}| \geq 1$.

From these facts, it can be immediately deduced that the MST can have at most five edges of length ≥ 1 (at most one > 1), and at most five edges of length ≤ 1. There are only seventeen cases of N satisfying these conditions up to rotation and reflection. Rubinstein and Thomas also proved that ρ can only achieve minimum when the SMT is full. They used this fact and the second derivative method to conclude that ρ can achieve minimum at none of the seventeen configurations. Thus the Graham's conjecture was proved. Namely,

Theorem 5.6 *Consider a set of cocircular terminals. If at most one pair of adjacent terminals has a distance greater than the radius, then an MST is an SMT.*

5.3 Co-path Terminals

Consider a simple path consisting of line segments such that there is a terminal at every turning point as well as the two ends of the path. We call such a set of terminals *co-path* terminals. Of course, any set of points in the plane can be construed as co-path terminals by running a proper path through them. So one will view a set of terminals as co-path terminals only when the path possesses some interesting properties.

Let $[p_1, p_n]$ denote the segment connecting the two ends of a path H. Then $[p_1, p_n]$ and H together enclose a sequence of m polygons where two adjacent polygons (a polygon can degenerate into a segment of H) meet at an intersection of $[p_1, p_n]$ and H. Let $T_i, i = 1, \ldots, m$, denote a full SMT for the vertices of the i^{th} polygon. Then $\cup_{i=1}^{m} T_i$ is called a *cut-and-patch tree* for N. Most existing results for co-path terminals are to determine conditions on the paths such that cut-and-patch trees are SMTs.

The first such result was given by Chung and Graham [3] for sets of co-path terminals called *ladders*. Let L_n denote a ladder of $2n$ terminals arranged in a rectangular 2 by n array with adjacent pairs of terminals forming a unit square. The path is an alternating sequence of vertical and horizontal segments wrapping around the squares in order.

Theorem 5.7 *A cut-and-patch tree for L_n is an SMT. Furthermore,*

$$|L_n| = \begin{cases} \sqrt{(n(1+\sqrt{3}/2)-1)^2 + 1/4} & \text{if } n \text{ is odd,} \\ n(1+\sqrt{3}/2) - 1 & \text{if } n \text{ is even.} \end{cases}$$

For n even, $[p_1, p_{2n}]$ is a side of the $2 \times n$ array and cuts H into $n/2$ 1×1 squares connected by $n/2 - 1$ unit segments. For n odd, $[p_1, p_{2n}]$ is a diagonal of the $2 \times n$ rectangle and cuts H into $m - 1$ trapezoids. The cut-and-patch trees for $n = 4$ and 5 are illustrated in Fig. 5.3.

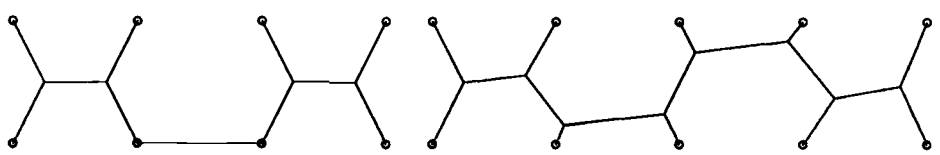


Figure 5.3: SMTs for ladders

Note that there are two SMTs for the four corners of a unit square and either one can be inserted into any one of the $n/2$ squares for the even n case.

All segments in the path of a ladder have the same length. Du and Hwang [4] allowed each segment to have arbitrary length and called the path a *rectangular wave*. Call a vertical segment a *bar* and let B_i denote the i^{th} bar. Let v_i denote the height of B_i, and let h_i denote the horizontal distance between B_i and B_{i+1}.

A rectangular wave is called *mild* if $h_i \geq \max\{v_i, v_{i+1}\}$ for all i. Furthermore, let v_0 denote the vertical distance between p_1 and p_{2n}. Du and Hwang proved:

Theorem 5.8 *Let P be a set of $2n$ co-path terminals where the path is a mild rectangular wave. If $[p_1, p_{2n}]$ intersects every bar, then a cut-and-patch tree is an SMT with length*

$$\sqrt{\left(\sum_{i=1}^{n-1} h_i + (\sqrt{3}/2)\sum_{i=1}^{n} v_i\right)^2 + v_0^2/4}$$

Note that Theorem 5.7 is a special case of Theorem 5.8 with $h_i = v_i = 1$ for all i, $v_0 = 0$ for n even and $v_0 = 1$ for n odd.

Du and Hwang made a further attempt of generalization by allowing the segments connecting two adjacent bars to be nonhorizontal. They called such a path a *bar wave*. A bar wave is *mild* if $h_i \geq \max\{v_i, v_{i+1}\}$ for all i and if every angle wrapped by the path is less than 120°. Consider an ST T for a terminal set P on a bar wave. For a given T a bar is called top-first (or bottom first) if T around it has the form shown in Fig. 5.4a (or Fig. 5.4b):

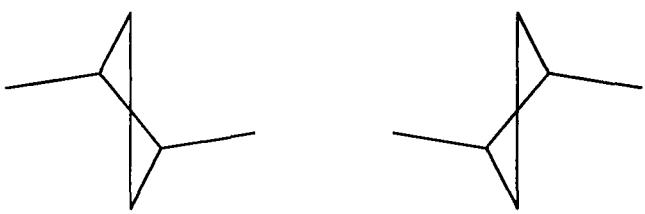


Figure 5.4: Top (bottom) first bars

Let T_i denote the subtree of T lying between B_i and B_{i+1}. Then T is called *formal* if every bar is either top-first or bottom-first, and every T_i is a 4-point FST. A formal tree can be characterized by a vector $(e_2, e_3, \ldots, e_{n-1})$ where $e_i = 1$ if B_i is top-first and $e_i = -1$ if B_i is bottom first. Let y_i and y_i' denote the y-coordinates of the upper and lower end, respectively, of B_i. Then the length of a full formal tree is

$$f(e_2, \ldots, e_{n-1}) = \sqrt{\left(\sum_{i=1}^{n-1} h_i + (\sqrt{3}/2)\sum_{i=1}^{n} v_i\right)^2 + \left(y_1 + y_1' - y_n - y_n' + \sum_{i=2}^{n-1} v_i e_i\right)^2}/4$$

A formal tree which minimizes $f(e_2, \ldots, e_{n-1})$ is an SMT. Note that the above minimization problem amounts to a partition problem, i.e., to partition the $n-2$ numbers v_i, $i = 2, \ldots, n-2$, into two sets headed by the two numbers $y_1 + y_1'$ and $y_n + y_n'$, respectively, so that the sums of the numbers in the two sets are as close as possible. The partition problem is an NP-complete problem. But in the case that for each $B_i, i = 2, \ldots, n-2$, there exist $j < i < k$ such that either $y_i = y_j$,

$y'_i = y'_k$ or $y_i = y_k$, $y'_i = y'_j$, then one can set $e_i = -1$ in the former case and $e_i = 1$ in the latter case to obtain

$$y_1 + y'_1 - y_n - y'_n + \sum_{i=2}^{n-1}(y_i - y'_1)e_i = 0$$

which obviously minimizes $f(e_2, \ldots, e_{n-1})$. One can view such a formal tree as a many-cuts-and-patch tree where each cut corresponds to a horizontal segment connecting a pair of bar-ends (see Fig. 5.5).

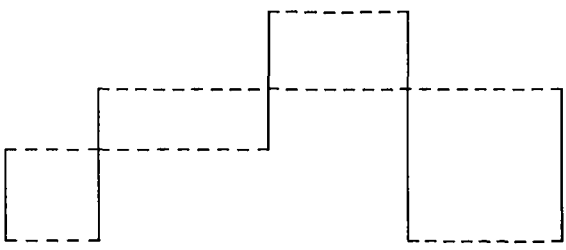


Figure 5.5: A special formal tree

Du, Hwang and Weng [7] considered paths, called *zigzag lines*, such that p_{i+2}, $i = 1, \ldots, n-2$, is to the right (left) of the directed line from p_i to p_{i+1} if i is odd (even). A zigzag line is *regular* if the smaller angle between two adjacent segments is a constant α. It is *convex* if all terminals lie on its convex hull $p_1p_3p_5 \cdots p_n \cdots p_4p_2$. It is *normal* if $|p_i, p_{i+1}| \leq |p_i, p_{i+2}|$ for $1 \leq i \leq n-3$ and $|p_{i+1}, p_{i+2}| \leq |p_i, p_{i+2}|$ for $2 \leq i \leq n-2$. They proved.

Theorem 5.9 *Let N be a terminal set on a regular, convex and normal zigzag line with $\alpha \geq 60°$. Then a cut-and-patch tree is an SMT for N whose length is*

$$\sqrt{\left(\sum_i |p_{2i-1}, p_{2i}|\right)^2 + \left(\sum_i |p_{2i}, p_{2i+1}|\right)^2 - 2\sum_i |p_{2i-1}, p_{2i}| \sum_i |p_{2i}, p_{2i+1}| \cos(60° + \alpha)}$$

Fig. 5.6 illustrates such an SMT.

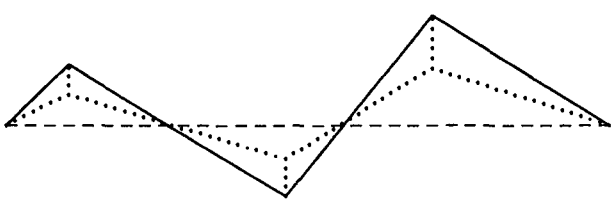


Figure 5.6: SMT for a zigzag line

Recently, Booth and Weng [1] generalized the above result by weakening the "regular" and "convex" condition. A zigzag line is called *Steiner* if all terminals lie on its Steiner hull.

Theorem 5.10 *Let P be a terminal set on a Steiner and normal zigzag line such that the first angle and the last angle wrapped by the path is at least 60°. Then there exists $p_{i_1}, p_{i_2}, \ldots, p_{i_{m-1}}$, $1 \equiv i_0 < i_1 < i_2 < \cdots < i_m - 1 < i_m \equiv n$, such that $\cup_{j=1}^{m} T_{i_j}$ is an SMT, where T_{i_j} is the full SMT on the zigzag line $p_{i_{j-1}}\, p_{i_{j-1}} \cdots p_{i_j}$.*

Fig. 5.7 illustrates such an SMT.

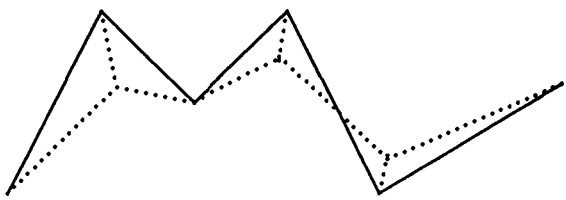


Figure 5.7: SMT for a generalized zigzag line

Note that regardless of where the points $p_{i_1}, \ldots, p_{i_{m-1}}$ are, the topology for the SMT is in the class $D(F)$, where F is the full topology whose circumference order is $p_1 p_2 p_4 p_6 \cdots p_n \cdots p_7 p_5 p_3$. Therefore, one can apply the luminary algorithm (see Section 2.8) to construct an SMT in $O(n^2)$ time.

5.4 Terminals on Lattice Points

Chung, Gardner and Graham [2] studied SMTs when terminals are arranged at regular lattices of unit squares, like the points at the corners of the cells of a checkerboard. They gave various constructions for square lattices but no proof of optimality exists except for the 2×2 square. A basic building block, denoted by X (see Fig. 5.8), in these constructions is a 4-point full SMT for the four corners of a unit square. Chung, Gardner and Graham proved:

Theorem 5.11 *If a rectangular array can be spanned by an ST made up entirely of Xs, then the array is a square of size 2^t by 2^t for some $t \geq 1$.*

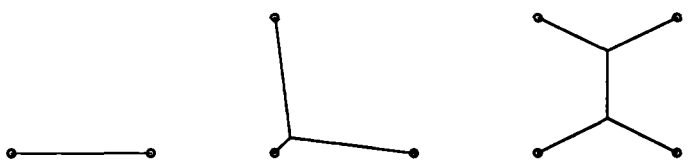


Figure 5.8: I, Y and X

For square arrays of size $n = 2^t$, they gave a recursive construction of such an ST B_n by connecting four $B_{n/2}$ by an X in the center. Thus B_n consists of $(4^t - 1)/3$ Xs and $|B_n| = [(4^t - 1)/3](1 + \sqrt{3})$.

For square arrays of size $n \neq 2^t$, three other building blocks, I: a segment between two adjacent terminals, Y: 3-point full SMT on three corners of a unit square (see Fig. 5.8) and Z: a full SMT on a 2×5 ladder (as shown in Fig. 5.3), are used to fill the gaps between Xs.

Let B_n denote the ST obtained by their recursive construction for an $n \times n$ array. Table 5.1 summarizes the constructions for $n \neq 2^t$.

n	B_n	$\lvert B_n \rvert$
$6k$	$(12k^2-1)X+Y$	$(12k^2-1)(1+\sqrt{3})+(1+\sqrt{3})/\sqrt{2}$
$6k+1$	$(12k^2+4k-1)X+3I$	$(12k^2+4k-1)(1+\sqrt{3})+3$
$6k+2$	$(12k^2+8k-2)X+Z$	$(12k^2+8k-2)(1+\sqrt{3})+\sqrt{35+20\sqrt{3}}$
$6k+3$	$(12k^2+12k+2)X+2I$	$(12k^2+12k+2)(1+\sqrt{3})+2$
$6k+4$	$(12k^2+16k+2)X+Z$	$(12k^2+16k+2)(1+\sqrt{3})+\sqrt{35+20\sqrt{3}}$
$6k+5$	$(12k^2+20k+7)X+3I$	$(12k^2+20k+7)(1+\sqrt{3})+3$

Table 5.1: B_n and its length

Chung, Gardner and Graham conjectured that B_n is an SMT for all n. They felt particularly confident when $n = 2^t$.

Hwang [11] studied regular triangular arrays and regular hexagonal arrays which consist of unit triangles and the lattice points are the corners of the triangles. Let $t_n(h_n)$ denote the former (latter) array with n lattice points on each of the three (six) sides. Note that a Chinese checkerboard c_n of size n consists of six t_n each attaches to a side of a h_n (see Fig. 5.9).

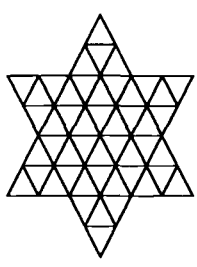


Figure 5.9: A Chinese checkerboard

Hwang gave recursive constructions of a tree T_n for t_n and a tree H_n for h_n, which uses only Y^*, the 3-point full SMT on the three corners of a unit triangle, except when $n \equiv 0$ or 3 (mod 4), then a single edge connecting a pair of adjacent terminals needs to be used. He also showed that for $n > 2$, six T_n and one H_n, with some possible modifications, can be properly connected to yield an ST C_n for c_n which uses Y^* only. By Theorem 2.3, it follows:

Theorem 5.12 *T_n for $n \equiv 1$ or 2 (mod 4), H_n for $n \geq 1$ and C_n for $n \neq 2$ are SMTs with lengths*

$$\begin{aligned} |T_n| &= [(n+2)(n-1)/4]\sqrt{3} \\ |H_n| &= 3[n(n-1)/2]\sqrt{3} \\ |C_n| &= [3n(n-1)]\sqrt{3} \end{aligned}$$

C_2 is given in Fig. 5.10 and was verified to be optimal by the computing algorithm of Smith (see Section 6.1).

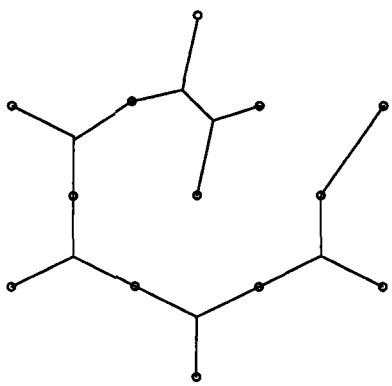


Figure 5.10: C_2

T_n for $n \equiv 0$ or 3 has length

$$|T_n| = [(n+1)n/4 - 1]\sqrt{3} + 1$$

and is conjectured to be an SMT.

5.5 Two Related Results

Hwang, Weng and Du [12] gave a method to grow an FST, called a *splitting tree*, such that the tree is always the unique SMT for its set of endpoints. Unlike the special cases in the previous sections where a terminal-set can be described by an explicit property, here the terminal-set is characterized by the implicit property that there exists a splitting tree with the terminal-set as its set of endpoints. The construction of splitting trees also answers affirmatively the question whether for any full Steiner topology there exists a set N such that an SMT(N) has that topology.

A splitting tree T_n with n terminals can be defined as follows:

(i) A T_3 is a 3-point full SMT for the three corners of an equilateral triangle.

(ii) A T_n, $n > 3$, is obtained from a T_{n-1} by splitting an endpoint w into two new edges $[w, u]$ and $[w, v]$, where u and v are new endpoints of T_n, such that

(a) The three edges at the splitting point meet at 120°.

(b) The lengths of the two new edges are equal and less than $\lambda_{n-1} \equiv \min_{R_{n-1}}\{|R_{n-1}| - T_{n-1}\}$, where R_{n-1} is any other tree interconnecting endpoints of T_{n-1}.

A splitting tree T_5 is illustrated in Fig. 5.11.

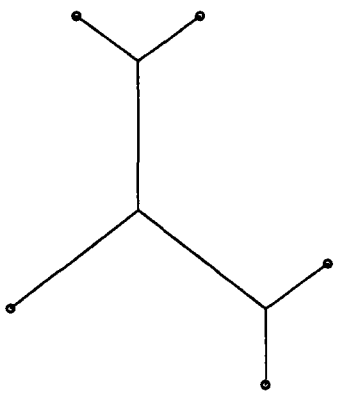


Figure 5.11: A splitting tree T_5

Hwang, Weng and Du proved:

Theorem 5.13 *T_n is the unique SMT for its endpoints.*

Proof: (sketch). Suppose that $R_n \neq T_n$ is an SMT for the set of endpoints. Let T_n be obtained from T_{n-1}. Modify R_n into a tree R_{n-1} interconnecting the endpoints of T_{n-1}. Then it can be shown that the topologies of R_{n-1} and T_{n-1} are different. Therefore,

$$|R_{n-1}| \geq |T_{n-1}| + \lambda_{n-1}$$

By using the relations between T_n and T_{n-1}, and between R_n and R_{n-1}, it can be shown that $|R_n| > |T_n|$, contradicting the claim that R_n is an SMT. □

Note that $\lambda_{n-1} > 0$ for all n since T_{n-1} is the unique SMT. Therefore T_n is well defined.

Weng [17] gave the following interesting result.

Theorem 5.14 *Let S be an ST on a set N of $2m$ terminals with a symmetrical topology G which maps terminal p_i to terminal p_{m+i}, $i = 1, \ldots, m$. Let N' be the set of $2m$ terminals obtained from N by shifting the segments $[p_i, p_{m+i}]$ parallelly such that the midpoints of all m segments coincide. Let S' be an ST on N' with the same topology G. Then $|S| = |S'|$.*

Proof: Theorem 5.14 is trivially true for $m = 1$. The general $m \geq 2$ case is proved by induction on m.

Without loss of generality assume that p_1 and p_2 are siblings. Then p_{m+1} and p_{m+2} are also siblings. Let p and q be the correct E-points for the pairs (p_1, p_2) and (p_{m+1}, p_{m+2}) with respect to G. Let o, o_1, o_2 be the midpoints of $[p, q]$, $[p_1, q_1]$, $[p_2, q_2]$. By the theorem of similarity [9, p. 118], $\Delta o o_1 o_2$ is equilateral.

Shift $[p_1, p_{m+1}]$ and $[p_2, p_{m+2}]$ to $[p'_1, p'_{m+1}]$ and $[p'_2, p'_{m+2}]$ parallelly such that o_1 and o_2 coincide with o. Then $\Delta p p_1 p'_1$ and $\Delta p p_2 p'_2$ are equivalent since $|p p_1| = |p p_2|$, $|p_1 p'_1| = |o_1 o| = |o_2 o| = |p_2 p'_2|$ and $\angle p p_1 p'_1 = \angle p p_2 p'_2$ from $\angle p_1 p p_2 = \angle o_1 o o_2$ (see Fig. 5.12). Therefore $|p p'_1| = |p p'_2|$ and $\angle p'_1 p p'_2 = \angle p_1 p p_2 = 60°$, i.e., p is also the correct E-point for the pair (p'_1, p'_2). Theorem 5.14 now follows by induction. □

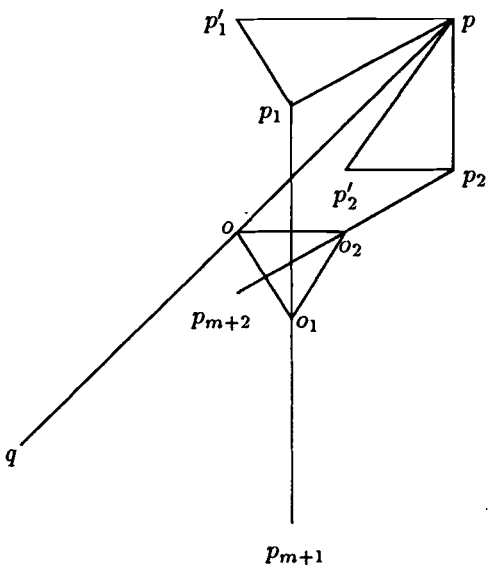


Figure 5.12: Parallel shifting

Weng gave an application of Theorem 5.14 to the 4-terminal case. Let N be a 4-terminal set and suppose both FSTs for N exist. Since both FSTs have symmetrical topologies, one may assume that the 4 terminals form a parallelogram by Theorem 5.14. Theorem 5.1 is then an easy consequence.

References

[1] R. S. Booth and J. F. Weng, Steiner minimal trees for a class of zigzag lines, *Algorithmica* **7** (1992) 231-246.

[2] F. R. K. Chung, M. Gardner and R. L. Graham, Steiner trees on a checkerboard, *Math. Mag.* **62** (1989) 83–96.

[3] F. R. K. Chung and R. L. Graham, Steiner trees for ladders, *Ann. Discrete Math.* **2** (1978) 173–200.

[4] D. Z. Du and F. K. Hwang, Steiner minimal trees for bar waves, *Acta Math. Appl. Sin.* **3** (1987) 246–256.

[5] D. Z. Du, F. K. Hwang and S. C. Chao, Steiner minimal trees for points on a circle, *Proc. Am. Math. Soc.* **95** (1985) 613–618.

[6] D. Z. Du, F. K. Hwang, G. D. Song and G. Y. Ting, Steiner minimal trees on sets of four points, *Discrete Comput. Geom.* **2** (1987) 401–414.

[7] D. Z. Du, F. K. Hwang and J. F. Weng, Steiner minimal trees on zigzag lines, *Trans. Am. Math. Soc.* **278** (1983) 149–156.

[8] D. Z. Du, F. K. Hwang and J. F. Weng, Steiner minimal trees on regular polygons, *Discrete Comput. Geom.* **2** (1987) 65–87.

[9] H. Eves, *A Survey of Geometry*, Allyn and Bacon, Boston, 1972.

[10] R. L. Graham, Some results on Steiner minimal trees, Bell Labs Memo (1967).

[11] F. K. Hwang, Steiner minimal trees on the Chinese checkerboard, *Math. Mag.* **64** (1991) 332-339.

[12] F. K. Hwang, J. F. Weng and D. Z. Du, A class of full Steiner minimal trees, *Discrete Math.* **45** (1983) 107–112.

[13] V. Jarník and O. Kössler, O minimálních grafech obsahujících n daných bodu, *Čas. Pêstování Mat.* **63** (1934) 223–235.

[14] K. Ollerenshaw, Minimum networks linking four points in a plane, *Inst. Math. Appl.* **15** (1978) 208–211.

[15] H. O. Pollak, Some remarks on the Steiner problem, *J. Comb. Theory, Ser. A* **24** (1978) 278–295.

[16] J. H. Rubinstein and D. A. Thomas, Graham's problem on shortest networks for points on a circle, *Algorithmica* **7** (1992) 193-218.

[17] J. F. Weng, Symmetrization theorem for full Steiner trees, Preprint (1991).

[18] J. F. Weng, Restudy of Steiner minimal trees on four points, Preprint (1991).

Chapter 6

Generalizations

In this chapter we discuss cases where the basic assumptions of the ESP are generalized in some way. We partition the cases according to whether the generalizations are related to (i) space, (ii) costs of edges, (iii) terminals, (iv) Steiner points, (v) the presence of obstacles in the space.

6.1 d-Dimensional Euclidean Spaces

Since two touching lines in a d-space can be embedded in a 2-dimensional plane, lines still cannot meet at less than 120° in a d-dimensional ST. Hence it is still true that a Steiner point has at least three edges. The argument used in proving Theorem 1.2 remains valid to bound the number of Steiner points by $n-2$. Gilbert and Pollak [18] gave the following argument to show that at most three vectors can meet at angles of 120° and more. Suppose $v_1, \ldots, v_k$ are k such unit vectors. Then

$$0 \leq |v_1 + \cdots + v_k|^2 = \sum_i |v_i|^2 + \sum_{i \neq j} v_i \cdot v_j$$

Since v_i and v_j meet at 120° or more, $v_i \cdot v_j \leq -1/2$. Hence

$$0 \leq k - k(k-1)/2 = k(3-k)/2$$

an inequality which is satisfied only by $k = 0, 1, 2, 3$.

Surprisingly, the Melzak FST algorithm cannot be extended to the d-dimension case even for $d = 3$. The reason is that for two given terminal points a and b there is an infinite number of E-points c such that Δabc is equilateral. Since the GEOSTEINER algorithm, the negative edge algorithm and the luminary algorithm either use the Melzak FST algorithm as a subroutine, or use the E-point directly, they are affected by the same problem. Nevertheless, as the location of a Steiner point is a function of the locations of its three adjacent points, the locations of Steiner points can be computed by an iterative method. Such a method was reported by Thompson [33] who attributed it to Edwards.

Another iterative method applicable to the d-space is the Smith numerical algorithm (see Section 2.4). Let G be the given full topology and let the coordinates of the k^{th} terminal be denoted by the d-vector $\vec{x}_k$, where $\vec{x}_1, \ldots, \vec{x}_n$ are the terminals and $\vec{x}_{n+1}, \ldots, \vec{x}_{2n-2}$ the Steiner points. Smith [30] proved the following convergence theorem:

Theorem 6.1 *At the i^{th} step in an iterative process, $i = 0, 1, \ldots$, solve the system of $n-2$ linear equations*

$$\vec{x}_k^{(i+1)} = \sum_{j:[j,k]\in G} \frac{\vec{x}_j^{(i+1)}}{|\vec{x}_j^{(i)} - \vec{x}_k^{(i)}|} \Bigg/ \sum_{j:[j,k]\in G} \frac{1}{|\vec{x}_j^{(i)} - \vec{x}_k^{(i)}|}$$

for $n+1 \le k \le 2n-2$. Then from all initial choices of Steiner point coordinates $\vec{x}_k^{(0)}$, except for a set of measure zero in $\mathcal{R}^{(n-2)d}$, the iteration converges to the unique optimum Steiner point coordinates. This convergence happens in such a way that the tree length is decreasing over the iterations.

Smith also showed that each iteration can be solved in $O(nd)$ time and the convergence is linear.

The annealing algorithm of Lundy (see Section 4.4) was actually proposed as a heuristic for the d-space. However, since the Hwang-Weng ST-algorithm with a given topology is applicable only for the 2-space, one has to go back to the reposition schemes given by Lundy, one of which is just the Edwards iterative method given earlier.

Gilbert and Pollak conjectured that the minimum Steiner ratio is achieved at the corners of the d-dimensional regular simplex. Chung and Gilbert [7] computed the Steiner ratio for the corners of the d-dimensional regular simplex to be .8130 for $d = 3$, .7837 for $d = 4$, and = .7645 for $d = 5$. They also showed that the ratio is upper bounded by .6698 for large d. Smith used his numerical algorithm to compute SMTs for the d-dimensional regular simplex and the d-dimensional regular octahedron for $d \le 9$. He found that the Steiner ratio of the former is larger than that of the latter for every $d = 3, \ldots, 9$, thus disproving Gilbert and Pollak's conjecture for those d. Smith suspected that the conjecture is false for all $d \ge 3$, but could not confirm this due to the limitation of his numerical algorithm in computing larger SMTs.

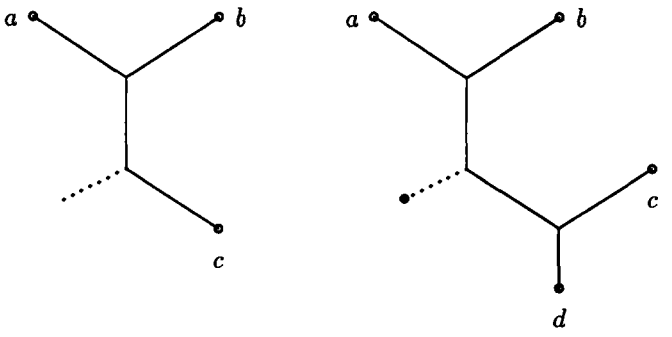
For a lower bound of the Steiner ratio in a d-space, note that the Graham and Hwang's $1/\sqrt{3}$ bound given in Section 3.1 remains valid for an arbitrary d-space. Du[13] examined some slightly more complicated local structures (see Fig. 6.1) and used a laborious analysis to prove

Theorem 6.2 *Let r^* denote the unique solution in the interval $(0.148, 0.15)$ of the equation*

$$\frac{2+r-e}{\sqrt{3}} = \frac{2+r}{\sqrt{8e^2+4er+2e}}$$

where $e = \sqrt{r^2+r+1}$. Then

$$\rho \ge \frac{2+r^*-\sqrt{(r^*)^2+r^*+1}}{\sqrt{3}} \sim 0.615$$



(a) (b)

Figure 6.1: Two local structures

Let $E_d(n)$ and $M_d(n)$ denote the expected length and the maximum length of an SMT over all sets of n points lying in a unit d-cube. Define

$$\beta_d(n) = E_d(n)n^{1-1/d}$$

and

$$\beta'_d(n) = M_d(n)n^{1-1/d}$$

Beardwood, Halton and Hammersley [3], using a partitioning argument, proved that $\beta_d(n)$ converges with probability 1 to a constant β_d. Steele [32] showed that the convergence to β_d follows from a result on subadditive Euclidean functionals (also true for the rectilinear metric). The convergence of $\beta'_d(n)$ is still an open problem. However, there exist several bounds on $\beta'_d(n)$.

Few [15], using an argument similar to the planar case (see Section 4.6), proved

$$\beta'_d(n) \leq d(8d-8)^{\frac{1-d}{2d}} + o(1)$$

Note that his construction for the upper bound is again a rectilinear tree. Smith [29], using a probabilistic construction, improved the first term of the upper bound to $\sqrt{d/2e\pi} \sim .2420\sqrt{d}$.

Let r_d denote the maximum radius of n identical d-spheres packed in a unit d-sphere. Moran [23] proved

$$r_d \leq \beta'_d(n) \leq \frac{2r_d d}{d-1} + o(r_d)$$

Smith noted that for d large,

$$.12098\sqrt{d} \leq r_d \leq .15975\sqrt{d}$$

This yields an upper bound of $.3195\sqrt{d}$ for $\beta'_d(n)$. He also noted that a lower bound can be obtained by multiplying a lower bound of the corresponding $\beta'_d(n)$ for MST

to a lower bound of ρ_d. Using an easily seen bound $2r_d$ for the former, and the .615 bound of Du for the latter, one obtains

$$\beta'_d(n) \geq 1.23r_d \geq .1488\sqrt{d}$$

Now consider terminals on the surface of a sphere. By using the fact that the azimuthal equidistant projection from sphere onto the Euclidean plane preserves distance, Litwhiler and Aly [22] proved (also see Cockayne [9]):

Theorem 6.3 *A point on the surface of a sphere can be a Steiner point (a local minimum) only if its planar image under the projection is a Steiner point.*

Furthermore, using the azimuthal equidistant projection with a Steiner point as pole, they noted that angles are preserved at the polar point. Hence the angle condition of the planar ST (see Section 1.3) also holds on the surface.

Dolan, Weiss and Smith [12] gave an $O(n \log n)$ heuristic for the spherical SMT which was derived from the planar heuristic using Delaunay triangulation (see Section 4.2).

6.2 Cost of Edge

Gilbert [17] considered a model where each edge e of a Steiner network is associated with a *capacity* c_e. Given a set of flows $w(i,j)$ for each pair (i,j) of terminals, the network consists of a set of capacitated edges which supports the flows between terminals i and j for all pairs (i,j). Let $f(c)$ denote the cost per unit length of an edge with capacity c. It is assumed that $f(c)$ is subadditive and increasing, i.e., $f(a)+f(b) \geq f(a+b) \geq \max\{f(a), f(b)\}$. The problem is to find a Steiner network which supports the flows and has minimum cost. One can also study a directed version of the Gilbert model, i.e., $w(i,j)$ is defined for every ordered pair (i,j). A special case of this model is when the n terminals consist of a unique source and $n-1$ sinks (or $n-1$ sources and one sink). Then $w(i,j)$ is zero except when i is the source (or j is the sink). Applications of this special case are pipeline networks or drainage networks.

Define a *capacitated topology* as a topology G and a set of capacities c_e for each edge e in G. A network which minimizes the cost for a given capacitated topology is called a *Gilbert network*. Vertices of a network which are not terminals are called *Gilbert points*.

For a given capacitated topology G, the cost of a network is

$$\sum_{e \in G} |e| \cdot f(c_e)$$

Since $f(c_e)$ is fixed for a given G, the cost is again a convex function of the locations of the Gilbert points. However, unlike the Steiner model, the case exceptional to strict convexity (as mentioned in the proof of Theorem 1.3) can occur. Fig. 6.2 illustrates a network which preserves the directions of all edges in perturbation.

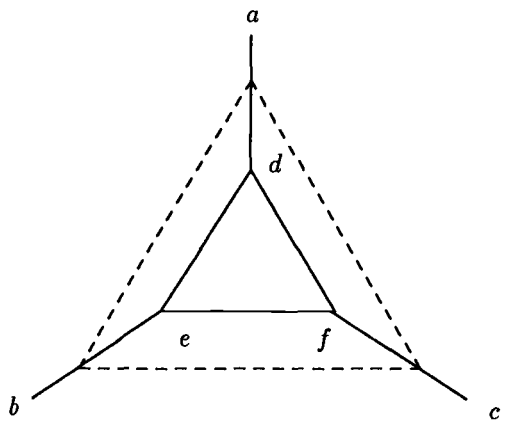


Figure 6.2: Invariant directions in perturbation

Gilbert gave the example shown in Fig. 6.2, with $c_{[a,d]} = c_{[b,e]} = c_{[c,f]} = 2$, $c_{[d,e]} = c_{[e,f]} = c_{[f,d]} = 1$, $f(1) = 1$, $f(2) = \sqrt{3}$. The network has minimum cost if abc and def are parallel equilateral triangles but network cost is invariant to the location of d, including the two extreme cases $d = a$ and $d = e = f$ (collapsing into one point). Minimizing networks may fail to be unique even when the topology is a tree, if the terminals are collinear. Gilbert showed that when n is even, $w(i,j) = 1$ for all i, j, and the tree consists of an edge from each terminal point to the Steiner point, then the cost is invariant to the location of the Steiner point as long as it lies between the two median terminals.

For a given N and $\{w(i,j)\}$, a Gilbert network with minimum cost is called a *minimum Gilbert network* (MGN). The minimum cost network not allowing any Gilbert point is called a *minimum regular network* (MRN). Clearly, if $f(c)$ is proportional to c, then an MRN is an MGN. On the other hand, if $f(c)$ is a constant, then an MGN is an SMT. Note that there are two reasons that an MGN does not have to be a tree. One is demonstrated by the fact that an MRN, which is not a tree in general, can be an MGN. The second is that it can be cost-saving to split $w(i,j)$ flows over two paths, thus creating a cycle. Gilbert proved:

Theorem 6.4 *If f is concave, then there exists an MGN without splitting paths. If f is strictly concave, then an MGN has no splitting paths.*

From now on we will confine our attention to networks whose topologies are Steiner, i.e., the topologies are trees and each Steiner point has three edges. (The reason for this restriction is simply that not much is known otherwise.) Such a network is called a *Gilbert-Steiner tree*. Correspondingly, we will be using the terms minimum Gilbert-Steiner tree and minimum Gilbert-spanning tree. Define the Gilbert-Steiner ratio analogous to the Steiner ratio, except that the former is a function of f and $\{w(i,j)\}$. Let ρ^0 denote the infimum ρ over all N, f and $\{w(i,j)\}$, Trietsch and Handler [35] proved that for $n = 3$, $\rho^0 \geq \sqrt{3}/2$ with computer-aided

computations. Du and Hwang [14] gave a simple geometric proof, and also showed that $\rho^0 \leq 2(\sqrt{2}-1) < \sqrt{3}/2$ for $n = 4$.

Suppose that three edges v_1, v_2, v_3 with costs per unit length $|v_1|, |v_2|$ and $|v_3|$ are adjacent to a Gilbert-Steiner point s. The law of cosines determines the angles between the edges. For example,

$$\cos(v_2, v_3) = \frac{(|v_1|^2 + |v_2|^2 - |v_3|^2)}{2|v_2||v_3|}$$

For the ordinary Steiner case one has $|v_1| = |v_2| = |v_3|$, and $\cos(v_1, v_2) = \cos(v_1, v_3) = \cos(v_2, v_3) = -1/2$. Hence the angles between the three edges are all 120°.

Gilbert observed that the Melzak FST algorithm can be generalized to construct full Gilbert-Steiner trees by simply generalizing the definitions of E-points. Let a and b be two sibling terminal points both adjacent to the Steiner point s. Then the E-point (ba) is determined such that $\Delta ab(ba)$ has outer angles $\alpha_1, \alpha_2, \alpha_3$ which are the angles between the edges of s as shown in Fig. 6.3. Note that the four points $a, s, b, (ba)$ are still cocircular.

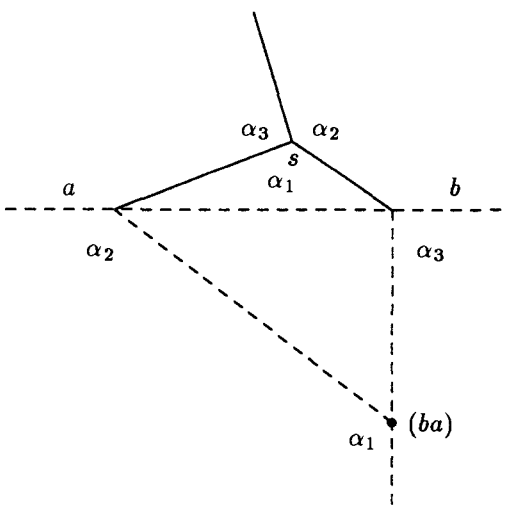


Figure 6.3: The generalized E-point

Clark [8] showed that the algebraic method given in Section 2.3 also works for the Gilbert model. van der Heyden [19] gave a heuristic to construct Gilbert-Steiner minimal trees which mimics Chang's heuristic for SMT (see Section 4.2). Similar heuristics for drainage networks and pipeline networks were given by Lee [20] and Bhaskaran and Salzborn [4]. In these two networks, c represents the pipe diameter and is continuous. A special complication for the pipeline model is that c_e depends on the length of e. If the sink (source) can be chosen optimally, Lin and Smith [21] gave an $O(n)$ procedure to find it. But the optimal sink can be a Steiner point.

When $f(c)$ is a linear function, Gilbert gave good bounds for the cost of a Gilbert-Steiner minimal trees. Werner [37] also studied this case for three terminals.

Next consider a model where edges are not capacitated but the cost of an edge is a nonlinear function of its length. Again, assume that the function f is nonnegative and subadditive so that a Steiner point has at least three edges. Soukup [31] found a way to upper bound the degree of a vertex. Call a differentiable function f (r,ϵ)-*increasing* in the interval $[0,p]$, for integer $r \geq 2$, $\cos(\pi/r+1) > \epsilon \geq 0$ and $p > 0$ if f is nonnegative increasing and

$$f'(x) \geq \frac{f'(0)}{2(\cos(\pi/(r+1)) - \epsilon)} \quad \text{for all} \quad x \in (0,p]$$

(the definition was also extended to nondifferentiable function). For example, every increasing convex function is $(3,\epsilon)$-increasing in the interval $[0,\text{inf})$ for any $\epsilon < (\sqrt{2}-1)/2$. Soukup proved:

Theorem 6.5 *Suppose that f is (r,ϵ)-increasing in $[0,p]$, where p is the diameter of the set N of terminals, and where $r \geq 3$ and $\epsilon > 0$ is an arbitrarily small number. Then a minimizing network exists which has maximum degree r. Furthermore, if f is $(r,0)$-increasing, then two edges meeting at a terminal must have angles at least $2\pi/(r+1)$.*

6.3 Terminal Clusters and New Terminals

Cockayne and Melzak [10] considered the model that each terminal is not a single point, but a compact set of points $N_1,\ldots,N_s$. They showed that Melzak's FST algorithm still works if the E-point for two terminal-sets A and B is interpreted as the compact set

$$(AB) = \{(a,b) : a \in A, b \in B\}$$

(Note that (AB) preserves the following properties of A and B: convexity, connectedness, being a smooth boundary of a region, being a simple polygon.) When only two terminal-sets are left, we connect them by a shortest line. However, the efficiency of the algorithm suffers on two accounts:

(i) Hwang's linear variant of the Melzak FST algorithm does not extend to the current case.

(ii) The complexity to construct (AB) and a shortest line connecting A and B depends on the properties of A and B.

A slight variation is when a terminal set is a finite set of points. Two terminal sets are considered connected if a point in one set is connected to a point in the other set. The Melzak FST algorithm applies just as in the compact set case.

In yet another model the locations of terminals are confined to regions, but not necessarily fixed. Chen [6] studied the case of three terminals a,b,x with a and b fixed and x confined to a straight line l. The problem involves the determination of

a possible Steiner point as well as the location of x. If a and b lie on opposite sides of l, then the line $[a, b]$ is clearly the SMT. So suppose a and b are on the same side of l with distance A and B, $A \leq B$, to l. Let $[a, o]$ and $[b, c]$ be perpendicular to l, and let C be the distance from o to c. Chen showed that:

condition	location of x	tree length
$\sqrt{3}(B-A) \geq C$	o	$A + \sqrt{C^2 + (B-A)^2}$
$\sqrt{3}(B+A) \geq C \geq \sqrt{3}(B-A)$	$\frac{C}{2} - \frac{\sqrt{3}}{2}(B-A)$ from o	$\frac{\sqrt{3}}{2}C + \frac{B+A}{2}$
$C \geq \sqrt{3}(B+A)$	$\frac{AC}{A+B}$ from o	$\sqrt{C^2 + (B+A)^2}$

Weng [36] considered the case that the third point lies on a continuously differentiable curve, and also the case that two terminals are confined to two lines. He used the hexagonal coordinate algorithm introduced in Section 2.3 to solve for the SMTs.

A related problem was studied by Trietsch [34] to augment an existing network by n new terminals. When $n = 2$, $C^2 \leq 2AB - A^2$ and the existing network is a line segment covering $[o, c]$, the problem is identical to the one studied by Chen. But, in general, the new terminals can be connected to different parts of the existing network. Trietsch gave a finite procedure to find the shortest augmenting network which essentially enumerates all possibilities and selects the shortest tree.

The concept of MST can also be extended to models discussed in this section. MSTs for terminals which are finite point-sets will play a crucial role in the construction of a heuristic whose performance ratio is provably better than the Steiner ratio (see Part IV, Section 1.4).

6.4 k-SMT

Recall from Section 4.9 that a shortest tree among all STs with at most k Steiner points is called a k-SMT. For $k < n - 2$ the maximum degree of an SMT can be more than three since one may not have the freedom to add an additional Steiner point within an angle of less than 120°, even though the addition will shorten the tree. This is also the case when each Steiner point costs something. Boyce [5] raised the question what is the maximum degree in a k-SMT for $k < n - 2$.

If a vertex has six or more edges, then there exist two edges with a subtending angle less than 60°, and the tree can be shortened by replacing one of the two edges. The only exception is when a vertex has six edges of equal lengths and the subtending angles are all 60°. But then none of its adjacent points can have more than four edges. By replacing an edge of the degree-6 point by an edge connecting its two adjacent points, another SMT with one fewer degree-6 points is obtained. Repeatedly doing so eventually yields an SMT with maximum degree 5.

No such simple argument is available to rid the degree-5 points from an SMT. Boyce conjectured, and Rubinstein, Thomas and Weng [25], using the variational approach (see Section 3.3), proved the following theorem.

Theorem 6.6 *The maximum degree of a Steiner point in a k-SMT for $k < n-2$ is four.*

By noting that the 1-SMT for four corners of a square consists of its two diagonals (whose intersection creates a degree-4 point), SMTs with degree-4 Steiner points do exist. Note that Theorem 6.6 does not apply to a degree-5 terminal whose existence is still an open problem.

Therefore it suffices to consider topologies in which Steiner points have either three or four edges. Clearly, a degree-4 Steiner point must lie at the intersection of the two diagonals of its four adjacent points (if they don't intersect, the corresponding degree-4 Steiner point does not exist) due to triangle inequality. So Melzak's FST algorithm can be easily modified to apply to the k-SMT case. This is very fortunate since Cockayne and Melzak [11] gave an example that the optimal location of a degree-5 point cannot be constructed by ruler and compass.

For constant k one can construct a k-SMT in $O(n^{4k+1} \log n)$ time since there are $\binom{n}{4} + \binom{n}{3}$ choices of sets of three and four terminals to add the first Steiner point, $\binom{n+1}{4} + \binom{n+1}{3}$ choices for the second, and so on. It takes constant time to find an SMT for three or four terminals; and then it takes $O(n \log n)$ time to find the MST for each $N \cup M$ where M is the chosen set of Steiner points.

Georgakopoulos and Papadimitriou [16] gave an $O(n^2)$ algorithm for 1-SMT by dividing the plane into $O(n^2)$ region such that the topology of the MST of $N \cup s$ depends only on the region of the Steiner point s, but not the exact location of s.

6.5 Obstacles

A shortest tree interconnecting a terminal set N in the Euclidean plane and whose edges avoid a set Ω of obstacles will be called an *obstacle-avoiding SMT* and denoted by $\mathrm{SMT}_\Omega(N)$. Applications of such trees range from the optimal design of oil and natural gas pipeline systems in regions with lakes, mountains and other natural phenomena, to the routing of mechanical and electrical systems around the architectural and structural elements within buildings [28].

If obstacles neither intersect nor touch each other, then the same reason that a Steiner point has exactly degree three in the regular ESP still prevails. Some other basic properties for routing SMTs around obstacles were given by Smith [26]. For easier analysis, obstacles are often assumed to be convex polygonal objects with a total of v extreme points (this is assumed throughout this section). Even so, exact algorithms have been proposed only for a single convex polygonal obstacle and a small number of terminals. Smith gave an $O(v \log v)$ time algorithm for three terminals and a single convex polygonal obstacle.

Winter and Smith [39] considered the same problem and used a computational geometry approach to cut down the time complexity to $O(v)$. If the obstacle is preprocessed in an appropriate way in $O(v)$ time, then $\mathrm{SMT}_\Omega(N)$ can be determined in $O(\log v)$ time for any set N of three terminals. The general idea behind the algorithm is as follows. Assume that the SMT for $N = \{p_i, p_j, p_k\}$ cuts across the obstacle. Consider an ordered pair (p_i, p_j) of the three terminals. Throw a

polygonal chain P_{ij} from p_i to p_j such that it embraces the obstacle and separates it from the third terminal p_k. Local properties of the SMT and of the obstacle permit the replacement of p_i and p_j by the extreme points on P_{ij}, one point at a time, until the SMT for the new set of points does not cut across the obstacle (see Fig. 6.4). Do this for all six ordered pairs of the three terminals (some of these six pairs can be ruled out right away). The shortest one among the six trees is an $\mathrm{SMT}_\Omega(N)$.

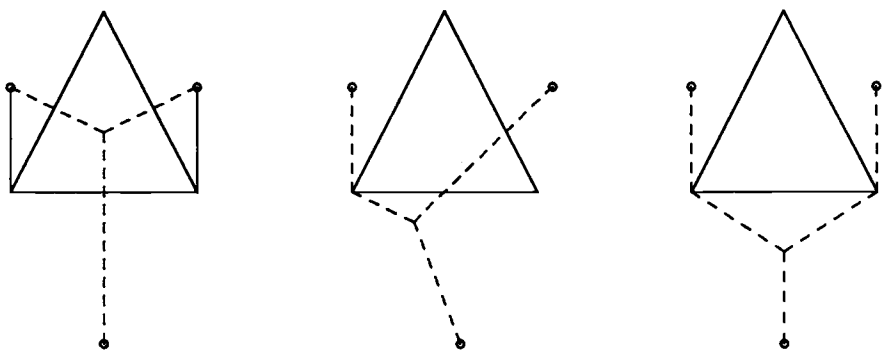


Figure 6.4: Moving around the obstacle

There is a straightforward $O(v^2)$ time algorithm for four terminals and a single convex polygonal obstacle. Back to the general n case, the ideas of many of the heuristics studied in Chapter 4 can also be used here. In particular, Armillotta and Mummolo [2] proposed an $O((n+v)^3)$ time heuristic by locally improving an obstacle avoiding MST. Smith [26,27] and Winter [38] considered heuristics by concatenating small obstacle-avoiding SMTs. However, there do not exist efficient obstacle-avoiding SMT algorithms when there is more than one obstacle. The only currently available method is by brute-force enumeration.

Provan [24] extended his ϵ-approximation scheme for $\mathrm{SMT}(N)$ (see Section 4.7) to the construction of $\mathrm{SMT}_\Omega(N)$. The main idea is to show that a *path-convex hull*, a minimum-perimeter region containing all terminals and avoiding all obstacles, is a Steiner hull. Provan gave an $O((n+v)^3)$ algorithm for path-convex hulls. The problem is then transformed to an ST problem on networks by introducing a lattice of nonterminals whose density depends on ϵ. In particular, if there exists a polynomial algorithm for the network problem, then the ϵ-approximation scheme is fully polynomial. One application of this result is a fully polynomial ϵ-approximation scheme for the case when all terminals except a constant number lie on the boundaries of obstacles.

Winter [38] introduced the notion of *Steiner visibility*. A point b is Steiner visible to another point a given the set Ω of obstacles if there exists two rays from a and b, unobstructed by any obstacles, meeting at a point s such that $\angle asb = 120°$ (asb is in the clockwise order). A Steiner visibility graph is a digraph with nodes $N \cup V$, V is the set of extreme points in Ω, and a link from a to b if and only if b

is Steiner visible from a. A plane sweeping algorithm similar to the algorithm by Alt and Welzl [1] for visibility graph can determine the Steiner visibility graph in $O(nv+v^2)$ time. Steiner visibility graphs can be used to quickly determine whether an FST for three nodes penetrate any of the obstacles. A generalized version can accommodate FSTs with more than three nodes.

References

[1] H. Alt and E. Welzl, Visibility graphs and obstacle avoiding shortest paths, *Z. Oper. Res.* **32** (1989) 145–164.

[2] A. Armillotta and G. Mummolo, A heuristic algorithm for the Steiner problem with obstacles, Technical Report, Dipt. di Pregettazione e Produzione Industriale, Univ. degli Studi di Bari, Italy (1988).

[3] J. Beardwood, J. H. Halton and J. M. Hammersley, The shortest path through many points, *Proc. Cambridge Philos. Soc.* **55** (1959) 299–327.

[4] S. Bhaskaran and F. J. M. Salzborn, Optimal design of gas pipeline networks, *J. Oper. Res. Soc.* **30** (1979) 1047–1060.

[5] W. M. Boyce, Can degree-five Steiner points reduce network costs for planar sets? Unpublished manuscript.

[6] G. X. Chen, The shortest path between two points with a (linear) constraint (in Chinese), *Knowledge and Appl. of Math.* **4** (1980) 1–8.

[7] F. R. K. Chung and E. N. Gilbert, Steiner trees for the regular simplex, *Bull. Inst. Math., Acad. Sin.* **4** (1976) 313–325.

[8] R. C. Clark, Communication networks, soap films and vectors, *Phys. Ed.* **16** (1981) 32–37.

[9] E. J. Cockayne, On the Steiner problem, *Canad. Math. Bull.* **10** (1967) 451–450.

[10] E. J. Cockayne and Z. A. Melzak, Steiner problems for set-terminals, *Q. Appl. Math.* **26** (1967) 213–218.

[11] E. J. Cockayne and Z. A. Melzak, Euclidean constructability in graph minimization problems, *Math. Mag.* **42** (1969) 206–208.

[12] J. Dolan, R. Weiss and J. M. Smith, Minimal length trees on the unit sphere, *Ann. Oper. Res.* **33** (1991) 503-535.

[13] D. Z. Du, On Steiner ratio conjectures, *Ann. Oper. Res.* **33** (1991) 437-449.

[14] D. Z. Du and F. K. Hwang, On a conjecture of Trietsch and Handler on the flow-dependent Steiner ratio, *Networks* **16** (1986) 47–50.

[15] L. Few, The shortest path and shortest road through n points, *Mathematika* **2** (1955) 141–144.

[16] G. Georgakopoulos and C. H. Papadimitriou, A 1-Steiner tree problem, *J. Algorithms* **8** (1987) 122–130.

[17] E. N. Gilbert, Minimum cost communication networks, *Bell System Tech. J.* **46** (1967) 2209–2227.

[18] E. N. Gilbert and H. O. Pollak, Steiner minimal trees, *SIAM J. Appl. Math.* **16** (1968) 1–29.

[19] W. P. A. van der Heyden, Some experiments with Steiner trees, ISNM 36 Birkhauser Verlag, Basel and Stuttgart (1977) 79–109.

[20] D. H. Lee, Low cost drainage and networks, *Networks* **6** (1976) 351–371.

[21] E. Lin and M. L. Smith, Capacitated Steiner network, problem and solution, unpublished manuscript.

[22] D. W. Litwhiler and A. A. Aly, Steiner's problem and Fagano's result on the sphere, *Math. Program.* **18** (1980) 286–290.

[23] S. Moran, On the length of optimal TSP circuits in sets of bounded diameter, *J. Comb. Theory, Ser. B* **37** (1984) 113–114.

[24] J. S. Provan, An approximation scheme for finding Steiner trees with obstacles, *SIAM J. Comput.* **17** (1988) 920–934.

[25] J. H. Rubinstein, D. A. Thomas and J. F. Wang, Degree five Steiner points cannot reduce network costs for planar sets, *Networks*, to appear.

[26] J. M. Smith, Steiner minimal trees with obstacles, Technical Report, Dept. of Ind. Eng. and Oper. Res., Univ. of Massachusetts (1982).

[27] J. M. Smith, Generalized Steiner network problems in engineering design, in J.S. Gero (ed.) *Design Optimization*, Academic Press (1985) 119-161.

[28] J. M. Smith and J. S. Liebman, Steiner trees, Steiner circuits, and the interference problem in building design, *Eng. Design* **4** (1979) 15–36.

[29] W. D. Smith, Studies in computational geometry motivated by mesh generation, Ph.D. Thesis, Princeton Univ. (1988).

[30] W. D. Smith, How to find Steiner minimal trees in Euclidean d-space, *Algorithmica* **7** (1992) 137-177.

[31] J. Soukup, On minimum cost networks with nonlinear cost, *SIAM J. Appl. Math.* **29** (1975) 571–581.

[32] J. M. Steele, Subadditive Euclidean functionals and nonlinear growth in geometric probability, *Ann. Probab.* **9** (1981) 365–376.

[33] E. A. Thompson, The method of minimum evolution, *Ann. Human Genetics* **36** (1973) 330-340.

[34] D. Trietsch, Augmenting Euclidean networks — the Steiner case, *SIAM J. Appl. Math.* **45** (1985) 855–860.

[35] D. Trietsch and G. Y. Handler, The Gilbert and Pollak conjecture - a generalization, *Networks* **15** (1983) 365–380.

[36] J. F. Weng, Generalized Steiner problem and hexagonal coordinate (in Chinese), *Acta Math. Appl. Sin.* **8** (1985) 383–397.

[37] C. Werner, The role of topology and geometry in optimal network design, *Papers of Regional Sci. Assoc.* **21** (1968) 173–189.

[38] P. Winter, Euclidean Steiner minimal tree with obstacles and Steiner visibility graphs, *Discrete Appl. Math.*, to appear.

[39] P. Winter and J. M. Smith, Steiner minimal trees for three points with one convex polygonal obstacle, *Ann. Oper. Res.* **33** (1991) 577-599.

This Page Intentionally Left Blank

Part II

Steiner Problem in Networks

This Page Intentionally Left Blank

Chapter 1

Introduction

The *Steiner problem in networks* is a combinatorial version of the Euclidean Steiner problem discussed in Part I. It can be formulated as follows:

- **GIVEN:** An undirected network $G = (V, E, c)$ where $c : E \rightarrow \mathcal{R}$ is an edge length function, and a non-empty set N, $N \subseteq V$, of *terminals*.
- **FIND:** A subnetwork $T_G(N)$ of G such that:
  - there is a path between every pair of terminals,
  - total length $|T_G(N)| = \sum_{e_l \in T_G(N)} c(e_l)$ is minimized.

The vertices in $V \setminus N$ are called *non-terminals*. Non-terminals that end up in $T_G(N)$ are called *Steiner vertices*.

If G is not connected and terminals appear in at least two components, then the Steiner problem has no solution. If G is not connected but all terminals appear in the same component, then the remaining components can be disregarded (with the exception of negative length edges that all must be in $T_G(N)$). It is assumed in the sequel that G is connected.

The subnetwork $T_G(N)$ is called a *Steiner minimal network for N in G*. [1] Contrary to the Euclidean version of the Steiner problem, Steiner vertices in the network version are selected from a finite set of non-terminal vertices.

If all edges in G have positive length, a Steiner minimal network $T_G(N)$ must be a tree. The problem is therefore often referred in the literature as the *Steiner tree problem*, and $T_G(N)$ is called a *Steiner minimal tree* for N in G. In particular, $T_G(V)$ denotes a *minimum spanning tree* for G.

It is assumed in the sequel that edge lengths are positive. However, it will be shown in Subsection 1.4.1 how an instance of the Steiner problem in a network with non-positive edge lengths can be transformed to an instance with positive edge lengths.

Terminals and non-terminals in the network shown in Fig. 1.1 are indicated by black and white circles respectively. The Steiner minimal tree is indicated by heavy line segments.

[1] $T_G(N)$ should in fact be referred to as a *Steiner minimum network*. However, for historical reasons, the less correct term widely used in the literature is preferred.

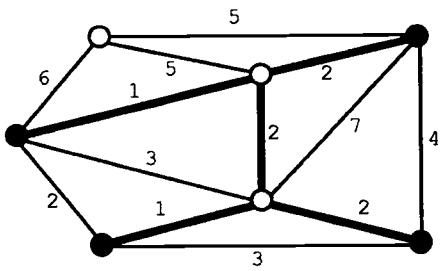


Figure 1.1: Steiner minimal tree

The Steiner tree problem in networks was originally formulated by Hakimi [7] and independently by Levin [14] in 1971. Since then, the problem received considerable attention in the literature. Several exact algorithms and heuristics have been suggested, implemented and compared. Surveys on the Steiner tree problem in networks have been given by Foulds and Rayward-Smith[4], Winter [24], Maculan [15], Korte, Prömel and Steger [11] , Voss [22] and Hwang and Richards [9].

In the remainder of this chapter some applications of the Steiner tree problem are pointed out. Basic definitions are given. Some special cases of the problem and various reformulations are discussed. The complexity of the problem is discussed in the final section of this chapter. Chapter 2 is concerned with various ways of reducing instances of the Steiner tree problem to smaller instances by identifying edges and non-terminals that either belong or do not belong to at least one Steiner minimal tree. Such reductions are of extreme importance in connection with the exponential exact algorithms for the Steiner tree problem discussed in Chapter 3. Heuristics for the Steiner tree problem are described in Chapter 4. Their worst-case time complexity and worst-case error ratio is discussed. There are certain classes of networks in which the Steiner tree problem can be solved optimally in polynomial time. They are discussed in Chapter 5. Finally, Chapter 6 gives an overview of problems in one way or another related to the Steiner tree problem in networks.

1.1 Applications

The Steiner tree problem in networks has obvious applications in the design of various communication, distribution and transportation systems.

An interesting real-life problem that can be formulated as the Steiner tree problem in networks arises in connection with the wire routing phase in physical VLSI design [13]. After the placement of components on a chip, sets of pins on the component boundaries are to be connected within the remaining free chip space. For each set of pins sharing the same electrical signal, a Steiner minimal tree for pins as terminals is sought. The underlying network is defined by the positions of pins and component boundaries.

Determining a customer's bill for renting a large communication network also

leads to the Steiner tree problem in networks [6]. The bill is not based on the actual circuits supplied. These circuits may be roundabouts and may change over time as other customers change their demands. Instead, the bill is obtained by applying some simple formula to an ideal network that would provide all the required facilities at minimum cost.

Many network design problems can be formulated as Steiner tree problems in undirected as well as directed networks. This is for example the case for the uncapacitated plant location problem [16,15]. A variety of network design problems can be viewed as generalizations of the Steiner tree problem [16].

1.2 Definitions

An *undirected graph* $G = (V, E)$ consists of a nonempty set V of v *vertices* and a set E, $E \subseteq V \times V$, of e *edges* connecting pairs of vertices. An *undirected network* $G = (V, E, c)$ has V and E as in undirected graphs. In addition, it has a *length function* $c : E \rightarrow \mathcal{R}$ associated with the edges. Graphs can be considered as networks with unit length edges. The reader is referred to Harary [8] for more on graph theory and graph-theoretical concepts defined below.

An edge e_l between a pair of vertices v_i and v_j is denoted by (v_i, v_j). The vertices v_i and v_j are the *end-vertices* of e_l. The number of edges incident with a vertex v_i in a network G is called the *degree* of v_i, denoted by $deg_G(v_i)$ or $deg(v_i)$ if there is no danger of confusion.

If an edge e_l connects a vertex v_i with itself, then e_l is called a *loop*. If two edges e_k and e_l connect the same pair of vertices, then e_k and e_l are said to be *parallel*. Unless explicitly stated, networks are assumed to be loopless and without parallel edges.

An alternating sequence of vertices and edges beginning and ending with vertices, in which each edge is preceded and followed by its end-vertices) is called a *walk*. If all vertices are distinct, then the walk is called a *path*. If all vertices apart from the first and last vertex are distinct, and the number of vertices is greater than 2, then the walk is called a *cycle*. A network with no cycles is called a *forest*.

A network is said to be *connected* if it has a path between every pair of vertices. Otherwise it is said to be *disconnected*. A connected forest is called a *tree*.

A network $H = (W, F, c_F)$ is called a *subnetwork* of $G = (V, E, c)$ if $W \subseteq V$, $F \subseteq E$, and $c_F(e_l) = c(e_l)$ for all $e_l \in F$. H is said to *span* $X \subseteq V$ if $X \subseteq W$. H is *induced* by a subset W of vertices in G if its edges are precisely all edges of G connecting vertices in W.

A disconnected network consists of two or more maximal (with respect to the number of vertices) connected induced subnetworks, called *components* of G. A *cut-vertex* of a network G is a vertex whose removal increases the number of components. A *bridge* is such an edge. A network G is *nonseparable* if it is connected and has no cut-vertices. A *block* of a network G is a maximal (w.r.t. the number of vertices) nonseparable subnetwork of G.

Let W denote a subset of vertices in a network $G = (V, E, c)$. Its *complementary* subset is $\bar{W} = V \setminus W$. A partition $\{W, \bar{W}\}$ of V, $\emptyset \subset W \subset V$, is called a *cut*. The

set of its *crossing edges* $\{(v_i, v_j) \in E \mid v_i \in W, v_j \in \bar{W}\}$ is called a *cut-set* between W and $\bar{W}$.

A *contraction* of a network G along an edge (v_i, v_j) results in a network $\bar{G}$ obtained in the following way (Fig. 1.2):

- the edge (v_i, v_j) is deleted,
- any edge (v_k, v_j) is replaced by an edge (v_k, v_i) (of the same length). The vertex v_j is deleted,
- if pairs of parallel edges occur as a result, only the shorter edge in each pair is retained.

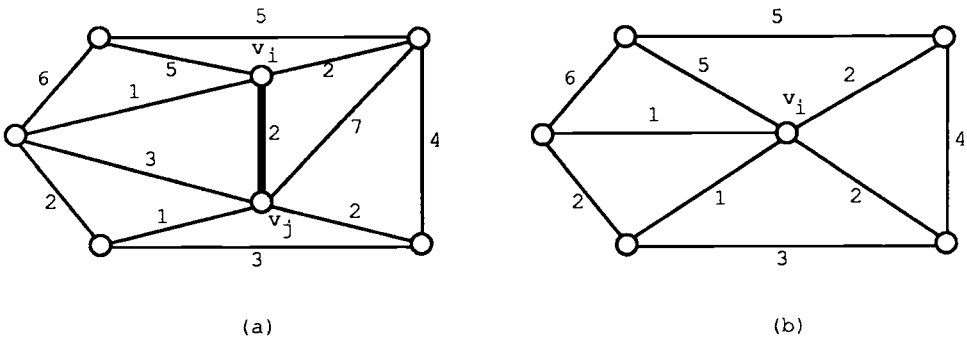


Figure 1.2: Contraction along (v_i, v_j)

A network with every pair of vertices being adjacent is said to be *complete*. A network in which the vertices can be partitioned into two sets W and $\bar{W}$ such that each edge is in the cut-set between W and $\bar{W}$ is said to be *bipartite*. A network that can be drawn in the plane such that (not necessarily straight) line segments corresponding to edges touch each other only at points corresponding to vertices is said to be *planar*.

For every edge $e_l = (v_i, v_j)$ in $G = (V, E, c)$, its *length* or *cost* $c(e_l)$ is also denoted by $c(v_i, v_j)$, $c_{v_i v_j}$ or c_{ij}. The length of any subnetwork H of G is defined by $\sum_{e_l \in H} c(e_l)$ and denoted by $|H|$ or $c(H)$.

A network G is said to satisfy the *triangle inequality* if G is complete, $c(e_l) \geq 0$ for all $e_l \in E$, and $c_{ij} \leq c_{ik} + c_{kj}$ for all $v_i, v_j, v_k \in V$.

A minimum length path $P_G(v_i, v_j)$ between two vertices v_i and v_j in G is also denoted by $P(v_i, v_j)$, $P_{v_i v_j}$ or P_{ij}. Its total length $|P_G(v_i, v_j)|$ is also denoted by $d_G(v_i, v_j)$, $d(v_i, v_j)$ or d_{ij}. A minimum length path between a subset W of V and a vertex $v_i \in V$ is denoted by $P_G(W, v_i)$. A complete network $D_G(W)$ with W, $W \subseteq V$, as its vertex set, and with $d_G(v_i, v_j)$ for each pair of vertices $v_i, v_j \in W$ as the length of the edge (v_i, v_j), is called a *distance network* of W in G. Clearly, $D_G(W)$ satisfies the triangle inequality. If $W = V$ and G is obvious from the context, $D_G(V)$ is denoted by D.

Most of the definitions for undirected networks carry over in natural way to *directed networks* where edges, referred to as *arcs*, are *ordered* pairs of vertices. An

arc from a vertex v_i to a vertex v_j is denoted by $[v_i, v_j]$. Note that $[v_i, v_j] \neq [v_j, v_i]$. Let $\vec{G}= (V, A, c)$ denote a directed network. An *arborescence* rooted at some vertex $v_k \in V$ is a subnetwork of $\vec{G}$ with no cycles and such that each of its vertices can be reached from v_k by a directed path.

The cardinality of any set S is denoted by $|S|$. Although the same notation is used to indicate lengths of networks, a particular meaning will be obvious from the context. Union of a set S and one-element set $\{t\}$ is denoted by $S \cup t$ rather than by formally correct $S \cup \{t\}$. Similarly, $S - t$ is used to denote the set difference $S \setminus \{t\}$.

1.3 Trivial Special Cases

Let $G = (V, E, c)$ be an undirected, connected network, and let $N = \{v_1, v_2, ..., v_n\}$ be a set of terminals in G. Whenever it is necessary to emphasize that $v_1, v_2, ..., v_n$ are terminals, symbols $z_1, z_2, ..., z_n$ are used. Under the assumption that all edges have positive length, there are the following special cases:

- If $n = 1$, then the unique Steiner minimal tree consists of the terminal alone.
- If $n = 2$, then the Steiner tree problem reduces to the well-known shortest path problem. Several polynomial time algorithms for this problem are known [1,3,21].
- If $n = v$, then the Steiner tree problem reduces to the well-known minimum spanning tree problem. Several polynomial time algorithms for this problem are known [12,19,21].

Suppose that G is separable. A block B_l of G is said to be *intermediate* if there is a pair of terminals z_i and z_j in G such that every path between z_i and z_j contains at least one edge of B_l. Let $B_1, B_2, ..., B_k$ denote the intermediate blocks of G. Let N_l denote the terminals in B_l, $1 \leq l \leq k$. Let C_l denote the cut-vertices of G in B_l, $1 \leq l \leq k$. Then $T_G(N) = \bigcup_{l=1}^{k} T_{B_l}(N_l \cup C_l)$. Hence, the instance of the Steiner tree problem for N in G reduces to k smaller Steiner tree problem instances.

A linear-time algorithm based on the depth-first search that finds all blocks in a network is available [20]. Pruning of non-intermediate blocks can also be done in linear time. Although the applicability of block decompositions is rather limited in larger, multiconnected networks, reductions described in Chapter 2 may result in smaller networks where cut-vertices do appear.

1.4 Problem Reformulations

In this section some ways of reformulating the Steiner tree problem are discussed.

1.4.1 Positive Length Edges

Every negative length edge must belong to every $T_G(N)$. Furthermore, there is an optimal solution containing all zero-length edges. Consequently, optimal solutions for problem instances with non-positive edge lengths are not necessarily connected (but all terminals will be in the same component) and can contain cycles.

A problem instance with arbitrary edge lengths can be transformed into a problem instance with positive edge lengths by contraction along non-positive-length edges. Given a Steiner minimal tree for the reduced problem instance, a Steiner minimal network for the original problem instance is obtained by the reintroduction of the contracted edges. Therefore positive edge lengths are assumed in the sequel.

1.4.2 Complete Networks

Let $G = (V, E, c)$ be a connected network. Consider a complete network G^* obtained by adding edges of length greater than $|G|$ between every pair of non-adjacent vertices. The added edges are assumed to have *infinite* length. Solving an instance of the Steiner tree problem for N in G is clearly equivalent to solving an instance of the Steiner tree problem for N in G^*. Although this reformulation is of no practical significance, it simplifies some of the proofs that follow in this and subsequent chapters.

1.4.3 Distance Networks

Solving an instance of the Steiner tree problem for N in G is equivalent to solving an instance of the Steiner tree problem for N in the distance network $D = D_G(V)$. This is a direct consequence of the following lemma.

Lemma 1.1 $|T_G(N)| = |T_D(N)|$.

Proof: Suppose that a Steiner minimal tree $T_G(N)$ is given. $|T_G(N)| = |T_{G^*}(N)|$. Furthermore, every edge in D is not longer than the corresponding edge in G^*. Thus, $|T_G(N)| = |T_{G^*}(N)| \geq |T_D(N)|$.

On the other hand, suppose that a Steiner minimal tree $T_D(N)$ is given. The network obtained by replacing each edge of $T_D(N)$ by the corresponding shortest path yields a connected subnetwork of G that spans all terminals. Hence, $|T_D(N)| \geq |T_G(N)|$. In conclusion, $|T_G(N)| = |T_D(N)|$. □

Given a Steiner minimal tree $T_D(N)$ for N in D, a Steiner minimal tree $T_G(N)$ for N in G is obtained by replacing edges in $T_D(N)$ by the corresponding shortest paths. Note in particular that edge lengths in D satisfy the triangle inequality.

Solving the Steiner tree problem for N in D rather than in G has both its advantages and disadvantages. One of the advantages is that any Steiner vertex in at least one $T_D(N)$ must have degree greater than 2. Consequently, such $T_D(N)$ can contain at most $n-2$ Steiner vertices. The proof of these facts will be given in Section 3.1. If G is dense and has few terminals, solving the Steiner tree problem for N in D by the spanning tree enumeration algorithm (Section 3.1) can be much more efficient than solving the Steiner tree problem for N in G. On the other hand, if G is sparse and has many terminals, structural properties of G will be lost when the problem is transformed.

1.4.4 Steiner Arborescence Problem

Mathematical programming formulations of the Steiner tree problem in undirected networks that will be given in Chapter 3 are for a more general *Steiner arborescence problem* in directed networks. This section first gives the formulation of the Steiner arborescence problem. It is then shown how every instance of the Steiner tree problem in an undirected network can be transformed into an instance of the Steiner arborescence problem in a directed network.

Let $\vec{G}= (V, A, c)$ be a directed network. Let N, $N \subseteq V$, be a set of terminals. Finally, suppose that one of the terminals (z_1, say) is designated as the *root*. Let $N_1 = N - z_1$. The *Steiner arborescence problem* for N in the directed network $\vec{G}$ is to find a least length arborescence $T_{\vec{G}}(N)$ rooted at z_1 and spanning all terminals. In other words, one looks for a set of paths from z_1 to all terminals in N_1 such that the total length of the arcs in these paths is as small as possible.

Any instance of the Steiner tree problem in an undirected network G can be transformed into an instance of the Steiner arborescence problem in a directed network $\vec{G}$. Replace every undirected edge (v_i, v_j) in G by two directed arcs $[v_i, v_j]$ and $[v_j, v_i]$. Associate the length of the edge (v_i, v_j) with the two opposite arcs $[v_i, v_j]$ and $[v_j, v_i]$. Select any terminal as the root z_1. Then the solution $T_{\vec{G}}(N)$ to the Steiner arborescence problem for N in $\vec{G}= (V, A, c)$ gives the solution $T_G(N)$ of the undirected Steiner tree problem for N in G: $T_G(N)$ consists of those edges whose end-vertices are connected by arcs in $T_{\vec{G}}(N)$.

1.4.5 Degree-Constrained Formulation

The Steiner tree problem for a set of terminals N in an undirected network $G = (V, E, c)$ can be formulated as a problem of determining least length tree spanning *all* vertices of a slightly larger network subject to simple degree constraints. The significance of this formulation stems from the fact that it directly leads to a couple of exact algorithms that will be discussed in Chapter 3.

Add a new vertex v_0 to G. Connect it to one, arbitrarily chosen, terminal z_1 and to all non-terminals by zero-length edges. Let $\bar{G}$ denote the new network (Fig. 1.3). Consider a minimum length tree $\bar{T}_{\bar{G}}(V \cup v_0)$ spanning $\bar{G}$ such that any non-terminal adjacent to v_0 in $\bar{T}_{\bar{G}}(V \cup v_0)$ has degree 1. The subtree $\bar{T}_{\bar{G}}(N)$ obtained from $\bar{T}_{\bar{G}}(V \cup v_0)$ when v_0 and all degree 1 non-terminals are removed, spans all terminals. It can be easily verified that $\bar{T}_{\bar{G}}(N)$ is a Steiner minimal tree for N in G.

The addition of a new vertex v_0 is not strictly necessary. Zero-length edges from an arbitrarily chosen terminal could be used. However, this would cause some complications due to the presence of parallel edges.

A similar reformulation is possible in connection with the Steiner arborescence problem. The only difference is that instead of edges from the vertex v_0, arcs directed from v_0 are added.

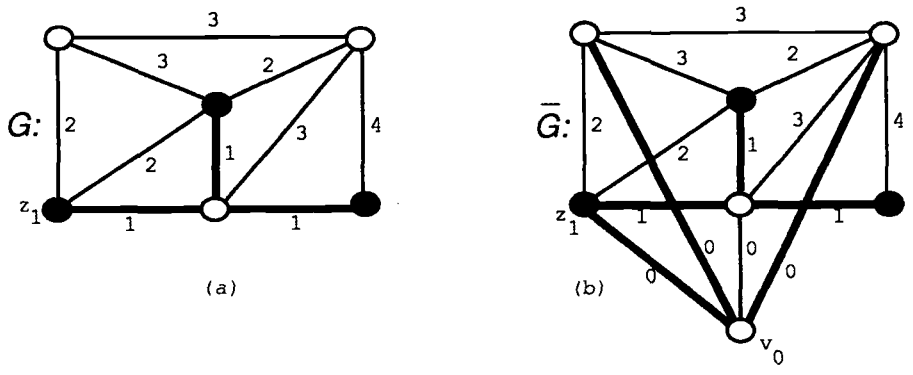


Figure 1.3: Networks G and $\bar{G}$

1.5 Complexity

The Steiner tree problem is an NP-hard optimization problem. Karp [10] stated (without proving) that the problem is NP-hard even if G is a bipartite network. In fact this is the case even if G is a bipartite graph with no edge joining a pair of terminals or a pair of non-terminals. In order to prove this result, consider the following Steiner tree decision problem:

- **GIVEN:** A bipartite graph $G = (V, E)$, a subset N of terminals, and an integer B. Assume furthermore that no pair of terminals or non-terminals is adjacent in G.
- **DECIDE:** Is there a tree T in G that spans all terminals and has at most B edges?

Theorem 1.1 *The Steiner tree decision problem is NP-complete.*

Proof: The Steiner tree decision problem in bipartite graphs is obviously in the class NP. In order to prove the completeness, an instance of the exact cover by 3-sets problem is transformed in polynomial time to an instance of the Steiner tree decision problem in a bipartite graph with no pair of terminals or non-terminals being adjacent.

The exact cover by 3-sets problem can be formulated as follows.

- **GIVEN:** A set $X = \{x_1, x_2, ..., x_{3p}\}$ and a family of subsets $C = \{C_1, C_2, ..., C_q\}$ such that $C_i \subseteq X$ and $|C_i| = 3$ for all $i = 1, 2, ..., q$.
- **DECIDE:** Is there an exact cover of X by subsets from C (i.e., is it possible to select some mutually disjoint subsets from C such that their union is X)?

Construct a problem instance of the Steiner tree decision problem in a graph $G = (V, E)$ as follows (Fig. 1.4).

- $V = v_0 \cup C \cup X$
- $E = \{(v_0, C_i) | 1 \leq i \leq q\} \cup \{(C_i, x_j) | x_j \in C_i, 1 \leq i \leq q, 1 \leq j \leq 3p\}$
- $N = v_0 \cup X$
- $B = 4p$

It is obvious that a Steiner minimal tree for N in $G = (V, E)$ of length B exists if C contains an exact cover for X. Suppose that C does not contain an exact cover of X. Each terminal in X must be incident with exactly one edge in $T_G(N)$. Consequently, $T_G(N)$ must contain at least $p + 1$ non-terminals in C. It follows that $T_G(N)$ must contain at least $4p + 2$ vertices. Since the number of edges in any tree is one less than the number of its vertices, $|T_G(N)| \geq 4p + 1$. □

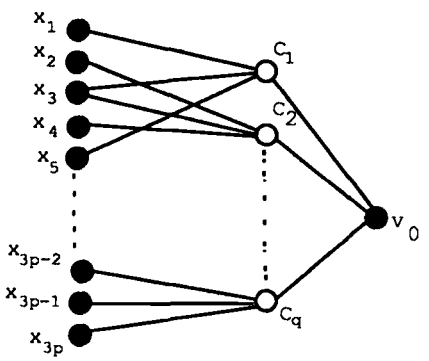


Figure 1.4: Transformed instance

An alternative proof of Theorem 1.1 was given by Plesník [18] where an instance of the vertex cover problem is transformed in polynomial time to an instance of the Steiner tree problem in a bipartite graph.

The Steiner tree problem remains NP-hard even if G is

- a chordal graph [23],
- a strongly chordal network [23],
- a chordal bipartite graph [17],
- a split graph [23],
- a planar network [5],
- a complete network with edge lengths either 1 or 2 [2].

As already indicated, the Steiner tree problem is NP-hard for planar *networks*. However, the complexity of the problem in planar *graphs* remains to be settled.

References

[1] R. E. Bellman, On a routing problem, *Q. Appl. Math.* **16** (1958) 87-90.

[2] M. Bern and P. Plassmann, The Steiner problem with edge lengths 1 and 2, *Inf. Process. Lett.* **32** (1989) 171-176.

[3] E. W. Dijkstra, A note on two problems in connection with graphs, *Numer. Math.* **1** (1959) 269-271.

[4] L. R. Foulds and V. J. Rayward-Smith, Steiner problems in graphs: Algorithms and applications, *Eng. Optimization* **7** (1983) 7-16.

[5] M. R. Garey and D. S. Johnson, The rectilinear Steiner tree problem is NP-complete, *SIAM J. Appl. Math.* **32** (1977) 826-834.

[6] E. N. Gilbert, A solvable routing problem, *Networks* **19** (1989) 587-594.

[7] S. L. Hakimi, Steiner's problem in graphs and its implications, *Networks* **1** (1971) 113-133.

[8] F. Harary, *Graph Theory*, Addison-Wesley, Reading, MA (1969).

[9] F. K. Hwang and D. S. Richards, Steiner tree problems *Networks* **22** (1992) 55-89.

[10] R. M. Karp, Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher (eds.) *Complexity of Computer Computations*, Plenum Press, New York, NY (1972) 85-103.

[11] B. Korte, H. J. Prömel and A. Steger, Steiner trees in VLSI-layout, in B. Korte, L. Lovasz, H. J. Prömel and A. Schrijver (eds.) *Paths, Flows, and VLSI-Layout*, Springer-Verlag, Berlin (1990) 185-214.

[12] J. B. Kruskal, On the shortest spanning subtree of a graph and the travelling salesman problem, *Proc. Am. Math. Soc.* **7** (1956) 48-50.

[13] T. Lengauer, *Combinatorial Algorithms for Integrated Circuit Layout*, John Wiley & Sons, Chichester, England (1990).

[14] A. Y. Levin, Algorithm for shortest connection of a group of graph vertices, *Sov. Math. Dokl.* **12** (1971) 1477-1481.

[15] N. Maculan, The Steiner problem in graphs, *Ann. Discrete Math.* **31** (1987) 185-212.

[16] T. L. Magnanti and R. T. Wong, Network design and transportation planning: Models and algorithms, *Transp. Sci.* **18** (1984) 1-55.

[17] H. Müller and A. Brandstädt, The NP-completeness of Steiner tree and dominating set for chordal bipartite graphs, *Theor. Comput. Sci.* **53** (1987) 257-265.

[18] J. Plesník, *Graph Algorithms*, Veda, Bratislava (1983) (in Slovak).

[19] R. C. Prim, Shortest connection networks and some generalizations, *Bell System Tech. J.* **36** (1957) 1389-1401.

[20] R. E. Tarjan, Depth-first search and linear graph algorithms, *SIAM J. Comput.* **1** (1972) 146-159.

[21] R. E. Tarjan, *Data Structures and Network Algorithms*, SIAM (1983).

[22] S. Voss, *Steiner-Probleme in Graphen*, Mathematical Systems in Economics **120**, Anton Hein, Frankfurt-am-Main (1990).

[23] K. White, M. Farber and W. R. Pulleyblank, Steiner trees, connected domination and strongly chordal graphs, *Networks* **15** (1985) 109-124.

[24] P. Winter, Steiner problem in networks: A survey, *Networks* **17** (1987) 129-167.

Chapter 2

Reductions

A particular instance of the Steiner tree problem often can be reduced to a smaller one by examining local properties of the network G. Reductions fall into two main categories:

- identification of edges or non-terminals that do not belong to at least one Steiner minimal tree.
- identification of edges or non-terminals that belong to at least one Steiner minimal tree.

It is assumed that the reductions are applied one at a time. When an edge (v_i, v_j) belonging to a Steiner minimal tree is identified, G is reduced to a smaller network by the contraction along (v_i, v_j). The vertex to which (v_i, v_j) is contracted is then regarded as a terminal. Deletions of edges and non-terminals are straightforward.

Reductions decrease the cardinality of the vertex and edge sets. They are important since the performance of exact algorithms and heuristics for the Steiner tree problem is closely related to these parameters. Furthermore, heuristics may produce better solutions in the reduced networks. Reductions can also be applied at intermediate stages of any branch-and-bound algorithm.

Computational experiments indicate that a large number of problem instances can be solved by reductions alone. The remaining problem instances are reduced substantially (on average to one fourth of the original sizes).

Tests identifying edges or non-terminals not belonging to at least one Steiner minimal tree are discussed in Section 2.1. Inclusion tests for edges or non-terminals are covered in Section 2.2. The important issue of how to integrate exclusion and inclusion tests is addressed in Section 2.3. Effectiveness of reductions is discussed in Section 2.4.

Reductions of instances of the Steiner arborescence problem in directed networks are of similar nature as the reductions described below. The reader is referred to Voss [12] for a review.

2.1 Exclusion Tests

This section discusses various tests that can be used to identify edges and non-terminals not belonging to at least one Steiner minimal tree. Some very simple tests are given first. More complicated tests described next in fact generalize the simpler ones. The reason for including simpler tests is partly historical and partly because they often (but not always) are faster to check than their generalizations.

2.1.1 Non-Terminals of Degree 1 (NTD1)

Suppose that a network G contains a non-terminal v_k of degree 1. Let (v_k, v_i) be the edge incident with v_k. This edge cannot be in any Steiner minimal tree. If it did, its deletion would result in two components, one containing v_k alone, the other containing all terminals. Since $c_{v_k v_i} > 0$, the component spanning all terminals has a smaller total length than any Steiner minimal tree, a contradiction. Hence v_k and its incident edge (v_k, v_i) can be removed from G.

2.1.2 Non-Terminals of Degree 2 (NTD2)

Suppose that a network G contains a non-terminal v_k of degree 2. Let (v_i, v_k) and (v_k, v_j) denote the two edges incident with v_k. If $c_{v_i v_k} + c_{v_k v_j} \geq c_{v_i v_j}$, then there is at least one Steiner minimal tree not containing v_k. Hence v_k (and its two incident edges) can be deleted from G. If $c_{v_i v_k} + c_{v_k v_j} < c_{v_i v_j}$, then the edge (v_i, v_j) cannot belong to any Steiner minimal tree and can be deleted from G. Furthermore, the edges (v_i, v_k) and (v_k, v_j) can be replaced by a new edge (v_i, v_j) of length $c_{v_i v_k} + c_{v_k v_j}$, and the non-terminal v_k can be deleted from G (Fig. 2.1).

Figure 2.1: Removal of a non-terminal of degree 2

Suppose that the added edge (v_i, v_j) is eventually found to belong to a Steiner minimal tree for the reduced problem instance. In order to obtain a Steiner minimal tree for the original problem instance, the edge (v_i, v_j) has to be replaced by the original two edges (v_i, v_k) and (v_k, v_j) whereby the non-terminal v_k is reintroduced.

2.1.3 Non-Terminals of Higher Degree (NTDk)

Let v_i and v_j denote two vertices in a network G. Any path P from v_i to v_j breaks down into one or more *elementary paths* between v_i, successive terminals and v_j. A *Steiner distance* between v_i and v_j along P is the length of the longest elementary path in P. The *bottleneck Steiner distance* $b_{v_iv_j}$ between v_i and v_j is the minimum Steiner distance between v_i and v_j taken over all paths from v_i to v_j in G. Furthermore, the *restricted bottleneck Steiner distance* $\bar{b}_{v_iv_j}$ between v_i and v_j is the bottleneck Steiner distance between v_i and v_j in $G-(v_i,v_j)$.

Every path between v_i and v_j in G contains at least one elementary path of length at least $b_{v_iv_j}$. Any path between v_i and v_j in G containing an elementary subpath of length $b_{v_iv_j}$ is called a *bottleneck Steiner path* between v_i and v_j. It will be shown in Subsection 2.3.1 that bottleneck Steiner distances between all pairs of vertices in G can be determined in $O(v^2n)$ time.

Let Γ_{v_k} denote the vertices of G adjacent to a non-terminal v_k. Let B_{v_k} denote a complete network with Γ_{v_k} as its vertex set (Fig. 2.2). The length of an edge (v_i,v_j) in B_{v_k} is defined as the bottleneck Steiner distance $b_{v_iv_j}$ between v_i and v_j in G. Let B'_{v_k} denote a subnetwork of B_{v_k} induced by a subset Γ'_{v_k} of Γ_{v_k}. Finally, let T'_{v_k} denote a minimum spanning tree for Γ'_{v_k} in B'_{v_k}, and let C'_{v_k} be the set of edges in G connecting v_k with the vertices in Γ'_{v_k}.

Lemma 2.1 ([6]) *If*

$$|T'_{v_k}| \leq |C'_{v_k}|$$

for every subset Γ'_{v_k} *of* Γ_{v_k}, $|\Gamma'_{v_k}| \geq 3$, *then the non-terminal* v_k *has degree at most 2 in at least one Steiner minimal tree.*

Proof: If $deg_G(v_k) \leq 2$, then the lemma clearly holds. It will be therefore in the following assumed that $deg_G(v_k) > 2$.

Suppose that v_k has degree at least 3 in every Steiner minimal tree for N in G. Consider a Steiner minimal tree $T_G(N)$ for N in G where v_k has lowest possible degree l, $l \geq 3$. Let Γ'_{v_k} denote the set of vertices adjacent to v_k in $T_G(N)$. Remove v_k from $T_G(N)$. It then breaks into l subtrees $T_{v_1}, T_{v_2}, ..., T_{v_l}$, each containing one of the vertices previously adjacent to v_k.

Pick a shortest edge (v_i,v_j) in T'_{v_k}. Consider a bottleneck Steiner path $P_{v_iv_j}$ in G (in Fig. 2.2 $v_i = v_1$, $v_j = v_3$ and $P_{v_1v_3}$ is indicated by the shaded region). Traverse $P_{v_iv_j}$ from v_i toward v_j. Whenever a path $P_{v'_pv'_q}$ between vertices in two different trees T_{v_p} and T_{v_q} is encountered, T_{v_p} and T_{v_q} are interconnected by it. This interconnecting path $P_{v'_pv'_q}$ is a subpath of one of the elementary paths in $P_{v_iv_j}$. Consequently, $|P_{v'_pv'_q}| \leq b_{v_iv_j}$.

The traversal of $P_{v_iv_j}$ can interconnect several subtrees in $T_{v_1}, T_{v_2}, ..., T_{v_l}$ (e.g., $P_{v_1v_3}$ in Fig. 2.2 goes through T_{v_2}). If there is more than one subtree left when v_j is reached during the traversal of $P_{v_iv_j}$, the second shortest edge of T'_{v_k} and the corresponding bottleneck Steiner path in G are processed in the same manner. This continues until all subtrees $T_{v_1}, T_{v_2}, ..., T_{v_l}$ become interconnected.

Exactly $l-1$ subpaths must be added to $T_G(N)-v_k$ before it becomes connected. Their total length L, is at most $|T'_{v_k}|$. Since $T_G(N)$ is a Steiner minimal tree, L

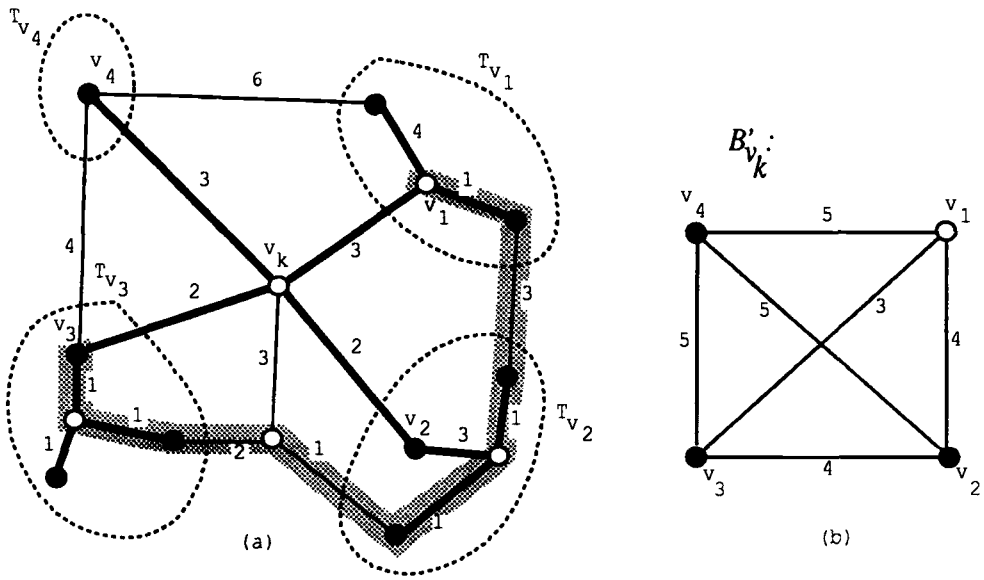


Figure 2.2: G, $T_G(N)$ and B'_{v_k}

must be at least $|C'_{v_k}|$. Hence, $L \leq |T'_{v_k}| \leq |C'_{v_k}| \leq L$. The tree obtained from $T_G(N) - v_k$ by adding the edges on the $l-1$ subpaths yields another Steiner minimal tree $T'_G(N)$.

Suppose that v_k has degree l in $T'_G(N)$. The length of l paths from v_k to $T_{v_1}, T_{v_2}, ..., T_{v_l}$ in $T'_G(N)$ is $|C'_{v_k}|$. Furthermore, $|C'_{v_k}| < L \leq |T'_{v_k}|$, a contradiction. Hence, $T'_G(N)$ is a Steiner minimal tree where v_k has degree less than l. This is in contradiction with the manner in which $T_G(N)$ was chosen. □

If the conditions of Lemma 2.1 are satisfied, then the non-terminal v_k appears in at least one Steiner minimal tree with degree at most 2. The original instance of the Steiner tree problem can be reduced in the following way. The non-terminal v_k is removed together with its all incident edges. The edges (v_i, v_j), $v_i, v_j \in \Gamma_{v_k}$, of length $c_{v_i v_k} + c_{v_k v_j}$ are added to G. If parallel edges arise as a result, only shorter ones are retained (Fig. 2.3). Once the smaller instance of the Steiner tree problem is solved, new edges appearing in the solution are replaced by the corresponding pairs of edges.

It can be easily verified that the NTDk-test is a generalization of the NTD1- and NTD2-tests.

The number of minimum spanning trees constructed in connection with the NTDk-test grows exponentially with $deg_G(v_k)$. Consequently, this test should be applied to non-terminals of low degree (typically not more than 4).

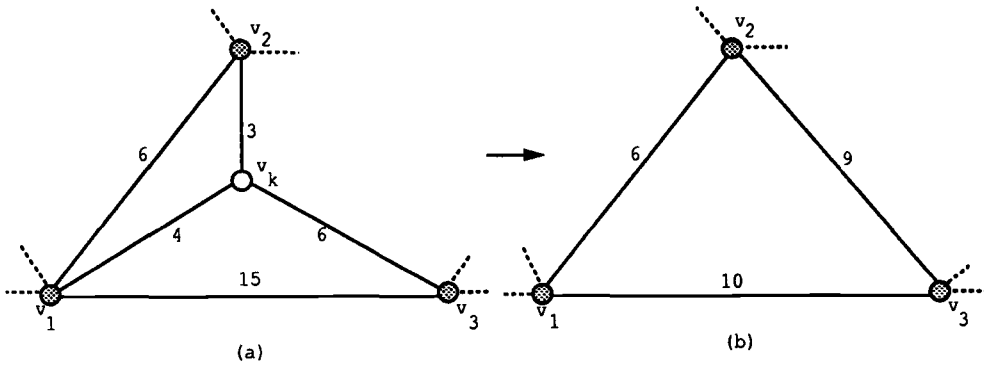


Figure 2.3: Removal of a non-terminal of degree 3

2.1.4 Long Edges (LE)

This test permits a deletion of long edges. The simple proof is omitted as the LE-test will be subsequently generalized.

Lemma 2.2 ([2]) *Any edge* (v_i, v_j) *with*

$$c_{v_i v_j} > d_{v_i v_j}$$

can be removed from G.

An application of the LE-test is shown in Fig. 2.4a where the edge indicated by the broken line segment can be removed. The LE-test can be generalized to cover the case when $c_{v_i v_j} = d_{v_i v_j}$ provided that there is a shortest path from v_i to v_j other than the shortest path consisting of the edge (v_i, v_j) alone.

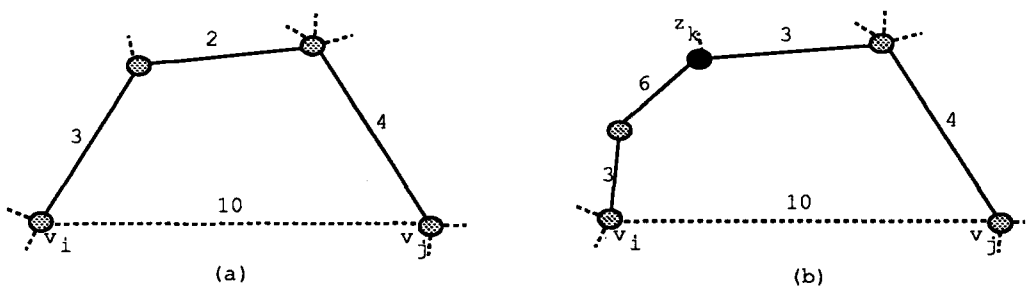


Figure 2.4: Removal of long edges

2.1.5 Paths with One Terminal (PT1)

This test is closely related to the LE-test. However, unlike the LE-test, it uses some information about the location of terminals in the network. Again, the simple proof

is omitted as the PT1-test will be subsequently generalized.

Lemma 2.3 ([6]) *If G contains an edge (v_i, v_j) and there is a terminal z_k such that*

$$c_{v_i v_j} > \max\{d_{v_i z_k}, d_{v_j z_k}\}$$

then (v_i, v_j) can be removed from G.

An application of the PT1-test is shown in Fig. 2.4b where the edge indicated by the broken line segment can be removed. Contrary to the LE-test, the PT1-test can be successful in networks satisfying the triangle inequality and in Euclidean networks in particular.

2.1.6 Minimum Spanning Trees (MST)

This test was originally given by Balakrishnan and Patel [1] in terms of three related tests. Only a unified version is presented here. Assume that G is a complete network. This is not a significant restriction as any instance of the Steiner tree problem can be transformed into an equivalent problem instance on a complete network by adding edges of "infinite" length (Subsection 1.4.2).

Lemma 2.4 ([7]) *Any edge (v_i, v_j) not in a minimum spanning tree for the subnetwork of G induced by $N \cup \{v_i, v_j\}$ can be removed from G.*

The proof is omitted as a more general test will be given below. An application of the MST-test to all edges in G is shown in Fig. 2.5. Not shown edges have "infinite" length. Edges that can be deleted by the MST-test are indicated by broken line segments.

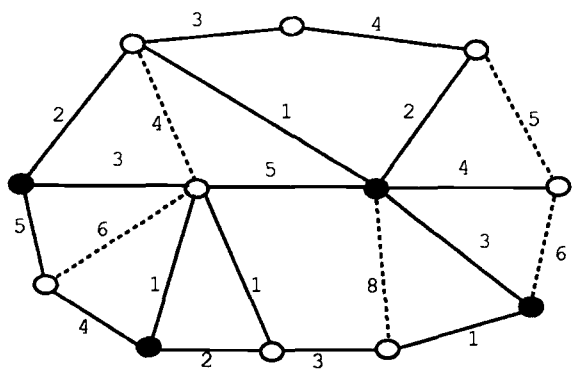


Figure 2.5: Removal of long edges based on the MST-test

2.1.7 Paths with Many Terminals (PTm)

This test is a generalization of both the LE- and PT1-tests. It can also be shown to be a generalization of the MST-test [7].

Lemma 2.5 ([6]) *Any edge* (v_i, v_j) *with*

$$c_{v_i v_j} > b_{v_i v_j}$$

can be removed from G.

Proof: Suppose that a Steiner minimal tree $T_G(N)$ contains an edge (v_i, v_j) with $c_{v_i v_j} > b_{v_i v_j}$. $T_G(N) - (v_i, v_j)$ consists of two components: C_{v_i} containing v_i and C_{v_j} containing v_j. Consider a bottleneck Steiner path $P_{v_i v_j}$ in G. Let P be the shortest subpath on $P_{v_i v_j}$ which has one end-vertex in C_{v_i}, and the other end-vertex in C_{v_j}. P always exists and is completely within one of the elementary paths of $P_{v_i v_j}$. Hence $|P| \leq b_{v_i v_j} < c_{v_i v_j}$. Reconnecting C_{v_i} and C_{v_j} by P yields a tree spanning all terminals and being shorter than $T_G(N)$, a contradiction. □

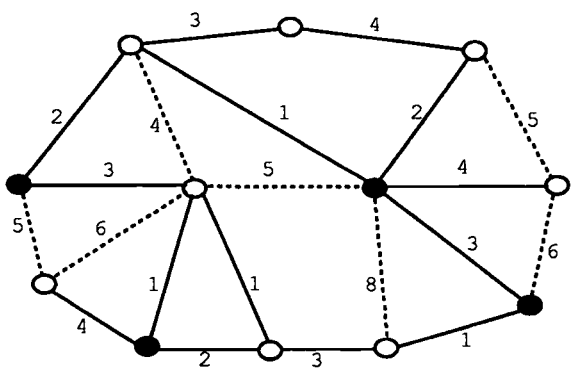


Figure 2.6: Removal of long edges based on the PTm-test

An application of the PTm-test to all edges of G is shown in Fig. 2.6. Edges that can be deleted by the PTm-test are indicated by broken line segments.

The PTm-test is also applicable when $c_{v_i v_j} = b_{v_i v_j} = \bar{b}_{v_i v_j}$. However, complications can arise if this extended PTm-test is applied more than once without the recalculation of the restricted bottleneck Steiner distances. The restricted bottleneck Steiner distances $\bar{b}_{z_1 z_2}, \bar{b}_{z_1 z_3}, \bar{b}_{z_2 z_3}$ in Fig. 2.7 are all equal to 4. Any of the edges (z_1, z_2), (z_1, z_3), (z_2, z_3) can be deleted from G. Suppose that (z_1, z_3) is deleted. This causes both $\bar{b}_{z_1 z_2}$ and $\bar{b}_{z_2 z_3}$ to increase from 4 to 5 (assuming that all not shown edges of G have length at least 5). Consequently, the removal of for example (z_1, z_2) is now not allowed.

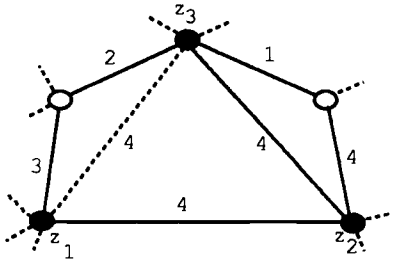


Figure 2.7: Improper removal of two edges

2.1.8 Reachability (R)

Suppose that a tree $T_G^*(N)$ spanning the terminals is available. $T_G^*(N)$ can for example be obtained by one of the heuristics for the Steiner tree problem discussed in Chapter 4. Let v_l denote any non-terminal of G not in $T_G^*(N)$. Let z_i and z_j denote terminals closest and second-closest to v_l in G. Let z_k denote a terminal farthest away from v_l in G.

Lemma 2.6 ([2,6]) *If*

$$d_{v_l z_i} + d_{v_l z_j} + d_{v_l z_k} \geq |T_G^*(N)|$$

then there exists at least one Steiner minimal tree where v_l has degree at most 2.

Proof: Assume first that the non-terminal v_l has degree at least 3 in some Steiner minimal tree $T_G(N)$. Let z_k' denote a terminal farthest away from v_l in $T_G(N)$. Since $T_G(N)$ is a subnetwork of G, z_k' is at least $d_{v_l z_k}$ away from v_l. Let z_i' denote a terminal closest to v_l in $T_G(N)$ and such that it can be reached from v_l by a path edge-disjoint from the path to z_k'. The length of this path from v_l to z_i' is at least $d_{v_l z_i}$. Let z_j' denote a terminal closest to v_l in $T_G(N)$ and such that it can be reached from v_l by a path edge-disjoint from the paths to z_k' and z_i'. The length of this path from v_l to z_j' is at least $d_{v_l z_j}$. The three edge-disjoint paths in $T_G(N)$ have therefore total length at least $d_{v_l z_i} + d_{v_l z_j} + d_{v_l z_k}$. Hence,

$$|T_G(N)| \geq d_{v_l z_i} + d_{v_l z_j} + d_{v_l z_k} \geq |T_G^*(N)|$$

implying that $T_G^*(N)$ is a Steiner minimal tree not containing v_l. □

If the condition of Lemma 2.6 is fulfilled, then v_l can be deleted from G in the way already described in Subsection 2.1.3. The addition of new edges between vertices adjacent to v_l can be completely avoided if

$$d_{v_l z_i} + d_{v_l z_k} \geq |T_G^*(N)|$$

To see this, suppose that there is a Steiner minimal tree $T_G(N)$ where v_l has degree 2. Let z_k' and z_i' be as defined above. The total length of paths from v_l to z_k' and z_i'

is at least $d_{v_l z_i} + d_{v_l z_k}$. Consequently, $|T_G(N)| \geq d_{v_l z_i} + d_{v_l z_k} \geq |T_G^*(N)|$, implying that $T_G^*(N)$ is a Steiner minimal tree not containing v_l.

$T_G^*(N)$ in Fig. 2.8 is indicated by thick line segments. In this particular example, all non-terminals not in $T_G^*(N)$ disappear due to the R-test. Consider for example the non-terminal v_5. Then $z_i = z_3$, $z_j = z_4$ and $z_k = z_1$. Furthermore $d_{v_5 z_3} = 2$, $d_{v_5 z_4} = 5$ and $d_{v_5 z_1} = 9$ while $|T_G^*(N)| = 10$. Hence v_5 can be deleted from G. For the non-terminal v_6, one has $z_i = z_2$, $z_j = z_4$ and $z_k = z_3$. Furthermore $d_{v_6 z_2} = 2$, $d_{v_6 z_4} = 4$ and $d_{v_6 z_3} = 6$. Hence v_6 can be deleted from G. However, in this case edges (z_2, v_7) and (v_7, v_8) of length 5 and 4 respectively must be added to G. There is no need of adding the edge (z_2, v_8) of length 3 since G already contains an edge between these two vertices of length 1.

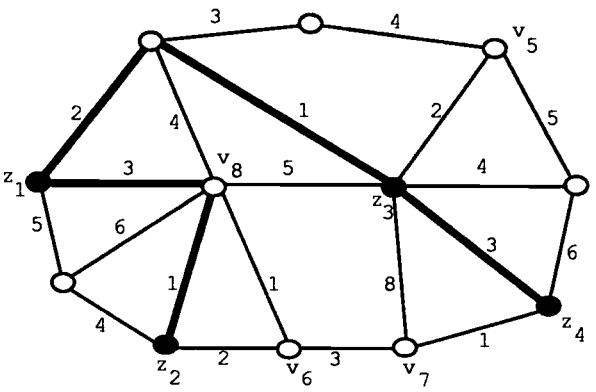


Figure 2.8: Removal of non-terminals based on the R-test

2.1.9 Cut Reachability (CR)

Let $T_G^*(N)$ and v_l be as in Subsection 2.1.8. Let $\bar{c}_{z_i}$ denote the length of the shortest edge in G incident with a terminal z_i. Select z_i and z_j such that $d_{v_l z_i} - \bar{c}_{z_i}$ is smallest possible, and $d_{v_l z_j} - \bar{c}_{z_j}$ is second smallest. Let $N_{ij} = N \setminus \{z_i, z_j\}$.

Lemma 2.7 ([9,6]) *If*

$$d_{v_l z_i} + d_{v_l z_j} + \sum_{z_k \in N_{ij}} \bar{c}_{z_k} \geq |T_G^*(N)| \tag{2.1}$$

then there exists a Steiner minimal tree not containing v_l.

Proof: Suppose that there is a Steiner minimal tree $T_G(N)$ containing v_l. At least one edge enters each terminal. There are at least two edge-disjoint paths leaving v_l toward two distinct terminals z_i' and z_j'. Furthermore, z_i' and z_j' can be chosen such that there is no terminal on the paths from v_l to z_i' and z_j', respectively. Hence,

$$|T_G(N)| \geq d_{v_l z_i'} - \bar{c}_{z_i'} + d_{v_l z_j'} - \bar{c}_{z_j'} + \sum_{z_k \in N} \bar{c}_{z_k} \geq d_{v_l z_i} + d_{v_l z_j} + \sum_{z_k \in N_{ij}} \bar{c}_{z_k}$$

Consequently, $T_G^*(N)$ is a Steiner minimal tree not containing v_l. □

There is also an edge version of this test. Let (v_i, v_j) denote an edge of G not in $T_G^*(N)$. Select $z_i \neq v_j$ and $z_j \neq v_i$ such that $d_{v_i z_i} - \bar{c}_{z_i} + d_{v_j z_j} - \bar{c}_{z_j}$ is smallest possible.

Lemma 2.8 ([9,6]) *If*

$$d_{v_i z_i} + c_{v_i v_j} + d_{v_j z_j} + \sum_{z_k \in N_{ij}} \bar{c}_{z_k} \geq |T_G^*(N)|$$

then there is a Steiner minimal tree not containing the edge (v_i, v_j).

The proof, analogous to the proof of Lemma 2.7, is omitted. An application of the vertex version of the CR-tests is shown in Fig. 2.9 where $\bar{c}_{z_1} = 2$ while $\bar{c}_{z_2} = \bar{c}_{z_3} = \bar{c}_{z_4} = 1$. Non-terminals and their incident edges indicated by broken line segments can be removed from G. For instance, consider the non-terminal v_5: $d_{v_5 z_1} = 5$, $d_{v_5 z_2} = 8$, $d_{v_5 z_3} = 4$, $d_{v_5 z_4} = 7$. Furthermore, $d_{v_5 z_1} - \bar{c}_{z_1} = 3$, $d_{v_5 z_2} - \bar{c}_{z_2} = 7$, $d_{v_5 z_3} - \bar{c}_{z_3} = 3$, $d_{v_5 z_4} - \bar{c}_{z_4} = 6$. Let $z_i = z_1$ and $z_j = z_3$. Then the left side of (2.1) is equal to 11 while $|T_G^*(N)| = 10$. It can be easily verified that the vertex version of the CR-test may fail for a non-terminal v_l while the edge version is applicable to one of the edges incident with v_l. In particular, the edge (v_7, z_3) would be removed by the application of the edge version of the CR-test.

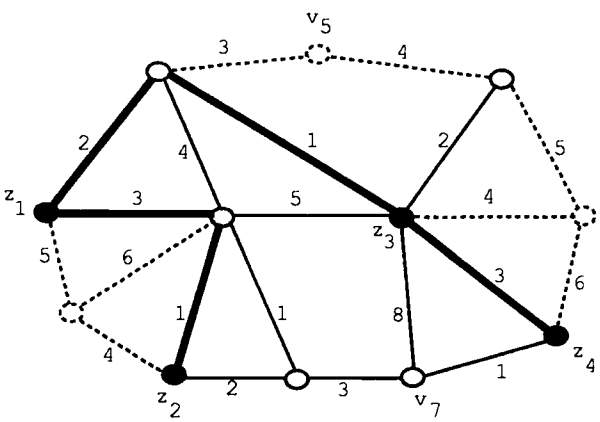


Figure 2.9: Removal of terminals based on the CR-test

2.2 Inclusion Tests

In this section tests that identify edges and non-terminals belonging to at least one Steiner minimal tree are discussed. Again, for similar reasons as when describing the exclusion tests, simple tests, which are subsequently generalized, are discussed first.

2.2.1 Terminals of Degree 1 (TD1)

Suppose that G contains a terminal z_i with $deg_G(z_i) = 1$. The unique edge (z_i, v_j) belongs to every Steiner minimal tree. G can be reduced by contracting it along (z_i, v_j).

2.2.2 Short Terminal-to-Terminal Edges (STTE)

If the nearest vertex adjacent to a terminal z_i also is a terminal, denoted by z_j, then the edge (z_i, z_j) must belong to at least one Steiner minimal tree. Note in particular that (z_i, z_j) belongs to a minimal spanning tree. In fact, even a more general result is valid. The straightforward proof is omitted as the STTE-test will be generalized.

Lemma 2.9 ([1]) *Any edge (z_i, z_j) of a minimum spanning tree belongs to at least one Steiner minimal tree.*

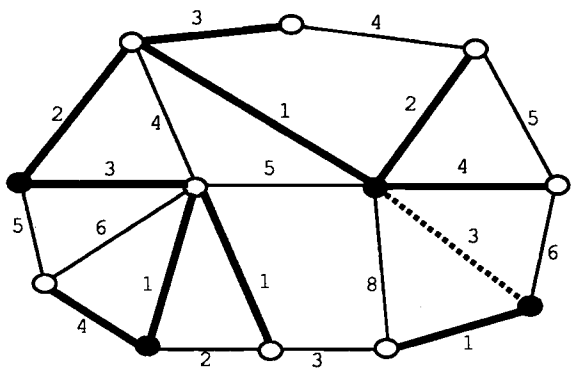


Figure 2.10: Inclusion of an edge based on the STTE-test

An application of the STTE-test is shown in Fig. 2.10. Edges indicated by heavy or broken line segments belong to the minimum spanning tree $T_G(V)$. The edge indicated by the broken line segment is the only edge identified by the STTE-test as belonging to at least one Steiner minimal tree.

The STTE-test leads to the following logical constraints [1]. Suppose that there is an edge (v_i, z_k) in a minimum spanning tree $T_G(V)$ between a terminal z_k and a non-terminal v_i. As soon as it is known that $v_i \in T_G(N)$, the edge (v_i, z_k) must belong to $T_G(N)$ as well. Unfortunately, this information about v_i is not available before $T_G(N)$ has been determined. But consider any integer programming formulation of the Steiner tree problem with x_{ij} denoting a binary variable with value 1 if $(v_i, v_j) \in T_G(N)$ and 0 otherwise. Then the above interdependence can be represented by the constraints

$$x_{ik} \geq x_{ij} \text{ for all } (v_i, z_k) \in T_G(V) \text{ and for all } (v_i, v_j) \in G$$

Although these inequalities are redundant in every integer programming formulation of the Steiner tree problem, they strengthen the linear programming and Lagrangean relaxations, and consequently lead to better lower bounds needed in connection with branch-and-bound algorithms for the Steiner tree problem (Chapter 3).

2.2.3 Nearest Vertex (NV)

This test identifies additional edges of a minimum spanning tree $T_G(V)$ as belonging to at least one Steiner minimal tree $T_G(N)$. The proof is omitted since the test will be generalized. Let z_k denote a terminal in G. Let (z_k, v_i) and (z_k, v_j) denote respectively the shortest and second shortest edge incident with z_k.

Lemma 2.10 ([2]) *If G contains a terminal z_l, $z_l \neq z_k$, satisfying*

$$c_{z_k v_j} \geq c_{z_k v_i} + d_{v_i z_l}$$

then the edge (z_k, v_i) belongs to at least one Steiner minimal tree.

An application of the NV-test is shown in Fig. 2.11 where $c_{z_k v_j} = 8$ while $c_{z_k v_i} + d_{v_i z_l} \leq 6$. Hence (z_k, v_i) belongs to every Steiner minimal tree.

If $deg_G(z_k) = 1$, then let $c_{z_k v_j} = \infty$ and conditions of Lemma 2.10 are fulfilled. The NV-test is therefore a generalization of the TD1-test (Subsection 2.2.1). If $v_i \in N$, then $z_l = v_i$ (i.e., $d_{v_i z_l} = 0$) and conditions of Lemma 2.10 are always satisfied. Hence, the NV-test is also a generalization of the STTE-test (Subsection 2.2.2).

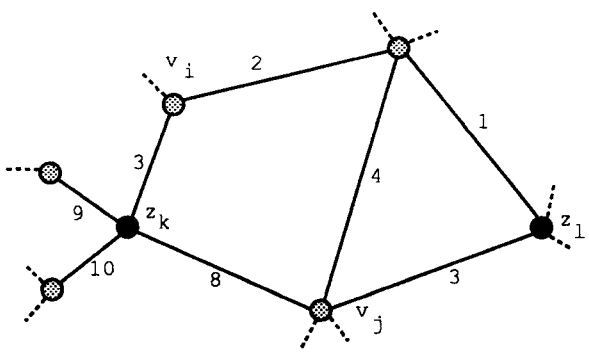


Figure 2.11: Inclusion of an edge based on the NV-test

2.2.4 Short Edges (SE)

The SE-test described in this subsection is a generalization of the TD1-, STTE- and NV-tests. Let (v_i, v_j) be an edge in a minimum spanning tree $T_G(V)$. Let $\bar{r}_{v_i v_j}$ denote the length of the shortest edge among longest edges taken over all paths

from v_i to v_j in $G-(v_i,v_j)$. $\bar{r}_{v_iv_j}$ is referred to as the *restricted bottleneck edge length* between v_i and v_j. Note that if $N=V$, then $\bar{r}_{v_iv_j}=\bar{b}_{v_iv_j}$.

Lemma 2.11 ([6]) *If there is a pair of terminals z_k and z_l such that at least one shortest path from z_k to z_l goes through the edge (v_i,v_j) and*

$$\bar{r}_{v_iv_j} \geq d_{z_kz_l}$$

then the edge (v_i,v_j) belongs to at least one Steiner minimal tree.

Proof: Suppose that the edge (v_i,v_j) is not in any Steiner minimal tree for N in G. $T_G(V)-(v_i,v_j)$ consists of two components C_i and C_j such that $v_i \in C_i$ and $v_j \in C_j$. Since one of the shortest paths $P_{z_kz_l}$ contains the edge (v_i,v_j) and $\bar{r}_{v_iv_j} \geq d_{z_kz_l}$, the path $P_{z_kz_l}$ crosses the cut $\{C_i,C_j\}$ exactly once (at the edge (v_i,v_j)). Hence $z_k \in C_i$ and $z_l \in C_j$ or vice versa.

Consider the path from z_k to z_l in the Steiner minimal tree $T_G(N)$. It must contain an edge (v_p,v_q) of length at least $\bar{r}_{v_iv_j}$. $T_G(N)-(v_p,v_q)$ consist of two components, one containing z_k and the other containing z_l. $T_G(N) \cup P_{z_kz_l}-(v_p,v_q)$ is a connected network that spans N, contains (v_i,v_j), and has total length not greater than $|T_G(N)|$, a contradiction. □

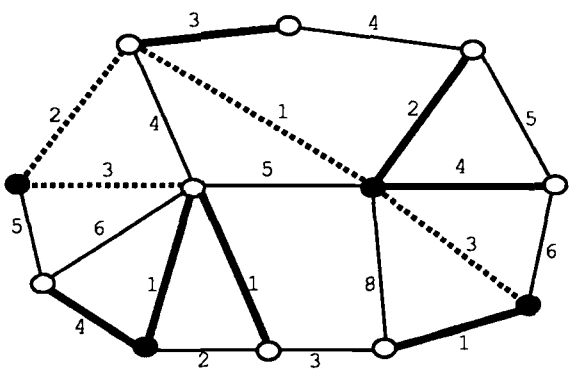


Figure 2.12: Inclusion of edges based on the SE-tests

An application of the SE-test is shown in Fig. 2.12. Edges indicated by heavy and broken line segments belong to the minimum spanning tree $T_G(V)$. Broken line segments indicate the edges that can be contracted.

The SE-test generalizes the NV-test: assume that v_i, v_j, z_k, z_l satisfy the conditions of Lemma 2.10. At least one shortest path from z_k to z_l is forced to go through the edge $(z_k,v_i) \in T_G(V)$. Furthermore, $\bar{r}_{z_kv_i} \geq c_{z_kv_j} \geq c_{z_kv_i}+d_{v_iz_l} \geq d_{z_kz_l}$.

To carry out the SE-test efficiently, one needs to restrict the number of candidates for z_k and z_l for a given edge $(v_i,v_j) \in T_G(V)$. In fact, it suffices to check the conditions of Lemma 2.11 for only one pair of terminals. If $d_{z_kv_i} < d_{z_kv_j}$ for

all $z_k \in N$ (or $d_{z_k v_j} < d_{z_k v_i}$ for all $z_k \in N$), then no shortest path between any pair of terminals goes through (v_i, v_j). Hence conditions of Lemma 2.11 are fulfilled only if there is a terminal z_k closer to v_i than to v_j, and a terminal z_l closer to v_j than to v_i. Suppose that one of the shortest paths between z_k and z_l goes through (v_i, v_j) and $\bar{r}_{v_i v_j} \geq d_{z_k z_l}$. Suppose furthermore that there is a terminal z_p closer to v_i than to v_j and such that $d_{z_k v_i} > d_{z_p v_i}$. It follows immediately that $\bar{r}_{v_i v_j} \geq d_{z_k z_l} > d_{z_p z_l}$. Hence any shortest path between z_p and z_l goes through (v_i, v_j) unless $z_p \in C_j$ where C_j is the component of $T_G(V) - (v_i, v_j)$ containing v_j. Suppose that $z_p \in C_j$. The shortest path from z_p to v_i is not allowed to go through v_j; this would imply that z_p is closer to v_j than to v_i. But the shortest path from z_p to v_i has to cross the cut $\{C_j, C_i\}$, implying that $d_{z_p v_i} \geq \bar{r}_{v_i v_j} \geq d_{z_k z_l} > d_{z_k v_i}$, a contradiction. It follows that z_k can be chosen from among terminals closer to v_i than to v_j such that $d_{z_k v_i}$ is minimized. By a similar argument, z_l can be chosen from among terminals closer to v_j than to v_i such that $d_{v_j z_l}$ is minimized.

An application of the SE-test to edges of a minimum spanning tree $T_G(V)$ only is not restrictive. The conditions of Lemma 2.11 cannot be fulfilled if $(v_i, v_j) \notin T_G(V)$. To see this, suppose that (v_i, v_j) is an edge not in any minimum spanning tree $T_G(V)$, and there is a pair of terminals z_k and z_l such that the shortest path from z_k to z_l goes through (v_i, v_j). Then, $\bar{r}_{v_i v_j} < c_{v_i v_j} \leq d_{z_k z_l}$.

Finally, logical constraints discussed in connection with the STTE-test can be generalized to all edges in a minimum spanning tree $T_G(V)$.

2.2.5 Length Transformations (LT)

Reductions can sometimes become applicable after a suitable modification of edge lengths. One such method was suggested by Duin and Volgenant [6]. Let v_k be a non-terminal. Let (v_k, v_i) and (v_k, v_j) denote two edges incident with v_k. Let G' be a network obtained from G by changing the lengths of (v_k, v_i) and (v_k, v_j) to

$$c'_{v_k v_i} = c_{v_k v_i} + \alpha \text{ and } c'_{v_k v_j} = c_{v_k v_j} - \alpha$$

where $\alpha > 0$. A simple proof of the following lemma is omitted.

Lemma 2.12 ([6]) *If the edges (v_k, v_i) and (v_k, v_j) belong to Steiner minimal trees for $N \cup v_k$ in both G and G', then Steiner minimal trees for N in G and in G' have the same length.*

Suppose that a minimum spanning tree $T_G(V)$ contains two edges (v_k, z_i) and (v_k, z_j), where z_i and z_j are terminals and v_k is a non-terminal. Suppose furthermore that the SE-test does not apply while it does apply to both (v_k, z_i) and (v_k, z_j) when the set of terminals is extended by v_k. Note that (v_k, z_i) and (v_k, z_j) are shortest paths from v_k to z_i and z_j respectively. The length of (v_k, z_i) can be reduced by some amount α, $0 < \alpha \leq c_{v_k z_i}$. This will not influence the applicability of the SE-test for (v_k, z_i) with respect to $N \cup v_k$: $\bar{r}_{v_k z_i}$ remains unchanged while (v_k, z_i) remains the shortest path (of smaller length). Also $\bar{r}_{v_k z_j}$ remains unchanged: every path from v_k to z_j over the edge (v_k, z_i) contains another edge of length at least $c_{v_k z_i}$ (otherwise (v_k, z_i) would not belong to $T_G(V)$). In addition, (v_k, z_j) remains the shortest path from v_k to z_j.

Let α in addition be chosen such that $\alpha < \bar{r}_{v_k z_j} - c_{v_k z_j}$. The increase of the length of (v_k, z_j) by α will not influence the applicability of the SE-test for (v_k, z_j) with respect to $N \cup v_k$. This is due to the fact that $\bar{r}_{v_k z_j}$ remains unchanged. The length of (v_k, z_j), even if it increases, will be strictly smaller than $\bar{r}_{v_k z_j}$. Hence, (v_k, z_j) remains the shortest path from v_k to z_j. Finally, $\bar{r}_{v_k z_i}$ can only increase as a result of increasing the length of (v_k, z_j) and (v_k, z_i) remains the shortest path from v_k to z_i.

Thus, if α is chosen as large as possible subject to $\alpha \leq c_{v_k z_i}$ and $\alpha < \bar{r}_{v_k z_j} - c_{v_k z_j}$, the decrease of the lenght of (v_k, z_i) can make the SE-test applicable to both (v_k, z_i) and (v_k, z_j) in G'. Any tree containing both these edges will have the same length in both G' and G. Hence, in view of Lemma 2.12, a Steiner minimal tree for N in G' will be a Steiner minimal tree for N in G. This is for example the case in Fig. 2.13 [6]. Before the transformation, the shortest path from z_1 to z_3 goes through (z_1, v_5). But $\bar{r}_{z_1 v_5} = 70$ while $d_{z_1 z_3} = 80$. So the SE-test for (z_1, v_5) based on terminals z_1 and z_3 will fail. If v_5 is regarded as a terminal, the shortest path from z_1 to v_5 goes through (z_1, v_5). Furthermore, $\bar{r}_{z_1 v_5} = 70$ and $d_{z_1 v_5} = 30$. Consequently, the SE-test is successful for (v_5, z_1). Similarly, the shortest path from v_5 to z_2 goes through (v_5, z_2). Furthermore, $\bar{r}_{v_5 z_2} = 70$ and $d_{v_5 z_2} = 50$. Consequently, the SE-test is successful for (v_5, z_2). Next, $\min\{c_{v_5 z_1}, \bar{r}_{v_5 z_2} - c_{v_5 z_2}\} = \{30, 70 - 50\} = 20$. If the length of (z_1, v_5) is reduced by less than 20 (e.g., by 19) and the length of (v_5, z_2) is increased by the same amount, then the SE-test remains successful for both (z_1, v_5) and (v_5, z_2). The search for a Steiner minimal tree can now continue in G'. The shortest path from z_1 to z_3 goes again through (v_5, z_1). Furthermore, $\bar{r}_{z_1 v_5} = 70$ while $d_{z_1 z_3} = 61$. Hence, the edge (z_1, v_5) belongs to at least one Steiner minimal tree for N in G'. The vertex v_5 can now be regarded as a terminal. This implies that (v_5, z_2) also belongs to a Steiner minimal tree for N in G'. The length of this Steiner minimal tree is the same in both G and G'. From Lemma 2.12 follows that there is a Steiner minimal tree for N in G' containing both (v_5, z_1) and (v_5, z_2).

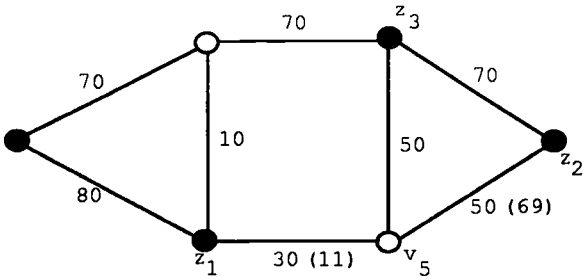


Figure 2.13: Length transformation enabling additional reductions

2.3 Integration of Tests

The tests described in this chapter vary from very simple tests of limited applicability to very general tests. Fig. 2.14 summarizes the interdependencies between various tests. Tests higher up in the hierarchies generalize all tests below.

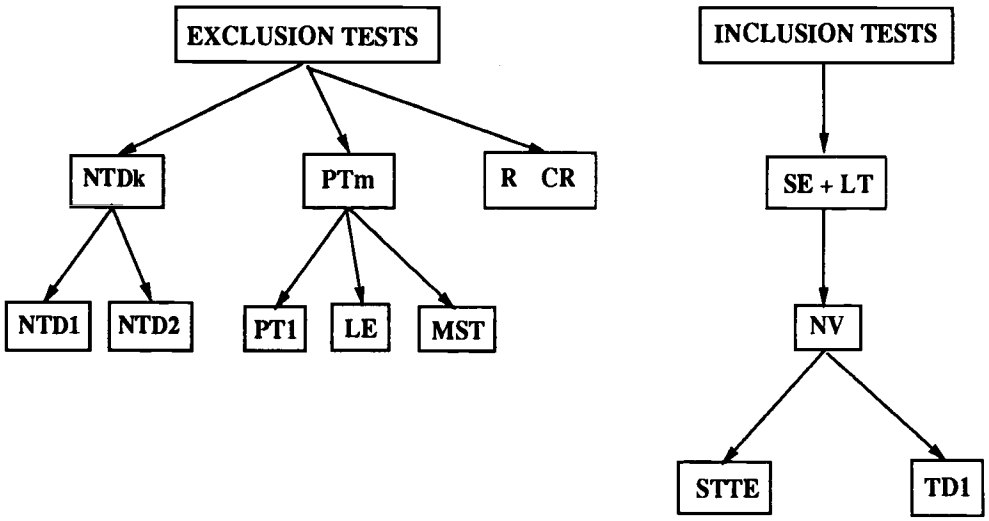


Figure 2.14: Test hierarchies

One reduction may trigger other reductions based on the same or different tests. It is therefore very important to establish the order in which the tests are to be applied. Furthermore, the reduction process should be terminated if the computational effort per test becomes too large. Another very important issue is what modifications of distance and bottleneck matrices are needed after each reduction type.

2.3.1 Data and Algorithms

When the applicability of some particular test is to be established, various types of information are required. Table 2.1 indicates what is required for each non-dominated test type. Rows correspond to the following information: shortest distances between vertices (represented by an $v \times v$ matrix D), bottleneck Steiner distances (represented by an $v \times v$ matrix B), a minimum spanning tree $T_G(V)$, and a suboptimal tree $T_G^*(N)$ spanning all terminals.

Some of the entries in D and B rows are marked with downward or upward arrows. This means that a particular test remains valid even if entries are respectively lower and upper bounds of the exact values. The issue of inexact information will be addressed in Subsection 2.3.3.

	NTDk	PTm	R	CR	SE+LT
D			$\bullet\downarrow$	$\bullet\downarrow$	$\bullet\uparrow$
B	$\bullet\uparrow$	$\bullet\uparrow$			
$T_G(V)$					$\bullet$
$T_G^*(N)$			$\bullet$	$\bullet$	

Table 2.1: Data requirements

In order to simplify the discussion below, assume that D and B are represented by $v \times v$ matrices. $T_G(V)$ and $T_G^*(N)$ are represented by edge lists. It should however be emphasized that more elaborate data structures such as forward stars, heaps and disjoint sets representation [10] could be employed to speed-up the reduction process.

The distance matrix D containing lengths of shortest paths between all pairs of vertices in V can be determined in $O(v^3)$ by for example Dijkstra's algorithm [4] (applied to each vertex as a source). More sophisticated algorithms are also available [10].

The bottleneck Steiner distance matrix B for G can be determined in $O(v^2 n)$ by a simple modification of Dijkstra's algorithm (applied to the distance network $D_G(V)$ rather than to G). More specifically, the following algorithm finds bottleneck Steiner distances from a vertex v_i to all other vertices in G.

- **STEP 1:** $b_{v_i v_j} := d_{v_i v_j}$ for all $v_j \in V$. $L := \{v_i\}$.
- **STEP 2:** If $N \subseteq L$ then **Stop**.
- **STEP 3:** Let $z \in N \setminus L$ with $b_{v_i z}$ smallest possible. $L := L \cup z$.
- **STEP 4:** $b_{v_i v_j} := \min\{b_{v_i v_j}, \max\{b_{v_i z}, d_{z v_j}\}\}$ for every $v_j \in V \setminus L$.
  If v_j is a non-terminal and $b_{v_i v_j} \leq b_{v_i z}$, then $L := L \cup v_j$.
  Go to **STEP 2**.

A minimum spanning tree $T_G(V)$ can be determined in $O(v^2)$ time using for example Prim's algorithm which is closely related to the Dijkstra's algorithm for shortest paths [10]. Alternatively, more sophisticated algorithms using Fibonacci heaps [8] or leftist heaps [10] could be used.

Several good polynomial heuristics for the Steiner tree problem in networks are described in Chapter 4. In particular, the shortest paths heuristic (Subsection 4.1.1) that requires $O(n(e + v \log v))$ time is well-suited for the reduction purposes as it basically requires the distance matrix D. Another more straightforward possibility is to use the simple heuristic which deletes (one at a time) non-terminals of degree 1 from the minimum spanning tree $T_G(V)$ (Subsection 4.2.1). However, this heuristic usually finds suboptimal solutions that are worse than those found by the shortest paths heuristic.

Some tests need additional information which can be easily derived from C (edge length matrix), D, and $T_G(V)$. More specifically, the NTDk-test needs to identify non-terminals of low degree. This information is readily available from C. In

addition, the NTDk-test requires the determination of minimum spanning trees for small complete networks with the bottleneck Steiner distances as edge lengths. The R- and CR-tests need to know the closest, second-closest, and farthest terminals from each non-terminal. In addition, the CR-test needs to know the closest vertex to each terminal. All this information can be obtained in $O(v)$ time from D. The SE-test requires the restricted bottleneck edge length $\bar{r}_{v_i v_j}$ between vertices v_i and v_j adjacent in the minimum spanning tree $T_G(V)$. It is equal to the length of the shortest edge in $G-(v_i, v_j)$ crossing the cut $\{C_i, C_j\}$ obtained by deleting the edge (v_i, v_j) from $T_G(V)$. It can be found in $O(e)$ time.

2.3.2 Updating

When edges and non-terminals are deleted or edges are contracted, it is not always necessary to determine D, B, $T_G(V)$ and $T_G^*(N)$ from the scratch. For instance, when an edge (v_i, v_j) is deleted from G and it does not belong to $T_G(V)$, the minimum spanning tree for the smaller network remains unchanged. If (v_i, v_j) is deleted but belongs to $T_G(V)$, new minimum spanning tree can be determined in $O(e)$ time by reconnecting the two components of $T_G(V) - (v_i, v_j)$. Similar simplifications can be applied when edges are contracted or non-terminals are deleted. Such simplifications are also possible in connection with suboptimal trees spanning the terminals. The determination of shortest paths when edges are deleted or contracted can also be speed-up by the use of distance information in the original network [5]. These methods can also be adapted in connection with the determination of new bottleneck Steiner distances.

2.3.3 Use of Inaccurate Information

The computational effort needed to obtain and update the information necessary to verify the applicability of reductions can be substantial. It can be advantageous to use inaccurate information even if it can fail to recognize some reductions. This is in particular the case during the initial phases of the reduction process when the network is relatively large. Furthermore, rather than updating the information after each reduction, original information can sometimes be used even if it becomes inaccurate.

A simple upper bound for $b_{v_i v_j}$ is $c_{v_i v_j}$. A slightly better upper bound is obtained as follows.

$$b^+_{v_i v_j} = \min\{c_{v_i v_j}, \min_{v_k \notin N}\{c_{v_i v_k} + c_{v_k v_j}\}, \min_{z_l \in N}\{\max\{c_{v_i z_l}, c_{z_l v_j}\}\}\}$$

In other words, $b^+_{v_i v_j}$ is the bottleneck Steiner distance with respect to paths from v_i to v_j with at most two edges.

Table 2.2 shows what happens to D, B, $T_G(V)$, and $T_G^*(N)$ when NTDk-, PTm-, R-, CR- and SE-tests are satisfied and the corresponding reductions are applied.

When applying the NTDk-, PTm-, and SE-tests, the bottleneck Steiner distances between the remaining vertices are either unchanged or become smaller [6]. Hence, the original bottleneck Steiner distances are upper bounds for the actual

	NTDk	PTm	R+CR	SE+LT
D	unchanged	increased	increased	decreased
B	unchanged	unchanged	increased	decreased
T	can change	unchanged	can change	unchanged
T*	can change	unchanged	unchanged	unchanged

Table 2.2: Changes caused by reductions

bottleneck Steiner distances. Updating B can be postponed for as long as desired when applying the NTDk-, PTm-, and SE-tests. However, updating should be done from time to time in order to catch the NTDk- and PTm-tests missed by the use of upper bounds.

PTm-tests do not use shortest distances (although their application can cause shortest distances between the remaining vertices to increase). Hence, there is no need to update D until no more PTm-tests can be applied.

NTDk-tests do not use shortest distances. When they apply, shortest distances between the remaining vertices remain unchanged.

SE-tests use shortest distances but when they apply, an edge is contracted. Consequently, the original shortest distances are upper bounds in the reduced network. Updating D can be postponed for as long as desired when applying the SE-tests. However, this can cause some SE-tests to be missed.

When a non-terminal or an edge is deleted by the R- or CR-tests, shortest distances between the remaining vertices may increase. Consequently, the original distances are lower bounds. Updating D can be postponed for as long as desired when applying the R- and CR-tests. Again, this can cause some R- and CR-tests to be missed.

The edges deleted by the PTm-tests cannot belong to any minimum spanning tree. Hence, $T_G(V)$ remains unchanged when such edges disappear. The edges contracted by the SE-test belong to $T_G(V)$. Hence, a minimum spanning tree for the contracted network can be obtained by contracting $T_G(V)$ along the same edge.

Non-terminals or edges deleted by the R- or CR-tests do not belong to the suboptimal tree $T_G^*(N)$. Hence, there is no need to recompute $T_G^*(N)$ as long as these tests are applicable. If $T_G^*(N)$ is determined by deleting degree 1 non-terminals from $T_G(V)$, then $T_G^*(N)$ is unchanged when PDm- and SE-tests are applied.

2.3.4 Ordering of Tests

The general ordering of tests is shown in Fig. 2.15. More detailed orderings (involving in particular NTD3- and PTm-tests with B^+ rather than B were suggested in [6,12]). For the sake of clarity, it is assumed that the distance matrix D, the bottleneck Steiner distance matrix B, the minimum spanning tree $T_G(V)$ and the suboptimal Steiner tree $T_G^*(N)$ are determined before the reduction process begins,

and they are updated after each reduction. But as discussed in Subsection 2.3.3, these updates could be delayed. All tests are applied as many times as possible before continuing with next test type. Upward-going arrows are followed only if at least one reduction was carried out since previous upward step.

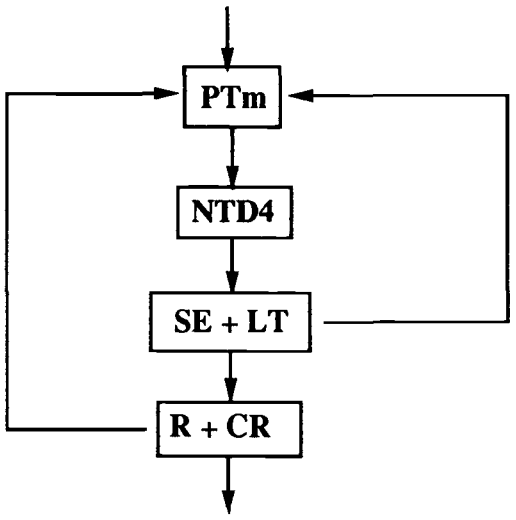


Figure 2.15: Ordering of reductions

The ordering of the tests is not accidental. The PTm-tests should in general be performed before the NTDk-tests since the application of the PTm-tests reduces the degrees of vertices. Since both the PTm- and NTDk-tests delete edges (and possibly replace pairs of edges by a single edge of length equal to the sum of lengths of the pair), these tests can only improve the performance of the SE-tests.

The R- and CR-tests are of rather limited applicability and should only be applied when all other tests fail. The same applies to the LT-transformations which can sometimes trigger additional reductions.

2.4 Effectiveness of Reductions

Although it is not difficult to find networks for which none (or few) of the tests apply (networks satisfying triangle inequality are usually of this type), there are on the other hand many networks for which considerable reductions can be attained (this is in particular the case for sparse networks and for networks with many terminals).

2.4.1 Theoretical Results

Theoretical results concerning the effectiveness of reductions are very limited. For complete graphs, it can be shown [1] that after the STTE-tests, the expected

number of contracted edges is $n(n-1)/v$. In other words, the STTE-tests reduce the number of terminals by a factor of $(n-1)/v$.

For complete graphs, it can be shown [1] that after the MST-tests, the expected number of edges in the reduced network is at most v^2/n. Similar results concerning the SE-tests are given in [7].

2.4.2 Computational Results

For the computational results concerning the effectiveness of reductions, the reader is referred to [1,6,12]. Below the conclusions derived from the test runs reported there are mentioned.

- The speed-up when solving the Steiner tree problem with the reductions is considerable. It can be up to a factor ten for sparse problem instances.
- Reductions perform better for problem instances with larger ratios of n/v. This is mainly due to the fact that the SE-tests are then applicable more often. Also, a good upper bound is usually obtained for this type of problem instances. This makes R- and CR-tests more powerful.
- Problem instances are often completely solved by the reductions alone. This is in particular the case for sparse problem instances ($e \leq 2v$) with many terminals ($n \geq v/2$). This observation also applies for networks with randomly generated edge lengths (with uniform distribution).
- The R- and CR-tests proved to be of limited importance. When applicable, they can often be replaced by other tests.

An important issue which so far has not been addressed adequately in the literature is that of suitable data structures to represent the information needed by various reduction tests. In particular, such data structures must be of dynamic nature as networks undergo non-trivial changes during the reduction process. Another important issue that remains to be addressed in more detail is the applicability of reductions before and in particular during any branch-and-bound algorithm. Developing suitable data structures in this context is even more complicated since networks undergo changes not only when reduced but also during the branching and in particular during the backtracking stages. The use of some reductions in branch-and-bound algorithms have been discussed by Beasley [3] and Voss [11].

References

[1] A. Balakrishnan and N. R. Patel, Problem reduction methods and a tree generation algorithm for the Steiner network problem, *Networks* **17** (1987) 65-85.

[2] J. E. Beasley, An algorithm for the Steiner problem in graphs, *Networks* **14** (1984) 147-159.

[3] J. E. Beasley, An SST-based algorithm for the Steiner problem in graphs, *Networks* **19** (1989) 1-16.

[4] E. W. Dijkstra, A note on two problems in connection with graphs, *Numer. Math.* **1** (1959) 269-271.

[5] R. Dionne and M. Florian, Exact and approximate algorithms for optimal network design, *Networks* **9** (1979) 37-59.

[6] C. W. Duin and A. Volgenant, Reduction tests for the Steiner problem in graphs, *Networks* **19** (1989) 549-567.

[7] C. W. Duin and A. Volgenant, An edge elimination test for the Steiner problem in graphs, *Oper. Res. Lett.* **8** (1989) 79-83.

[8] M. L. Fredman and R. E. Tarjan, Fibonacci heaps and their uses in improved network optimization algorithms, *J. Assoc. Comput. Mach.* **34** (1987) 596-615.

[9] A. Iwainsky, E. Canuto, O. Taraszow and A. Villa, Network decomposition for the optimization of connection structures, *Networks* **16** (1986) 205-235.

[10] R. E. Tarjan, *Data Structures and Network Algorithms*, SIAM (1983).

[11] S. Voss, A reduction based algorithm for the Steiner problem in graphs, *Methods Oper. Res.* **58** (1987) 239-252.

[12] S. Voss, *Steiner-Probleme in Graphen*, Mathematical Systems in Economics **120**, Anton Hein, Frankfurt-am-Main (1990).

Chapter 3

Exact Algorithms

This chapter discusses various exact algorithms for the Steiner tree problem. Sections 3.1-3.3 discuss some complete enumeration algorithms. A dynamic programming algorithm is described in Section 3.4. Branch-and-bound algorithms are discussed in Section 3.5. Several mathematical formulations of the Steiner arborescence problem are given in Section 3.6. The use of linear relaxations of these formulations to obtain lower bounds in branch-and-bound algorithms is discussed in Section 3.7. Lower bounds obtained by Lagrangean relaxations are discussed in Section 3.8. Benders' decomposition algorithm is described in Section 3.9. A set covering algorithm is discussed in Section 3.10. Finally, computational experience concerning these algorithms is mentioned in Section 3.11.

3.1 Spanning Tree Enumeration Algorithm

Hakimi [30] provided the following straightforward *spanning tree enumeration algorithm*: A Steiner minimal tree $T_G(N)$ can be found by the enumeration of minimum spanning trees of subnetworks of G induced by supersets of terminals.

Lawler [34] suggested a modification of the spanning tree enumeration algorithm based on the observation that finding a Steiner minimal tree $T_G(N)$ for N in G is equivalent to finding a Steiner minimal tree $T_D(N)$ for N in the distance network $D = D_G(V)$ (Subsection 1.4.3).

Lemma 3.1 *There exists a Steiner minimal tree $T_D(N)$ for N in the distance network D where each Steiner vertex has degree at least 3.*

Proof: $T_D(N)$ cannot have Steiner vertices of degree 0 or 1. If it has a Steiner vertex v_k of degree 2, with (v_k, v_i) and (v_k, v_j) being the edges incident with v_k, then one can replace these edges by the edge (v_i, v_j) and delete v_k. Since D satisfies the triangle inequality, a tree spanning all terminals with length at most $|T_D(N)|$ is obtained. Steiner vertices of degree 2 can be replaced in this way one by one. Hence, there exists a Steiner minimal tree for N in D where all Steiner vertices have degree at least 3. □

Lemma 3.2 *There exists a Steiner minimal tree $T_D(N)$ for N in D with at most $n-2$ Steiner vertices.*

Proof: Let s denote the number of Steiner vertices in $T_D(N)$ with all Steiner vertices having degree at least 3. Let d_s and d_n denote the mean number of edges incident with a Steiner vertex and a terminal respectively. The number of edges in $T_D(N)$ is $n+s-1 = (d_n n + d_s s)/2$. Since $d_s \geq 3$ and $d_n \geq 1$, it follows that

$$n+s-1 = (d_n n + d_s s)/2 \geq (n+3s)/2$$

implying that $n-2 \geq s$. □

In order to determine $T_D(N)$, it suffices to enumerate minimum spanning trees of subnetworks of D induced by supersets of terminals of size at most $2n-2$.

If $2n-2 < v$, and in particular if G is dense, it is easier to find $T_D(N)$ than $T_G(N)$. On the other hand, if $2n-2 \geq v$ or G is sparse, the structural properties of G are lost when D is formed. Consequently, the effort of determining minimum spanning trees of induced subnetworks of D may be greater than when G is used.

The number of minimum spanning trees that must be determined is

$$\sum_{i=0}^{n-2} \binom{v-n}{i} \leq 2^{v-n}$$

The time complexity of the algorithm is therefore $O(n^2 2^{v-n} + v^3)$ where the term v^3 stems from the shortest paths computation using Floyd's algorithm [19]. The spanning tree enumeration algorithm is polynomial in the number of terminals and exponential in the number of non-terminals. An algorithm polynomial in the number of non-terminals while exponential in the number of terminals will be described in Section 3.4.

3.2 Degree-Constrained Tree Enumeration Algorithm

Another complete enumeration algorithm was suggested by Balakrishnan and Patel [3]. It is based on the *degree-constrained formulation* of the Steiner tree problem. Let $\bar{G}$ and $\bar{T}_G(V)$ be as defined in Subsection 1.4.5.

$\bar{T}_{\bar{G}}(V)$ can be determined by the enumeration of spanning trees of $\bar{G}$ containing the edge (v_0, z_1) in order of non-decreasing length. The first spanning tree with all non-terminals adjacent to v_0 having degree 1 yields $\bar{T}_G(V)$. Gabow's edge-exchange algorithm [24] for the enumeration of spanning trees in order of their nondecreasing length can be used (with modifications preventing the generation of infeasible spanning trees).

3.3 Topology Enumeration Algorithm

Hakimi [30] presented an enumeration algorithm that to some extent is related to the Melzak algorithm for the Euclidean Steiner tree problem [40].

Any Steiner minimal tree $T_G(N)$ must have at least two terminals z_i and z_j satisfying one of the following conditions:

(i) $deg(z_i) = deg(z_j) = 1$, and the path $P(z_i, z_j)$ in $T_G(N)$ contains Steiner vertices only; exactly one of them has degree greater than 2. The remaining Steiner vertices (if any) have degree 2.

(ii) either $deg(z_i) = 1$ or $deg(z_j) = 1$, and the path $P(z_i, z_j)$ in $T_G(N)$ contains Steiner vertices only (if any); their degrees are 2.

If z_i and z_j satisfy either (i) or (ii), then they are called ***pseudoadjacent*** in $T_G(N)$.

The ***topology enumeration algorithm*** determines (by recursion) the minimum length tree spanning N in G with z_i and z_j being pseudoadjacent. Unfortunately, there is no way of knowing beforehand which terminals will be pseudoadjacent in $T_G(N)$. Thus, one has to consider each choice of z_i and z_j in turn. For each choice a minimum length tree spanning N with z_i and z_j being pseudoadjacent is constructed. One with the overall minimum length is $T_G(N)$.

Assume that z_i and z_j are pseudoadjacent in $T_G(N)$. Construct a network $\bar{G} = (\bar{V}, \bar{E}, \bar{c})$ (Fig. 3.1) where

$$\bar{V} = V \cup v_0, v_0 \notin V$$

$$\bar{E} = E \cup \{(v_0, v_k) | v_k \in V\}$$

$$\bar{c}(e_m) = \begin{cases} c(e_m) & \text{if } e_m \in E \\ d_G(z_i, v_k) + d_G(z_j, v_k) & \text{if } e_m = (v_0, v_k), v_k \in V \end{cases}$$

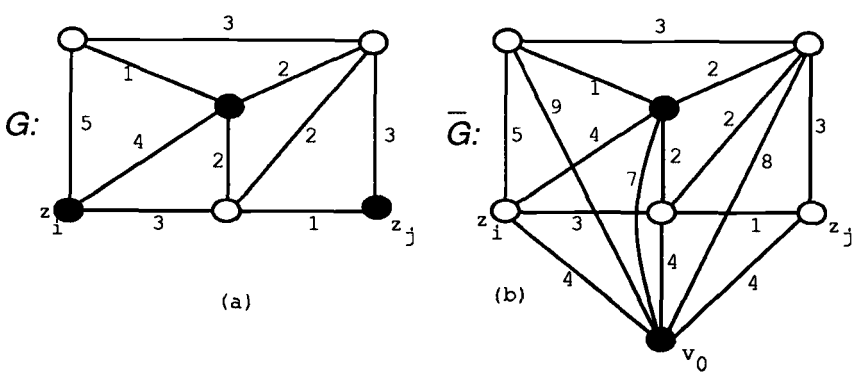


Figure 3.1: Networks G and $\bar{G}$

Let $\bar{T}_{\bar{G}}(\bar{N})$ denote a Steiner minimal tree for $\bar{N} = N \cup v_0 \setminus \{z_i, z_j\}$ in $\bar{G}$ (determined recursively). $\bar{T}_{\bar{G}}(\bar{N})$ can always be chosen such that v_0 is incident with just one edge (v_o, v_k). Let

$$\tilde{T}_{\bar{G}}(\bar{N}) = \bar{T}_{\bar{G}}(\bar{N}) \cup P_G(v_k, z_i) \cup P_G(v_k, z_j) - (v_0, v_k)$$

where v_k is the unique vertex in $\bar{T}_{\bar{G}}(\bar{N})$ adjacent to v_0.

If $\bar{T}_{\bar{G}}(\bar{N})$ is a tree spanning N in G with z_i and z_j being pseudoadjacent, then it is of minimum length among such trees. Otherwise no tree spanning N in G with z_i and z_j being pseudoadjacent can be a Steiner minimal tree [30].

3.4 Dynamic Programming Algorithm

Another approach to the Steiner tree problem, based on the dynamic programming methodology, was presented by Dreyfus and Wagner [13], and independently (in a slightly different form) by Levin [36].

Suppose that v_i is a vertex in $T_G(N)$ incident with more than one edge. Decompose $T_G(N)$ into two subtrees $\bar{T}$ and $\tilde{T}$ as follows (Fig. 3.2):

- replace v_i by two new vertices $\bar{v}_i$ and $\tilde{v}_i$ disconnected from one another,
- connect some arbitrarily chosen edges, previously incident with v_i, to $\bar{v}_i$,
- connect the remaining edges previously incident with v_i, to $\tilde{v}_i$.

Let $\bar{N}$ and $\tilde{N}$ denote the set of terminals in $\bar{T}$ and $\tilde{T}$ respectively. Then $\bar{T}$ and $\tilde{T}$ are Steiner minimal trees for respectively $\bar{N} \cup \bar{v}_i$ and $\tilde{N} \cup \tilde{v}_i$ in G [13].

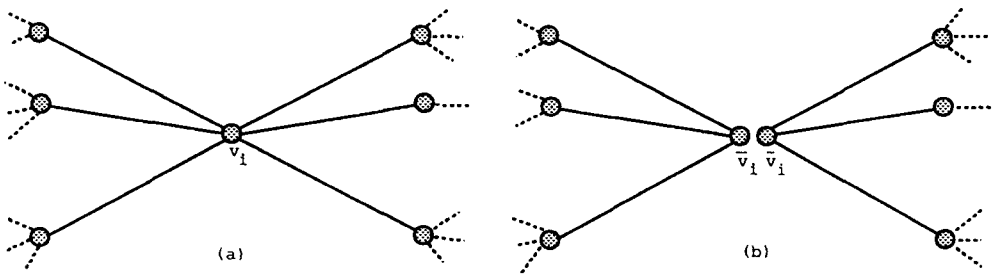


Figure 3.2: Splitting at v_i

This *optimal decomposition property* is the basis of the *dynamic programming algorithm*. Optimal solutions of smaller instances are found and retained for use in solving larger instances (smaller instances are never solved again). Nonoptimal solutions to smaller instances are discarded when solved; they need not be considered when solving larger instances.

Let Y be a nonempty subset of N. Let $v_i \in V \setminus Y$. Let $T_G(v_i \cup Y)$ denote a Steiner minimal tree for $v_i \cup Y$ in G. Let $T_G^2(v_i \cup Y)$ be a minimum length union of two Steiner minimal trees, one spanning $v_i \cup X$ and the other spanning $v_i \cup Y \setminus X$, where $\emptyset \subset X \subset Y$. Consequently,

$$|T_G^2(v_i \cup Y)| = \min_{\emptyset \subset X \subset Y} \{|T_G(v_i \cup X)| + |T_G(v_i \cup Y \setminus X)|\}$$

To obtain a functional relationship for $|T_G(v_i \cup Y)|$, note that there are the following possible configurations of $T_G(v_i \cup Y)$ (Fig. 3.3):

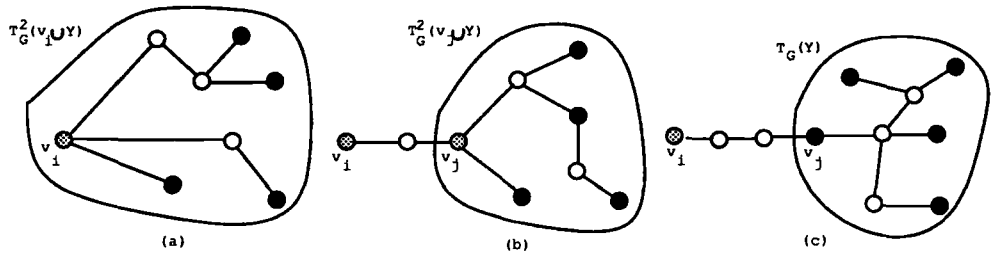


Figure 3.3: Functional relationship for $|T_G(v_i \cup Y)|$

(i) $deg(v_i) > 1$. Then $T_G(v_i \cup Y) = T_G^2(v_i \cup Y)$.

(ii) $deg(v_i) = 1$. $T_G(v_i \cup Y)$ must be a union of $T_G^2(u \cup Y)$ with a shortest path from v_i to some $v_j \notin Y$ of degree at least 3, or a union of $T_G(Y)$ with a shortest path from v_i to some $v_j \in Y$. Since $T_G(v_i \cup Y)$ is a Steiner minimal tree, v_j must be chosen such that the length of the union is minimized. Hence, $|T_G(v_i \cup Y)|$ is the smallest of the following three terms:

$$|T_G^2(v_i \cup Y)| \tag{3.1}$$

$$\min_{v_j \notin Y} \{d_{v_i v_j} + |T_G^2(v_j \cup Y)|\} \tag{3.2}$$

$$\min_{v_j \in Y} \{d_{v_i v_j} + |T_G(Y)|\} \tag{3.3}$$

Note that (3.1) is redundant; it occurs in (3.2) for $v_j = v_i$.

By defining $|T_G(v_i \cup z_k)| = d_{v_i z_k}$ for all $z_k \in N$ and $v_i \in V$, $T_G^2(v_i \cup Y)$ can be determined for every two-terminal subset Y of N and every $v_i \notin Y$. Then $T_G(v_i \cup Y)$ can be determined for every two-terminal subset Y of N and every vertex $v_i \notin Y$. In particular, this will yield all Steiner minimal trees with exactly 3 terminals. Continuing in this manner, will eventually yield $T_G(N)$.

Theorem 3.1 *The worst-case time complexity of the dynamic programming algorithm is* $O(3^n v + 2^n v^2 + v^3)$.

Proof: The total number of computational steps when determining $|T_G^2(v_i \cup Y)|$ is of the same order as the number of possible choices of v_i, Y and X subject to $Y \subseteq N$, $v_i \in V \setminus Y$ and $\emptyset \subset X \subset Y$. There are v choices of v_j. Each terminal belongs to exactly one of the sets $N \setminus Y$, $Y \setminus X$ or X. Thus, the number of choices of Y and X is $O(3^n)$. The number of computational steps is therefore $O(3^n v)$.

The number of times $|T_G(v_i \cup Y)|$ has to be determined is $2^n v$. Each time involves the minimization over at most v terms. Hence, the total number of computational steps is $O(2^n v^2)$. Initial shortest paths computations require $O(v^3)$ time using e.g., Floyd's algorithm [19]. □

It can be seen that the algorithm is polynomial in v (and therefore also polynomial in the number of non-terminals) while exponential in the number of terminals.

As noted by Erickson, Monma and Veinott [16] and Bern [8], the determination of $|T_G(v_i \cup Y)|$ for different choices of v_i but for the same choice of Y can be done simultaneously. Consider (3.2). Build a new network where all terminals in Y are contracted to a supernode v_Y. Add edges from v_Y to all vertices $v_j \notin Y$. One can determine $\min_{v_j \notin Y}\{d_{v_i v_j} + |T_G^2(v_j \cup Y)|\}$ for all choices of v_i simultaneously using Dijkstra's shortest path algorithm [12] with v_Y acting as the root. A similar technique can be used in connection with (3.3).

Dijkstra's shortest path algorithm can be implemented using Fibonacci heaps so that it requires $O(v \log v + e)$ time [23]. In particular, for planar networks, the overall complexity of the algorithm reduces to $O(3^n v + 2^n v \log v + v^2)$. Note that the shortest paths between all pairs of vertices can be determined in $O(v^2)$ time in planar networks [22].

Erickson, Monma and Veinott [16] considered a more general *minimum-concave-cost network flow problem.*

- **GIVEN:** a directed network $\vec{G} = (V, A, c)$ where $c : A \rightarrow \mathcal{R}$ defines the cost of sending a unit of flow through arcs, and a set N, $N \subseteq V$, of terminals with non-zero (positive or negative) flow demands.
- **FIND:** a minimum flow pattern satisfying these demands.

Let Y be a nonempty subset of terminals, and let $v_i \notin Y$. Consider a subproblem obtained by setting the demands of vertices in $V \setminus Y$ to zero, and then reducing the demand of v_i by the sum of demands at Y. The algorithm solves larger and larger subproblems of this type dynamically (using functional relations analogous to those for the Steiner tree problem).

The Steiner tree problem is a special case of the minimum-concave-cost network flow problem: Choose one of the terminals to have demand $-n+1$ and let the remaining $n-1$ terminals have unit demands. Furthermore, replace each (undirected) edge by two arcs with opposite directions. The (concave) unit flow cost is zero if the flow is zero, and is equal to the corresponding arc cost otherwise. If a solution to this problem exists, then there exists an optimal solution with integral flow pattern. Disregarding the orientation of the arcs with non-zero flows yields $T_G(N)$.

3.5 Branch-and-Bound Algorithm

An edge-based *branch-and-bound algorithm* has been developed by Shore, Foulds and Gibbons [42] and in a slightly less efficient form by Yang and Wing [46]. The set F_0 of all feasible solutions (i.e., trees spanning N) is in a systematic way partitioned into smaller subsets. Each subset is analyzed with use of upper and lower bounds, whether or not it can contain a Steiner minimal tree for N in G.

Each subset F_i of trees spanning terminals is characterized by a set IN_i of edges required to be in every tree, and a set OUT_i of edges not permitted in any tree. $IN_0 = \emptyset$ and $OUT_0 = \emptyset$ for F_0.

Consider a subset F_i. Let G_i denote the network obtained from G by the removal of the edges in OUT_i, and by contraction along the edges in IN_i (regarding the vertices incident with edges in IN_i as terminals).

Let N_i denote the terminals in G_i. Determining the optimal solution in F_i (if it exists) is equivalent to the problem of determining a Steiner minimal tree $T_{G_i}(N_i)$ for N_i in G_i. Given $T_{G_i}(N_i)$, the optimal solution in F_i consists of the edges in $T_{G_i}(N_i)$ together with the edges in IN_i.

An upper bound $\overline{b}_i$ for F_i can be determined by adding the lengths of edges in IN_i to the length of a tree spanning N_i in G_i obtained by any of the heuristics for the Steiner tree problem described in Chapter 4. If the shortest tree found so far spanning all terminals has length b_0 (initially $b_0 = \infty$) and $b_0 > \overline{b}_i$, then b_0 is set to $\overline{b}_i$.

In order to determine a lower bound $\underline{b}_i$ for F_i, consider a terminal z_k in G_i. Let

- $\overline{c}_{z_k}$: the length of the shortest edge incident with z_k. If no such edge exists, then $\overline{c}_{z_k} = \infty$.
- $\hat{c}_{z_k}$: the length of the second-shortest edge incident with z_k. If no such edge exists, then $\hat{c}_{z_k} = \infty$.
- $\tilde{c}_{z_k}$: the length of the shortest edge between z_k and another terminal in N_i. If no such edge exists, then $\tilde{c}_{z_k} = \infty$.

Suppose first that $T_{G_i}(N_i)$ contains at least one Steiner vertex. Thus, $T_{G_i}(N_i)$ has at least $|N_i|$ edges. Each terminal in N_i is incident with at least one edge in $T_{G_i}(N_i)$. Furthermore, these edges can be chosen in such a way that they are mutually distinct. Hence,

$$\overline{t}_i = \sum_{z_j \in N_i} \overline{c}_{z_j} \leq |T_{G_i}(N_i)|$$

Suppose next that $T_{G_i}(N_i)$ contains only terminals. Then $T_{G_i}(N_i)$ has exactly $|N_i| - 1$ edges and

$$\tilde{t}_i = \sum_{z_j \in N_i} \tilde{c}_{z_j} - \min\{\tilde{c}_{z_j} \mid z_j \in N_i\} \leq |T_{G_i}(N_i)|$$

It follows that

$$\underline{b}_i = \min\{\overline{t}_i, \tilde{t}_i\} + \sum_{e_m \in IN_i} c(e_m)$$

is a lower bound for F_i. If $b_0 \leq \underline{b}_i$, then no tree in F_i can be shorter than the best tree found so far. Consequently, there is no need to partition F_i any further. If $b_0 < \underline{b}_i$ and $\underline{b}_i = \overline{b}_i$, then the feasible solution obtained when determining $\overline{b}_i$ is the shortest possible in F_i. In this case, $b_0 = \overline{b}_i$, and once again there is no need to partition F_i any further.

Whenever F_i cannot be disregarded (i.e., it is uncertain whether or not F_i contains a Steiner minimal tree), it is partitioned into two subsets:

- F_j with $IN_j = IN_i \cup e_m$, $OUT_j = OUT_i$,
- F_k with $IN_k = IN_i$, $OUT_k = OUT_i \cup e_m$,

for some suitably chosen edge $e_m \in E \setminus (IN_i \cup OUT_i)$. The choice of the edge e_m is crucial for the performance of the algorithm. Shore, Foulds and Gibbons [42]

select as e_m the minimum length edge incident with a terminal $z_k \in N_i$ for which $\hat{c}_{z_k} - \overline{c}_{z_k}$ is maximum (ties are broken arbitrarily).

The subset F_j is examined first (by successive partitioning of F_j). When the optimal feasible solution for F_j has been found or it has been established that the optimal feasible solution for F_0 cannot be in F_j, the subset F_k is examined, and then the algorithm backtracks to F_i (completing the examination of F_i). The enumeration is completed when the algorithm backtracks to F_0. The minimum length feasible solution found during the search yields $T_G(N)$.

A *vertex-based branch-and-bound algorithm* has been developed (among others) by Beasley [5,6]. Each subset F_i of feasible solutions is characterized by having included a set IN_i of non-terminals, and excluded a set OUT_i of non-terminals. Let G_i denote the network obtained from G by deleting the non-terminals in OUT_i and with vertices in IN_i regarded as terminals). Let N_i denote the terminals in G_i.

As in the edge-based branch-and-bound algorithm, the problem of determining the optimal solution in F_i (if it exists) is equivalent to the problem of determining a Steiner minimal tree $T_{G_i}(N_i)$ for N_i in G_i. Branch-and-bound algorithms where the determination of lower bounds is based on various relaxations of mathematical programming formulations (discussed in the following three sections) are typically vertex-based.

3.6 Mathematical Programming Formulations

As already mentioned in Chapter 1, the Steiner arborescence problem in directed networks is a generalization of the Steiner tree problem in undirected networks. This section discusses several mathematical programming formulations of the Steiner arborescence problem. Some of these formulations were originally given for the Steiner tree problem.

3.6.1 Flow Formulations

The first mathematical programming formulation of the Steiner arborescence problem is as follows. Let $N_1 = N - z_1$. For any arc $[v_i, v_j] \in A$, let x_{ij} be a binary variable indicating whether the arc $[v_i, v_j]$ should be included ($x_{ij} = 1$) or excluded ($x_{ij} = 0$) from the solution. Furthermore, for any arc $[v_i, v_j] \in A$ and any terminal $z_k \in N_1$, let y_{ij}^k denote the amount of commodity k to be sent from the terminal z_1 to the terminal z_k through the arc $[v_i, v_j]$. Consider the following mixed integer programming problem $\mathcal{P}_1$, originally introduced by Claus and Maculan [11], Wong [45] and (for undirected case) by Beasley [5]:

$$\min \sum_{[v_i,v_j]\in A} c_{ij} x_{ij} \tag{3.4}$$

subject to

$$x_{ij} \geq y_{ij}^k, \qquad \forall [v_i, v_j] \in A, \forall v_k \in N_1 \tag{3.5}$$

$$\sum_{[v_j,v_i]\in A} y_{ji}^k - \sum_{[v_i,v_j]\in A} y_{ij}^k = \begin{cases} 1, & \forall v_k \in N_1, v_i = v_k \\ 0, & \forall v_k \in N_1, \forall v_i \neq z_1, v_k \end{cases} \quad (3.6)$$

$$x_{ij} \in \{0,1\}, \qquad \forall [v_i,v_j] \in A \quad (3.7)$$

$$y_{ij}^k \geq 0, \qquad \forall [v_i,v_j] \in A, \forall v_k \in N_1 \quad (3.8)$$

Flow conservation constraints (3.6) together with constraints (3.5) force the arcs $\{[v_i,v_j] \mid x_{ij} = 1\}$ in any solution to yield an arborescence rooted at z_1 and spanning all terminals.

Beasley [5] suggested that the following constraint should be added to $\mathcal{P}_1$:

$$n - 1 \leq \sum_{[v_i,v_j]\in A} x_{ij} \leq v - 1 \quad (3.9)$$

This constraint limits the number of arcs in the solution. Although redundant, it does strengthen the lower bounds that will be described in Section 3.7 and in Section 3.8.

A more compact formulation $\mathcal{P}'_1$ is obtained by aggregating constraints (3.5) for the same arc $[v_i,v_j]$.

$$(n-1)x_{ij} \geq \sum_{z_k\in N_1} y_{ij}^k, \qquad \forall [v_i,v_j] \in A$$

The disaggregated formulation $\mathcal{P}_1$ and the aggregated formulation $\mathcal{P}'_1$ are equivalent in the sense that they yield the same optimal solutions. However, the experimental results indicate that optimal solutions are obtained considerably faster for $\mathcal{P}_1$. This observation is in fact supported by a more formal argument. Let $\mathcal{LP}_1$ and $\mathcal{LP}'_1$ denote the linear relaxations of $\mathcal{P}_1$ and $\mathcal{P}'_1$ respectively. Let $v(\mathcal{LP}_1)$ and $v(\mathcal{LP}'_1)$ denote the optimal solution values for $\mathcal{LP}_1$ and $\mathcal{LP}'_1$ respectively.

Lemma 3.3 ([14]) $v(\mathcal{LP}'_1) \leq v(\mathcal{LP}_1)$.

Proof: It is clear that any feasible solution of $\mathcal{LP}_1$ is also a feasible solution of $\mathcal{LP}'_1$. In the reverse direction, this is not true in general. Consider the problem instance in Fig. 3.4. Let $x_{12} = x_{14} = x_{23} = x_{31} = x_{42} = 1$ and $y_{14}^2 = y_{42}^2 = 2$, $y_{23}^2 = y_{31}^2 = y_{12}^3 = y_{23}^3 = 1$ while keeping the remaining x- and y-variables at 0. This solution is feasible for $\mathcal{LP}'_1$ but infeasible for $\mathcal{LP}_1$. □

Many techniques for solving mixed integer programming problems first solve the linear programming relaxations. Because the linear programming version of the disaggregated formulation is more tightly constrained than its aggregated counterpart, better lower bounds are obtained for the former. As pointed out by among others Magnanti and Wong [39], the disaggregated formulation has more constraints. Consequently, it has a richer collection of linear programming dual variables. The enhanced set of dual variables provides more flexibility in the algorithmic development. Furthermore, Jain [33] used probabilistic arguments to show that the linear programming relaxation of $\mathcal{P}'_1$ provides lower bounds rapidly diverging from the optimal values. He also characterized the rate of divergence.

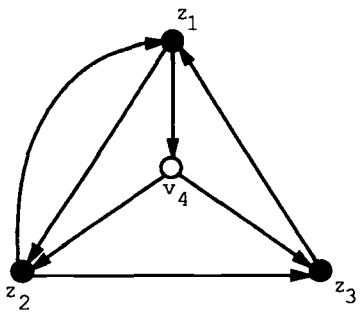


Figure 3.4: $\vec{G}$ with a solution feasible for $\mathcal{LP}'_1$ and infeasible for $\mathcal{LP}_1$

When examining the formulation $\mathcal{P}'_1$, one observes that there is no need for flow variables with respect to each terminal in N_1. This leads to the following formulation $\mathcal{P}''_1$ originally introduced by Arpin, Maculan and Nguyen [2]:

$$\min \sum_{[v_i,v_j]\in A} c_{ij}x_{ij}$$

subject to

$$\begin{aligned}
(n-1)x_{ij} &\geq \textstyle\sum_{z_k\in N_1} y_{ij}, && \forall [v_i,v_j]\in A \\
\textstyle\sum_{[v_j,v_i]\in A} y_{ji} - \sum_{[v_i,v_j]\in A} y_{ij} &= \begin{cases} 1, & \forall v_i \in N_1 \\ 0, & \forall v_i \notin N \end{cases} \\
x_{ij} &\in \{0,1\}, && \forall [v_i,v_j]\in A \\
y_{ij} &\geq 0, && \forall [v_i,v_j]\in A
\end{aligned}$$

The formulations $\mathcal{P}'_1$ and $\mathcal{P}''_1$ are equivalent both in terms of determining the same optimal solution and in terms of identical linear relaxations. In view of Lemma 3.3, $v(\mathcal{LP}_1) \geq v(\mathcal{LP}''_1)$.

3.6.2 Two-Terminal Formulation

A special case of the Steiner arborescence problem arises when $n = 3$. In this case the optimal solution is a union of three paths: one path from the root z_1 to a *splitter vertex* v_p, and two paths from the splitter vertex v_p to terminals z_k and z_l respectively. The mathematical programming formulation of this *two-terminal* Steiner arborescence problem was studied by Ball, Liu and Pulleyblank [4]. It is as follows.

$$\min \sum_{[v_i,v_j]\in A} c_{ij}x_{ij}$$

subject to

$$x_{ij} \geq y^{kl}_{ij} + u^{kl}_{ij} + w^{kl}_{ij}, \qquad \forall [v_i,v_j]\in A$$

$$\sum_{[v_j,v_i]\in A}(y_{ji}^{kl}+u_{ji}^{kl})-\sum_{[v_i,v_j]\in A}(y_{ij}^{kl}+u_{ij}^{kl})=\begin{cases}1, & v_i=z_k\\ 0, & \forall v_i\neq z_1,z_k\end{cases}$$

$$\sum_{[v_j,v_i]\in A}(y_{ji}^{kl}+w_{ji}^{kl})-\sum_{[v_i,v_j]\in A}(y_{ij}^{kl}+w_{ij}^{kl})=\begin{cases}1, & v_i=z_l\\ 0, & \forall v_i\neq z_1,z_l\end{cases}$$

$$x_{ij}\in\{0,1\}, \qquad \forall[v_i,v_j]\in A$$

$$y_{ij}^{kl},u_{ij}^{kl},w_{ij}^{kl}\geq 0, \qquad \forall[v_i,v_j]\in A$$

This leads directly to the *two-terminal* formulation $\mathcal{P}_2$ of the general Steiner arborescence problem with flow constraints for every pair of terminals $z_k, z_l \in N_1$, $z_k \neq z_l$. Liu [37] proved that $v(\mathcal{LP}_2) \geq v(\mathcal{LP}_1)$.

3.6.3 Degree-Constrained Formulation

The next formulation $\mathcal{P}_3$ of the Steiner arborescence problem was presented by Beasley [6]. It is the degree-constrained formulation of the problem discussed in Subsection 1.4.5 stated in mathematical form. While the formulation given by Beasley was for the Steiner tree problem, the formulation for the Steiner arborescence problem is given here.

$$\min \sum_{[v_i,v_j]\in A} c_{ij}x_{ij} \tag{3.10}$$

subject to

$$\{[v_i,v_j] \mid x_{ij}=1\} \text{ is a spanning arborescence of } \bar{G} \tag{3.11}$$

$$x_{0i}+x_{ij}\leq 1, \quad \forall v_i\in V\setminus N, \forall[v_i,v_j]\in A \tag{3.12}$$

$$x_{ij}\in\{0,1\}, \quad \forall[v_i,v_j]\in A \tag{3.13}$$

The constraint (3.11) can be written in mathematical form. However, the corresponding constraints are not essential in the sequel. The implicit formulation is preferred for the sake of clarity.

3.6.4 Set Covering Formulation

Aneja [1] formulated the Steiner tree problem as a set covering problem. The Steiner arborescence version $\mathcal{P}_4$ that is stated below was presented by Wong [45].

$$\min \sum_{[v_i,v_j]\in A} c_{ij}x_{ij} \tag{3.14}$$

subject to

$$\sum_{[v_i,v_j]\in C} x_{ij}\geq 1, \quad \forall C=\{W,\bar{W}\}, z_1\in W, N_1\cap\bar{W}\neq\emptyset \tag{3.15}$$

$$x_{ij}\in\{0,1\}, \quad \forall[v_i,v_j]\in A \tag{3.16}$$

where $C=\{W,\bar{W}\}$ denotes a cut of $\vec{G}$.

Suppose that $X = (x_{ij} | [v_i, v_j] \in A)$ is an optimal solution to $\mathcal{P}_4$. Let T be a subnetwork of $\vec{G}$ whose arc set is $\{[v_i, v_j] | x_{ij} = 1\}$. T is connected and spans N. If not, a constraint corresponding to some cut C would be violated. Furthermore, the set of arcs of any arborescence rooted at z_1 and spanning terminals is a feasible solution of $\mathcal{P}_4$. Since $c_{ij} > 0$ for all $[v_i, v_j] \in A$, it follows immediately that $T = T_{\vec{G}}(N)$.

Lemma 3.4 ([45]) $v(\mathcal{LP}_1) = v(\mathcal{LP}_4)$.

Proof: It is proved first that $v(\mathcal{LP}_1) \geq v(\mathcal{LP}_4)$. Let $(\bar{x}_{ij}, \bar{y}_{ij}^k)$ be a feasible solution of $\mathcal{LP}_1$. Let z_k denote any terminal in N_1. Let $C = \{W, \bar{W}\}$ be a cut of V with $z_1 \in W$ and $z_k \in \bar{W}$.

Since $(\bar{x}_{ij}, \bar{y}_{ij}^k)$ is a feasible solution of $\mathcal{LP}_1$, it is possible to send one unit of commodity k from z_1 to z_k on a network where the capacity of an arc $[v_i, v_j]$ is $\bar{x}_{ij}$. By the max flow-min cut theorem, the total capacity $\sum_{[v_i,v_j] \in C} \bar{x}_{ij}$ of the cut C must be at least one. Since the cut C was chosen arbitrarily, any cut separating z_1 from z_k must have total capacity at least one. Since z_k was chosen arbitrarily, any cut separating z_1 from at least one other terminal must have total capacity at least one. It follows that $(\bar{x}_{ij})$ is a feasible solution to $\mathcal{LP}_4$. Consequently, $v(\mathcal{LP}_1) \geq v(\mathcal{LP}_4)$.

Next it is proved that $v(\mathcal{LP}_1) \leq v(\mathcal{LP}_4)$. The proof is essentially a reversal of the above argument. Let $(\bar{x}_{ij})$ be a feasible solution of $\mathcal{LP}_4$. Consider $\bar{x}_{ij}$ to be the capacity of an arc $[v_i, v_j]$. Let $\bar{y}_{ij}^k = \bar{x}_{ij}$ for all arcs $[v_i, v_j]$ and all $z_k \in N_1$. By the max flow-min cut theorem, it must be possible to send one unit of commodity k from z_1 to every $z_k \in N_1$. Consequently, $(\bar{x}_{ij}, \bar{y}_{ij}^k)$ is a feasible solution to $\mathcal{LP}_1$ and $v(\mathcal{LP}_1) \leq v(\mathcal{LP}_4)$. □

Goemans and Bertsimas [27] and Goemans [26] considered the undirected version of $\mathcal{P}_4$ and obtained some interesting results. For any feasible solution $\bar{X} = (\bar{x}_{ij})$ of $\mathcal{P}_4$, the degree of a vertex v_i, defined by

$$d_{\bar{X}}(v_i) = \sum_{[v_i,v_j] \in E} \bar{x}_{ij}$$

is at least 1 if $v_i \in N$ and at least 0 if $v_i \in V - N$. If $v_i \in N$ and $d_{\bar{X}}(v_i) = 1$ or $v_i \in V - N$ and $d_{\bar{X}}(v_i) = 0$, the $\bar{X}$ is said to be *parsimonious* at the vertex v_i. Let W be any subset of V. Consider the problem $\mathcal{P}_4^W$ obtained from $\mathcal{P}_4$ by adding additional constraints forcing every feasible solution to be parsimonious at every vertex in W. These constraints are

$$\sum_{[v_i,v_j] \in E} x_{ij} = \begin{cases} 1 & \text{if } v_i \in W \cap N \\ 0 & \text{if } v_i \in W - N \end{cases}$$

$\mathcal{P}_4^W$ has the optimal solution value which usually is different from the optimal solution value for $\mathcal{P}_4$. However, if the underlying network G satisfies the triangle inequality, Goemans and Bertsimas [27,26] proved the the linear relaxations of $\mathcal{P}_4$ and $\mathcal{P}_4^W$ have the same optimal solution values. Note that in view of Lemma 1.1

in Subsection 1.4.3, the triangle inequality requirement can be easily satisfied for every instance of the Steiner tree problem.

Since $v(\mathcal{LP}_4) = v(\mathcal{LP}_4^W)$ for any choice of W, this is in particular the case when $W = V$. This in itself allows considerable reduction of the size of $\mathcal{LP}_4^V$ when compared with $\mathcal{LP}_4$. However, Goemans and Bertsimas used the parsimonious property to relate $\mathcal{LP}_4^V$ (and therefore $\mathcal{LP}_4$) with the 1-tree relaxation with the Lagrangean objective function of the traveling salesman problem [31,32,35]. Let $D_G(N)$ denote the distance network induced by the set of terminals N. Geomans and Bertsimas [27,26] proved that the 1-tree relaxation of the traveling salesman is equal to $2v(\mathcal{LP}_4^V) = 2v(\mathcal{LP}_4)$. Moreover, efficient implementations yielding very close approximations to the optimal solutions of 1-tree relaxations are available.

3.7 Linear Relaxations

A linear relaxation of any of the mathematical programming formulations given in Section 3.6 would yield a lower bound for the Steiner arborescence problem. In particular, Arpin, Maculan and Nguyen [2] applied the variable upper bound simplex method for solving the linear relaxation of $\mathcal{P}_1$. The variables x_{ij} in the constraints (3.5) were considered as upper bounds (beginning with $x_{ij} = 1$ if and only if the arc $[v_i, v_j]$ is on a shortest path from z_1 to any other vertex).

Solving linear relaxations has two major disadvantages. First of all, lower bounds can be weak. Secondly, even though linear programming problems can be solved efficiently by various standard methods, their use in branch-and-bound algorithms can be time-consuming. It is therefore often advantageous to solve linear relaxations by heuristic methods. Such approaches are described in the remainder of this section.

Consider the dual $\mathcal{DLP}_1$ of the linear relaxation $\mathcal{LP}_1$ of $\mathcal{P}_1$:

$$\max \sum_{v_k \in N_1} u_{kk}$$

subject to

$$\begin{aligned}
u_{jk} - u_{ik} - w_{ij}^k &\leq 0, && \forall [v_i, v_j] \in A, \forall v_k \in N_1, v_i \neq z_1 \\
u_{jk} - w_{ij}^k &\leq 0, && \forall [v_i, v_j] \in A, \forall v_k \in N_1, v_i = z_1 \\
\textstyle\sum_{v_k \in N_1} w_{ij}^k &\leq c_{ij}, && \forall [v_i, v_j] \in A \\
w_{ij}^k &\geq 0, && \forall [v_i, v_j] \in A, \forall v_k \in N_1
\end{aligned}$$

Solving $\mathcal{DLP}_1$ (heuristically or optimally) would yield a lower bound for $\mathcal{P}_1$. Wong [45] suggested a simple dual ascent heuristic for computing a near optimal solution to $\mathcal{DLP}_1$. Another version of this approach was suggested by Prodon, Liebling and Groflin [41]. As will be seen in Chapter 4, this heuristic also generates (with little extra effort) a feasible solution to $\mathcal{P}_1$.

Let $H = (V, \bar{A})$ denote an auxiliary directed graph (initially $\bar{A} = \emptyset$). Let $C(v_k)$ denote a set of vertices in H for which there exists a directed path to v_k.

Furthermore, let

$$\bar{A}(v_k) = \{[v_i, v_j] \in A | v_j \in C(v_k), v_i \notin C(v_k)\}$$

Thus, addition of the arc $[v_i, v_j] \in \bar{A}(v_k)$ to H would create a directed path from v_i to v_k. The dual ascent heuristic is as follows.

- **Step 1:**

$$\begin{aligned} u_{kk} &:= 0, & \forall v_k \in N_1 \\ w_{ij}^k &:= 0, & \forall [v_i, v_j] \in A, \forall v_k \in N_1 \\ \sigma(v_i, v_j) &:= c_{ij}, & \forall [v_i, v_j] \in A \end{aligned}$$

- **Step 2:** Select a component C of H containing at least one N_1-terminal, and such that all its terminals are strongly connected. If no such C exists, then **Stop**. Note that the heuristic will not terminate unless all terminals in N_1 are connected via a path to z_1.
- **Step 3:** Let v_k be an arbitrarily chosen N_1-terminal in C.

$$\begin{aligned} \sigma(v_{i^*}, v_{j^*}) &= \min\{\sigma(v_i, v_j) \mid [v_i, v_j] \in \bar{A}(v_k)\} \\ u_{hk} &:= u_{hk} + \sigma(v_{i^*}, v_{j^*}), & \forall v_h \in C(v_k) \\ w_{ij}^k &:= w_{ij}^k + \sigma(v_{i^*}, v_{j^*}), & \forall [v_i, v_j] \in \bar{A}(v_k) \\ \sigma(v_i, v_j) &:= \sigma(v_i, v_j) - \sigma(v_{i^*}, v_{j^*}), & \forall [v_i, v_j] \in \bar{A}(v_k) \end{aligned}$$

- **Step 4:** Add the arc $[v_{i^*}, v_{j^*}]$ to H. Go to **Step 2**.

The initial solution to the $\mathcal{DLP}_1$ (with all variables having zero values) is feasible. At each iteration, one of the variables u_{kk}, $v_k \in N_1$, has its value increased. The other variables are also modified. Wong [45] proved that this modification of variable values at Step 2 indeed preserves the dual feasibility of the new solution. One arc is added to H during each iteration. The heuristic therefore terminates after a finite number of iterations.

Wong [45] also observed that the dual ascent approach is a generalization of the spanning arborescence algorithm by Chu and Liu [10] and Edmonds [15]. It is also a generalization of the dual ascent algorithm for the uncapacitated plant location problem by Bilde and Krarup [9] and Erlenkotter [17].

Liu [37] developed a similar dual ascent method with respect to the two-terminal formulation $\mathcal{P}_2$. The initial dual feasible solution is obtained by means of the dual ascent method for $\mathcal{P}_1$.

3.8 Lagrangean Relaxations

This section describes several Lagrangean relaxations for the mathematical programming formulations given in Section 3.6. Common for all variants is that appropriately chosen complicating constraints are removed (and incorporated into the objective function). The relaxed problems are typically very simple and solvable in polynomial time. Furthermore, their solutions provide lower bounds for the optimal solutions of the original problem instances. The lower bounds are

iteratively improved via subgradient optimization. Although the Lagrangean relaxation approaches do not solve original problem instances, they can be integrated into branch-and-bound algorithms.

3.8.1 Lagrangean Relaxations of $\mathcal{P}_1$

The first Lagrangean relaxation $\mathcal{LGR}_1^a$ due to Beasley [5] is based on the flow formulation $\mathcal{P}_1$. It is obtained by regarding (3.5) as complicating constraints. Let u_{ij}^k ($u_{ij}^k \geq 0$ for all $[v_i, v_j] \in A$, $v_k \in N_1$), be Lagrangean multipliers for (3.5). The Lagrangean relaxation $\mathcal{LGR}_1^a$ is then given by

$$\min \sum_{[v_i,v_j]\in A} (c_{ij} - \sum_{v_k \in N_1} u_{ij}^k) x_{ij} + \sum_{v_k \in N_1} \sum_{[v_i,v_j]\in A} u_{ij}^k y_{ij}^k$$

subject to (3.6-3.9).

The optimal solution to $\mathcal{LGR}_1^a$ for any non-negative Lagrangean multipliers is a lower bound on the value of the optimal solution to $\mathcal{P}_1$.

The $\mathcal{LGR}_1^a$ can be easily solved for a given set of non-negative Lagrangean multipliers since it decomposes into n separate subproblems. The first subproblem is

$$\min \sum_{[v_i,v_j]\in A} (c_{ij} - \sum_{z_k \in N_1} u_{ij}^k) x_{ij}$$

subject to (3.7) and (3.9). The optimal solution to this problem is obtained by setting $x_{ij} = 1$ if and only if $c_{ij} - \sum_{[v_i,v_j]\in A} u_{ij}^k$ either is among $n-1$ smallest coefficients or is negative and among $v-1$ smallest coefficients. Each of the remaining $n-1$ subproblems is of the form

$$\min \sum_{[v_i,v_j]\in A} u_{ij}^k y_{ij}^k$$

subject to (3.6) and (3.8) where v_k is a fixed terminal in N_1. Each subproblem can be solved by finding the shortest path in $\vec{G}$ from z_1 to $v_k \in N_1$ with u_{ij}^k interpreted as arc lengths.

In order to improve the lower bound obtained from $\mathcal{LGR}_1^a$, Beasley uses the subgradient optimization algorithm [18] (which is also used in connection with the other lower bounds described below). Since $\mathcal{LGR}_1^a$ has the integrality property, the lower bound will never be better than the lower bound obtained from the linear relaxation $\mathcal{LP}_1$.

The second Lagrangean relaxation $\mathcal{LGR}_1^b$ due to Beasley [5] is also based on the flow formulation $\mathcal{P}_1$. It is obtained by regarding (3.6) as complicating constraints. Let u_{ik}, $v_i \in V$, $v_k \in N_1$, be Lagrangean multipliers for (3.6). The Lagrangean relaxation $\mathcal{LGR}_1^b$ is then given by

$$\min \sum_{[v_i,v_j]\in A} c_{ij} x_{ij} + \sum_{[v_i,v_j]\in A, v_i \neq z_1} \sum_{v_k \in N_1} C_{ij}^k y_{ij}^k - \sum_{v_k \in N_1} u_{kk}$$

subject to (3.5) and (3.7-3.9), where $C_{ij}^k = u_{jk} - u_{ik}$, and the rightmost term is a constant. If x_{ij} is set to 1, then the best contribution from $[v_i, v_j] \in A$ is

$$b_{ij} = \begin{cases} c_{ij} + \sum_{v_k \in N_1} \min\{0, C_{ij}^k\} & \text{if } v_i \neq z_1 \\ c_{ij} & \text{if } v_i = z_1 \end{cases}$$

Thus, $\mathcal{LGR}_1^b$ can be rewritten as

$$\min \sum_{[v_i,v_j] \in A} b_{ij} x_{ij} - \sum_{v_k \in N_1} u_{kk}$$

subject to (3.5) and (3.7-3.9). The optimal solution to $\mathcal{LGR}_1^b$ is obtained by setting $x_{ij} = 1$ if and only if b_{ij} either is among $n-1$ smallest coefficients or is negative and among $v-1$ smallest coefficients. Since $\mathcal{LGR}_1^b$ has the integrality property, the lower bound will never be better than the lower bound obtained from the linear relaxation $\mathcal{LP}_1$.

3.8.2 Lagrangean Relaxation of $\mathcal{P}_3$

The third Lagrangean relaxation $\mathcal{LGR}_3$ due to Beasley [6] is based on the degree-constrained formulation $\mathcal{P}_3$. The lower bound $\underline{b}$ is obtained by regarding (3.12) as the complicating constraints. Let u_{ij} ($u_{ij} \geq 0$ for all $v_i \notin N$), $[v_i, v_j] \in A$) be Lagrangean multipliers for (3.12). The Lagrangean relaxation $\mathcal{LGR}_3$ is then given by

$$\min \sum_{[v_i,v_j] \in A} c_{ij} x_{ij} + \sum_{v_i \notin N} \sum_{[v_i,v_j] \in A} u_{ij}(x_{0i} + x_{ij}) - \sum_{v_i \notin N} \sum_{[v_i,v_j] \in A} u_{ij}$$

subject to (3.11) and (3.13). By rearranging and letting

$$C_{ij} = \begin{cases} \sum_{[v_i,v_j] \in A} u_{ij}, & \text{if } v_i = v_0, v_j \notin N \\ c_{ij} + u_{ij}, & \text{if } [v_i, v_j] \in A, v_i \notin N \\ c_{ij}, & \text{otherwise} \end{cases}$$

the $\mathcal{LGR}_3$ becomes

$$\min \sum_{[v_i,v_j] \in A} C_{ij} x_{ij} - \sum_{v_i \notin N} \sum_{[v_i,v_j] \in A} u_{ij}$$

subject to (3.11) and (3.13). Note that the rightmost term is a constant, and $C_{ij} \geq 0$ for all $[v_i, v_j] \in A$. For any given set of non-negative Lagrangean multipliers, this is the unconstrained minimum spanning arborescence problem. A similar Lagrangean relaxation for the Steiner tree problem in an undirected network would result in the unconstrained minimum spanning tree problem. Both can be solved in polynomial time by any of the well-known algorithms [43,44].

The $\mathcal{LGR}_3$ is very sensitive to the choice of the terminal which is made adjacent to the artificial vertex v_0. A rule suggested by Beasley [6] in the first version of the paper was to select a terminal whose average distance to other terminals is a minimum. In the revised version, a simpler strategy which however seems to yield reasonable results was suggested: select a terminal with the largest degree in $\vec{G}$ (ties are broken arbitrarily).

3.8.3 Lagrangean Relaxation of $\mathcal{P}_4$

The fourth Lagrangean relaxation is based on the set covering formulation $\mathcal{P}_4$. As observed by Dror, Gavish and Choquette [14], there is too little structure in this formulation to be useful in Lagrangean relaxation schemes. In order to amend this weakness, they add the following redundant constraints to $\mathcal{P}_4$.

$$\sum_{[v_i,v_j]\in A} x_{ik} = 1, \quad \forall v_k \in N_1 \tag{3.17}$$

$$\sum_{[v_i,v_j]\in A} x_{ij} \geq n-1, \tag{3.18}$$

$$y_i - y_j + v x_{ij} \leq v-1, \quad \forall [v_i,v_j] \in A \tag{3.19}$$

where y_i is a continuous variable for every $v_i \in V$. Constraints (3.17-3.18) are redundant. Constraints (3.19) rule out every cycle. To see this, consider any cycle C in G and assume that $x_{ij} = 1$ for all $[v_i,v_j] \in C$. Summing up both sides of constraints (3.19) for all $[v_i,v_j] \in C$ yields $v|C| \leq (v-1)|C|$, a contradiction. To see that constraints (3.19) do not rule out any terminal-spanning arborescence rooted at z_1, interpret y_i as the depth of v_i in the arborescence. If v_i is a non-terminal not spanned by the arborescence, let $y_i = 0$. For any arc $[v_i,v_j]$ in the arborescence, $y_i - y_j = -1$. For any arc $[v_i,v_j]$ not in the arborescence, $y_i - y_j \leq v-1$ (since $0 \leq y_i, y_j \leq v-1$). Thus, the corresponding constraint (3.19) is satisfied in both cases.

Dror, Gavish and Choquette [14] relax the extended formulation of $\mathcal{P}_4$ in Lagrangean fashion by regarding the cut constraints (3.15) as complicating. Let u_C ($u_C \geq 0$ for every cut $C = \{W, \bar{W}\}$, $z_1 \in W, N_1 \cap \bar{W} \neq \emptyset, W \cup \bar{W} = V$) be Lagrangean multipliers for (3.15). The Lagrangean relaxation $\mathcal{LGR}_4$ is then given by

$$\min \sum_{[v_i,v_j]\in A} c_{ij}x_{ij} + \sum_C u_C(1 - \sum_{[v_i,v_j]\in C} x_{ij})$$

subject to (3.16) and (3.17-3.19). By rearranging and letting

$$C_{ij} = c_{ij} - \sum_{\{C|[v_i,v_j]\}\in C} u_C, \quad \forall [v_i,v_j] \in A$$

the $\mathcal{LGR}_4$ becomes

$$\min \sum_C u_C + \sum_{[v_i,v_j]\in A} C_{ij}x_{ij}$$

subject to (3.16) and (3.17-3.19). For fixed Lagrangean multipliers, this is a minimum arborescence problem. $\mathcal{LGR}_4$ does not have the integrality property. Using the subgradient optimization can therefore lead to lower bounds better than $v(\mathcal{LP}_4)$. Since $v(\mathcal{LP}_4) = v(\mathcal{LP}_1)$ (Lemma 3.4), Dror, Gavish and Choquette [14] suggested the dual ascent heuristic (Section 3.7) to obtain good initial Lagrangean multipliers.

3.9 Benders' Decomposition Algorithm

Maculan [38] suggested an application of the Benders' decomposition algorithm to the flow formulation $\mathcal{P}_1$ of the Steiner arborescence problem.

Consider the flow formulation $\mathcal{P}_1$ of the Steiner arborescence problem. The linear relaxation $\mathcal{LP}_1$ is bounded. Under the assumption that every terminal in $\vec{G}$ can be reached from z_1, the $\mathcal{LP}_1$ is also feasible.

Suppose that the complicating binary variables are fixed, i.e., $x_{ij} = \bar{x}_{ij} \in \{0,1\}$ for every $[v_i, v_j] \in A$. Then $\mathcal{P}_1$ reduces to a linear programming problem $\mathcal{LP}_1(\bar{X})$. Its dual $\mathcal{DLP}_1(\bar{X})$ is

$$\max \sum_{v_k \in N_1} (u_{kk} - \sum_{[v_i,v_j] \in A} \bar{x}_{ij} w_{ij}^k) + \sum_{[v_i,v_j] \in A} c_{ij} \bar{x}_{ij} \tag{3.20}$$

subject to

$$u_{jk} - u_{ik} - w_{ij}^k \leq 0, \quad \forall [v_i, v_j] \in A, \forall v_k \in N_1, v_i \neq z_1 \tag{3.21}$$

$$u_{jk} - w_{ij}^k \leq 0, \quad \forall [v_i, v_j] \in A, \forall v_k \in N_1, v_i = z_1 \tag{3.22}$$

$$w_{ij}^k \geq 0, \quad \forall [v_i, v_j] \in A, \forall v_k \in N_1 \tag{3.23}$$

Constraints in every $\mathcal{DLP}_1(\bar{X})$ are also constraints in $\mathcal{DLP}_1$. The fact that $\mathcal{LP}_1$ is bounded and feasible implies that $\mathcal{DLP}_1$ is feasible. Hence every $\mathcal{DLP}_1(\bar{X})$ is feasible and $v(\mathcal{LP}_1(\bar{X})) = v(\mathcal{DLP}_1(\bar{X}))$. It follows that $\mathcal{P}_1$ is equivalent to

$$\min \beta$$

subject to

$$\beta \geq \textstyle\sum_{v_k \in N_1} (u_{kk} - \sum_{[v_i,v_j] \in A} x_{ij} w_{ij}^k) + \sum_{[v_i,v_j] \in A} c_{ij} x_{ij}$$
$$x_{ij} \in \{0,1\}, \forall [v_i, v_j] \in A$$

and (3.21-3.23). Consider the relaxation of the above formulation of $\mathcal{P}_1$ obtained by dropping the constraints (3.21-3.23). It will be referred to it as *Benders' master problem.* Let $\bar{X} = \{\bar{x}_{ij} | [v_i, v_j] \in A\}$ denote the values of binary variables in the optimal solution to the Benders' master problem. The solution of $\mathcal{DLP}_1(\bar{X})$ can be obtained by solving separately $n-1$ $\mathcal{DLP}_1^k$ problems, one for each $v_k \in N_1$:

$$\max u_{kk} - \sum_{[v_i,v_j] \in A} \bar{x}_{ij} w_{ij}^k$$

subject to

$$u_{jk} - u_{ik} - w_{ij}^k \leq 0, \quad \forall [v_i, v_j] \in A, v_i \neq z_1$$
$$u_{jk} - w_{ij}^k \leq 0, \quad \forall [v_i, v_j] \in A, v_i = z_1$$
$$w_{ij}^k \geq 0, \quad \forall [v_i, v_j] \in A,$$

If $v(\mathcal{DLP}_1^k(\bar{X})) < \infty$ for all $v_k \in N_1$ then $v(\mathcal{DLP}_1(\bar{X})) = \sum_{v_k \in N_1} v(\mathcal{DLP}_1^k(\bar{X})) + \sum_{[v_i,v_j] \in A} \bar{x}_{ij} w_{ij}^k$. If $\mathcal{DLP}_1^k(\bar{X})$ is unbounded for at least one $v_k \in N_1$, then $\mathcal{DLP}_1(\bar{X})$ is also unbounded.

Suppose that $\mathcal{DLP}_1^k(\bar{X})$ is bounded for every $v_k \in N_1$. Then $\bar{X}$ is the optimal solution to $\mathcal{P}_1$. Suppose that $\mathcal{DLP}_1^k(\bar{X})$ is unbounded for some $v_k \in A$. This implies that $\mathcal{LP}_1(\bar{X})$ is infeasible. Regard $\bar{x}_{ij}$ as the capacity of the arc $[v_i, v_j]$. Unboundedness of $\mathcal{DLP}_1^k(\bar{X})$ implies that it is impossible to send one unit of flow from z_1 to v_k. Let $\{W, \bar{W}\}$ denote the cut in $\vec{G}$ such that W contains all vertices that can receive one unit of flow from z_1. This partition can be determined by solving a maximum flow problem. Let

$$\bar{u}_{ik} = \begin{cases} 0 & \text{if } v_i \in W \setminus \{z_1\} \\ \lambda & \text{if } v_i \in \bar{W} \end{cases}$$

and

$$\bar{w}_{ij}^k = \begin{cases} \lambda|\bar{u}_{jk} - \bar{u}_{ik}| & \text{if } [v_i, v_j] \in A, v_i \neq z_1 \\ \lambda \bar{u}_{jk} & \text{if } [v_i, v_j] \in A, v_i = z_1 \end{cases}$$

for any $\lambda \geq 0$. This is a feasible solution to $\mathcal{DLP}_1^k(\bar{X})$ for every $\lambda \geq 0$. Hence, an extreme ray of the polyhedral convex cone defined by the constraints of $\mathcal{DLP}_1^k(\bar{X})$ has been identified. Add the constraint

$$\bar{u}_{kk} - \sum_{[v_i, v_j] \in A} x_{ij} \bar{w}_{ij}^k \leq 0$$

to the Benders' master problem and solve it again. Note that $\bar{w}_{ij}^k \neq 0$ if and only if $[v_i, v_j]$ is crossing the cut $\{W, \bar{W}\}$. Substituting $\bar{u}_{kk}$ and $\bar{w}_{ij}^k$ by their values therefore yields

$$\sum_{[v_i, v_j] \in A} x_{ij} \geq 1, \quad v_i \in W, v_j \in \bar{W}$$

This is one of the constraints occurring in the set covering formulation $\mathcal{P}_4$ of the Steiner arborescence problem.

A new solution $\tilde{X} = \{\tilde{x}_{ij} | [v_i, v_j] \in A\}$ leads to a new dual $\mathcal{DLP}_1(\tilde{X})$. If any of its $n - 1$ separable subproblems $\mathcal{DLP}_1^k(\tilde{X})$, $v_k \in N_1$, is again unbounded, a new constraint corresponding to an extreme ray will be identified. Furthermore, it must be different from all extreme rays identified previously. Since the number of extreme rays in the polyhedral cone defined by (3.21-3.23) is finite, (3.20-3.23) will sooner or later be bounded, yielding a solution to $\mathcal{P}_1$. Usually the algorithm will terminate after a small number of iterations.

If the Benders' decomposition algorithm described above required the identification of all extreme rays, Benders' master problem would be identical with the set covering formulation $\mathcal{P}_4$.

3.10 Set Covering Algorithm

The number of constraints in the set covering formulation $\mathcal{P}_4$ grows exponentially with the size of problem instances. However, the *set covering algorithm* suggested by Aneja [1] is able to handle the constraints implicitly. This algorithm is a modification of the cutting plane algorithm for the general set covering problem proposed by Bellmore and Ratliff [7] and also described in [25].

Consider the linear relaxation $\mathcal{LP}_4$ of $\mathcal{P}_4$. It can be solved by the dual simplex algorithm. However, if the number of constraints is large (as is the case for the Steiner arborescence problem), this approach is not attractive. Aneja has shown how to determine variables entering and leaving a current basis (initially slack variables are introduced into the basis) without the explicit knowledge of constraints. Essentially, these variables are determined by identifying the minimum length cuts among all cuts separating z_1 from at least one other terminal and with simplex multipliers interpreted as edge lengths. This problem can be solved in polynomial time by the algorithm of Gomory and Hu [28].

The details of the set covering algorithm will not be given here as the approach is closely related to the Benders' decomposition algorithm described in Section 3.9. The reader is referred to [1,7,25] for more on the set covering algorithm.

3.11 Summary and Computational Experience

Table 3.1 gives an overview of the exact algorithms for the Steiner tree (and arborescence) problem discussed in this chapter.

In general, it is very difficult to say anything definite about the performance of the algorithms. One of the reasons is that the computational results presented by the authors are very limited. Secondly, algorithms have been implemented in different programming languages, run on different computers, and applied to different problem instances.

Shore, Foulds and Gibbons [42] reported on computational experience for the spanning tree enumeration algorithm (Section 3.1), dynamic programming algorithm (Section 3.4), and the branch-and-bound algorithm (Section 3.5). The same results are also reported in [20,21]. Complete networks with $v = 10, 20, 30$ and for several values of n were randomly generated. The main conclusions that can be drawn from the average computation times are as follows.

- If the number of terminals is small (less than $v/2$), then the dynamic programming algorithm performs best.
- If the number of terminals is around $v/2$, then the branch-and-bound algorithm performs best.
- If the number of terminals is large (more than $v/2$), then the spanning tree enumeration algorithm performs best.

Furthermore, the spanning tree enumeration algorithm and dynamic programming algorithms remain largely unaffected by the change of the range of lengths. On the other hand, the branch-and-bound algorithm exhibited widely fluctuating times, making it difficult to draw any conclusions about the influence of edge lengths on its performance. However, it appears that the distribution of edge lengths, rather than merely their variance is the crucial factor affecting the performance of this algorithm.

Wong [45], Arpin, Maculan and Nguyen [2] and Maculan [38] observed that the linear programming relaxation of $\mathcal{P}_1$ very frequently (at least for sparse networks) yields integer solutions. In particular, all 80 linear relaxations considered by Arpin,

Method	Submethod	Model	Reference	Section
Complete Enumeration	Spanning Tree Enumeration		Hakimi [30]	3.1
			Lawler [34]	3.1
	Degree-Constrained Enumeration		Balakrishnan and Patel [3]	3.2
	Topology Enumeration		Hakimi [30]	3.3
Dynamic Programming			Dreyfus and Wagner [13]	3.4
			Levin [36]	3.4
Branch-and-Bound			Shore et al. [42]	3.5
			Yang and Wing [46]	3.5
	Linear Relaxations	$\mathcal{P}_1$	Arpin et al. [2]	3.7
		$\mathcal{P}_1$	Wong [45]	3.7
		$\mathcal{P}_1$	Prodon et al. [41]	3.7
		$\mathcal{P}_2$	Liu [37]	3.7
	Lagrangean Relaxations	$\mathcal{P}_1$	Beasley [5]	3.8
		$\mathcal{P}_3$	Beasley [6]	3.8
		$\mathcal{P}_4$	Dror et al. [14]	3.8
Benders' Decomposition		$\mathcal{P}_1$	Maculan [38]	3.9
Set Covering		$\mathcal{P}_4$	Aneja [1]	3.10

Table 3.1: Exact algorithms for the Steiner tree/arborescence problem

Maculan and Nguyen [2] with $v \in [10, 40]$, $n \in [5, 8]$ and $e \in [20, 50]$ had integer optimal solutions. The average computation times (using CDC CYBER 835) were between 0.857 and 8.910 seconds. Aneja [1] made a similar observation for the (equivalent) linear relaxation of $\mathcal{P}_4$.

The dual ascent method used to determine the lower bound for $\mathcal{P}_1$ (Section 3.7) was implemented by Wong [45] (in FORTRAN using IBM 3033). Its efficiency was tested for 24 problem instances within the following four groups (6 in each group): $v = 40$, $n = 20$, $e = 120/160$, and $v = 60$, $n = 30$, $e = 180/240$. Edge lengths were randomly selected from the interval $[0, 1]$. Lower bounds for 22 of these problem instances turned out to be equal the value of upper bounds determined by the dual ascent heuristic which will be described in Chapter 4. For the remaining 2 problem instances, the gap between the upper and lower bounds was extremely small. Average CPU times for each group of problem instances were well below 1 second. Similar results were reported by Guyard [29].

The dual ascent method used to determine the lower bound for $\mathcal{P}_2$ (Section 3.7) was implemented by Liu [37] (in C using Micro VAX II). Problem instances with up to 100 vertices and up to 40 terminals were solved in less than 2.344 seconds. More importantly, lower bounds obtained were in most cases strictly better than those obtained by the dual ascent method applied to $\mathcal{P}_1$. Also, the upper bounds (which

can be derived with little extra effort) were usually better. Thus, a branch-and-bound algorithm would have little difficulty in determining an optimal solution. Such an algorithm remains to be developed.

Vertex-based branch-and-bound algorithms employing Lagrangean relaxations $\mathcal{LGR}_1^a$, $\mathcal{LGR}_1^b$, and $\mathcal{LGR}_3$ (Section 3.8) proved (at least for sparse networks) to be very efficient. For the Steiner tree problem (undirected networks), Beasley [5,6] observed that $\mathcal{LGR}_1^a$ and $\mathcal{LGR}_1^b$ are more or less comparable with respect to their efficiency. In particular, the duality gap for $\mathcal{LGR}_1^a$ is lower than when using $\mathcal{LGR}_1^b$. On the other hand, the use of $\mathcal{LGR}_1^b$ requires less time in total to obtain optimal solutions. However, the difference in time is essential only for few of the randomly generated problem instances. Similar results for the Steiner arborescence problem were reported by Dror, Gavish and Choquette [14].

$\mathcal{LGR}_3$ is superior to $\mathcal{LGR}_1^a$ and $\mathcal{LGR}_1^b$ both in terms of the duality gap at F_0 and in terms of execution times. Computational evidence presented in the first version of [6] indicated that $\mathcal{LGR}_3$ is able to solve problem instances with very large but sparse networks. For example, among 12 randomly generated problem instances with $v = 500$, $e = 625, 1000, 2500$, and $n = 5, 83, 125, 250$, all but one were solved within 20 minutes. Not surprisingly, the efficiency of $\mathcal{LGR}_3$ improves as n becomes large. In the second version of [6], even better computational times were reported for the same instances (using CRAY-X-MP/48). In particular, it was reported that (a subset of) reduction tests described in Chapter 2 have not been very effective, and were as a consequence not used. Also larger instances with edge-densities reaching 0.1 have been tested. These results indicate that $\mathcal{LGR}_3$ is able to solve large, randomly generated problem instances.

Computational results given by Beasley [6] are for relatively sparse networks. In particular, reductions involving computation of shortest paths could prove more useful for dense graphs. Furthermore, no information about the distribution and variance of edge lengths is given. Further analysis of $\mathcal{LGR}_3$ would therefore be of interest.

Since $\mathcal{LGR}_4$ uses the dual ascent heuristic to obtain initial Lagrangean multipliers, it is likely that subgradient optimization would produce good quality lower bounds. However, Dror, Gavish and Choquette [14] did not address this issue.

The set covering algorithm (Section 3.10) was implemented by Aneja [1] (in FORTRAN IV using IBM 370/158). 10 sparse problem instances for each of the following $v|e$ pairs: $10|20, 20|30, 20|40, 30|40, 30|50$, $40|50, 40|60, 50|60$, and $n = v/2$ were tested for relatively sparse networks with random edge lengths from the interval [1,10]. Solutions to the linear relaxation of $\mathcal{P}_4$ yielded very good bounds. For 15 of 57 problem instances that were solved within 60 seconds, $\mathcal{LP}_4$ yielded integer solutions during the first iteration. For 12 more problem instances, the cost of the nonredundant cover obtained by rounding up the values of the optimal solution to $\mathcal{LP}_4$ was equal to the value of this optimal solution. Furthermore, for 13 of the remaining 30 problem instances, the first nonredundant cover was in subsequent iterations discovered to be optimal. Among 23 problem instances that did not terminate within 60 seconds, 20 did not complete solving $\mathcal{LP}_4$ at the first iteration. This suggests that solving $\mathcal{LP}_4$ is a bottleneck of the set covering

algorithm.

No computational results for the Benders' decomposition algorithm are available. But this algorithm is closely related to the set covering algorithm. It most likely has similar performance characteristics.

References

[1] Y. P. Aneja, An integer linear programming approach to the Steiner problem in graphs, *Networks* **10** (1980) 167-178.

[2] D. Arpin, N. Maculan and S. Nguyen, Le problème de Steiner sur un graphe orienté: formulations et relaxations, Technical Report 315, Centre de Recherche sur les Transports, Université Montréal (1983).

[3] A. Balakrishnan and N. R. Patel, Problem reduction methods and a tree generation algorithm for the Steiner network problem, *Networks* **17** (1987) 65-85.

[4] M. O. Ball, W. G. Liu and W. R. Pulleyblank, Two terminal Steiner tree polyhedra, *Proc. of CORE 20th Year Anniversary Conference* (1988).

[5] J. E. Beasley, An algorithm for the Steiner problem in graphs, *Networks* **14** (1984) 147-159.

[6] J. E. Beasley, An SST-based algorithm for the Steiner problem in graphs, *Networks* **19** (1989) 1-16.

[7] M. Bellmore and H. D. Ratliff, Set-covering and involutary bases, *Manage. Sci.* **18** (1971) 194-206.

[8] M. W. Bern, Faster exact algorithm for Steiner trees in planar networks, *Networks* **20** (1990) 109-120.

[9] O. Bilde and J. Krarup, Sharp lower bounds and efficient algorithms for the simple plant location problem, *Ann. Discrete Math.* **1** (1977) 79-97.

[10] Y. J. Chu and T. H. Liu, On the shortest arborescences of a directed graph, *Scientia Sinica* **14** (1965) 1396-1400.

[11] A. Claus and N. Maculan, Une nouvelle formulation du problème de Steiner sur un graphe, Technical Report 280, Centre de Recherche sur les Transports, Université Montréal (1983).

[12] E. W. Dijkstra, A note on two problems in connection with graphs, *Numer. Math.* **1** (1959) 269-271.

[13] S. E. Dreyfus and R. A. Wagner, The Steiner problem in graphs, *Networks* **1** (1971) 195-207.

[14] M. Dror, B. Gavish and J. Choquette, Directed Steiner tree problem on a graph: models, relaxations and algorithms, *INFOR* **28** (1990) 266-281.

[15] J. Edmonds, Optimum branchings, *J. Res. Natl. Bur. Stand. - B. Mathematics and Mathematical Physics* **71B** (1967) 233-240.

[16] R. E. Erickson, C. L. Monma and A. F. Veinott Jr., Send-and-split method for minimum-concave-cost network flows, *Math. Oper. Res.* **12** (1987) 634-664.

[17] D. Erlenkotter, A dual-based procedure for uncapacitated facility location, *Oper. Res.* **26** (1978) 992-1009.

[18] M. L. Fisher, The Lagrangian relaxation method for solving integer programming problems, *Manage. Sci.* **27** (1981) 1-18.

[19] R. W. Floyd, Algorithm 97: Shortest path, *Commun. ACM* **5** (1962) 345.

[20] L. R. Foulds, P. B. Gibbons and M. L. Shore, Algorithms for the Steiner problem in graphs, *J. Comb. Inf. Syst. Sci.* **6** (1981) 215-219.

[21] L. R. Foulds and V. J. Rayward-Smith, Steiner problems in graphs: Algorithms and applications, *Eng. Optimization* **7** (1983) 7-16.

[22] G. N. Frederickson, Fast algorithms for shortest paths in planar graphs, with applications, *SIAM J. Comput.* **16** (1987) 1004-1023.

[23] M. L. Fredman and R. E. Tarjan, Fibonacci heaps and their uses in improved network optimization algorithms, *J. Assoc. Comput. Mach.* **34** (1987) 596-615.

[24] H. N. Gabow, Two algorithms for generating weighted spanning trees in order, *SIAM J. Comput.* **6** (1977) 139-150.

[25] R. S. Garfinkel and G. L. Nemhauser, *Integer Programming*, J. Wiley, N.Y. (1972).

[26] M. X. Goemans, Survivable networks and parsimonious property, Technical Report, Oper. Res. Center, MIT (1990).

[27] M. X. Goemans and D. J. Bertsimas, On the parsimonious property of connectivity problems, Technical Report, Oper. Res. Center, MIT (1989).

[28] R. E. Gomory and T. C. Hu, Multi-terminal network flows, *J. SIAM* **9** (1961) 551-570.

[29] L. Guyard, Le problème de l'arbre de Steiner: modélization par programmation linéaire et résolution par des techniques de decomposition (application à un modèle de bases de données relationnelles), PhD Thesis, Ecole Nationale Supérieure des Télécommunications, Paris (1985).

[30] S. L. Hakimi, Steiner's problem in graphs and its implications, *Networks* **1** (1971) 113-133.

[31] M. Held and R. M. Karp, The traveling-salesman problem and minimum spanning trees, *Oper. Res.* **18** (1970) 1138-1162.

[32] M. Held and R. M. Karp, The traveling-salesman problem and minimum spanning trees: Part II, *Math. Program.* **1** (1971) 6-25.

[33] A. Jain, Probabilistic analysis of LP relaxation bounds for the Steiner problem in networks, *Networks* **19** (1989) 793-801.

[34] E. L. Lawler, *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart, and Winston, N.Y. (1976).

[35] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan and D. B. Shmoys (eds.), *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, J. Wiley & Sons (1985).

[36] A. Y. Levin, Algorithm for the shortest connection of a group of graph vertices, *Sov. Math. Dokl.* **12** (1971) 1477-1481.

[37] W. G. Liu, A lower bound for the Steiner tree problem in directed graphs, *Networks* **20** (1990) 765-778.

[38] N. Maculan, The Steiner problem in graphs, *Ann. Discrete Math.* **31** (1987) 185-212.

[39] T. L. Magnanti and R. T. Wong, Network design and transportation planning: models and algorithms, *Transp. Sci.* **18** (1984) 1-55.

[40] Z. A. Melzak, On the problem of Steiner, *Canad. Math. Bull.* **4** (1961) 143-148.

[41] A. Prodon, T. M. Liebling and H. Groflin, Steiner's problem on two-trees, Technical Report RO 850315, Dept. de Math. Ecole Polytechnique Federale de Lausanne, Suisse (1985).

[42] M. L. Shore, L. R. Foulds and P. B. Gibbons, An algorithm for the Steiner problem in graphs, *Networks* **12** (1982) 323-333.

[43] R. E. Tarjan, Finding optimum branchings, *Networks* **7** (1977) 25-35.

[44] R. E. Tarjan, *Data Structures and Network Algorithms*, SIAM (1983).

[45] R. T. Wong, A dual ascent approach for Steiner tree problem on a directed graph, *Math. Program.* **28** (1984) 271-287.

[46] Y. Y. Yang and O. Wing, An algorithm for the wiring problem, *Digest of the IEEE Int. Symp. on Electrical Networks* (1971) 14-15.

This Page Intentionally Left Blank

Chapter 4

Heuristics

As already mentioned in Chapter 1, the decision version of the Steiner tree problem is NP-complete. Consequently, no polynomial time algorithm for the Steiner tree problem is likely to exist. In view of this inherent intractability of the problem, it is of practical importance to develop heuristics that quickly find low-length trees spanning the set of terminals N. In this chapter available heuristics for the Steiner tree problem are reviewed. Section 4.1 covers path heuristics. They gradually add appropriately chosen paths between a tree constructed so far and terminals not yet in the tree. Tree heuristics are described in Section 4.2. They start with a tree spanning all terminals. Various strategies are then applied to obtain a shorter tree. Vertex heuristics are discussed in Section 4.3. Their common characteristic is that they select a subset of "good" non-terminals. Once they are selected, a minimum spanning tree of the subnetwork induced by terminals and selected non-terminals yields a suboptimal solution. Heuristics that do not fall into any of the above three classes are discussed in the subsequent sections: contraction heuristic in Section 4.4, dual ascent heuristic in Section 4.5 and set covering heuristic in Section 4.6. Finally, computational experience and comparison of heuristics is mentioned in Section 4.7.

4.1 Path Heuristics

Several heuristics for the Steiner tree problem can be characterized as path heuristics. Starting from an arbitrarily chosen terminal (or some other subnetwork of G), the tree is gradually grown until it spans all terminals. The expansion is typically based on the addition of (shortest) paths between vertices already in the tree and terminals not yet in the tree.

4.1.1 Shortest Paths Heuristic

Takahashi and Matsuyama [32] suggested a heuristic for the Steiner tree problem related to Prim's minimum spanning tree algorithm [27]: when a partial tree containing a subset N_k of terminals has been built up, an appropriately chosen

terminal $z_{k+1} \notin N_k$ is connected to it by a shortest path. More specifically, the ***shortest paths heuristic*** (SPH) is as follows.

- **Step 1:** Begin with a subtree T_{SPH} of G consisting of a single, arbitrarily chosen, terminal z_1 (Fig. 4.1a). $k = 1$.
- **Step 2:** If $k = n$, then **Stop**.
- **Step 3:** Determine a terminal $z_{k+1} \notin T_{SPH}$ closest to T_{SPH} (ties are broken arbitrarily). Add to T_{SPH} a shortest path joining it with z_{k+1} (Fig. 4.1b and Fig. 4.1c). $k = k + 1$. Go to **Step 2**.

As noted by Rayward-Smith and Clare [29], T_{SPH} can be further improved by two additional steps.

- **Step 4:** Determine a minimum spanning tree for the subnetwork of G induced by the vertices in T_{SPH} (Fig. 4.1d).
- **Step 5:** Delete from this minimum spanning tree non-terminals of degree 1 (one at a time). The resulting tree is the (suboptimal) solution. **Stop**.

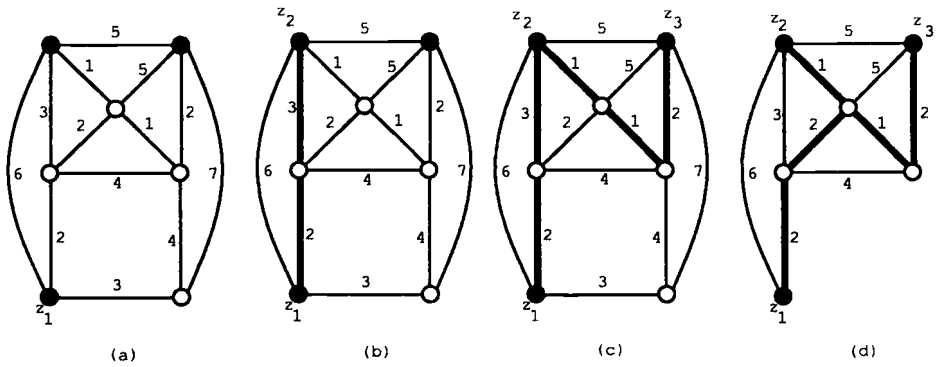


Figure 4.1: Shortest paths heuristic

The shortest paths heuristic can be implemented by a straightforward modification of Prim's algorithm for minimum spanning trees [27], given shortest paths from every terminal to all other vertices. Furthermore, the determination of these shortest paths is the bottleneck of the heuristic. Consequently, the shortest paths heuristic requires $O(nv^2)$ time, since shortest paths from each terminal to all other vertices can be determined by for example Dijkstra's algorithm [8] in $O(v^2)$ time. Alternatively, use of Fibonacci heaps to compute shortest paths [12] reduces the worst-case time complexity of the shortest paths heuristic for sparse networks to $O(n(e + v \log v))$.

A very important issue associated with heuristics for the Steiner tree problem is how bad can suboptimal solutions be in comparison with optimal solutions. Let $T_G(N)$ denote a Steiner minimal tree for N in G. Let L denote a walk around $T_G(N)$ traversing each edge exactly twice. One can regard L as composed of n ***terminal paths***, each connecting a terminal to a "next" terminal (Fig. 4.2). The

longest terminal path P^* of L must have length at least $2|T_G(N)|/n$. When P^* is deleted from L, a walk L' with $|L'| \leq |T_G(N)|(2-2/n)$ is obtained.

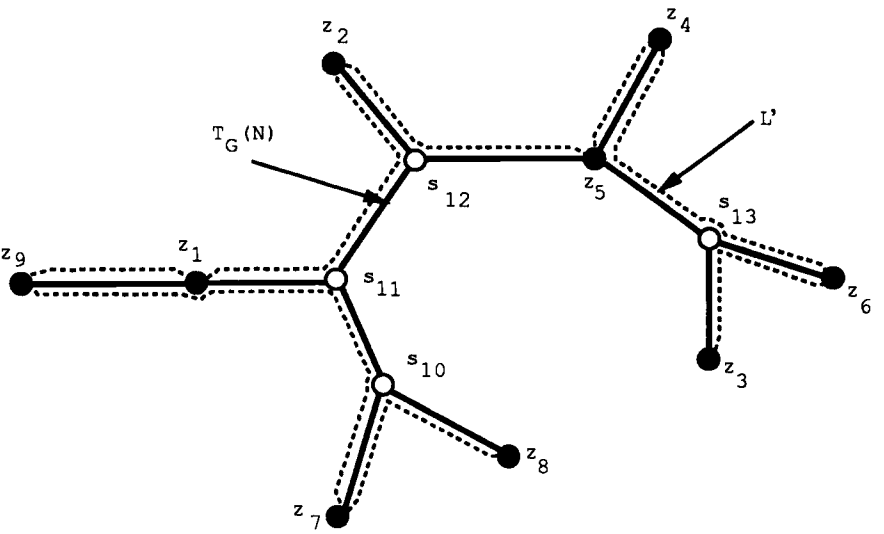


Figure 4.2: Walk L' around $T_G(N)$

Theorem 4.1 ([32]) *$|T_{SPH}|/|T_G(N)| \leq 2-2/n$ for any network G and any set of terminals N. Furthermore, this bound is tight, i.e., for any ε, $\varepsilon > 0$, there is an instance of the Steiner tree problem such that $|T_{SPH}|/|T_G(N)| > 2-\varepsilon$.*

Proof. In order to prove the first part of the theorem, one needs to show that $|T_{SPH}| \leq |L'|$. Let $z_1, z_2, ..., z_n$ be the ordering in which the terminals are selected by the shortest paths heuristic. Let P_k, $k = 2, 3, ..., n$, denote the shortest paths used to construct T_{SPH}. It will be shown that to each P_k corresponds a different, not shorter terminal path in L'. For each terminal z_k, $2 \leq k \leq n$:

- traverse L' backward (counterclockwise) beginning at z_k until reaching the first terminal path, denoted by B_k, from a terminal z_{k_b}, $k_b < k$ to a terminal z_i, $i \geq k$. Let $k_b = 0$ if B_k does not exist.
- traverse L' forward (clockwise) beginning at z_k until reaching the first terminal path, denoted by F_k, from a terminal z_j, $j \geq k$, to a terminal z_{k_f}, $k_f < k$. Let $k_f = 0$ if F_k does not exist.

Note that at least one of the paths B_k and F_k exists. Let L_k denote B_k if $k_b > k_f$ and F_k otherwise. Referring to Fig. 4.2 where $P^* = \{z_3, s_{13}, z_5, s_{12}, s_{11}, s_{10}, z_8\}$:

$$B_2 = \{z_1, s_{11}, s_{12}, z_2\}, F_2 = \emptyset, L_2 = B_2$$

$$B_3 = \{z_2, s_{12}, z_5\}, F_3 = \emptyset, L_3 = B_3$$

$$B_4 = \{z_2, s_{12}, z_5\}, F_4 = \{z_6, s_{13}, z_3\}, L_4 = F_4$$

$$B_5 = \{z_2, s_{12}, z_5\}, F_5 = \{z_5, z_4\}, L_5 = F_5$$
$$B_6 = \{z_4, z_5, s_{13}, z_6\}, F_6 = \{z_6, s_{13}, z_3\}, L_6 = B_6$$
$$B_7 = \emptyset, F_7 = \{z_9, z_1\}, L_7 = F_7$$
$$B_8 = \emptyset, F_8 = \{z_8, s_{10}, z_7\}, L_8 = F_8$$
$$B_9 = \{z_7, s_{10}, s_{11}, z_1, z_9\}, F_9 = \{z_9, z_1\}, L_9 = B_9$$

Let k and k' be two distinct integers satisfying $2 \leq k < k' \leq n$. Suppose that z_k precedes $z_{k'}$ on L'. The opposite case is proved in similar manner. Assume first that all terminals between z_k and z'_k on L' have indices higher than k. Then $B_{k'}$ begins somewhere between z_k and $z_{k'}$. Furthermore, B_k either does not exist or begins at a terminal preceding z_k. Hence $B_k \neq B_{k'}$. There are three possibilities for $F_{k'}$:

- $F_{k'}$ does not exist. Then F_k does not exist. Hence, $L_k = B_k$ and $L_{k'} = B_{k'}$. As already remarked, $B_k \neq B_{k'}$. Thus, $L_k \neq L_{k'}$.
- $F_{k'}$ ends in a terminal with index $k'_f > k$. F_k either does not exist, or ends in a terminal with index k_f satisfying $k'_f > k > k_f$. So both $B_k \neq B_{k'}$ and $F_k \neq F_{k'}$, implying that $L_k \neq L_{k'}$.
- $F_{k'}$ ends in a terminal with index $k'_f < k$. In this case, $F_k = F_{k'}$. However, $k'_b > k$ implying that $L_{k'} = B_{k'}$. Since $B_{k'} \neq B_k$ and $B_{k'} \neq F_k$, it follows again that $L_k \neq L_{k'}$.

Assume next that there is at least one terminal between z_k and $z_{k'}$ with index less than k. Then it follows immediately that $B_k, F_k, B_{k'}, F_{k'}$ are all distinct. Consequently, $L_k \neq L_{k'}$.

So far it has been proved that the paths $L_2, L_3, ..., L_n$ are mutually distinct. Each path L_k, $k = 2, 3, ..., n$, goes from a terminal with index less than k to a terminal with index greater than or equal to k. Consequently, L_k is not shorter than the path to z_k selected by the shortest paths heuristic.

It remains to show that the bound is tight. Consider the network shown in Fig. 4.3. Its n terminals form a path with edges of length 2. Its only non-terminal is adjacent to all terminals. All edges incident with the non-terminal have length $1 + \delta$, $\delta > 0$. T_{SPH} is the path through all terminals and $|T_{SPH}| = 2n - 2$. For sufficiently small δ, $T_G(N)$ consists of the edges incident with the non-terminal and $|T_G(N)| = n + n\delta$. As δ goes to 0, the ratio goes to $2 - 2/n$. For sufficiently large n, the ratio will be greater than $2 - \varepsilon$. □

4.1.2 Repetitive Shortest Paths Heuristics

The shortest paths heuristic is sensitive to the choice of the initial vertex from which the solution is constructed. Thus, it can be worthwhile to apply the shortest paths heuristic more than once, each time beginning with a different initial vertex (or subnetwork). Several such repetitive strategies have been investigated by Winter and Smith [39].

- SPH-N: determine T_{SPH} n times, each time beginning with a different terminal.
- SPH-V: determine T_{SPH} v times, each time beginning with a different vertex.

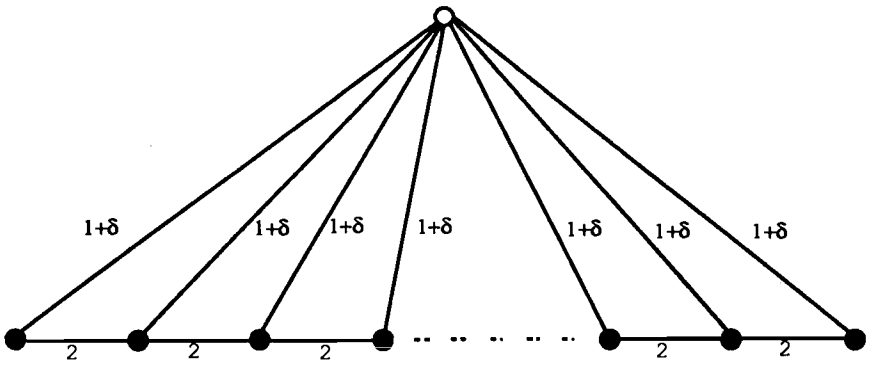


Figure 4.3: Tightness of the error ratio bound of T_{SPH}

- SPH-zN: determine T_{SPH} $n-1$ times, each time beginning with a shortest path from a fixed terminal z_1 to a terminal z_i, $i = 2, 3, ..., n$.
- SPH-NN: determine T_{SPH} $n(n-1)/2$ times, each time beginning with a shortest path between a different pair of terminals.

The repetitive applications of the shortest paths heuristic will yield better solutions. In particular, the SPH-V and SPH-NN variants seem to perform well. The improvement in terms of quality of obtained solutions is of course paid for by longer computational times. However, the repetitive schemes become much more attractive if the reductions described in Chapter 2 are applied first.

4.1.3 Shortest Paths with Origin Heuristic

Takahashi and Matsuyama [32] also studied the *shortest paths with origin heuristic* (SPOH). T_{SPOH} consists of the shortest paths from an arbitrarily chosen terminal z_i to all other terminals. The worst-case time complexity of this heuristic is $O(e + v \log v)$ using Fibonacci heaps. The worst-case error ratio $|T_{SPOH}|/|T_G(N)|$ is tightly bounded by $n-1$. Hence, the shortest paths with origin heuristic can be considered as inferior to the shortest paths heuristic also in the worst-case sense. It performs poorly on average.

4.1.4 Kruskal-Based Heuristic

The shortest paths heuristic (Subsection 4.1.1) is closely related to Prim's algorithm for minimmum spanning trees [27]. Wang [35] suggested a heuristic which is closely related to Kruskal's algorithm for minimum spanning trees. The *Kruskal-based heuristic* (KBH) is as follows.

- **Step 1:** Begin with a forest T_{KBH} consisting of all isolated terminals.
- **Step 2:** If T_{KBH} is connected, then **Stop**.
- **Step 3:** Find two trees in T_{KBH} closest to each other (ties are broken arbitrarily). Add to T_{KBH} a shortest path connecting those two trees. Go to **Step 2**.

The worst-case time complexity of this heuristic is $O(nv^2)$. Widmayer [37] showed that $|T_{KBH}|/|T_G(N)| \leq 2 - 2/n$. Plesník [25] showed that this bound is tight in the same sense as the bound for the error ratio of T_{SPH}.

4.1.5 Y-Heuristic

A variant of the shortest paths heuristic was suggested by Chen [5]. It will be referred to as the *Y-heuristic* (YH). Rather than begin with an arbitrary terminal, Steiner minimal trees for all triplets of terminals are determined. Such Steiner minimal trees consist of at most one non-terminal of degree 3 from which three paths to the terminals leave. The topology of such Steiner minimal trees can be represented by the letter Y. Hence the name of the heuristic. The shortest Steiner minimal tree taken over all triples of terminals is used as a starting tree (instead of an arbitrary terminal chosen by the shortest paths heuristic).

The expansion phase of the Y-heuristic is also slightly different than in the shortest paths heuristic. Suppose that a tree T_k spanning k terminals, $3 \leq k < n$, has been constructed. A terminal $z_{k+1} \notin T_k$ closest to T_k is determined. Let $v_{k+1} \in T_k$ denote the other end-vertex of the corresponding shortest path. Let q denote the degree of v_{k+1} in T_k. T_k contains q edge-disjoint elementary paths, all beginning at v_{k+1} and with intermediate non-terminals (if any) of degree 2. Each such elementary path is removed in turn. This splits T_k into two components (temporarily contracted to two terminals). They are reconnected together with z_{k+1} using the Steiner minimal tree construction for triplets. The shortest among the q trees generated is selected as the tree T_{k+1} spanning $k+1$ terminals to be used in the next iteration.

Chen [5] showed that a Steiner minimal tree for a triplet of terminals can be determined in $O(e \log v)$ time. Furthermore, a shortest Steiner minimal tree for all $O(n^3)$ triplets of terminals can be found in $O(ne \log v)$ time. Each iteration of the Y-heuristic involves the determination of $O(n)$ Steiner minimal trees for triplets of vertices. Before a shortest among these $O(n)$ Steiner minimal trees is determined, up to $O(v)$ edges will be temporarily contracted or deleted from G in $O(e)$ time. Hence, the k-th iteration requires $O(ne \log v + e) + O(e + v \log v)$ time where the second term is the time needed to identify z_{k+1}. This reduces to $O(ne \log v)$. There are $n-3$ iterations. Hence, the Y-heuristic requires in total $O(n^2 e \log v)$ time. Not surprisingly, on average, solutions are better than those obtained by the shortest paths heuristic. Widmayer [38] showed that $|T_{YH}|/|T_G(N)| \leq 2 - 2/n$. Plesník [25] showed that this bound is tight in the same sense as the bound for the error ratio of T_{SPH}.

4.2 Tree Heuristics

Another class of heuristics is based on the idea of constructing a tree spanning all terminals. Usually a variant of the minimum spanning tree algorithm is used to obtain this initial tree. Once given, various strategies to improve it can be applied.

4.2.1 Minimum Spanning Tree Heuristic

In the *minimum spanning tree heuristic* (MSTH) suggested by Takahashi and Matsuyama [32], the solution T_{MSTH} is obtained by deleting from the minimum spanning tree for G non-terminals of degree 1 (one at a time). The worst-case time complexity of the minimum spanning tree heuristic is $O(e+v\log v)$. The worst-case error ratio $|T_{MSTH}|/|T_G(N)|$ is tightly bounded by $v-n+1$. Hence, the minimum spanning tree heuristic can be considered as inferior to the shortest paths heuristic in the worst-case sense. It also performs poorly on average.

4.2.2 Greedy Tree Heuristic

Minoux [23] suggested a *greedy tree heuristic* (GTH). It constructs various minimum spanning trees of subnetworks of G induced by appropriate supersets of N. In order to ensure existence of such minimum spanning trees, G should be complete. If this is not the case, G can be transformed into a complete network G^* by the addition of infinite length edges as explained in Subsection 1.4.2. Alternatively, the problem can be solved in the distance network $D = D_G(V)$ rather than in G. However, this will involve the determination of all shortest paths. The heuristic is as follows.

- **Step 1:** Let $k = 0$ and $N_k = N$. Let T_k denote a minimum spanning tree of the subnetwork of G^* induced by N_k.
- **Step 2:** If $N_k = V$, then **Stop**. Otherwise, determine minimum spanning trees $T_k^{v_i}$ of subnetworks of G^* induced by $N_k \cup v_i$, $v_i \notin N_k$. Let $T_k^{v_j}$ denote the shortest of these minimum spanning trees.
- **Step 3:** If $|T_k| \leq |T_k^{v_j}|$, then let $T_{GTH} = T_k$ and **Stop**. Otherwise, $N_{k+1} = N_k \cup v_j$, $T_{k+1} = T_k^{v_j}$, $k = k+1$, and go to **Step 2**.

The number of iterations is $O(v-n)$. The number of minimum spanning trees determined during each iteration is also $O(v-n)$. These minimum spanning trees do not need to be determined from scratch. In fact, given a minimum spanning tree of a subnetwork of G^* induced by a vertex set N_k, $0 \leq k < v-n$, any minimum spanning tree of a subnetwork of G^* induced by a vertex set $N_k \cup v_i$, $v_i \notin N_k$, can be determined in $O(|N_k|)$ time [23]. It follows that the complexity of the greedy tree heuristic is $O((v-n)^2 v + n^2)$. If the greedy tree heuristic is solved in D, additional $O(v^3)$ time is required to determine all shortest paths. Note that the heuristic can completely fail to find a solution if T_0 contains infinite length edges. Even when solved in D, the heuristic can halt with T_0 as T_{GTH}. Hence, its worst-case error ratio is $v-n+1$.

Minoux [23] noticed that the efficiency of the greedy tree heuristic can be considerably improved if each pair of non-terminals is non-adjacent in the network G. Let $N_0 \subset N_1 \subset \ldots \subset N_k$, $0 \leq k < v-n$, be a sequence of subsets generated by the heuristic. Let $v_i \notin N_k$. Let $T_0^{v_i}, T_1^{v_i}, \ldots, T_k^{v_i}$ denote minimum spanning trees induced by $N_j \cup v_i$, $j = 0, 1, \ldots, k$. It can be shown [23] that

$$|T_0^{v_i}| - |T_0| \leq |T_1^{v_i}| - |T_1| \leq \ldots \leq |T_k^{v_i}| - |T_k|$$

The modified heuristic is then as follows.

- **Step 1:** Let $k = 0$ and $N_k = N$. Let T_0 denote a minimum spanning tree of the subnetwork of G^* induced by N_k.
- **Step 2:** If $N_k = V$, then **Stop**. Otherwise determine minimum spanning trees T^{v_i} of subnetworks of G induced by $N_k \cup v_i$, $v_i \notin N_k$. Place these trees on a queue Q in non-decreasing order of their length.
- **Step 3:** Let T^{v_j} denote the first minimum spanning tree on Q. Remove T^{v_j} from Q.
- **Step 4:** If T^{v_j} does not span $N_k \cup v_j$, then redefine T^{v_j} to be the minimum spanning tree of the subnetwork of G induced by $N_k \cup v_j$, add it to Q (preserving the non-decreasing order), and go to **Step 2**.
- **Step 5:** If $|T_k| \leq |T^{v_j}|$, then let $T_{GTH} = T_k$ and **Stop**. Otherwise, $N_{k+1} = N_k \cup v_j$, $T_{k+1} = T^{v_j}$, $k = k + 1$, and go to **Step 2**.

Computational experience reported by Minoux [23] indicates that the number of minimum spanning trees generated by the modified version is reduced approximately 4 times. However, the worst-case time complexity of the modified heuristic remains unchanged.

4.2.3 Distance Network Heuristic

The *distance network heuristic* (DNH) was suggested independently by Choukhmane [6], Kou, Markowsky and Berman [19], Plesník [24], and Iwainsky, Canuto, Taraszow and Villa [16]. It is as follows.

- **Step 1:** Construct the distance network $D_G(N)$ for N in G (Fig. 4.4a and Fig. 4.4b).
- **Step 2:** Determine a minimum spanning tree of $D_G(N)$ (indicated by heavy line segments in Fig. 4.4b).
- **Step 3:** Replace each edge in the minimum spanning tree by a corresponding shortest path in G (Fig. 4.4c). Let T_D denote this network. Note that shortest paths can be selected in such a way that T_D is a tree.
- **Step 4:** Determine a minimum spanning tree T_{DNH} of the subnetwork of G induced by the vertices of T_D (Fig. 4.4d).
- **Step 5:** Delete from T_{DNH} non-terminals of degree 1 (one at a time). **Stop.**

The overall worst-case time complexity of the distance network heuristic is $O(nv^2)$. It is Step 1 that dominates all the remaining steps. It involves solving n shortest path problems. The bound can be reduced for sparse networks to $O(n(e + v \log v))$ if Fibonacci heaps are used [12].

Theorem 4.2 *$|T_{DNH}|/|T_G(N)| \leq 2 - 2/n$ for any network G and any set N of terminals. Furthermore, this bound is tight, i.e., for any ε, $\varepsilon > 0$, there is an instance of the Steiner tree problem such that $|T_{DNH}|/|T_G(N)| > 2 - \varepsilon$.*

Proof. Let L be the walk around $T_G(N)$ as defined in Subsection 4.1.1. One may regard L as composed of at most n leaf-connecting paths. If the longest of these paths is deleted from L, a walk L'' such that $|L''| \leq |T_G(N)|(2 - 2/n)$ is obtained. On the other hand, $|T_{DNH}| \leq |T_D| \leq |L''|$, completing the proof of the first part of the theorem.

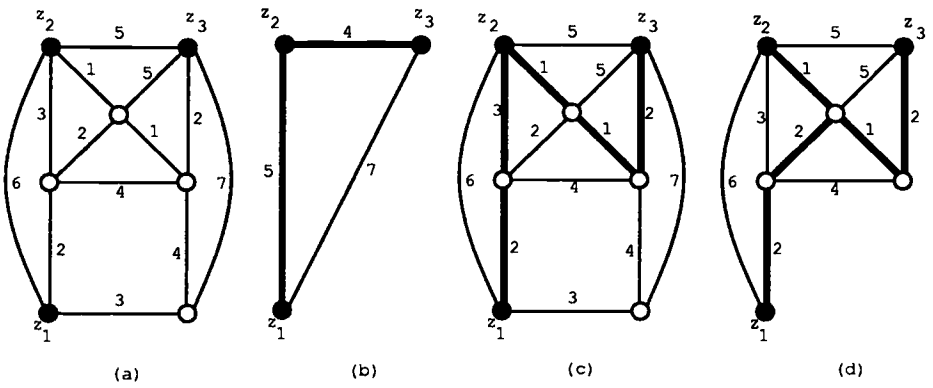


Figure 4.4: Distance network heuristic

The second part follows immediately using the same networks as when proving the tightness of the bound for the shortest paths heuristic in Subsection 4.1.1. □

Goemans and Bertsimas [15,14] proved a stronger result by showing that $|T_G(N)|$ in Theorem 4.2 can be replaced by $v(\mathcal{LP}_4)$; the optimal solution value of the linear relaxation of the mathematical programming formulation $\mathcal{P}_4$ of the Steiner tree problem (see Subsection 3.6.4).

The error bound for the distance network heuristic and for the shortest paths heuristic is the same. However, the latter often (but not always) produces better solutions. For instance, T_{SPH} indicated in Fig. 4.5a by heavy line segments has length 15 (if either z_1 or z_2 is chosen as the initial terminal) while T_{DNH} in Fig. 4.5b has length 16. On the other hand, T_{SPH} in Fig. 4.5c has length 45 while T_{DNH} in Fig. 4.5d has length 39.

It has been shown by Kucera, Marchetti-Spaccamela, Protasi and Talamo [21] that the distance network heuristic is nearly optimal for random graphs (i.e., networks where an edge of unit length is present with probability p). In that case the expected length of optimal solutions is approximately $n \log v / \log(vp)$. But even one of the most trivial heuristics, namely the shortest paths with origin heuristic (Subsection 4.1.3) is also nearly optimal for random graphs.

The inferiority of the solutions obtained by the distance network heuristic when compared with the shortest paths heuristic, together with the fact that their worst-case error ratios and worst-case time complexities are comparable, makes the distance network heuristic a poor choice among the heuristics for the Steiner tree problem. However, as will be seen in the remainder of this subsection, the worst-case time complexity of the distance network heuristic can be reduced substantially.

Wu, Widmayer and Wong [41], and independently Wang [35] suggested a method of determining T_D directly from G without explicit determination of the distance network $D_G(N)$. The construction of $D_G(N)$ is avoided by simultaneously growing shortest path trees from each terminal. During each iteration, the shortest, not yet

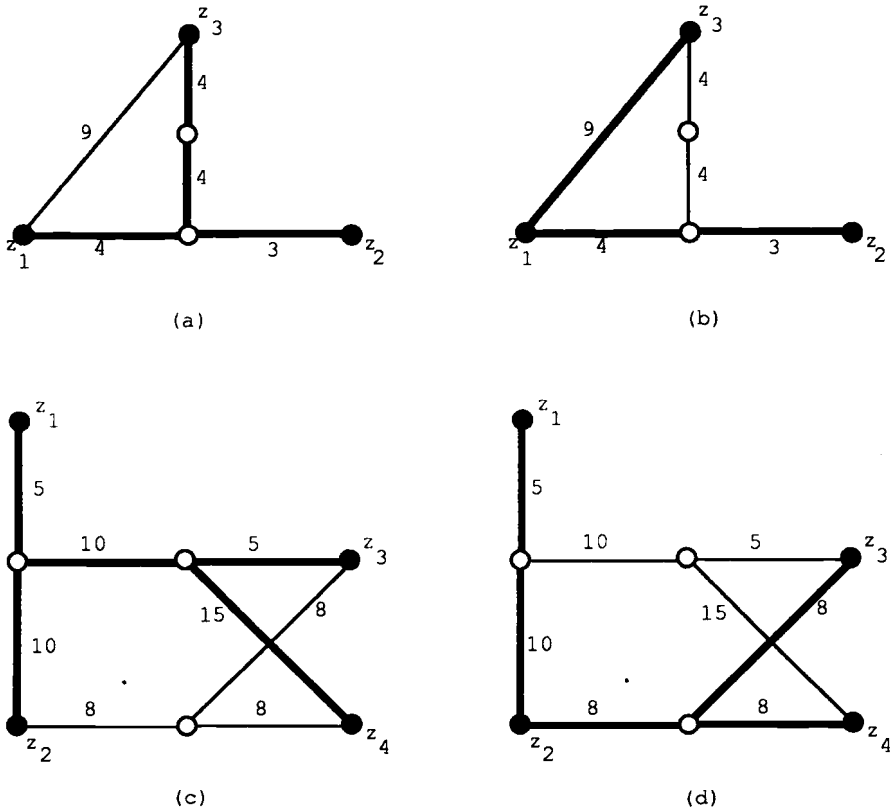


Figure 4.5: T_{SPH} outperforming T_{DNH} and vice versa

included, edge incident to one of the shortest path trees is added. When two trees meet, a path between their roots is formed (corresponding to a path in T_D). Two joined shortest path trees are restructured to a single tree with one root, and the expansion continues (until only one tree is left). The shortest path trees computation requires $O(e \log v)$ time. This is an improvement over the bound $O(nv^2)$ of the distance network heuristic, unless the network is dense and has few terminals. Unfortunately, the penalty is the need of rather complex data structures.

Further worst-case time complexity reduction when determining T_D, based on even more complex data structures [12,13] was given by Widmayer [37]. The worst-case time complexity of the distance network heuristic is then $O(e + (v + \min\{e, n^2\}) \log v)$. Recently, Kou [17] provided yet another algorithm for T_D. In the worst-case, his algorithm runs in $O(e + n \log n + (v - n) \log(v - n))$ time. If G is sparse, the time complexity reduces to $O((v-n)\log(v-n) + q \log \beta(q, n))$ where $q = \min\{e, (n-1)^2/2\}$ and $\beta(x, y) = \min\{i | \log^i y \leq x/y\}$.

Mehlhorn [22] suggested yet another modification of the distance network heuristic. Furthermore, this modification requires very simple data structures and has

superior worst-case time complexity. T_D is determined not from the distance network $D_G(N)$, but from another, easier to find, network G'. More specifically, G' is obtained as follows. Let $N(z_i)$, $z_i \in N$, denote non-terminals of G closer to z_i than to any other terminal (Fig. 4.6a). Consider the network $G' = (N, E', c')$ where

$$E' = \{(z_i, z_j) \mid \exists (v_k, v_l) \in E : v_k \in N(z_i), v_l \in N(z_j)\}$$

and the length $c'(z_i, z_j)$ of the edge (z_i, z_j) in G' is

$$\min\{d(z_i, v_k) + c(v_k, v_l) + d(v_l, z_j) \mid (v_k, v_l) \in E, v_k \in N(z_i), v_l \in N(z_j)\}$$

The network G' obtained from G (Fig. 4.6a) is shown in Fig. 4.6b.

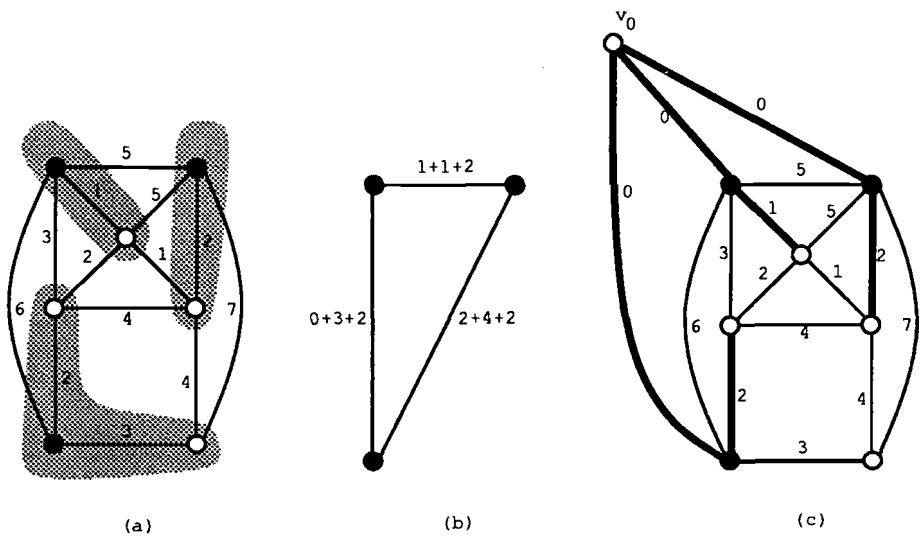


Figure 4.6: Neighborhoods of terminals

The neighborhoods $N(z_i)$, $z_i \in N$, can be found by a single shortest path computation. Add an artificial vertex v_0 to G, connect it by zero-length edges to all terminals. Find the shortest paths tree with v_0 as the root (Fig. 4.6c). When v_0 is deleted from this tree, each of the resulting components contains exactly one terminal; its non-terminals belong to the neighborhood of the terminal. All this requires $O(e + v \log v)$ time.

Theorem 4.3 ([22]) *Every minimum spanning tree of G' is also a minimum spanning tree of $D_G(N)$.*

A straightforward but technical proof is omitted. From Theorem 4.3 follows that Step 1 and Step 2 of the distance network heuristic can be replaced by the determination of a minimum spanning tree $T_{G'}(N)$. This requires $O(e + v \log v)$ time. This term still dominates the worst-case time complexity of the remaining

three steps. Floren [10] observed that when replacing the edges of $T_{G'}(N)$ by the corresponding paths in G, the resulting network will always be a tree. So in principle Step 4 and Step 5 can be avoided. However, if applied, these steps can result in a better solution. This is for example the case for the problem instance shown in Fig. 4.6.

4.2.4 Repetitive Distance Network Heuristic

Plesník [24] and independently Sullivan [31] suggested a modification of the distance network heuristic which in general yields better solutions but has higher worst-case time complexity bounds. For a given q, $0 \leq q \leq n-2$, form distance networks $D_G(Q \cup N)$ for each $Q \subseteq V \setminus N$ with $|Q| \leq q$. Use the distance network heuristic for each of these networks and take as T_{DNH} the one with minimum length. This repetitive heuristic is stronger than the distance network heuristic. As shown by Sullivan [31], the worst-case error ratio tends to $2 - q/(n-2)$ which is better than $2 - 2/n$ for $q \geq 2$. On the other hand, the worst-case time complexity of this heuristic is $O((v-n)^q n^2 + v^3)$. Note that for $q = n-2$, the heuristic is guaranteed to obtain a Steiner minimal tree. It is then identical to the enumeration algorithm described in Section 3.1.

4.2.5 Multiple Distance Network Heuristic

Diané and Plesník [7] suggested the following *multiple distance network heuristic* (MDNH). Let $T_{DNH}^{v_k}$ denote T_{DNH} for $N \cup v_k$, $v_k \notin N$. Let $T_{DNH}^{\leq v_k}$ denote T_{DNH} for $N \cup \{v_i \notin N \mid |T_{DNH}^{v_i}| \leq |T_{DNH}^{v_k}|\}$. Select as T_{MDNH} the shortest tree from among T_{DNH} and $T_{DNH}^{\leq v_k}$, $v_k \notin N$.

This heuristic performs at least as well as the distance network heuristic. Diané and Plesník [7] showed that $|T_{MDNH}|/|T_G(N)| \leq 2 - 2/n$. The worst-case time complexity of the heuristic is $O(ev + v^2 \log v)$.

4.2.6 Simulated Annealing Heuristics

The reader is assumed to be familiar with the general framework of the simulated annealing algorithm. It will be only briefly discussed below. The reader is referred to e.g., van Laarhoven and Aarts [33] for theoretical and practical aspects of this algorithm.

The simulated annealing algorithm starts with a feasible solution T at some initial *temperature* t. Another feasible solution T' in the *neighborhood* of T is selected. If T' is better than T, T' replaces T in the next iteration. Otherwise, T' replaces T with probability $e^{-(|T'|-|T|)/t}$. After a certain number $n(t)$ of iterations, the temperature is appropriately reduced. If the neighborhood structure is appropriately chosen (i.e., every feasible solution can reach any other feasible solution in a finite number of neighborhood transformations) and the cooling schedule reduces t at an appropriate rate (with the initial t sufficiently large), then it can be shown that the simulated annealing algorithm obtains a global optimum with probability

converging to 1 as the number of iterations goes to ∞. A *simulated annealing heuristic* (SAH) is derived from the algorithm by fixing the number of iterations.

Schiemangk [30] suggested a simulated annealing heuristic for the Steiner arborescence problem. Its counterpart for the Steiner tree problem is outlined below. A solution T is feasible if T is a tree spanning all terminals and where all non-terminals have degree at least 2. A neighbor of a feasible solution T is any tree T' that can be obtained from T by

- removing an edge e from T,
- reconnecting two components of $T - e$ by a path with minimum number of edges,
- removing non-terminals of degree 1 (one at a time).

Dowsland [9] suggested another simulated annealing heuristic for the Steiner tree problem. Feasible solutions are again trees spanning all terminals (with no non-terminals of degree 1). The neighborhood structure is however more elaborate. Current feasible solution T is broken into two subtrees T_i and T_j by removal of a randomly chosen elementary path (i.e., a path with end-vertices being either terminals or non-terminals of degree more than 2, and with intermediate vertices being non-terminals of degree 2). Three vertices are then selected at random: v_i belonging to T_i, v_j belonging to T_j and v_s belonging to non-terminals not in T extended by an artificial vertex v_0. If $v_s = v_0$, then T_i and T_j are reconnected by a shortest path from v_i to v_j. Otherwise, T_i and T_j are reconnected by shortest paths from v_i to v_s and from v_j to v_s. If these two shortest paths overlap before reaching v_s, only their disjoint parts are included in the new feasible solution. Dowsland [9] showed that the neighborhood structure defined in this way is sound (i.e., a Steiner minimal tree can be reached by a finite number of transformation from any feasible solution). In order to reduce the number of transformations, some of the reductions described in Chapter 2 are incorporated into the heuristic (i.e., some elementary paths are never chosen as they belong to at least one Steiner minimal tree, and some shortest paths are never used to interconnect subtrees as they do not belong to any Steiner minimal tree).

4.2.7 Hill-Climbing Heuristics

Dowsland [9] also suggested three *hill-climbing heuristics* (HCH). The first of them examines each elementary path in the current feasible solution T. Each elementary path is removed (one at a time) and the two subtrees are reconnected by a shortest possible path. T is replaced by the shortest tree found in this way. The heuristic stops when no improvements are possible.

The second hill-climbing heuristic examines all local transformations based on the sequential removal of two elementary paths followed by an appropriate reconnection scheme.

The third variant uses hybrid approach. The first hill-climbing heuristic is applied. Then one iteration of the second hill-climbing heuristic is applied. If this results in any improvement, the whole sequence is repeated.

4.3 Vertex Heuristics

The major difficulty when solving the Steiner tree problem is to identify non-terminals that belong to the Steiner minimal tree. Once given, the Steiner minimal tree can be found easily; it is a minimum spanning tree for the subnetwork induced by the terminals and selected non-terminals. The general idea behind vertex heuristics is to identify "good" non-terminals.

4.3.1 Average Distance Heuristic

The *average distance heuristic* (ADH) was suggested by Rayward-Smith [28]. It is based on the idea of connecting already constructed subtrees of a solution by shortest paths through some centrally located vertex. The degree of centrality of a vertex $v_i \in V$ is measured by some appropriately chosen measure $f(v_i)$ (to be defined below).

- **Step 1:** Begin with T_{ADH} consisting of the isolated terminals (Fig. 4.7a). Let $k = 1$.
- **Step 2:** If $k = n$, then **Stop.** Otherwise determine $f(v_i)$ for each $v_i \in V$, and select a vertex v_l with the smallest f-value.
- **Step 3:** Add the edges on the paths from v_l to the closest and second-closest components of T_{ADH} (Fig. 4.7b and Fig. 4.7c). $k := k + 1$. Go to **Step 2.**

Two additional steps, identical to Step 4 and Step 5 of the shortest paths heuristic, can be applied to further improve the solution.

One way of defining f during the k-th iteration, $1 \leq k \leq n-1$, is as follows [29]. Let $C_r^{v_i}$, $2 \leq r \leq n-k+1$, denote r trees in the partially constructed T_{ADH} that are closest to a vertex $v_i \in V$ (ties are broken arbitrarily). Define

$$f(v_i) = \min_r \{ \sum_{T_j \in C_r^{v_i}} d(v_i, T_j)/(r-1) \}$$

The worst-case time complexity of the average distance heuristic is dominated by the computation of shortest paths between all pair of vertices in G. Hence, its worst-case time complexity is $O(v^3)$ [11]. This bound can be reduced to $O(ve + v^2 \log v)$ for sparse networks.

Theorem 4.4 ([36]) *$|T_{ADH}|/|T_G(N)| \leq 2 - 2/l$ for any network G and any set N of terminals. Furthermore, for any ε, $\varepsilon > 0$ there exists a problem instance such that $|T_{ADH}|/|T_G(N)| > 2 - \varepsilon$.*

Proof. Let $T_D(N)$ denote the minimum spanning tree of $D_G(N)$. From Theorem 4.2 follows immediately that $|T_D(N)|/|T_G(N)| \leq 2 - 2/l$ where l is the number of leaves in $T_G(N)$. Waxman and Imase [36] proved that $|T_{ADH}| \leq |T_D(N)|$ (a tedious technical proof is omitted). The first part of the theorem follows.

In order to prove the second part, let k be any positive integer. Consider a full binary tree B_k of depth k with an additional path through the leaves (Fig. 4.8).

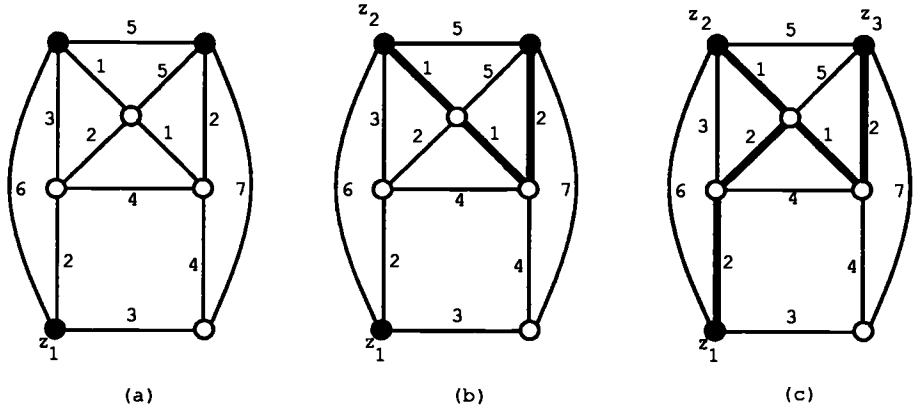


Figure 4.7: Average distance heuristic

Consider any non-leaf vertex v_i. Let h denote its height (number of edges which must be traversed to reach any leaf). Define the length of the two edges leaving v_i downward to be $2^{h-1} + \delta$, $0 < \delta \leq 1$. Define the length of the bottom edge connecting the two subtrees of v_i to be $2^{h+1} - 2$. Let N be the set of leaf-vertices.

For a fixed k, $k > 0$, T_{ADH} consists of the path through the leaves. Its length is $2(k2^k - 2^k + 1)$ On the other hand, $T_G(N)$ is the binary tree B_k. Its length is $k2^k + (2^{k+1} - 1)\delta$. Then

$$|T_{ADH}|/|T_G(N)| > \frac{2}{1 + 2\delta/k} - \frac{2}{k}$$

Given any $\varepsilon > 0$, then $|T_{ADH}|/|T_G(N)| > 2 - \varepsilon$ for any fixed δ, $0 < \delta < 1$, and sufficiently large k. □

Bern and Plassmann [4] analyzed the average distance heuristic for complete networks with edge lengths either 1 or 2. They proved that the worst-case error ratio is then tightly bounded by 4/3.

If $n = v$, the average distance heuristic reduces to the well-known minimum spanning tree algorithm of Kruskal [20]. If $n = 2$, the solution will be the minimum length path between the two terminals. Thus, in those two "boundary" cases, the average distance heuristic will yield a Steiner minimal tree.

4.3.2 Fast Average Distance Heuristic

The average distance heuristic connects two trees during each iteration. However, $f(v_l)$ can be attained for any subset $C_r^{v_l}$, $2 \leq r \leq n - k + 1$, of trees in the partially constructed T_{ADH}. It seems therefore natural to connect all trees in $C_r^{v_l}$ defining $f(v_l)$ to v_l via shortest paths. This usually reduces the number of iterations. Although this *fast average distance heuristic* (FADH) is less "cautious" than the

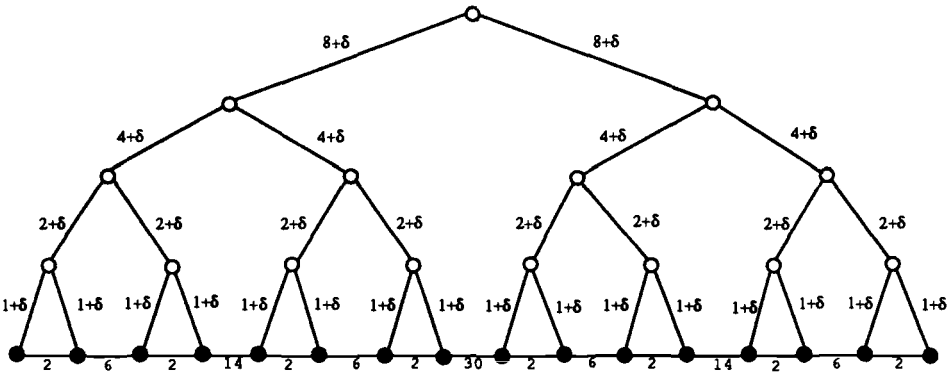


Figure 4.8: Tightness of the error ratio bound of T_{ADH}

average distance heuristic, Plesník [25] observed that this usually does not affect the quality of the solutions. He also proved that $|T_{FADH}|/|T_G(N)| \leq 2 - 2/n$ for any network G and any set N of terminals. Moreover, this bound is tight in the same sense as the bound for the error ratio of T_{ADH}. The worst-case time complexity of the fast average distance heuristic is $O(v^3)$.

4.3.3 Restricted Average Distance Heuristic

Chen [5] suggested a variant of the average distance heuristic that will be referred to as the *restricted average distance heuristic* (RADH). It joins three trees per iteration. Initially, T_{RADH} consists of n trees, each consisting of a terminal only. The selection is based on Steiner minimal trees for triplets of terminals. More specifically, let

$$f'(v_i) = \sum_{j=1}^{3} d(v_i, T_j^{v_i})$$

where $T_1^{v_i}$, $T_2^{v_i}$, $T_3^{v_i}$ are three subtrees of T_{RADH} closest to $v_i \in V$. A vertex v_l with the smallest f'-value is then selected, and the corresponding trees are joined together by a Steiner minimal tree. This is repeated as long as there are more than two subtrees left. If there are two subtrees left, they are joined by a shortest interconnecting path. The worst-case time complexity of the restricted average distance heuristic is $O(n^2 e \log v)$. Widmayer [38] showed that $|T_{RADH}|/|T_G(N)| \leq 2 - 2/n$. Plesník [25] showed that this bound is tight in the same sense as the bound for the error ratio of T_{ADH}.

4.3.4 Median Heuristic

Diané and Plesník [7] suggested the f ollowing *median heuristic* (MH). Determine

$$f''(v_i) = \sum_{z_k \in N} d(v_i, z_k)$$

for each $v_i \notin N$. Select v_l with the smallest f''-value. Consider the subnetwork of G induced by shortest paths from v_l to all terminals. Find a minimum spanning tree for this subnetwork. Remove (one at a time) non-terminals of degree one. Let $T^{v_l}_{DNH}$ denote the resulting tree. Let

$$T_{MH} = \begin{cases} T^{v_l}_{DNH} & \text{if } |T^{v_l}_{DNH}| < |T_{DNH}| \\ T_{DNH} & \text{otherwise} \end{cases}$$

Diané and Plesník [7] proved that $|T_{MH}|/|T_G(N)| \leq 2 - 2/n$. This bound is tight in the same sense as the bound for the error ratio of T_{ADH}. The worst-case time complexity of the median heuristic is $O((v-n)v^2)$.

4.3.5 Antimedian Heuristic

Diané and Plesník [7] suggested also the following *antimedian heuristic* (AMH). As in the median heuristic, determine $f''(v_i)$ for every $v_i \notin N$. Remove from G (one at a time) non-terminals in non-increasing order of their f''-values provided that the resulting network contains all terminals in the same component. Upon completion, determine a minimum spanning tree of the component containing all terminals. Delete from this tree (one at a time) non-terminals of degree one. Let T^*_{AMH} denote the resulting tree. Let

$$T_{AMH} = \begin{cases} T^*_{AMH} & \text{if } |T^*_{AMH}| < |T_{DNH}| \\ T_{DNH} & \text{otherwise} \end{cases}$$

Diané and Plesník [7] proved that $|T_{AMH}|/|T_G(N)| \leq 2 - 2/n$. This bound is tight in the same sense as the bound for the error ratio of T_{ADH}. The worst-case time complexity of the antimedian heuristic is $O((v-n)v^2)$.

4.3.6 Upgrading Heuristic

Zelikovsky [42] suggested a vertex heuristic that is very important from the theoretical point of view. It is the first polynomial heuristic with the worst-case error ratio bounded by a constant less than 2. Furthermore, it can be generalized to a heuristic for the Steiner problem in an arbitrary metric space (see Part IV, Section 1.4). This is in particular the case for the Euclidean metric (see Part I, Section 4.8) and for the rectilinear metric (see Part III, Subsection 2.2.6).

The main idea behind the heuristic is to identify some non-terminals, include them into the set of terminals, and apply the distance network heuristic to this extended set of terminals. This heuristic will be referred to as the *upgrading heuristic* (UH) since some non-terminals are "upgraded" to terminals. The upgrading heuristic is as follows.

- **Step 1:** $W = \emptyset$. $w = \infty$. $D = D_G(N)$.
- **Step 2:** Identify a set of three terminals $N_i = \{z_i, z_j, z_k\}$ in D such that

$$w = |T_D(N)| - |T_D(\bar{N})| - |T_G(N_i)|$$

is maximized and where $\bar{D}$ denotes a network obtained from D by contraction of the three edges (z_i, z_j), (z_j, z_k), (z_k, z_i) while $\bar{N}$ denotes the vertices of $\bar{D}$.

- **Step 3:** If $w = 0$, then let T_{UH} be T_{DNH} for $N \cup W$ and **Stop**. Otherwise add to W the non-terminal v_p of degree 3 in $T_G(N_i)$ (it always exists). Let $D = \bar{D}$, and go to **Step 2**.

Theorem 4.5 ([42]) $|T_{UH}|/|T_G(N)| \leq 11/6$ *for any network* G *and any set* N *of terminals.*

The complicated proof of this theorem requiring several technical lemmas is omitted. The worst-case time complexity of the heuristic is $O(v^3 + vn^3)$.

4.3.7 Modified Upgrading Heuristic

During each iteration of the upgrading heuristic one non-terminal is identified and added to the set of terminals for which T_{DNH} is determined in the final stage. The inclusion of a non-terminal is irreversible and may therefore cause omission of some other non-terminals whose impact on the final solution would be even greater. Furthermore, the identification of non-terminals upgraded to terminals is based on the determination of Steiner minimal trees for triples of original terminals in G.

Berman and Ramaiyer [2,3] suggested a *modified upgrading heuristic* (MUH). Non-terminals are not upgraded to terminals immediately. Instead, they are pushed on a stack. Only after Steiner minimal trees for all small (not only triplets) subsets of original terminals in G have been considered, final decision which non-terminals should be upgraded is made.

Let N_i denote a subset of terminals, $3 \leq |N_i| \leq k$. Let $E_i = \{(z_j, z_l) | z_j, z_l \in N_i\}$. Let

$$w = |T_D(N)| - |T_{\bar{D}}(\bar{N})| - |T_G(N_i)|$$

where $D = D_G(N)$ and $\bar{D}$ is obtained by contracting D along $|N_i| - 1$ edges of E_i. If $w > 0$, then push $T_G(N)$ and E_i on the stack. With each edge $(z_j, z_l) \in E_i$ is associated a modified length $d(z_j, z_l) - w$. Apart from pushing E_i on the stack, it is also added to D (whereby parallel edges may occur). This process is repeated for all subsets with up to k terminals.

In the second phase of the modified upgrading heuristic, pairs of Steiner minimal trees and edge sets are removed from the stack one by one. Suppose that some $T_G(N_i)$ and E_i is removed from the stack. If $T_D(N) \cap E_i$ contains $|N_i| - 1$ edges, all non-terminals of $T_G(N_i)$ are upgraded to terminals. Otherwise E_i is removed from D.

When the stack becomes empty, T_{DNH} for N and the upgraded non-terminals in G is determined and returned as T_{MUH}.

If k is fixed, Steiner minimal trees for subsets with up to k terminals can be determined in polynomial time (using e.g., the exact dynamic programming algorithm discussed in Section 3.4). Since the number of subsets is also polynomial, the modified upgrading heuristic is polynomial. A detailed time complexity analysis is omitted as the modified upgrading heuristic cannot compete in this respect with other heuristics. The importance of the modified upgrading heuristic is due to the following theorem (the complicated technical proof is omitted).

Theorem 4.6 ([2,3]) $|T_{MUH}|/|T_G(N)| \leq 16/9$ *for* $k = 4$ *and for any network* G *and any set* N *of terminals.*

4.4 Contraction Heuristic

As it was pointed out in Subsection 2.2.2, if a network G contains a terminal z_i, and the shortest edge incident to it has a terminal z_j as the other endpoint, then (z_i, z_j) belongs to the Steiner minimal tree. It then suffices to consider the Steiner tree problem arising after the contraction of G along (z_i, z_j). The *contraction heuristic* (CH) given by Plesník [24] is based on a more general idea of contraction.

In order to describe the contraction heuristic, some definitions are needed. Let $z_i \in N$ and $r > 0$ be given. A neighborhood $N_r(z_i)$ of z_i with radius r is the set $\{x \in G \mid d_G(x, z_i) \leq r\}$. Note that x does not need to be a vertex belonging to V, but any point on the edges (regarded as simple curves) of G. Two neighborhoods $N_r(z_i)$ and $N_r(z_j)$, $z_i, z_j \in N$, are said to be reachable from one another if they are adjacent or there is a path from z_i to z_j entirely within a union of neighborhoods of some terminals. A *neighborhood class* C_p is the maximal union of neighborhoods that are reachable from one another. Classes arising in connection with the network shown in Fig. 4.9 (for $r = 1$) are indicated by shaded regions.

A class contraction of a network G that produces a network $\bar{G}$ is obtained in the following way.

- Each class C_p is contracted to a single vertex z_p regarded as a terminal in $\bar{G}$. Non-terminals not belonging to any class are left unchanged in $\bar{G}$.
- An edge $e_m = (v_i, v_j)$ in G with end-vertices not belonging to the same class generates an edge e_m in $\bar{G}$ according to the following rules (Fig. 4.9b):
  - If neither v_i nor v_j belongs to any class, then the edge e_m remains unchanged in $\bar{G}$.
  - If v_i belongs to a class C_p and v_j belongs to no class (i.e., $v_j \notin N$), then $\bar{G}$ contains the edge $e_m = (z_p, v_j)$ of length

$$c(z_p, v_j) = \begin{cases} c(v_i, v_j) - r & \text{if } v_i \in N \\ c(v_i, v_j) & \text{if } v_i \notin N \end{cases}$$

  - If v_i belongs to a class C_p and v_j to another class C_q, then $\bar{G}$ contains the edge $e_m = (z_p, z_q)$ of length

$$c(z_p, z_q) = \begin{cases} c(v_i, v_j) - 2r & \text{if } v_i, v_j \in N \\ c(v_i, v_j) & \text{if } v_i, v_j \notin N \\ c(v_i, v_j) - r & \text{otherwise} \end{cases}$$

- All but the minimum length edge connecting any pair of vertices in $\bar{G}$ are deleted (Fig. 4.9c).

The recursive heuristic based on the class contractions is as follows.

- **Step 1:** Determine the minimum length edge in G incident to a terminal. Let r denote its length. Form neighborhoods $N_r(z_i)$, $z_i \in N$, and the corresponding classes C_p, $p = 1, 2, ..., t$.

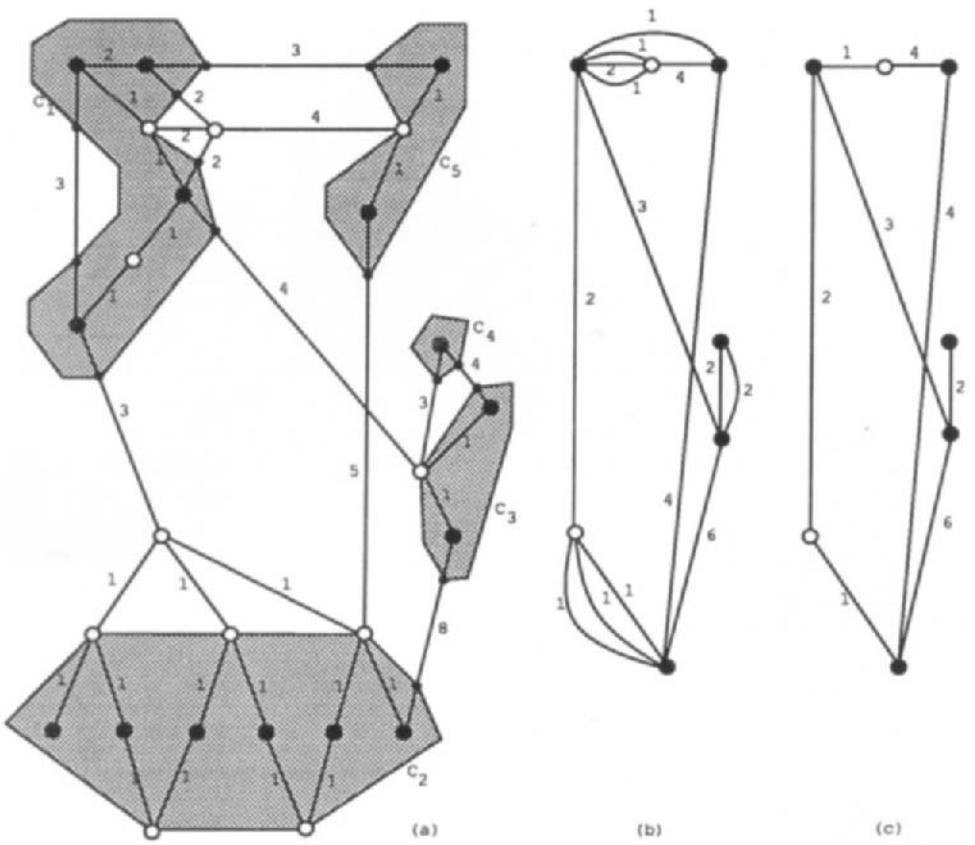


Figure 4.9: Contraction heuristic

- **Step 2:** For each class C_p, $p = 1, 2, ..., t$, let N_p denote the terminals in C_p. Note that for any terminal in N_p, there is another terminal in N_p (unless $|N_p| = 1$) not farther away than $2r$. Thus a tree T_p spanning N_p with $|T_p| \leq 2r(|N_p| - 1)$ can be determined by the shortest paths heuristic (Subsection 4.1.1).
- **Step 3:** If $t = 1$, then T_1 is a tree spanning N in G; **Return.** Otherwise, let $\bar{G}$ be the network obtained from G by class contraction. Let $\bar{N}$ denote the terminals of $\bar{G}$.
- **Step 4:** Determine a tree $\bar{T}$ spanning $\bar{N}$ in $\bar{G}$ (by recursion).
- **Step 5:** Replace each $z_p \in \bar{N}$ by T_p for $p = 1, 2, ..., t$. Reconnect $\bar{T}$ and $T_1, T_2, ..., T_t$ by adding at most $\sum_{z_p \in \bar{N}} deg_G(z_p)$ edges, each of length r; the nature of class contractions makes such reconnection always possible. The resulting tree T_{CH} is the solution. **Stop.**

None of the steps in the heuristic requires more than $O(v^2)$ time. Since at most v recursions are needed, it follows that its worst-case time complexity is $O(v^3)$.

Theorem 4.7 ([24]) $|T_{CH}|/|T_G(N)| \leq 2 - 2/n$ *for any network* G *and any set of*

terminals N. Furthermore, this bound is tight, i.e., for any ϵ, $\epsilon > 0$, there is an instance of the Steiner tree problem such that $|T_{CH}|/|T_G(N)| > 2 - \epsilon$.

Plesník [26] analyzed the contraction heuristic in more detail. In particular, he provided a more precise worst-case error ratio depending on additional parameters. He also suggested a modification of the contraction heuristic where Step 2 and Step 5 are united. While this modification has the same worst-case time complexity and error ratio, it sometimes yields better solutions.

4.5 Dual Ascent Heuristic

The dual ascent method for the determination of a lower bound for the Steiner arborescence problem was described in Section 3.7. Wong [40] suggested how this method can be used to find a suboptimal solution. The *dual ascent heuristic* (DAH) is as follows.

- **Step 1:** Determine the directed network H as described in Section 3.7. Disregard the orientation of arcs. Let Q denote the set of vertices in H connected by a path with the terminal z_1. Note that $N \subseteq Q$.
- **Step 2:** Determine a minimum spanning tree of the subnetwork of G induced by Q. Denote it by T_{DAH}.
- **Step 3:** Remove from T_{DAH} non-terminals of degree 1 (one at a time) and **Stop**.

Wong [40] did not analyze this heuristic in terms of the worst-case time complexity. Furthermore, nothing is known about the worst-case error-ratio of T_{DAH}. However, experimental results (see Section 4.7) indicate that the dual ascent heuristic yields very good suboptimal solutions.

4.6 Set Covering Heuristic

The set covering algorithm for the Steiner tree problem was described in Section 3.10. Aneja [1] suggested to use this method to obtain a good feasible solution. The *set covering heuristic* (SCH) is as follows.

- **Step 1:** Determine the optimal solution $X = (x_{ij} | [v_i, v_j] \in A)$ to the linear relaxation of the set covering formulation $\mathcal{P}_4$. Let T_{SCH} denote a subnetwork of G containing every arc $[v_i, v_j]$ with $x_{ij} > 0$.
- **Step 2:** Let $[v_i, v_j]$ denote an arc of T_{SCH} such that $T_{SCH} - [v_i, v_j]$ is connected and x_{ij} is as small as possible. If no such $[v_i, v_j]$ exists, then **Stop**.
- **Step 3:** Let $T_{SCH} = T_{SCH} - [v_i, v_j]$ and go to **Step 2**.

No worst-case time complexity bound for this heuristic is available. Nothing is known about the worst-case error ratio of T_{SCH}. Limited computational results do not indicate that the set covering heuristic yields exceptionally good suboptimal solutions.

4.7 Summary and Computational Experience

Table 4.1 gives an overview of the heuristics for the Steiner tree problem discussed in this chapter.

Comparative analysis of most heuristics described in this chapter carried out by Rayward-Smith [28] and Plesník [25,26] shows that there is no clear dominance relation between them. More specifically, Plesník [25,26] examined the following 8 heuristics: shortest paths heuristic (SPH), Kruskal-based heuristic (KBH), Y-heuristic (YH), distance network heuristic (DNH), average distance heuristic (ADH), fast average distance heuristic (FADH), restricted average distance heuristic (RADH), and contraction heuristic (CH). For any pair H_a and H_b of these heuristics (except DNH versus CH), he constructed problem instances where H_a beats H_b and problem instances where H_b beats H_a.

Rayward-Smith and Clare [29] compared the shortest paths heuristic (SPH), distance network heuristic (DNH), and average distance heuristic (ADH). Distance network heuristic turned out to be less accurate than both shortest paths and average distance heuristics for various kinds of networks. On average, the average distance heuristic performed better than the shortest paths heuristic, although there were cases where this was not the case. However, the differences between lengths of obtained solutions were not large. Similar pattern of behavior of these three heuristics was observed by Voss [34] and Winter and Smith [39].

The error ratio for all three heuristics was far below the worst-case error ratio $2 - 2/n$. However, this was observed only for very sparse networks. For larger and denser networks, the optimal solutions were not available.

The repetitive variants of the shortest paths heuristic were compared by Winter and Smith [39]. In particular, the SPH-V and SPH-NN variants turned out to perform very well. They outperformed the average distance heuristic for all kinds of tested networks. This in particular indicates that there might be some improvements available in connection with the average distance heuristic if a more suitable centrality measure f is used.

Voss [34] carried out an extensive comparison which in addition to the shortest paths heuristic (SPH), distance network heuristic (DNH) and average distance heuristic (ADH) also included Y-heuristic (YH), restricted average distance heuristic (RADH), repetitive shortest paths heuristic (SPH-zN) and dual ascent heuristic (DAH). In particular, the dual ascent heuristic seemed to perform better than other heuristics. The only exception occurred in connection with the grid networks (vertical and horizontal lines through terminals). For such networks, the repetitive shortest paths heuristic (SPH-zN) performed better. It remains to be seen whether the most efficient repetitive shortest paths heuristics (SPH-V and SPH-NN) can compete with the dual ascent heuristic on other types of networks. They performed better than the SPH-zN on grid networks [39].

Simulated annealing (SAH) and hill-climbing (HCH) heuristics seem to be poor alternatives. They typically require good starting solutions. Unless elaborate local exchanges are used, these heuristics require many iterations. Furthermore, in the case of hill-climbing heuristics, the quality of obtained solution is usually poor.

Method	Submethod	Acronym	Complexity	Error Ratio	Reference	Section
Path heuristics	shortest paths	SPH	$O(n(e + v \log v))$	$2 - 2/n$	Takahashi and Matsuyama [32]	4.1.1
	repetitive	SPH-N	$O(n^2v^2)$	$2 - 2/n$	Winter and Smith [39]	4.1.2
		SPH-V	$O(nv^3)$			
		SPH-zN	$O(n^2v^2)$			
		SPH-NN	$O(n^3v^2)$			
	with origin	SPOH	$O(e + v \log v)$	$n - 1$	Takahashi and Matsuyama [32]	4.1.3
	Kruskal	KBH	$O(nv^2)$	$2 - 2/n$	Wang [35]	4.1.4
	Y	YH	$O(n^2e \log v)$	$2 - 2/n$	Chen [5]	4.1.5
Tree heuristics	min. spanning trees	MSTH	$O(e + v \log v)$	$v - n + 1$	Takahashi and Matsuyama [32]	4.2.1
	greedy	GTH	$O((v-n)^2v + n^2)$	$v - n + 1$	Minoux [23]	4.2.2
	distance network	DNH	$O(e + v \log v)$	$2 - 2/n$	Mehlhorn [22] and [6,16,17,19,24,35,41]	4.2.3
	repetitive	RDNH	$O((v-n)^q n^2 + v^3)$	$2 - q/(n-2)$	Plesník [24] Sullivan [31]	4.2.4
	multiple	MDNH	$O(ev + v^2 \log v)$	$2 - 2/n$	Diané and Plesník [7]	4.2.5
	simulated annealing	SAH			Schiemangk [30] Dowsland [9]	4.2.6
	hill-climbing	HCH			Dowsland [9]	4.2.7
Vertex heuristics	average distance	ADH	$O(v^3)$	$2 - 2/n$	Rayward-Smith and Clare [28,29]	4.3.1
	fast	FADH	$O(v^3)$	$2 - 2/n$	Plesník [26]	4.3.2
	restricted	RADH	$O(n^2e \log v)$	$2 - 2/n$	Chen [5]	4.3.3
	median	MH	$O((v-n)v^2)$	$2 - 2/n$	Diané and Plesník [7]	4.3.4
	antimedian	AMH	$O((v-n)v^2)$	$2 - 2/n$	Diané and Plesník [7]	4.3.5
	upgrading	UH	$O(v^3 + vn^3)$	11/6	Zelikovsky [42]	4.3.6
	modified upgrading	MUH		16/9	Berman and Ramaiyer [2,3]	4.3.7
Contraction		CH	$O(v^3)$	$2 - 2/n$	Plesník [24,26]	4.4
Dual Ascent		DAH			Wong [40]	4.5
Set Covering		SCH			Aneja [1]	4.6

Table 4.1: Heuristics for the Steiner tree problem

No computational experience for the upgrading heuristic (UH) and modified upgrading heuristic (MUH) is currently available. Although these heuristics have worst-case error bounds below 2, it is uncertain whether they can produce solutions that on average are better than the solutions obtained by the best repetitive shortest paths heuristics or by the dual ascent heuristic. Furthermore, upgrading heuristics are far from efficient as their computational load is very high.

References

[1] Y. P. Aneja, An integer linear programming approach to the Steiner problem in graphs *Networks* **10** (1980) 167-178.

[2] P. Berman and V. Ramaiyer, An approximation algorithm for the Steiner tree problem, Technical Report CS-91-05, The Pennsylvania State University (1991).

[3] P. Berman and V. Ramaiyer, Improved approximations for the Steiner tree problem, Technical Report CS-91-10, The Pennsylvania State University (1991).

[4] M. W. Bern and P. Plassmann, The Steiner problem with edge lengths 1 and 2, *Inf. Process. Lett.* **32** (1989) 171-176.

[5] N. P. Chen, New algorithm for Steiner tree on graphs, *IEEE Symp. on Circuits and Systems* (1983) 1217-1219.

[6] E.-A. Choukhmane, Une heuristique pour le probleme de l'arbre de Steiner, *RAIRO Rech. Opér.* **12** (1978) 207-212.

[7] M. Diané and J. Plesník, Three new heuristics for the Steiner problem in graphs, *Acta Math. Univ. Comenianae* **60** (1991) 105-121.

[8] E. W. Dijkstra, A note on two problems in connection with graphs, *Numer. Math.* **1** (1959) 269-271.

[9] K. A. Dowsland, Hill-climbing, simulated annealing and the Steiner problem in graphs, *Eng. Optimization* **17** (1991) 91-107.

[10] R. Floren, A note on "A faster approximation algorithm for the Steiner problem in graphs", *Inf. Process. Lett.* **38** (1991) 177-178.

[11] R. W. Floyd, Algorithm 97: Shortest path, *Comm. ACM* **5** (1962) 344-345.

[12] M. L. Fredman and R. E. Tarjan, Fibonacci heaps and their uses in improved network optimization algorithms, *J. Assoc. Comput. Mach.* **34** (1987) 596-615.

[13] H. N. Gabow, Z. Galil and T. H. Spencer, Efficient implementation of graph algorithms using contraction, *J. Assoc. Comput. Mach.* **36** (1989) 540-572.

[14] M. X. Goemans, Survivable networks and parsimonious property, Technical Report, Oper. Res. Center, MIT (1990).

[15] M. X. Goemans and D. J. Bertsimas, On the parsimonious property of connectivity problems, Technical Report, Oper. Res. Center, MIT (1989).

[16] A. Iwainsky, E. Canuto, O. Taraszow and A. Villa, Network decomposition for the optimization of connection structures, *Networks* **16** (1986) 205-235.

[17] L. T. Kou, On efficient implementation of an approximation algorithm for the Steiner tree problem, *Acta Inf.* **27** (1990) 369-380.

[18] L. T. Kou and K. Makki, An even faster approximation algorithm for the Steiner tree problem in graphs, *Congr. Numerantium* **59** (1987) 147-154.

[19] L. T. Kou, G. Markowsky and L. Berman, A fast algorithm for Steiner trees, *Acta Inf.* **15** (1981) 141-145.

[20] J. B. Kruskal, On the shortest spanning subtree of a graph and the traveling salesman problem, *Proc. Am. Math. Soc.* **7** (1956) 48-50.

[21] L. Kucera, A. Marchetti-Spaccamela, M. Protasi and M. Talamo, Near optimal algorithms for finding minimum Steiner trees on random graphs, in J. Gruska (ed.) *Mathematical Foundations of Computer Science*, **LNCS 233**, Springer-Verlag (1986) 501-511.

[22] K. Mehlhorn, A faster approximation algorithm for the Steiner problem in graphs, *Inf. Process. Lett.* **27** (1988) 125-128.

[23] M. Minoux, Efficient greedy heuristics for Steiner tree problems using reoptimization and supermodularity, *INFOR* **28** (1990) 221-233.

[24] J. Plesník, A bound for the Steiner problem in graphs, *Math. Slovaca* **31** (1981) 155-163.

[25] J. Plesník, Worst-case relative performances of heuristics for the Steiner problem in graphs, *Acta Math. Univ. Comenianae* **60** (1991) 269-284.

[26] J. Plesník, On heuristics for the Steiner problem in graphs (to appear).

[27] R. C. Prim, Shortest connection networks and some generalizations, *Bell System Tech. J.* **36** (1957) 1389-1401.

[28] V. J. Rayward-Smith, The computation of nearly minimal Steiner trees in graphs, *Int. J. Math. Educ. Sci. Technol.* **14** (1983) 15-23.

[29] V. J. Rayward-Smith and A. Clare, On finding Steiner vertices, *Networks* **16** (1986) 283-294.

[30] C. Schiemangk, Thermodynamically motivated simulation for solving the Steiner tree problem and the optimization of interacting path systems, in A. Iwainsky (ed.), *Optimization of Connection Structures in Graphs* , CICIP, East Berlin, GDR (1985) 74-90.

[31] G. F. Sullivan, Approximation algorithms for Steiner tree problems, Tech. Report 249, Dept. of Computer Science, Yale Univ. (1982).

[32] H. Takahashi and A. Matsuyama, An approximate solution for the Steiner problem in graphs, *Math. Jap.* **24** (1980) 573-577.

[33] P. J. M. van Laarhoven and E. H. L. Aarts, *Simulated Annealing: Theory and Applications*, D. Reidel Publishing Company, Dordrecht, Holland (1987).

[34] S. Voss, Steiner's problem in graphs: Heuristic methods, to appear in *Discrete Appl. Math.*

[35] S.M. Wang, A multiple source algorithm for suboptimum Steiner trees in graphs, in H. Noltemeier (ed.) *Proc. Int. Workshop on Graph-Theoretic Concepts in Computer Science*, Würzburg (1985) 387-396.

[36] B. M. Waxman and M. Imase, Worst-case performance of Rayward-Smith's Steiner tree heuristics, *Inf. Process. Lett.* **29** (1988) 283-287.

[37] P. Widmayer, On approximation algorithms for Steiner's problem in graphs, in G. Tinhofer and G. Schmidt (eds.), *Graph-Theoretic Concepts in Computer Science*, **LNCS 246**, Springer-Verlag (1986) 17-28.

[38] P. Widmayer, Fast approximation algorithms for Steiner's problem in graphs, (Habilitation Thesis, 1986), Inst. für Angewandte Informatik und Formale Beschreibungsverfahren, Univ. Karlsruhe (1987).

[39] P. Winter and J. MacGregor Smith, Path-distance heuristics for the Steiner problem in undirected networks, *Algorithmica* **7** (1992) 309-327.

[40] R. T. Wong, A dual ascent approach for the Steiner tree problem on a directed graph, *Math. Program.* **28** (1984) 271–287.

[41] Y. F. Wu, P. Widmayer and C. K. Wong, A faster approximation algorithm for the Steiner problem in graphs, *Acta Inf.* **23** (1986) 223-229.

[42] A. Z. Zelikovsky, An 11/6-approximation algorithm for the Steiner problem on networks, to appear in *Proc. of the 4-th Czechoslovak Symp. on Combinatorics.*

Chapter 5

Polynomially Solvable Cases

As noted in Chapter 1, the decision version of the Steiner tree problem in networks is NP-complete. In particular, it has been shown that the problem remains NP-complete when G is planar. On the other hand, if G is a tree, the problem can be solved in linear time. This raises an interesting question whether there exist classes of planar networks for which the Steiner tree problem is solvable in polynomial (or perhaps even linear) time. Linear time algorithms for series-parallel and Halin networks are briefly described in this chapter. Also polynomial algorithms for k-planar networks and strongly chordal graphs (which are not planar) are discussed. Other classes of networks and graphs for which polynomial algorithms are available are also mentioned.

5.1 Series-Parallel Networks

A network $G = (V, E, c)$ is called *series-parallel* if and only if it contains no subnetwork homeomorphic to K_4. From Kuratowski's theorem, it follows immediately that every series-parallel network is planar [16]. A series-parallel network is said to be *maximal series-parallel* if no edge between nonadjacent vertices can be added to G so that it remains series-parallel. Every maximal series-parallel network with at least 2 vertices can be obtained using the following recursive construction:

(i) a network consisting of a single edge (v_i, v_j) is maximal series-parallel,

(ii) select any edge (v_i, v_j) of a maximal series-parallel network with $k-1$ vertices ($k \geq 3$). Add a new vertex v_k together with the edges (v_i, v_k) and (v_j, v_k). The extended network is maximal series-parallel (Fig. 5.1).

A maximal series-parallel network with v vertices has $2v-3$ edges. A maximal series-parallel network is sometimes referred to as a *2-tree* network. The reader is referred to [22,21] for more on series-parallel networks (which are in particular very common in circuits).

Wald and Colbourn [24], and in a slightly different versions, Prodon, Liebling and Groflin [17] and Rardin, Parker and Richey [20], developed an algorithm that first completes any 2-connected network $G = (V, E, c)$ to a maximal series-parallel

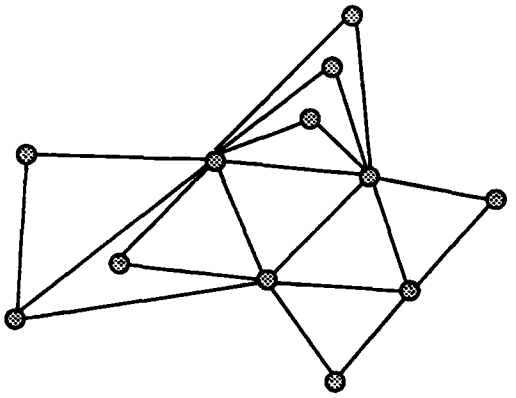


Figure 5.1: Maximal series-parallel network

network G' or decides that it is impossible in $O(v)$ time. If the edges added to G have sufficiently large lengths, it is sufficient to consider the Steiner tree problem in maximal series-parallel networks.

The completion of a 2-connected network G to a maximal series-parallel network is done as follows.

- **Step 1:** Let G' and G'' be two copies of G.
- **Step 2:** If G' consists of a single edge, then **Stop**; G'' is maximal series-parallel and has G as a partial spanning subnetwork.
- **Step 3:** Select a vertex v_k of degree 2 in G'. If no such v_k exists, then **Stop**; G is not series-parallel.
- **Step 4:** Let v_i and v_j denote vertices adjacent to v_k in G'. If $(v_i, v_j) \notin G'$, add (v_i, v_j) to G' and G''. Delete the vertex v_k and its two incident edges from G'. Go to **Step 2**.

Let G be a maximal series-parallel network. Let G' denote its copy. The algorithm for the Steiner tree problem for N in G iteratively reduces G' by removing vertices of degree 2. As a vertex v_k of degree 2 is removed together with its two incident edges (v_k, v_i) and (v_k, v_j), certain minimum length subnetworks are associated with the edge (v_i, v_j). These subnetworks are obtained by combining least length subnetworks associated with edges (v_k, v_i), (v_k, v_j) and (v_i, v_j) determined during the initialization phase or during preceding iterations. When G' is reduced to a single edge, one of the subnetworks associated with this edge yields $T_G(N)$. Due to the particular structure of G, the number of subnetworks associated with any edge is constant and small. Furthermore, the determination of these subnetworks for an edge (v_i, v_j) when v_k is removed can be done very efficiently.

Let (v_i, v_j) denote any edge of G. Let G_{ij} denote a connected subnetwork of G containing (v_i, v_j). G_{ij} will be further constrained below. Define the following subnetworks of G_{ij} spanning all its terminals:

- B_{ij} is a minimum length subtree containing v_i and v_j.

- F_{ij} is a minimum length subtree containing v_i but not containing v_j. If $v_j \in N$, then F_{ij} does not exist.
- L_{ij} is a minimum length subtree containing v_j but not containing v_i. If $v_i \in N$, then L_{ij} does not exist.
- N_{ij} is a minimum length subtree containing neither v_i nor v_j. If $v_i \in N$ or $v_j \in N$, then N_{ij} does not exist.
- U_{ij} is a minimum length union of two subtrees, containing respectively v_i and v_j.

Let $b_{ij}, f_{ij}, l_{ij}, n_{ij}, u_{ij}$ denote lengths of these subnetworks. If a particular subnetwork does not exist, its length is set to ∞. Furthermore, let $x_{ij} = 0$ if G_{ij} contains no terminals, and ∞ otherwise.

Before the first elimination, each G_{ij} consists of the edge (v_i, v_j). The lengths of subnetworks associated with G_{ij} are therefore:

$$b_{ij} = c_{ij} \qquad n_{ij} = \infty \qquad u_{ij} = 0$$

$$f_{ij} = \begin{cases} 0 & \text{if } v_j \notin N \\ \infty & \text{otherwise} \end{cases} \qquad l_{ij} = \begin{cases} 0 & \text{if } v_i \notin N \\ \infty & \text{otherwise} \end{cases} \qquad x_{ij} = \begin{cases} 0 & \text{if } v_i, v_j \notin N \\ \infty & \text{otherwise} \end{cases}$$

As a vertex v_k of degree 2 is about to be eliminated from G', G_{ij} is redefined to be a subnetwork of G induced by the vertices in G_{ij}, G_{ik} and G_{jk}. The lengths $b_{ij}, f_{ij}, l_{ij}, n_{ij}, u_{ij}, x_{ij}$ in the new G_{ij} are determined by the following recurrence relations (the straightforward verification is omitted):

- $b_{ij} := \min\{b_{ij} + u_{ik} + b_{jk}, b_{ij} + b_{ik} + u_{jk}, b_{ij} + f_{ik} + f_{jk}, u_{ij} + b_{ik} + b_{jk}\}$
- $f_{ij} := \min\{f_{ij} + f_{ik} + x_{jk}, f_{ij} + b_{ik} + l_{jk}\}$
- $l_{ij} := \min\{l_{ij} + l_{ik} + b_{jk}, l_{ij} + x_{ik} + f_{jk}\}$
- $n_{ij} := \min\{n_{ij} + x_{ik} + x_{jk}, x_{ij} + n_{ik} + x_{jk}, x_{ij} + x_{ik} + n_{jk}, x_{ij} + l_{ik} + l_{jk}\}$
- $u_{ij} := \min\{u_{ij} + u_{ik} + b_{jk}, u_{ij} + b_{ik} + u_{jk}, u_{ij} + f_{ik} + f_{jk}\}$
- $x_{ij} := x_{ij} + x_{ik} + x_{jk}$

The elimination process is completed when G' consists of a single edge (v_i, v_j). In particular, $G_{ij} = G$ and $|T_G(N)| = \min\{b_{ij}, f_{ij}, l_{ij}, n_{ij}\}$. By maintaining the information about the minimizing term for each recurrence relation, the edges of $T_G(N)$ can be recovered.

The initialization requires $O(v)$ time. Each iteration of the algorithm requires $O(1)$ time. Since each iteration eliminates one vertex, $v-2$ iterations are required. Retrieval of edges belonging to $T_G(N)$ can be done in $O(v)$ time. Consequently, the algorithm requires $O(v)$ time.

Wald and Colbourn [23] developed a similar algorithm for outerplanar networks. A network is *outerplanar* if it is planar and has a planar embedding with all vertices on the boundary of the exterior face. The linear algorithm for the Steiner tree problem in outerplanar networks is in fact a special case of the algorithm for series-parallel networks; every outerplanar network is series-parallel.

5.2 Halin Networks

A network $H = (V, E, c)$ is called a *Halin network* if (Fig. 5.2):

- no vertex has degree less than 3,
- H can be decomposed into a tree T spanning all vertices and a cycle C through the leaves of T,
- there is a planar embedding of H with C forming the boundary of the exterior face.

If T is a star (i.e., it contains a vertex v_i adjacent to all other vertices) then H is called a *wheel*.

Let Γ_i denote the set of vertices adjacent to a vertex v_i not on the cycle C of a Halin network $H = T \cup C$. If Γ_i contains exactly one vertex (say, v_j) not in C, then the subnetwork F_i of H induced by $\Gamma_i \cup v_i - v_j$ is called a *fan*. Every Halin network which is not a wheel has at least two fans.

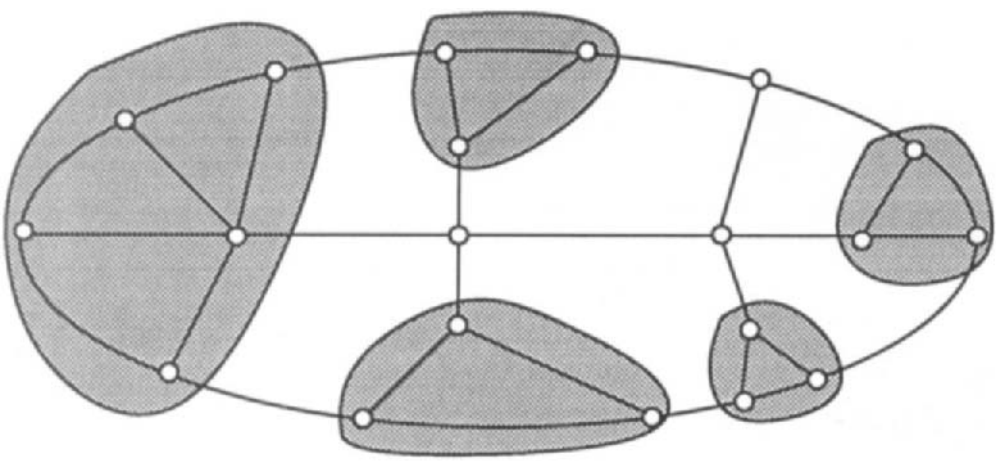


Figure 5.2: A Halin network and its fans

A *fan contraction* of a Halin network H along some fan F_i is obtained as follows:

- replace F_i by a new vertex w,
- the edges previously incident with exactly one vertex of F_i are made adjacent to w.

The linear time algorithm for the Steiner tree problem in a Halin network H is based on fan contractions. As a fan is contracted, appropriate least length subnetworks are associated with the vertex replacing the fan. When H is contracted to a wheel, the information associated with vertices of its cycle are adequate to determine $T_H(N)$. The technique is closely related to that used in connection with maximal series-parallel networks. The reader is referred to [26] for details.

Linear algorithms for the Steiner tree problem in maximal series-parallel networks and in Halin networks are based on determining a fixed number least length subnetworks of certain networks. These networks are defined recursively from some limited set of basic components. Composition rules for the generation of larger networks are of restricted nature: smaller networks can be attached to one another

at a bounded number of vertices. As larger networks are generated, their least length subnetworks are determined in constant time from least length subnetworks of smaller networks. As pointed out by Bern, Lawler and Wong [6], this dynamic programming technique applies to networks other than maximal series-parallel and Halin networks. Furthermore, it applies to many other combinatorial optimization problems apart from the Steiner tree problem.

5.3 k-Planar Networks

Let G be a 2-connected planar network with a given planar embedding and with all terminals on the boundary of the exterior face. Such planar networks are referred to as *1-planar* (Fig 5.3).

Any subset of terminals visited consecutively when traversing the boundary of the exterior face is called an *interval.*

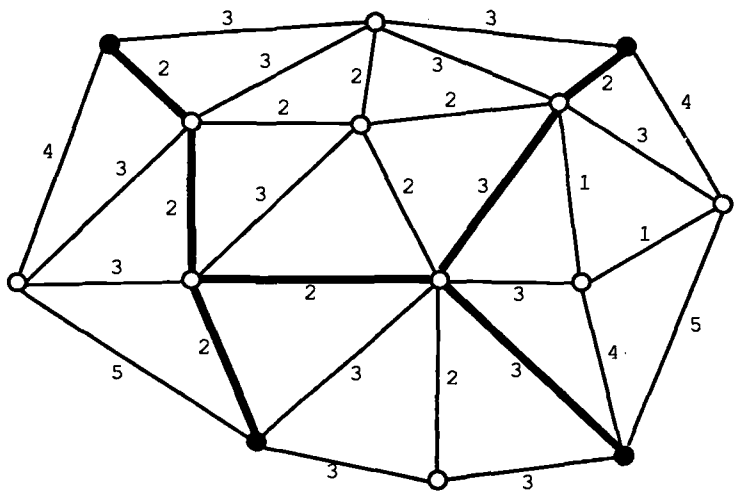


Figure 5.3: 1-planar network and its Steiner minimal tree

Suppose that v_i is a vertex in the Steiner minimal tree $T_G(N)$ incident with more than one edge. $T_G(N)$ can be decomposed at v_i (as explained in Section 3.4) in several different ways so that it breaks into two subtrees $\bar{T}$ and $\tilde{T}$. It is always possible to carry out this decomposition at v_i so that the set of terminals spanned by $\bar{T}$ and $\tilde{T}$ are intervals. It follows that when determining $T_G^2(v_i \cup Y)$, where Y is a subset of terminals, the general dynamic programming algorithm described in Section 3.4 only needs to consider those Y that are intervals. Furthermore, $T_G^2(v_i \cup Y)$ is a union of two Steiner minimal trees, one spanning $v_i \cup X$ and the other spanning $v_i \cup Y \setminus X$ for some X, $\emptyset \subset X \subset Y$. Hence, X and $Y \setminus X$ can be restricted to intervals. In conclusion, if G is 1-planar, the dynamic programming algorithm only needs to generate $T_G^2(v_i \cup Y)$ and $T_G(v_i \cup Y)$ for every interval Y in

G. This reduces the complexity of the dynamic programming algorithm drastically.

Theorem 5.1 *A Steiner minimal tree for N in a 1-planar network can be determined in $O(n^2v^2)$ time.*

Proof. There are only $O(n^2)$ choices of Y. For each choice of Y, there are $O(v)$ choices of v_i and $O(n)$ choices of X (since $Y \setminus X$ is required to be an interval). Hence, $O(n^3v)$ time is needed to determine all $T_G^2(v_i \cup Y)$. $T_G(v_i \cup Y)$ has to be determined for $O(n^2)$ choices of Y and $O(v)$ choices of v_i; in total $O(n^2v)$ times. Each time requires the minimization over $O(v)$ terms. Hence, $O(n^2v^2)$ time is needed to determine all $T_G(v_i \cup Y)$. This term dominates the complexity of determining all $T_G^2(v_i \cup Y)$ as well as the term $O(n^2)$ needed to determine all shortest paths in a planar network. □

If the determination of $T_G(v_i \cup Y)$ is carried out simultaneously for all choices of Y and fixed v_i as explained in Section 3.4, the overall complexity of the algorithm reduces to $O(n^3v + n^2v \log v + v^2)$.

Provan [19] extended the applicability of this result in the following way. Let G be a planar network given together with its planar embedding. Let R be a region enclosed by a polygon consisting of edges $e_1, e_2, ..., e_r$ of G. Note that the edges of the polygon are not necessarily distinct; any edge can be traversed once in each direction. The *perimeter* $\rho(R)$ of R is defined by

$$\rho(R) = \sum_{i=1}^{r} c(e_i)$$

The region R is said to be *path-convex* if any other region R' containing R satisfies $\rho(R') \geq \rho(R)$. If R is path-convex and all terminals are within R, then there is a Steiner minimal tree for N completely within R. Thus, all non-terminals and edges outside of R can be disregarded when searching for $T_G(N)$. Furthermore, if all terminals are on the polygon enclosing R, the original problem instance is reduced to a problem instance in a 1-planar network (Fig. 5.4).

Bienstock and Monma [8] proved that if there is a path-convex region R with all terminals on its boundary, then it is of minimum perimeter among all regions enclosing N. They also gave a polynomial algorithm that finds a minimum perimeter region enclosing all terminals. This leads to a polynomial algorithm that test whether or not a path-convex region containing all terminals on its boundary exists, and produces one in the affirmative case.

A planar network G with a fixed planar embedding is called *nearly 1-planar* if all terminals but at most one are on the boundary of the exterior face of G. The general dynamic programming algorithm for the Steiner tree problem also requires polynomial time in this case. Suppose that z_1 is in the interior of G. Let $N_1 = N - z_1$. Given $T_G^2(v_i \cup N_1)$ for all $v_i \notin N_1$ and $T_G(N_1)$ (which can be obtained in polynomial time since N_1 is on the boundary of the exterior face), $T_G(N)$ can be obtained by an additional minimization over n terms.

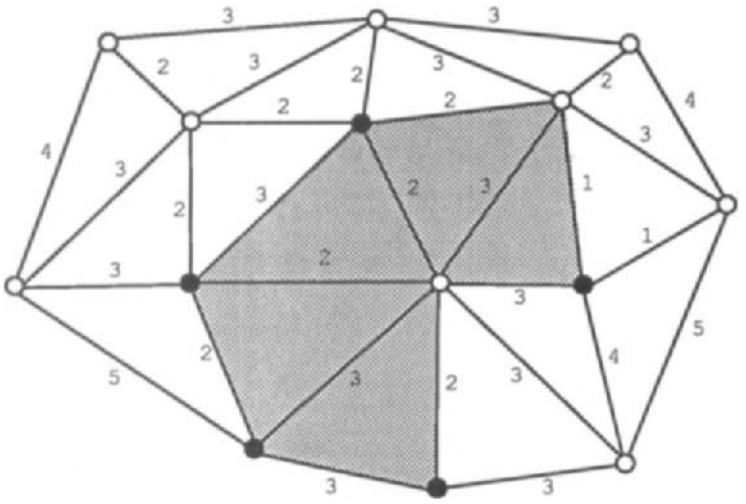


Figure 5.4: Minimum perimeter path-convex region enclosing all terminals

The general dynamic programming algorithm can be made more efficient even for a larger class of planar networks. A planar network G with a fixed planar embedding is said to be *k-planar* (respectively *nearly k-planar*) if all (respectively all but one) terminals are on the boundaries of k faces $B_1, B_2, ..., B_k$ (Fig. 5.5). It can be assumed without loss of generality that B_1 is the exterior face. Let $N_1, N_2, ..., N_k$ denote terminals on the boundaries of $B_1, B_2, ..., B_k$. The first polynomial algorithm for the Steiner tree problem in k-planar networks was given by Erickson, Monma and Veinott [14] where it was argued that it suffices to run the dynamic programming algorithm for the sets Y and X chosen such that $Y \cap N_i$, $X \cap N_i$, and $(Y \setminus X) \cap N_i$ are intervals for all $i = 1, 2, ..., k$. A modified version of this algorithm due to Bern [3,4] is described below.

Consider the geometric dual G_d of the k-planar network G. G_d is obtained by associating a node with each face of G and connecting two nodes by a branch if their corresponding faces share a boundary edge. It is always possible to embed G_d so that it is planar and each branch (not necessarily a straight line segment in the embedding) intersects the edge shared by the faces corresponding to the end-nodes of the branch. Note that in order to distinguish between G_d and G, terms *node* and *branch* rather than vertex and edge are used. Consider a tree T_d in G_d spanning nodes corresponding to $B_1, B_2, ..., B_k$, and with leaves being a subset of these nodes. If the edges of G intersected by the branches of T_d are deleted, a 1-planar network is obtained. Furthermore, at least one of the trees constructed in this way will not intersect any edge of $T_G(N)$. Consequently, if all possible T_d are enumerated, intersected edges are deleted and then the dynamic programming algorithm is applied, the shortest Steiner minimal tree generated will be the sought $T_G(N)$.

Given T_d, one in fact does not need to delete intersected edges. It is the ordering

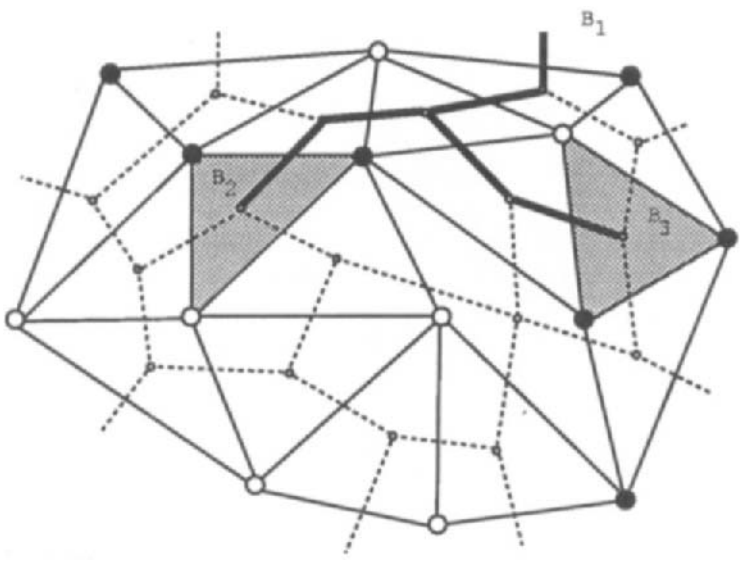


Figure 5.5: k-planar network G, its geometric dual G_d and T_d

of the terminals induced by T_d that matters. Applying the dynamic programming algorithm to 1-planar networks with the given ordering (as if G was 1-planar) will yield a tree spanning all terminals which might be shorter than the tree that would be obtained if intersected edges were deleted. But there is at least one T_d which does not intersect $T_G(N)$. Consequently, there will be at least one ordering for which the application of the dynamic programming algorithm will yield $T_G(N)$.

Every T_d can be broken down into $k-1$ paths with end-nodes corresponding to faces $B_1, B_2, ..., B_k$ and with no such nodes in the interior of these paths. Some of these paths can use the same branches in their front part. Each path cuts terminals on faces corresponding to its end-nodes. Hence, the total number of different subdivisions of terminals caused by a single path is $O(n^2)$. Since there are $k-1$ such paths, the total number of orderings is $O(n^{2k-2})$. Each ordering requires an application of the dynamic programming algorithm with the time complexity $O(n^3v + n^2v\log v)$.

Theorem 5.2 *The Steiner minimal tree for N in a k-planar network can be determined in $O(n^{2k+1}v + n^{2k}v\log v + v^2)$ time.*

The above approach can be applied to nearly k-planar networks using the same method as in connection with the nearly 1-planar networks.

In view of Theorem 5.2, the following *face cover* problem is of crucial importance.

- **GIVEN:** a planar network G with a fixed planar embedding and a subset N of terminals.
- **FIND:** minimum number of faces needed to cover all terminals.

Bienstock and Monma [7] proved that the decision version of this problem is NP-complete and gave an exponential exact algorithm. They also suggested a polynomial heuristic with the relative error converging to zero for a class of planar networks for which the problem remains NP-complete. Furthermore, they gave an $O(vc^k)$ time algorithm (where c is a constant) which checks if a planar graph with a fixed embedding is k-planar. Hence the algorithm runs in linear time for any fixed k.

Bienstock and Monma [7] also considered a more difficult problem where the planar embedding of G is not given beforehand. The problem is then to find a planar embedding with the minimum face cover of terminals. The decision version of this problem is also NP-complete. But they showed that it is still possible to generate a k-planar embedding of G provided that it exists in $O(vc^k)$ time.

Bern [3] showed that his cutline approach used in connection with k-planar networks can be adapted to the case when G is a network embedded on a torus so that no edges cross and all terminals are on the boundary of one face. In this case, $T_G(N)$ can be found in $O(n^7v + n^6v\log v)$ time. The approach is also applicable to higher genus surfaces.

Bern [4] also considered the problem when most terminals in a planar network are on the boundary of the exterior face while the remaining terminals are scattered arbitrarily in the interior of G. Let N_B denote the terminals on the exterior boundary. Let $b = |N_B|$. Bern proved that it suffices to run the general dynamic programming algorithm for the sets Y and X chosen such that $Y \cap N_B$, $X \cap N_B$, and $(Y \setminus X) \cap N_B$ are intervals. With these restrictions, the general dynamic programming algorithm requires $O(b^3 3^{n-b}v + b^2 2^{n-b}v\log v)$ time.

Similar polynomial algorithms based on the dynamic programming approach have been developed for k-outerplanar networks [2] and k-layered planar networks [3,5] where the sequential removal of k exterior boundaries leaves a network respectively empty and with no terminals.

Although the dynamic programming algorithms discussed in this section are polynomial and apply to classes of non-trivial planar networks, they are most likely too slow to be of practical importance. Development of faster algorithms for the Steiner tree problem in such networks is an important open research problem.

5.4 Strongly Chordal Graphs

White, Farber and Pulleyblank [25] developed an $O(v^3)$ time algorithm for the Steiner tree problem in strongly chordal graphs. A graph is called *chordal* if every cycle with four or more edges has a chord. A graph is called *strongly chordal* if it is chordal and every even cycle with six or more edges has an "odd" chord (i.e., a chord dividing the cycle into two paths, each containing an odd number of edges).

In order to describe the algorithm, some additional definitions are needed. For any vertex v_i, its neighborhood $\bar{\Gamma}_i$ consists of v_i and the set Γ_i of vertices adjacent to v_i. A vertex v_i is called *simple* if for each pair of vertices $v_k, v_l \in \bar{\Gamma}_i$, either $\bar{\Gamma}_k \subseteq \bar{\Gamma}_l$ or $\bar{\Gamma}_l \subseteq \bar{\Gamma}_k$. It can be shown that a graph G is strongly chordal if and only if it is possible to relabel vertices such that every vertex $v_i \in V$ is simple

in the subgraph induced by $W = \{v_j | j \geq i\}$. Such a labeling is called a ***simple elimination ordering*** (Fig. 5.6). It can be determined in $O(v^3)$ time [15].

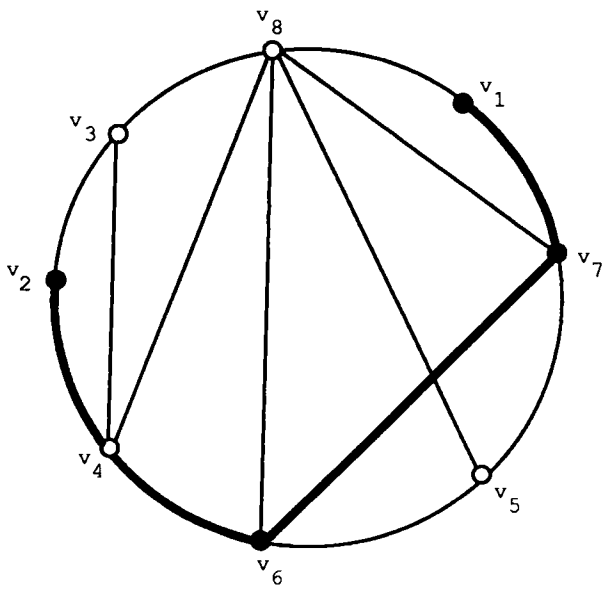


Figure 5.6: Strongly chordal graph and its simple elimination ordering

Given a simple elimination ordering, $T_G(N)$ can be determined in $O(v^2)$ time: Let T be a subnetwork of G. Initially T contains all terminals but no edges. Vertices of G are examined in their simple elimination order. If $v_i \notin T$, then v_i is deleted from G. If $v_i \in T$, the edge of G connecting v_i with some other vertex of T is added to T. If no such edge exists, the vertex v_j adjacent to v_i and with the highest degree in G is added to T. The edge (v_i, v_j) is also added to T. Finally, v_i is deleted from G. Upon termination, $T_G(N) = T$. Heavy edges in Fig. 5.6 indicate $T_G(N)$ for $N = \{v_1, v_2, v_6, v_7\}$.

The $O(v^3)$ time algorithm for the Steiner tree problem in strongly chordal graphs is perhaps of less theoretical importance than the fact that the decision version of the Steiner tree problem in chordal graphs is NP-complete. Furthermore, the decision version of the Steiner tree problem in strongly chordal networks is NP-complete even if edge lengths are strictly triangular (i.e., length of an edge is less than the length of any path between the end-vertices of the edge). Hence, here one has closely related problems where one is in P while others are NP-hard.

Polynomial time algorithms for the Steiner tree problem in other types of graphs have been developed. In particular, the class of perfect graphs and its various subclasses have been studied. Polynomial algorithms have been given for permutation graphs (and cographs) [11], distance-hereditary graphs [12], homogeneous graphs [13], (6,2)-chordal bipartite graphs [1] and series-parallel block graphs [9,10].

References

[1] G. Ausiello, A. D'Atri and M. Moscarini, Properties on graphs and minimal conceptual connections in semantic data models, *J. Comput. Syst. Sci.* **33** (1986) 179-202.

[2] B. S. Baker, Approximation algorithms for NP-complete problems on planar graphs, *Proc. of the 24-th Ann. Symp. on Foundations of Computer Science* (1983) 265-273.

[3] M. W. Bern, Network design problems: Steiner trees and spanning k-trees, Ph.D Thesis, Dept. of Computer Science, Univ. of California, Berkeley (1987).

[4] M. W. Bern, Faster exact algorithm for Steiner trees in planar networks, *Networks* **20** (1990) 109-120.

[5] M. W. Bern and D. Bienstock, Polynomially solvable special cases of the Steiner problem in planar networks, *Ann. Oper. Res.* **33** (1991) 405-418.

[6] M. W. Bern, E. L. Lawler and A. L. Wong, Why certain subgraph computations require only linear time? *Proc. of the 26-th Ann. Symp. on Foundations of Computer Science* (1985) 117-125.

[7] D. Bienstock and C. L. Monma, On the complexity of covering vertices by faces in a planar graph, *SIAM J. Comput.* **17** (1988) 53-76.

[8] D. Bienstock and C. L. Monma, Optimal enclosing regions in planar graphs, *Networks* **19** (1989) 79-94.

[9] B. A. Campbell and R. L. Rardin, Steiner tree problems on series-parallel block graphs I: Polynomial recognition and solution, Technical Report CC-87-17, School of Ind. Eng., Purdue Univ. (1987).

[10] B. A. Campbell, Steiner problems on series-parallel block graphs: Polyhedral characterization, Technical Report CC-87-25, School of Ind. Eng., Purdue Univ. (1987).

[11] C. J. Colbourn and L. K. Stewart, Permutation graph: Connected domination and Steiner trees, Technical Report CS-85-02, Dept. of Comp. Science, Univ. of Saskatchewan (1985).

[12] A. D'Atri and M. Moscarini, Distance-hereditary graphs, Steiner trees, and connected domination, *SIAM J. Comput.* **17** (1988) 521-538.

[13] A. D'Atri, M. Moscarini and A. Sassano, The Steiner tree problem and homogeneous sets, *Math. Foundations of Computer Science*, Lecture Notes in Computer Science, Springer-Verlag **324** (1988) 249-261.

[14] R.E. Erickson, C.L. Monma and A.F. Veinott Jr., Send-and-split method for minimum-concave-cost network flows, *Math. Oper. Res.* **12** (1987) 634-664.

[15] M. Farber, Characterization of strongly chordal graphs, *Discrete Math.* **43** (1983) 173-189.

[16] F. Harary, *Graph Theory*, Addison-Wesley, Reading, MA (1969).

[17] A. Prodon, T. M. Liebling and H. Groflin, Steiner's problem on two-trees, Technical Report RO 850315, Dept. de Math. Ecole Polytechnique Federale de Lausanne, Suisse (1985).

[18] J. S. Provan, A polynomial algorithm for the Steiner tree problem on terminal-planar graphs, Technical Report 83/10, Dept. of Operation Research, Univ. of North Carolina, Chapel Hill (1983).

[19] J. S. Provan, Convexity and the Steiner tree problem, *Networks* **18** (1988) 55-72.

[20] R. L. Rardin, R. G. Parker and M. B. Richey, A polynomial algorithm for a class of Steiner tree problems on graphs, Technical Report, Dept. of Ind. and Syst. Eng. J-82-5, Georgia Inst. Tech. (1982).

[21] M. M. Sysło, Series-parallel graphs and depth-first search trees, *IEEE Trans. Circuits Syst.* **CAS-31** (1984) 1029-1033.

[22] K. Takamizawa, T. Nishizeki and N. Saito, Linear-time computability of combinatorial problems on series-parallel graphs, *J. Assoc. Comput. Mach.* **29** (1982) 623-641.

[23] J. A. Wald and C. J. Colbourn, Steiner trees in outerplanar networks, *Congr. Numerantium* **36** (1982) 15-22.

[24] J. A. Wald and C. J. Colbourn, Steiner trees, partial 2-trees and minimum IFI networks, *Networks* **13** (1983) 159-167.

[25] K. White, M. Farber and W. R. Pulleyblank, Steiner trees, connected domination and strongly chordal graphs, *Networks* **15** (1985) 109-124.

[26] P. Winter, Steiner problem in Halin networks, Technical Report, Dept. of Computer Science, Univ. of Copenhagen (1983), also appeared in *Discrete Appl. Math.* **17** (1987) 281-294.

Chapter 6

Generalizations

A large variety of generalizations and variants of the Steiner tree problem have been considered in the literature. Some of the most important problems of this type are described in this chapter.

6.1 Steiner Trees in Directed Networks

The *Steiner arborescence problem* can be formulated as follows.

- **GIVEN:** A directed network $\vec{G}= (V, A, c)$, a non-empty subset N, $N \subseteq V$, of terminals, and a *root* terminal z_1.
- **FIND:** A directed subnetwork $T_{\vec{G}}(N)$ of $\vec{G}$ such that:
  - there is a path from z_1 to every other terminal,
  - $|T_{\vec{G}}(N)|$ is minimized.

As already mentioned in Subsection 1.4.4, the Steiner arborescence problem is a generalization of the Steiner tree problem in undirected networks. Several mathematical programming formulations of the Steiner arborescence problem were given in Section 3.6.

An important special case of the Steiner arborescence problem arises when $\vec{G}$ is acyclic. Such networks occur often in practical applications. Consider for example the design of an optimal drainage system. Here the root is defined by the location of the sewer which collects the waste from the drainage system, and the acyclicity is imposed by the law of gravity. Nastansky, Selow and Stewart [29] developed a specialized enumeration algorithm for the Steiner arborescence problem in acyclic networks.

Another problem in directed networks closely related to the Steiner arborescence problem is the *Steiner equivalent network problem* suggested by Hakimi [15].

- **GIVEN:** A directed network $\vec{G}= (V, A, c)$, a non-empty subset N, $N \subseteq V$, of terminals, and a *root* terminal z_1.

- **FIND:** A directed subnetwork $T_{\vec{G}}(N)$ of $\vec{G}$ such that:
  - there is a path between every pair of terminals provided that such a path exists in $\vec{G}$,
  - $|T_{\vec{G}}(N)|$ is minimized.

Note that the solution to this problem will usually not be an arborescence. The decision version of the Steiner equivalent network problem is NP-complete even if $N = V$ [35]. Relatively little work has been done on this problem. Furthermore, all work has been concerned with the case $N = V$ (although algorithms can easily be extended to the Steiner version). A branch-and-bound algorithm has been given by Martello and Toth [26] where also references to earlier algorithms can be found. A linear time algorithm in directed series-parallel networks (i.e., networks which are series-parallel when arc orientations are ignored) has been developed by Richey, Parker and Rardin [34].

6.2 Weighted Steiner Tree Problem

The *weighted Steiner tree problem* can be formulated as follows.

- **GIVEN:** An undirected network $G = (V, E, c, w)$ where $w : V \rightarrow \mathcal{R}$, and a non-empty subset N, $N \subseteq V$, of terminals.
- **FIND:** A subnetwork $T_G(N)$ of G such that:
  - there is a path between every pair of terminals,
  - total length of $T_G(N)$ defined by

$$\sum_{v_i \in T_G(N)} w_i + \sum_{(v_i, v_j) \in T_G(N)} c_{ij}$$

  is minimized.

The weighted Steiner tree problem is clearly a generalization of the Steiner tree problem where the vertices are of zero weight. If the weights of vertices are nonnegative, the weighted Steiner tree problem can be transformed into the Steiner arborescence problem: replace every edge (v_i, v_j) by two opposite arcs with lengths $c_{ij} + w_j$ and $c_{ji} + w_i$ respectively, and select an arbitrary terminal as the root. The weighted Steiner tree problem was originally introduced by Segev [36] where lower bounds and heuristics for the special (but still NP-complete) case with negative vertex weights and only one terminal were developed. This special case is of practical importance. The only terminal may represent a service center. The remaining non-terminals may represent potential customers. The length of an edge may be interpreted as the cost associated with the use of the connection in order to attach some customers to the center. Vertex weights can be interpreted as revenues that can be obtained by connecting a customer to the center.

The weighted Steiner tree problem can be formulated as a more general version of the degree-constrained formulation of the Steiner tree problem discussed in Subsection 1.4.5 and Subsection 3.6.3. Rather than associate length 0 to the

edges from the artificial vertex to all non-terminals, the length $c_{0s} = -w_s, s \in S$, is assigned to these edges. The problem then is to minimize

$$\sum_{v_i \in V \setminus N} w_i - \sum_{v_i \in V \setminus N} w_i x_{0i} + \sum_{(v_i,v_j) \in E} c_{ij} x_{ij}$$

subject to the same constraints as for the degree-constrained formulation of the Steiner tree problem given in Subsection 3.6.3. It was shown by Duin and Volgenant [11] that the reduction tests for the Steiner tree problem can be extended to also apply to the weighted Steiner tree problem.

6.3 Steiner Forest Problem

In some applications it may be appropriate to relax the requirement that the solution must be connected. When designing an offshore oil pumping network it is for example realistic to assume that there are several terminals ashore. This leads to the following *Steiner forest problem* which is clearly a generalization of the Steiner tree problem.

- **GIVEN:** An undirected network $G = (V, E, c)$, a non-empty subset N, $N \subseteq V$, of vertices, and an integer q, $1 \leq q \leq n$.
- **FIND:** A subnetwork $F_G(N)$ of G such that:
  - it has at most q components, each containing at least one terminal,
  - $|F_G(N)|$ is minimized.

Duin and Volgenant [11] formulated this problem as a degree-constrained minimum spanning tree problem discussed in Subsection 1.4.5 and Subsection 3.6.3 with the additional constraint restricting the number of edges incident with the artificial vertex to be at most q. When the problem is relaxed in Lagrangean fashion by moving *all* degree-constraints into the objective function, unconstrained minimum spanning tree problem is obtained. No computational experience concerning the quality of the lower bounds obtained by this kind of Lagrangean relaxation is available. Duin and Volgenant argued that the reductions applicable to the Steiner tree problem essentially carry over to this more general Steiner forest problem.

6.4 Hierarchical Steiner Tree Problem

In some practical applications it is unrealistic to assume that all terminals are of equal importance. Consider for instance problems involving the design of transportation networks. Usually few major cities are to be interconnected by primary roads. Then only secondary roads are needed to connect villages to primary roads. Similar types of hierarchies occur in many practical network design applications.

The *hierarchical Steiner tree problem* has been formulated by Iwainsky [20] in the following way.

- **GIVEN:** An undirected network $G = (V, E, C)$ where $C(e_i)$ is a k-dimensional length vector $(c_1(e_i), c_2(e_i), ..., c_k(e_i))$ satisfying $c_1(e_i) > c_2(e_i) > ... > c_k(e_i)$, a non-empty subset N, $N \subseteq V$, of terminals, and a partition $\{N_1, N_2, ..., N_k\}$ of N with $|N_1| \geq 2$,
- **FIND:** A subnetwork $T_G(N)$ of G such that:
  - the length of any edge e_i on a path between a pair of N_j-terminals is at least $c_j(e_i)$,
  - $|T_G(N)|$ is minimized.

This problem has also been investigated by Duin and Volgenant [12] for the special case with $k = 2$. They proved that every instance of that hierarchical Steiner tree problem can be transformed into an instance of the Steiner arborescence problem with kv vertices and $2ke + (k-1)v$ arcs. The transformation is as follows. Replace every edge of G by two opposite arcs. Take k copies $G_1, G_2, ..., G_k$ of this directed network and associate lengths c_i with the i-th network, $1 \leq i \leq k$. Connect a vertex v in G_i, $1 \leq i \leq k-1$, with its copy in G_{i+1} by a directed arc of length 0. Select any N_1-terminal as the root. The optimal solution to this instance of the Steiner arborescence problem will provide the optimal solution to the instance of the hierarchical Steiner tree problem.

Duin and Volgenant [12] identified various reduction tests which make it possible to identify non-terminals and edges which under no circumstances can belong to any optimal solution. Also some reduction tests identifying edges that must belong to an optimal solution were given. These reductions are of the same flavor as the reductions for the Steiner tree problem discussed in Chapter 2.

Another type of hierarchical Steiner tree problem arises when the network itself is hierarchically structured (i.e., portions of the network can be replaced by a single "macro-vertex") so there is an obvious way of decomposing the problem into smaller subproblems. Approaches to this kind of the hierarchical Steiner tree problem were discussed in [4,43,1,2].

6.5 Degree-Dependent Steiner Tree Problem

In some practical applications, it is unrealistic to assume that the total cost of the network is proportional to the sum of its edge lengths. One type of complications arises when the cost of the network depends also on the number of edges incident to vertices. Usually one would penalize vertices with more than 2 incident edges. Iwainsky, Canuto, Taraszow and Villa [21] considered the *degree-dependent Steiner tree problem* which captures this kind of complications.

- **GIVEN:** An undirected network $G = (V, E, c)$, a non-empty subset N, $N \subseteq V$, of terminals, a function b assigning to each vertex $v_i \in V$ and to each integer q, $1 \leq q < v$, a nonnegative real number $b(v_i, q)$ (with $b(v_i, q') \geq b(v_i, q)$ for $v > q' > q$, and $b(v_i, 1) = b(v_i, 2) = 0$ for all $v_i \in V$).
- **FIND:** A subnetwork $T_G(N)$ of G such that:
  - there is a path between every pair of terminals in $T_G(N)$,

- $|T_G(N)| + \sum_{v_i \in T_G(N)} b(v_i, deg_T(v_i))$ is minimized.

Iwainsky, Canuta, Taraszow and Villa [21] gave a reduction test which can be used to identify some of the non-terminals as not belonging to $T_G(N)$.

6.6 Group Steiner Tree Problem

As mentioned in Chapter 1, the Steiner tree problem has applications in connection with the wire routing phase in physical VLSI design. After the placement of components on a chip, sets of pins on the component boundaries are to be connected withing the remaining free chip space. However, even for components with predetermined interior layout, they can be flipped or rotated by the routing algorithm. Consequently, each pin can take one out of several given positions. This motivates the *group Steiner tree problem* where one is interested in finding a minimum length network which spans at least one terminal from each subset of a partition of all terminals. More formally, the problem can be formulated as follows.

- **GIVEN:** An undirected network $G = (V, E, c)$, a non-empty subset N, $N \subseteq V$, of terminals, and a partition $\{N_1, N_2, ..., N_k\}$ of N,
- **FIND:** A subnetwork $T_G(N)$ of G such that:
  - at least one terminal from each N_i, $i = 1, 2, ..., k$, is in $T_G(N)$,
  - $|T_G(N)|$ is minimized.

When $|N_i| = 1$ for every $i = 1, 2, ...k$, the problem reduces to the Steiner tree problem. The decision version of the generalized Steiner tree problem is NP-complete even if there is no non-terminals and G is a tree [32]. This is in sharp contrast with the Steiner tree problem where the restriction to either of cases makes the problem easy to solve. The generalized Steiner tree problem becomes polynomially solvable if vertex degrees are bounded by 2 [19]. Suppose for instance that G is a path embedded in the plane along the x-axis. For each $z_i \in N$, scan the path to the right beginning at z until at least one terminal from every group has been scanned (or when the end of the path has been reached). Among successfully generated subpaths (containing at least one terminal from each group), select the shortest one.

Any problem instance of the generalized Steiner tree problem with an arbitrary partition can be transformed to a problem instance of the Steiner tree problem. Consider a network $G' = (V', E', c')$ obtained from G in the following manner. For each subset N_i, $i = 1, 2, ..., k$, let u_i denote a new vertex representing the whole subset N_i. Let $U = \{u_1, u_2, ..., u_k\}$ and let

$$V' = V \cup U$$

$$E' = E \cup \bigcup_{i=1}^{k} \{(z, u_i) | z \in N_i\}$$

$$c'(e_j) = \begin{cases} c(e_j) & \text{if } e_j \in E \\ INF & \text{otherwise} \end{cases}$$

where INF is a constant greater than $|G|$. Consider the Steiner minimal tree for U in G'. Every vertex u_i, $i = 1, 2, ..., k$, has degree 1. Furthermore, it is adjacent

to a vertex in N_i. If the k edges incident with vertices of U are deleted, a tree spanning at least one terminal from each partition subset is obtained. This tree is bound to be the shortest among all such trees. A similar transformation can be used in connection with the *directed* generalized Steiner tree problem.

From the theoretical point of view the above transformation makes the generalized Steiner tree problem of limited importance. The transformed network G' has at most twice as many vertices as G'. Furthermore, the number of terminals to be spanned is reduced to k. Consequently, exact algorithms for the Steiner tree problem should work as well (or rather as bad) for the generalized Steiner tree problem.

The situation becomes different with respect to heuristics. Polynomial heuristics for the Steiner tree problem such as the shortest paths heuristic, distance network heuristic, average distance heuristic and their variants (see Chapter 4) produce solutions which never are twice the length of the optimal solution. Since the Steiner minimal tree for U in G' must contain k edges of length INF, the application of any of the heuristics to the transformed network may result in solutions which are far from the optimal solution. In other words, suppose that any of the above heuristics is applied to G' with U as the terminals required to be spanned. When long edges incident with U-vertices are deleted, the solution obtained will not necessarily be at most twice the length of the optimal solution to the generalized Steiner tree problem.

So far only three heuristics for the generalized Steiner tree problem have been suggested. They all work on the original network G. However, they have equivalent counterparts (producing the same solutions) defined on G'. When considered in such setting, they turn out to be identical with some of the heuristics discussed in Chapter 4.

Reich and Widmayer [32] suggested a heuristic which is identical to the minimum spanning tree heuristic for U in G' (see Subsection 4.2.1) with subsequent *clean-up*. The clean-up consists of deleting one by one vertices which have degree 1 provided that they leave the remaining tree with at least one terminal from each group. Furthermore, among all vertices eligible for clean-up, the one with a longest incident edge is always selected. This will in particular ensure that all U-vertices disappear. The clean-up step can be performed in $O(v \log v)$ time so the overall time complexity of the heuristic is dominated by the complexity of the minimum spanning tree heuristic that is $O(e+v \log v)$. Unfortunately, the minimum spanning tree heuristic is known to produce solutions that can be far from the optimum.

Reich and Widmayer [32] suggested another heuristic that is identical to the shortest paths heuristic for U in G' (see Subsection 4.1.1) followed by the clean-up. The worst-case time complexity of this heuristic is $O(k(e + v \log v))$. Ihler [18] proved that the worst-case error ratio of this heuristic is $k - 1$, and the bound is tight.

Ihler [18] suggested a heuristic that is identical to the shortest paths with origin heuristic (see Subsection 4.1.3) for U in G' followed by the clean-up. The worst-case time complexity of this heuristic is $O(e + v \log v)$. Ihler proved that its worst-case error ratio is also $k-1$, and the bound is tight. However, similarly to the minimum

spanning tree heuristic, the shortest paths with origin heuristic is known to produce solutions which can be far from the optimum.

An interesting question which arises in connection with the heuristics for the group Steiner tree problem is whether it is possible to have a polynomial heuristic with the worst-case error ratio bounded by a constant. Ihler [19] proved that even if G is a tree and all edges have unit lengths, the problem is MAX SNP-hard [30]. Consequently, it is rather unlikely that an approximation scheme for the generalized Steiner tree problem exists. Furthermore, Ihler proved that this restricted problem has a "constant" heuristic (i.e., with worst-case error ratio bounded by a constant) only if the minimum set cover problem has a "constant" heuristic. It is known the the minimum set cover problem is MAX SNP-hard (so it is unlikely that an approximation scheme for it exists). On the other hand, it is a longstanding open problem whether the minimum set cover problem has a "constant" heuristic [30].

Consider a slightly less general version of the generalized Steiner tree problem where the number of subsets with more than one terminal is bounded by a constant [19]. The heuristics for the Steiner tree problem can be easily modified to yield polynomial heuristics with the worst-case error ratio bounded by a constant. This problem is still MAX SNP-hard as it generalizes the Steiner tree problem.

6.7 Multiple Steiner Trees Problem

The *multiple Steiner trees problem* can be formulated as follows

- **GIVEN:** An undirected network $G = (V, E, c)$, and non-empty subsets $N_1, N_2, ..., N_k$ of V.
- **FIND:** k edge-disjoint trees $T_G(N_1), T_G(N_2), ..., T_G(N_k)$ such that $\sum_{i=1}^{k} |T_G(N_i)|$ is minimized.

Richey and Parker [33] proved that the decision version of this problem is NP-complete even if G is a series-parallel graph while being easy on trees. If $N_1 = N_2 = ... = N_k$, then the multiple Steiner tree problem is known to be NP-complete in general but is polynomially solvable in series-parallel graphs [8,33]. It is unsettled whether the variant of the multiple Steiner trees problem where trees $T_G(N_1), T_G(N_2), ..., T_G(N_k)$ are permitted to share edges (each edge contributes only once to the total cost) is NP-complete [33]. However, the decision version of the following fixed-charge multicommodity flow problem which is closely related to the edge-sharing multiple Steiner trees problem is NP-complete in series-parallel graphs.

- **GIVEN:** An undirected network $G = (V, E, c)$, non-empty pairs of subsets (N_1, D_1), $(N_2, D_2), ..., (N_k, D_k)$ of V where $|N_i| \leq 2$, $|D_i| \leq 2$, $i = 1, 2, ..., k$.
- **FIND:** k edge-disjoint trees $T_G(N_1 \cup D_1), T_G(N_2 \cup D_2), ..., T_G(N_k \cup D_k)$ such that $\sum_{i=1}^{k} |T_G(N_i \cup D_i)|$ is minimized.

Suppose that G is a directed network, and $T_G(N_i)$, $i = 1, 2, ..., k$, are required to contain a directed path between two N_i-terminals if such a path exists in G. The problem of finding minimum length network of this type is known to be NP-complete even if G is a directed graph with the underlying undirected structure being series-parallel [33]. This result is valid both when $T_G(N_1), T_G(N_2), ..., T_G(N_k)$

are required to be arc-disjoint and when arc-sharing is permitted. For $k = 1$, these problems are identical and can be solved in polynomial time when the underlying undirected structure is series-parallel [33].

A straightforward generalization of the multiple Steiner trees problem arises when the disjointness requirement is dropped (and edges contribute to the total weight only once). This generalization has some applications in connection with the design of vacuum systems. The decision version of this problem is NP-complete even if edge-lengths are uniform and $N = N_1 \cup N_2 \cup ... \cup N_k$ [10].

6.8 Multiconnected Steiner Network Problem

When designing water or electricity supply networks, communication networks and other types of networks, two antagonistic goals are often present: the minimization of their total cost and the maximization of their reliability. The reliability is often expressed by the degree of connectivity of the network.

Although Steiner minimal trees for a set of terminals satisfy the primary goal of minimizing the total cost while being connected, they are extremely vulnerable if operational failures of its vertices and edges, due to scheduled maintenance, error, overload, or any other kind of destruction, are likely to occur. Only one vertex or edge failure causes a tree network to fail in its main objective of enabling communication between every pair of vertices. Thus, when failures are likely to occur, the objective may be to determine a minimum cost network with some specified connectivity.

A network $G = (V, E, c)$ is said to be *k-connected* (respectively *k-edge-connected*), $1 \leq k \leq v - 1$, if for any pair of vertices v_i and v_j, there are at least k disjoint (edge-disjoint) paths from v_i to v_j. Two vertices v_i and v_j in G are said to be *locally k-connected* (*locally k-edge-connected*) if there are at least k disjoint (edge-disjoint) paths from v_i to v_j.

The *multiconnected Steiner network problem* originally formulated by Krarup [22] is as follows.

- **GIVEN:** An undirected network $G = (V, E, c)$, a non-empty subset N, $N \subseteq V$, of terminals, and an integer $n \times n$ matrix $R = (r_{ij})$ of required local connectivities (or edge-connectivities) between the terminals.
- **FIND:** A subnetwork $T_G(N)$ of G such that
  - every pair of terminals z_i and z_j is locally r_{ij}-connected (r_{ij}-edge-connected),
  - $|T_G(N)|$ is minimized.

The special case of the multiconnected Steiner network problem arising when $N = V$ received considerable attention. Both connected and edge-connected decision versions of this problem are known to be NP-complete [14]. A survey of the multiconnected Steiner network problem for $N = V$ has been given by Christofides and Whitlock [7], indicating that in particular uniform length networks received much attention.

The edge-connected version of the multiconnected Steiner network problem can be cast in the form of a mixed integer linear programming problem. Furthermore,

the *multiconnected Steiner arborescence problem* can be shown to be a generalization of the multiconnected Steiner network problem by a transformation similar to that given in Subsection 1.4.4. Hence, the formulation below is for the arborescence case.

Define a binary variable x_{ij} for every arc $[v_i, v_j]$ in $\vec{G}$. Furthermore, for any $[v_i, v_j] \in A$, and any $v_k, v_l \in N$, $v_k \neq v_l$, let y_{ij}^{kl} denote the amount of commodity to be sent from v_k to v_l along the arc $[v_i, v_j]$. Consider the following mixed integer linear programming problem (compare with the problem $\mathcal{P}_1$ in Subsection 3.6.1):

$$\min \sum_{[v_i,v_j]\in A} c_{ij} x_{ij}$$

subject to

$$\begin{aligned}
x_{ij} &\geq y_{ij}^{kl}, && \forall [v_i, v_j] \in A, \forall v_k, v_l \in N, v_k \neq v_l \\
\textstyle\sum_{[v_j,v_i]\in A} y_{ji}^{kl} - \sum_{[v_i,v_j]\in A} y_{ij}^{kl} &= \begin{cases} r_{kl}, & \forall v_k, v_l \in N, v_i = v_l \\ 0, & \forall v_k, v_l \in N, \forall v_i \neq v_k, v_l \end{cases} \\
x_{ij} &\in \{0, 1\}, && \forall [v_i, v_j] \in A \\
y_{ij}^{kl} &\geq 0, && \forall [v_i, v_j] \in A, \forall v_k, v_l \in N, v_k \neq v_l
\end{aligned}$$

Let $U = (u_{ij})$ be the optimal solution to this problem. It can be easily verified that the set of arcs determined by $\{[v_i, v_j] \in A | u_{ij} = 1\}$ yields $T_{\overline{G}}(N)$.

Analogous formulation for the vertex-connected version of the multiconnected Steiner network problem can be obtained by transforming G to a related network using the well-known vertex-splitting technique.

The above mathematical programming model is a special case of a more general, so called *network design problem* for which various exact algorithms and heuristics are available. A survey on the network design problem has been given by Magnanti and Wong [25].

To the best of our knowledge, no specialized algorithm for the multiconnected Steiner network problem has been developed. Linear time algorithms to determine 2-connected and 2-edge-connected $T_G(N)$ when G is outerplanar or series-parallel were given by Winter [41,42]. Similar techniques are also applicable to other "tree-structured" networks such as Halin networks. These techniques are closely related to the methods described in Chapter 5.

It remains an open problem whether or not the 2-connected and 2-edge-connected multiconnected Steiner network problem can be solved efficiently in 1-planar networks. However, it can be shown that an optimal 2-connected solution will be a cycle passing through terminals in the order in which they appear on the common (exterior) face in the plane embedding [27].

Monma, Munson and Pulleyblank [27] considered the 2-edge-connected multiconnected Steiner network problem when the vertices are embedded in a metric space, and edges satisfy the triangle inequality. The optimal solution is bound to be 2-connected. Furthermore, there exists an optimal solution with all involved vertices having degree at most 3 (non-terminals must have degree at least 3; otherwise they are superfluous). The length of an optimal traveling salesman cycle

through the terminals can be shown to be not greater than 4/3 times the length of an optimal solution to the 2-edge-connected multiconnected Steiner network problem for N. Thus, solving the traveling salesman problem either exactly or by means of one of many available heuristics [23], yields a reasonable feasible solution to the 2-edge-connected multiconnected Steiner network problem in networks satisfying the triangle inequality.

The multiconnected Steiner network problem with the required local connectivities equal to 2 plays an important role in connection with the design of fiber optic communication networks. Such networks are of low cost, reliable, and provide almost unlimited capacity. Heuristics for this special case were investigated by Voss and Mehr [39] and Monma and Shallcross [28].

Any 2-connected subnetwork can be constructed by an *ear composition* approach:

- Find a cycle through a subset of terminals. It forms the initial partial solution T.
- Add to T (repeatedly) a path (ear) which starts at a vertex in T, goes through any number of terminals not yet in T, and ends in a vertex of T.

Various strategies of finding the initial cycle and subsequent ears are available.

Given a solution T obtained by any ear composition, it can be further improved by local transformations. Monma and Shallcross [28] suggested several types of local transformations. For details on ear compositions, local transformations, and for the computational comparisons of these heuristics on different kinds of networks, the reader is referred to [28].

6.9 Steiner Problem in Probabilistic Networks

Consider a network $G = (V, E, c)$ where $c : E \rightarrow [0, 1]$ denotes the probability function defined on E (i.e., $c(e_i)$ is the probability that the edge e is operational). Edge probabilities are assumed to be statistically independent.

Let N be a set of terminals in G. One of the most important reliability problems arising in broadcast networks is that of determining the probability that terminals are connected. Equivalently, this is the probability that G contains a tree spanning all terminals. Various exact algorithms and heuristics for this and other reliability problems were proposed in the literature. Although usually developed for the case $N = V$, they can be easily modified to the more general case $N \subseteq V$. For a survey, the reader is referred to Hwang, Tillman and Lee [17].

Essentially, methods to solve connectedness reliability problems fall into two classes. One class is based on a factoring approach where the reliability of a network is expressed in terms of the reliabilities of smaller networks. Another class is based on path enumeration approach. All exact methods to determine the probability that all terminals are connected require exponential time. This is in agreement with the result due to Valiant [37] that the problem is $\#P$-complete. However if G is restricted to series-parallel networks, Wald and Colbourn [40] developed a linear time algorithm. It is similar to the linear algorithm for the Steiner tree problem in series-parallel networks (Section 5.1). The same vertex elimination procedure is used but recurrence rules are modified so that they compute probabilities for the

existence of subnetworks associated with the edges during the vertex elimination process.

The linear time algorithm for the Steiner tree problem in Halin networks (Section 5.2) can in a similar manner be modified to determine the probability of connectedness of its terminals.

6.10 Realization of Distance Matrices

Another interesting problem related to the Steiner tree problem is that of the *realization of distance matrices*. Suppose that a symmetric $n \times n$ matrix D of nonnegative distances is given. Is there a network $G = (V, E, c)$ with n vertices, and with edges having appropriately chosen costs so that distances between them agree with the entries of D. Necessary and sufficient conditions for the realizability of D were given by Hakimi and Yau [16].

If D is realizable, there can be several different realizations. It is therefore natural to look for one with the smallest possible total length. Let G be a realization of D. Remove from G (one at a time) redundant edges (i.e., edges whose removal does not affect distances between vertices). Hakimi and Yau [16] proved that a network realizing D, and having no redundant edges is unique (and therefore of minimum total length).

When looking for low total length realizations of D, it can be advantageous to introduce additional non-terminals. The general problem of determining minimum length Steiner network realizing D is unsolved. Hakimi and Yau [16] suggested a heuristic based on so called *triangle transformations*. Suppose that some realization of D contains a triangle based on vertices v_i, v_j and v_k, such that every pair of the edges satisfies the triangle inequality with respect to the third edge. Delete the edges of the triangle, and introduce a non-terminal s connected to v_i, v_j and v_k by edges of length $c_{is} = (c_{ij} + c_{ik} - c_{jk})/2$, $c_{js} = (c_{ij} + c_{jk} - c_{ik})/2$, and $c_{ks} = (c_{ik} + c_{jk} - c_{ij})/2$. It can be easily verified that the new network realizes D, and its cost has been reduced by $(c_{ij} + c_{ik} + c_{jk})/2$. The heuristic starts with a complete network K_n with edge lengths equal to the required distances between the terminals. Then triangle transformations (as well as some other transformations) are applied as many times as possible.

Sometimes a network realizing D may be required to have some particular structure. For example it may be required to be a tree with exactly n vertices. Unfortunately, no characterization of distance matrices realizable by trees (or any other types of networks) is known. On the other hand, there are $O(n^2)$ algorithms generating tree realizations if they exist [3,9]. To the best of our knowledge, nothing is known about the realizability of D by Steiner trees.

6.11 Other Steiner-Like Problems

Trees spanning a given network are fundamental structures often occurring in practical applications. Many spanning tree problems where the total cost of the tree

is other than the sum of the lengths of its edges have been extensively studied in the literature. All these problems could be generalized to Steiner tree problems. Maffioli [24] and Camerini, Galbiati and Maffioli [5,6] investigated these generalizations with respect to their computational complexity. They considered both *labeled* generalizations (where terminals are given) and *unlabeled* generalizations (where only the number of terminals is given). Not surprisingly, most of these problems is NP-complete. The most important exception is the *max Steiner tree problem* where a tree spanning all terminals with the length of the longest edge being minimized is sought. Both labelled and unlabelled versions are solvable in polynomial time.

References

[1] M. Ancona, E. Bruzzone and L. De Floriani, Minimal weak Steiner trees of a structured graph, Technical Report, Univ. of Genova (1988).

[2] M. Ancona, E. Bruzzone, L. De Floriani and P. Zenere, The Steiner problem in structured graphs, Technical Report, Univ. of Genova (1988).

[3] F. T. Boesch, Properties of the distance matrix of a graph, *Q. Appl. Math.* **26** (1969) 607-610.

[4] E. Bruzzone, The Steiner problem in structured graphs, Ph.D. Thesis, Univ. of Genova (in Italian) (1987).

[5] P. M. Camerini, G.Galbiati and F. Maffioli, On the complexity of Steiner-like problems, *Proc. of the 7-th Ann. Allerton Conf. on Comm., Control and Computing* (1979) 969-977.

[6] P. M. Camerini, G.Galbiati and F. Maffioli, Complexity of spanning tree problems: Part I, *Eur. J. Oper.Res.* **5** (1980) 346-352.

[7] N. Christofides and C. A. Whitlock, Network synthesis with connectivity constraints - a survey, in J.P. Brans (ed.) *Operational Research '81*, North-Holland (1981) 705-723.

[8] C. Colbourn, The complexity of edge packing in reliability problems, Technical Report, Dept. of Computer Science, Univ. of Auckland, New Zealand (1986).

[9] J. C. Culberson and P. Rudnicki, A fast algorithm for constructing tree from distance matrices, *Inf. Process. Lett.* **30** (1989) 215-220.

[10] D. Z. Du, An optimization problem on graphs, *Discrete Appl. Math.* **14** (1986) 101-104.

[11] C. W. Duin and A. Volgenant, Some generalizations of the Steiner problem in graphs, *Networks* **17** (1987) 353-364.

[12] C. W. Duin and A. Volgenant, Reducing the hierarchical network design problem, *Eur. J. Oper. Res.* **39** (1989) 332-344.

[13] C. W. Duin and A. Volgenant, The multi-weighted Steiner tree problem, *Ann. Oper. Res.* **33** (1991) 451-469.

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

[15] S. L. Hakimi, Steiner's problem in graphs and its applications, *Networks* **1** (1971) 113-133.

[16] S. L. Hakimi and S. S. Yau, Distance matrix of a graph and its realizability, *Q. Appl. Math.* **22** (1965) 305-317.

[17] C. L. Hwang, F. A. Tillman and M. H. Lee, System-reliability evaluation: Techniques for complex/large systems - a review, *IEEE Trans. Reliab.* **R30** (1981) 416-423.

[18] E. Ihler, Bounds on the quality of approximate solutions to the group Steiner problem, in *Proc. of the 16-th Int. Workshop on Graph-Theoretic Concepts in Computer Science, Lecture Notes in Computer Science* **484** (1991) 109-118.

[19] E. Ihler, The complexity of approximating the class Steiner tree problem, Technical Report, Inst. für Informatik, Univ. Freiburg (1991).

[20] A. Iwainsky, Optimal trees - a short overview on problem formulations, in A. Iwainsky (ed.), *Optimization of Connection Structures in Graphs* , CICIP, East Berlin, GDR (1985) 121-133.

[21] A. Iwainsky, E. Canuto, O. Taraszow and A. Villa, Network decomposition for the optimization of connection structures, *Networks* **16** (1986) 205-235.

[22] J. Krarup, Generalized Steiner problem, Unpublished note (1978).

[23] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan and D. B. Shmoys (eds.), *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, John Wiley & Sons, Chichester (1985).

[24] F. Maffioli, Complexity of optimum undirected tree problems: A survey of recent results, in G. Ausiello and M. Lucertini (eds.), *Analysis and Design of Algorithms in Combinatorial Optimization*, CISM 266, Springer-Verlag (1981) 107-128.

[25] T. L. Magnanti and R. T. Wong, Network design and transportation planning: Models and algorithms, *Transp. Sci.* **18** (1984) 1-55.

[26] S. Martello and P. Toth, Finding a minimum equivalent graph of a digraph, *Networks* **12** (1982) 89-100.

[27] C.L. Monma, B.S. Munson and W.R. Pulleyblank, Minimum weight two-connected spanning networks, *Math. Program.* **46** (1990) 153-171.

[28] C. L. Monma and D. F. Shallcross, Methods for designing communications networks with certain two-connected survivability constraints, *Oper. Res.* **37** (1989) 531-541.

[29] L. Nastansky, S. M. Selkow and N. F. Stewart, Cost minimal trees in directed acyclic graphs, *Z. Oper. Res.* **18** (1974) 59-67.

[30] C. H. Papadimitriou and M. Yannakakis, Optimization, approximation, and complexity classes, *Proc. of the 20-th Annual ACM Symp. on Theory of Computing* (1988) 229-234.

[31] G. Reich and P. Widmayer, Beyond Steiner's problem: A VLSI oriented generalization, *Proc. of the 15-th Int. Workshop on Graph-Theoretic Concepts in Computer Science, Lecture Notes in Computer Science* **411** (1989).

[32] G. Reich and P. Widmayer, Approximate minimum spanning trees for vertex classes, Technical Report, Inst. für Informatik, Freiburg Univ. (1991).

[33] M. B. Richey and R. G. Parker, On multiple Steiner subgraph problems, *Networks* **16** (1986) 423-438.

[34] M. B. Richey, R. G. Parker and R. L. Rardin, An efficiently solvable case of the minimum weight equivalent subgraph problem, *Networks* **15** (1985) 217-228.

[35] S. Sahni, Computationally related problems, *SIAM J. Comput.* **3** (1974) 262-279.

[36] A. Segev, The node-weighted Steiner tree problem, *Networks* **17** (1987) 1-17.

[37] L. G. Valiant, The complexity of enumeration and reliability problem, *SIAM J. Comput.* **8** (1979) 410-421.

[38] S. Voss, A survey on some generalizations of Steiner's problem, Paper presented at 1-st Balkan Conf. on Operational Research (1988).

[39] S. Voss and K. D. Mehr, On a generalized Steiner problem with 2-edge-connectivity, *Methods Oper. Res.* **57** (1987) 161-172.

[40] J. A. Wald and C. J. Colbourn, Steiner trees in probabilistic networks, Technical Report 82-7, Dept. of Comp. Science, Univ. of Saskatchewan (1982).

[41] P. Winter, Generalized Steiner problem in outerplanar networks, *BIT* **25** (1985) 486-496.

[42] P. Winter, Generalized Steiner problem in series-parallel networks, *J. Algorithms* **7** (1986) 549-566.

[43] P. Zenere, Algorithms for the Steiner problem in structured graphs, Ph.D. Thesis, Univ. of Genova (in Italian) (1987).

Part III

Rectilinear Steiner Problem

This Page Intentionally Left Blank

Chapter 1

Introduction

Hanan [8] first considered the rectilinear version after the Euclidean problem was attacked and before the network version was defined. However it needs to be discussed after both of the other versions since it builds on the techniques used for both the Euclidean and network versions, primarily the latter. The rectilinear problem is defined analogously to the Euclidean setting with only a change to the distance metric. However that dramatically changes the set of possible solutions, producing a situation analogous to the network setting.

The first section introduces many definitions. Section 1.2 discusses basic properties of special trees that are used in the next section to give a characterization of rectilinear Steiner minimal trees. Section 1.4 introduces several important problem reduction techniques. Extremal results and computational complexity are discussed in the next two sections. Section 1.7 presents some exact algorithms. De Souza and Ribeiro presented a survey of the rectilinear Steiner problem [15].

1.1 Definitions

Unless otherwise specified, this Part inherits terminology from the two preceding Parts of this book. For example, the input is N, a given set of n terminals in the plane. Since the rectilinear setting sits between the other two settings, occasionally it is convenient to introduce new terminology to avoid confusion.

Formally, a (rectilinear) *segment* is a horizontal or vertical line segment connecting its two endpoints in the plane. A *rectilinear tree* is a connected acyclic collection of segments, which intersect only at their endpoints. The *degree* of a point is the number of segments incident on it. A *rectilinear Steiner tree* for a given set of terminals is a rectilinear tree such that each terminal is an endpoint of a segment in the tree. The length of a tree is the sum of the lengths of its segments and a *rectilinear Steiner minimal tree* (RSMT) is such a tree of minimum length. Without loss of generality, if a degree two point has collinear incident segments then that point must be a terminal.

Clearly a point has degree ≤ 4. A point with degree ≥ 3 that is not a terminal is called a *Steiner point*. A *corner-point* is a degree two point with non-collinear incident segments that is not a terminal. Hence each endpoint of a segment is either a terminal, a Steiner point, or a corner-point. Two points are *directly connected* if they are connected by a sequence of one or more segments properly containing only corner-points; such a sequence of edges is called a *wire* (as suggested by the prevalent wire-routing application). In an RSMT two points i and j that are directly connected are connected by a wire of total length

$$d(i,j) = |x_i - x_j| + |y_i - y_j|$$

and such a wire has a "staircase" shape. In the following chapters "point" refers to the endpoint of a segment, unless the context clearly indicates a more generic usage.

A *line* is defined to be a sequence of one or more adjacent, collinear segments with no terminals in its relative interior. (Relative interior is used in its usual geometric sense.) A *complete line* is a line of maximal length; it does not have terminals in its relative interior, but it may have terminals as endpoints. A corner-point is incident to exactly one horizontal complete line and exactly one vertical complete line. These complete lines are the *legs* of the *corner*. If the remaining two endpoints of the leg are terminals, the corner is called a *complete corner*. A T-*point* is a degree three Steiner point. The two collinear segments are part of a complete line called the *head* of the T, and the remaining segment is part of a complete line called the *body*. Finally, a *cross-point* is a degree four Steiner point. The *spokes* of a cross-point are the four lines, of maximal length, meeting at the cross-point, which do not have terminals in their relative interior. See Fig. 1.1 for an illustration of these definitions. A line connects g and k, while a complete line connects e and g (but not d and g). The corner point a has legs ab and ak and is a complete corner. The T-point f has body cf and head eg. Finally, i is a cross-point, with spokes ih, ia, ij, and ik.

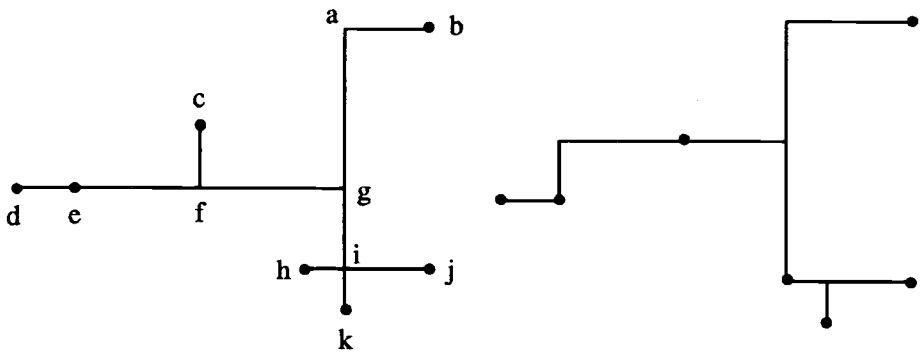


Figure 1.1: An RSMT and a fulsome RSMT for the same points

1.1.1 Full and Fulsome Trees

For any given RSMT T there is a corresponding tree graph $G(T)$ which, as in the Euclidean case, is called a *topology*. The vertices of $G(T)$ correspond to the terminals and Steiner points of T. Each edge $[i,j]$ of $G(T)$ connects the vertices i and j iff the corresponding points are directly connected in T by a wire.

By analogy with in the Euclidean case, a *full topology* is defined as a topology with each terminal having degree one. Clearly for each RSMT T the topology $G(T)$ is the union of full topologies joined at terminals with degree ≥ 2. Let $deg(i)$ be the degree of terminal i. The number of full topologies is $1 + \sum_{i \in N}(deg(i) - 1)$. Clearly the number of full topologies is maximized when $s(T) = \sum_{i \in N} deg(i)$ is maximized. It turns out that as there are more full topologies, each one tends to be simpler in structure. Therefore the set of RSMTs that have the maximum value of $s(T)$ are explored; such an RSMT is (whimsically) called a *fulsome RSMT*. A *full RSMT* is an RSMT with a full topology; in this case each terminal has degree one. In Fig. 1.1 an RSMT is shown for a set of seven terminals that is not fulsome, together with a fulsome RSMT for the same terminals. Note that the fulsome RSMT shown is the union of four subtrees, each a full RSMT.

1.1.2 Canonical Trees

Two transformations for rectilinear Steiner trees are defined: flipping and sliding. Each transformation maps one such tree to another without moving the positions of terminals and without increasing the length of the tree. Depending on the direction, there are four slides and four flips; example flips and slides are depicted in Fig. 1.2. More formally, in a slide, a segment e, perpendicular and incident to parallel lines l_1 and l_2, is replaced by a segment e'. This segment e' is also incident to l_1 and l_2, has the same length as e, and is parallel to e. The slide is said to be applied to segment e. In a flip, two segments e and f meeting at a corner-point are replaced by two other segments e' and f' such that e, f, e' and f' form a rectangle. The flip is said to be applied to the corner where segments e and f meet at a corner-point. Each slide and flip is named according to the position of the new segments relative to the original segments; see Fig. 1.2.

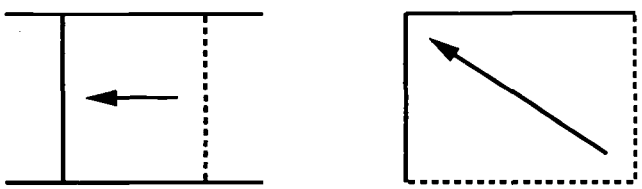


Figure 1.2: Examples of transformations: a W-slide and NW-flip

Unlike the Euclidean case, the locations of the Steiner points are not necessarily restricted to finite number of positions. This is easily seen for the case of just four terminals at the corners of a square; there are an infinite number of H-shaped

RSMTs. Further the embedding of the segments of a wire is not unique. Even if the number of corners in a wire is restricted, there can always be connections with a corner that can be flipped. For such reasons a canonical representation of an RSMT should be defined. There are several possible definitions; the following was found useful.

A tree is said to be *canonical* if it is impossible to apply any one of the following transformations without changing $s(T)$: (1) a W-slide, (2) a NW-flip or SW-flip, or (3) a N-slide or S-slide followed by transformation (2). An example of the third transformation is seen in Fig. 1.1; a N-slide of segment fg exposes a corner at f that allows a NW-flip. There must be at least one canonical Steiner tree since it is impossible to start with any tree T and apply one or more of the three transformations to obtain T, as each transformation replaces a vertical segment by another further to the left.

1.2 Basic Properties

In this section several characterizations of fulsome canonical trees are proved that will be used in the next section to characterize the topologies of general RSMTs.

Lemma 1.1 *In a canonical fulsome RSMT, no segment is incident to two corner-points. Further, the horizontal leg of a corner must be to the right of the vertical leg.*

Proof: An RSMT having a segment e incident to two corner-points either can be made shorter by sliding e (when the corners bend in the same direction) or is not canonical since one of the corners incident to e can be flipped (when the corners bend in opposite direction). The second proposition follows since any such corner could have been flipped. □

Lemma 1.2 *Let m be a segment with non-terminal endpoints a and b. In a canonical fulsome RSMT no two perpendicular segments incident to a and b, respectively, are on the same side of m.*

Proof: Suppose to the contrary that there are segments e and f incident to a and b on the same side of m (see Fig. 1.3). Let e be a segment in a complete line extending from point u to point s, and let f be a segment in a complete line extending from point v to point t. Also let u and v be on the same side of m as e and f are. Note that there may be other segments w, x, y, and z incident to a and b as shown. Note that neither a nor b can be corner-points, otherwise a flip could be used to decrease length.

One of u and v, say u, is not further away from m than the other, as measured by their perpendicular distance to m. Edge m can be slid towards u without increasing length, so no segments of the Steiner tree are in the rectangle bounded by e, f, m, and a line parallel to m passing through u. This includes segments at u, so u cannot be a T-point, a cross-point, or a corner-point whose other leg points towards f. Also, u cannot be a terminal, otherwise sliding m to u increases

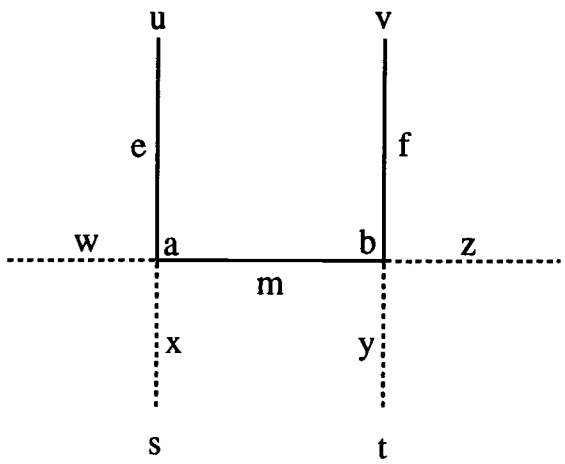


Figure 1.3: Two adjacent lines on the same side of segment m

terminal degree. (Note that this is an essential place where the fact that the tree is fulsome is used.) Therefore, u must be a corner-point whose other leg points away from f.

Since u is a corner-point, no segment can be incident on the line from u to a on the same side as w, including w itself, otherwise the corner at u could be flipped to decrease length. Since a is not a corner-point the segment x must exist and $s \neq a$. Since b is also not a corner-point, one of y and z must exist. Suppose that y exists; therefore, the complete line containing f connects v to t, where $t \neq b$. One of s and t is not further from m than the other, and the argument above implies that this point must be a corner-point whose other leg is not between x and y. If s is no further from m than t then s can be flipped to decrease length or the corners at u and s can be flipped together to decrease length. Therefore, if y exists, t must be closer to m than s, t must be a corner-point, and there are legs incident to u and t pointing in opposite directions. Further, note that a flipping argument, as above, implies that if y does exist, z cannot exist.

Therefore two cases are left: (1) point u is a corner-point no further from m than v, t is a corner-point closer to m than s, w and z do not exist, and there are legs incident to u and t pointing in opposite directions, and (2) point u is a corner-point no further from m than v, y does not exist, and z must exist. Our description so far has been independent of the orientation. There are eight orientations of each case and none is canonical, giving a contradiction. □

A segment e is *properly incident* to a perpendicular line l if e intersects the relative interior of l but does not intersect at a terminal; it can intersect at an endpoint of l that is a Steiner point or a corner-point. Two segments e and f properly incident to l are said to be *neighboring segments* if the line contained in l connecting $e \cap l$ and $f \cap l$ is a single segment. Neighboring segments are said to *alternate* along l if no two neighboring segments are on the same side of l and no

cross-point is in the relative interior of l.

Corollary 1.1 *In a canonical fulsome RSMT, neighboring segments incident to a leg of a corner must alternate. In particular, the segment closest to the corner-point must point in the opposite direction as the other leg of the corner.*

Corollary 1.2 *In a canonical fulsome RSMT, the body of a* T *must end at a terminal, and no segments are properly incident to the body of a* T. *Furthermore, each spoke of a cross-point ends at a terminal, and no segments are properly incident to the spoke,*

Lemma 1.3 *In a canonical fulsome RSMT, corners must be complete corners, and at most one of the legs can have more than one properly incident segment.*

Proof: Corollary 1.1 implies that a leg of a corner cannot end in a T-point or a cross-point. Leftness or minimal tree length implies a leg of a corner cannot end at a second corner, so it ends at a terminal; this is essentially the same argument as for Lemma 1.1.

To show that only one leg can have more than one incident segment, suppose to the contrary that each leg has at least two incident segments. Corollary 1.2 implies that any segment incident to a leg must end at a terminal and not properly intersect any other segments. Suppose the corner has the orientation depicted in Fig. 1.4; a symmetric argument can be made for the other orientation.

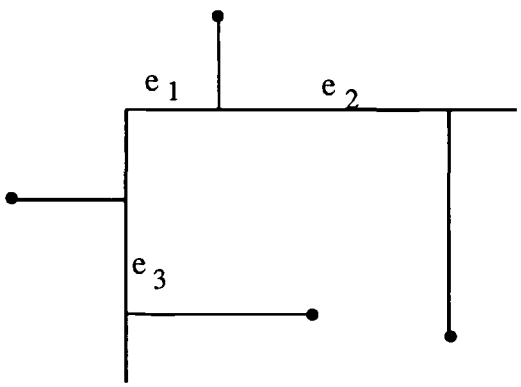


Figure 1.4: A corner with two lines incident to both legs

Terminal a must be below c and d must be to the right of b, or else a SE-flip on the corner and either an S-slide on e_2 or an E-slide on e_3 increases the sum of the degrees of the terminals. In this case, however, a SE-flip on the corner, an S-slide on e_2, and an E-slide on e_3 until e_3 hits terminals a or d increases the sum of the terminal degrees without increasing length; see Fig. 1.5. □

Lemma 1.4 *In a canonical fulsome RSMT, the head of a* T *is either a leg of a complete corner or a complete line with terminals as endpoints.*

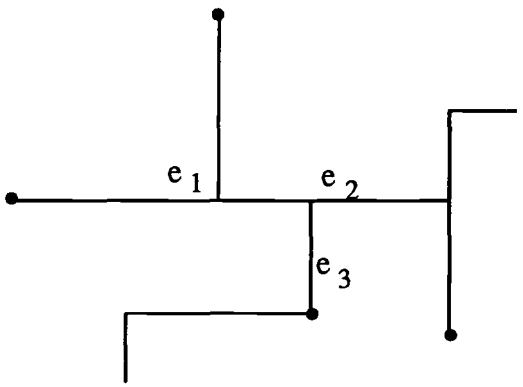


Figure 1.5: A transformation of the corner in Fig. 1.4

Proof: If one side of the head ends at a corner-point, then Lemma 1.3 implies that the head is one leg of a complete corner. Corollary 1.2 states that it is not possible for the head of a T to be the body of another T-point; similarly it cannot intersect a cross-point. So the only other possibility is that both sides of the head intersect terminals. □

1.3 A Characterization of RSMTs

In the preceding section the form of canonical fulsome trees was discussed. They are somewhat specialized. However for every set of terminals there clearly exists an RSMT which is the union of such trees. This observation is made in the corollary to the following theorem. The theorem is due to Hwang [9] and the proof is by Richards and Salowe [13].

Theorem 1.1 *Let $n > 4$. Suppose every RSMT for a given set of terminals is full. Then there exists an RSMT that either consists of a single complete line with $n-2$ alternating incident segments, a complete corner with $n-2$ alternating segments incident to a single leg, or a complete corner with $n-3$ alternating segments incident to one leg and a single segment incident to the other leg.*

Proof: Note that such a tree is both full and fulsome. Since each terminal must be a leaf, there is a single complete line incident to the topmost terminal. If this complete line is horizontal, it must end in a corner-point, forming a complete corner by Lemma 1.3. (If the complete line ended at a second terminal, then the assumption on terminal degree implies that it is the head of a T which, by Corollary 1.2, contains only three terminals.) The $n-2$ terminals not on the legs of the complete corner must connect to this complete corner. Since there are no terminals above the horizontal leg of the corner, Lemma 1.3 implies that all incident segments alternate off the vertical leg. Corollary 1.2 implies each incident segment

ends at a terminal and no other terminals can be reached from the leg across such a segment. Hence there are $n-2$ incident segments.

If the segment incident to the topmost terminal is vertical, then it either forms a complete line ending at another terminal, or it ends at a corner-point, or it ends at a T-point. Lemma 1.4 implies that if the complete line ends at a T-point, the head of the T forms either a complete corner or a complete line having terminals as endpoints. In each case, the analysis is similar to that given in the previous paragraph and is omitted. □

In other words, if a set of terminals has a fulsome RSMT that is also a full RSMT then it has an RSMT of one of the (generic) forms in Fig. 1.6.

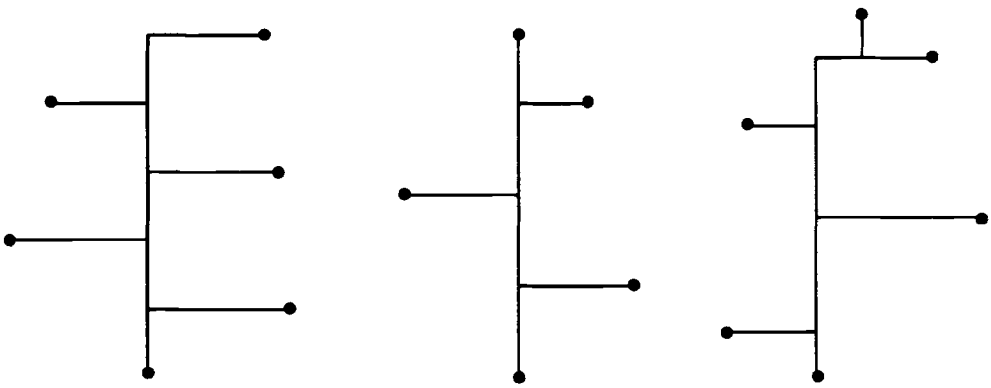


Figure 1.6: The only generic forms for a full RSMT that is fulsome, $n > 4$

Note that every fulsome RSMT is the union of subtrees, each of which is full (and fulsome) relative to the terminals it spans. These subtrees are joined at terminals of degree 2, 3, or 4.

Corollary 1.3 *For any set of terminals, $n > 4$, there exists an RSMT that is the union of full subtrees, each subtree in one of the forms shown in Fig. 1.6, joined at terminals of degree 2, 3, or 4.*

Hanan [8] analyzed the smallest RSMTs and proved the following Lemma using simple case analyses. It indicates a situation where an RSMT has a topology with a Steiner point of degree four. This did not arise in the Euclidean case and this is the only instance of it in RSMTs.

Lemma 1.5 *Theorem 1.1 holds for $n = 2, 3,$ or 4 except for when $n = 4$ and the four terminals are the endpoints of a cross as shown in Fig. 1.7.*

1.4 Problem Reductions

Hanan [8], in the first paper on RSMTs, gave a strong restriction on the class of possible solutions. He showed that there is a set of $O(n^2)$ points in the plane such

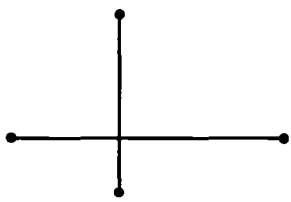


Figure 1.7: A cross is the only exception to Theorem 1.1

that there exists an RSMT with all of its Steiner points in that set.

Theorem 1.2 *Let the coordinates of the terminals be* $(x_1, y_1), ..., (x_n, y_n)$. *There exists an RSMT* T *where if* (x', y') *is a Steiner point of* T *then* $x' = x_i$ *and* $y' = y_j$, $1 \leq i, j \leq n$.

Proof: Recall that there exists an RSMT that is the union of subtrees, each a canonical full RSMT for the terminal set of its leaves. Therefore, since each Steiner point is in one of these subtrees, Theorem 1.1 says that it is connected by a line to at least one terminal in a horizontal direction and at least one terminal in a vertical direction. □

Actually the above argument proves a stronger characterization which has the following corollary. A *trombone wire* is a segment of an RSMT T that can be slid back and forth between two perpendicular segments without altering the length of T, which is not incident on a terminal. Snyder [14] argued that the presence of trombone wires is what made Hanan's original proof detailed and hard to generalize to higher dimensions; see Section 4.4.

Corollary 1.4 *There exists an RSMT for any set of terminals without trombone wires.*

Proof: Add to the argument in the above proof the fact that Theorem 1.1 guarantees alternating segments on each line. The only candidate segments in a fulsome RSMT are the segments within each full subtree that are not incident with a terminal. If such a candidate could slide, it would violate the alternation condition. □

Hanan's result can be phrased in graph-theoretic terms. A *grid graph* $GG(N) = (V, E)$ for a terminal set N is defined. Begin by constructing vertical and horizontal lines through each terminal. Let V be identified with the set of intersection points; $N \subseteq V$. There is an edge in E between two vertices iff the corresponding intersection points are directly connected by a single horizontal or vertical segment; the length of the edge is the length of the segment. It follows from Theorem 1.2 that any RSMT can be mapped to a Steiner minimal tree of the network $GG(N)$. Note that vertices corresponding to N would be the terminals of $GG(N)$. Further any Steiner minimal tree of $GG(N)$ corresponds to an RSMT. Hence the results for the Steiner problem in networks from Part II can be immediately applied to

the rectilinear Steiner problem. Most of the network-based algorithms in this Part of the book are based on $GG(N)$, but many of these only use $GG(N)$ implicitly (by constraining their attention to vertices in V without generating V completely). Note that since $|V| = O(n^2)$ any sub-quadratic algorithm must use the grid graph implicitly. In fact only a subnetwork of $GG(N)$ needs to be considered.

The *rectilinear convex hull* of N, $Rconv(N)$, is a smallest-area simply-connected figure containing a shortest rectilinear path between every pair of terminals in N. Equivalently, if one places a coordinate axis at any point on the boundary of $Rconv(N)$, at least one of the quadrants is empty. The boundary $B(N)$ of $Rconv(N)$ consists of vertical and horizontal segments.

Theorem 1.3 *There exists an RSMT T for N such that all of the segments in T lie on or inside $Rconv(N)$.*

Proof: The theorem is shown for a set N with a full RSMT. Clearly the union of the rectilinear convex hulls of various subsets of terminals (corresponding to full subtrees of canonical fulsome tree) must be contained in $Rconv(N)$.

That it is true for a full RSMT follows from Theorem 1.1. Consider, for example, the full RSMT shown in Fig. 1.8. Its corner can be flipped to remain inside its rectilinear convex hull. (Of course, such a tree is not necessarily canonical.) The other cases in Theorem 1.1 are similar. □

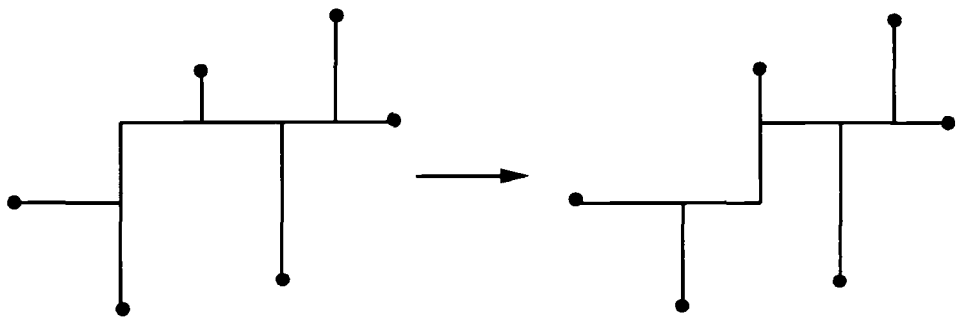


Figure 1.8: A full RSMT folded to fit in its rectilinear convex hull

As in the Euclidean case, the Steiner hull is a region known to contain a Steiner tree. In the rectilinear case, the Steiner hull would just be the set of segments of $GG(N)$ that are on or within $Rconv(N)$.

The boundary $B(N)$ contains a series of *boundary segments*. Boundary segments can meet at *boundary corners*. With respect to the interior of $Rconv(N)$, an *inner boundary corner* is a reflex angle, and an *outer boundary corner* is a convex angle. Note that all outer boundary corners are terminals. A *boundary line* is a series of collinear adjacent boundary segments. There are special boundary lines called *tabs*. Tabs connect two outer boundary corners and are at the positions as far to the north, south, east, and west as possible on $B(N)$. There are at most four tabs, by convexity. If, for example, there is a unique terminal with maximum

ordinate then the north tab is degenerate and of zero length. Each consecutive pair of tabs, around the boundary, is connected by a sequence of zero or more boundary lines forming *staircases*.

The rectilinear convex hulls are more confusing than the Euclidean case because of degeneracies. However these cases present no theoretical difficulties and can be easily handled in practice by preprocessing.

Tabs may be degenerate and contain only one point. There is another type of degeneracy where regions in $Rconv(N)$ are connected by rectilinear lines. These two degeneracies are depicted in Fig. 1.9. In order to remove these degeneracies, the rectilinear convex hull is preprocessed. Suppose the left tab is degenerate, and let a be the single leftmost point in $Rconv(N)$. By Theorem 1.3, one can add a new terminal b as depicted and remove the line to a to get a new problem with a nondegenerate tab. All tabs can be processed in this manner. Similarly, if regions are separated by rectilinear lines, Theorem 1.3 implies that the rectilinear convex hull can be separated into two or more nondegenerate subproblems; Fu [5] gives the details. (Note that the regions can be identified with a simple linear-time scan [12], given the points in sorted order by their coordinates.) Therefore it is assumed in the text below that no degeneracies are present.

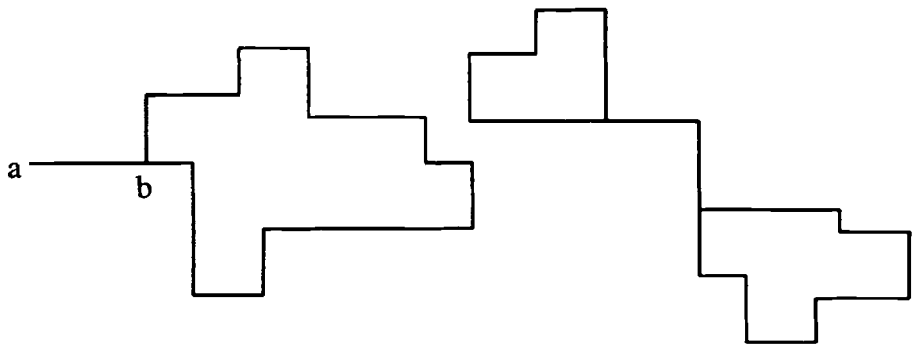


Figure 1.9: Two degeneracies: a degenerate tab and two separated regions

1.5 Extremal Results

In this section several bounds are given on the length of an RSMT, in various contexts. Let $RSMT(N)$ be an RSMT for N, $RMST(N)$ is the rectilinear minimum spanning tree (RMST) for N, and $|T|$ be the total rectilinear length of tree T. Clearly $|RMST(N)| \geq |RSMT(N)|$. Note that in an RMST only terminals can be connected to each other via wires, and these various wires may have overlapping segments. In computing $|RMST(N)|$, of course, a segment in two overlapping wires would be double-counted.

The *Steiner ratio* for the rectilinear case was settled by Hwang [9]. This is useful as many algorithms are based on some RMST.

Theorem 1.4 *For any terminal set N the Steiner ratio*

$$\rho(N) = \frac{|RSMT(N)|}{|RMST(N)|} \geq \frac{2}{3}$$

Proof: A proof by Salowe (private communication) is sketched which captures the main idea of the longer proof in [9]. The bound only needs to be established for each full RSMT; the sum of the lengths of the RMSTs for the terminal sets of each full subtree (of a fulsome RSMT) is lower bounded by $|RMST(N)|$. The argument is by induction on n. The basis, $n \leq 4$ is easily established [8].

In any full RSMT with $n > 4$ one can find a subtree as shown in Fig. 1.10, where the edge lengths satisfy $f > d$ and $e > g$. Let A be the set of terminals above x and B is the set of terminals below y. (There can be degeneracies in A or B but the argument is unchanged.) Let $C = \{w, x, y, z\}$. One can easily show $|RMST(C)| \leq \frac{3}{2}(a + b + c + e + f)$, by considering the rectangle bounding C. Let S_A be the sum of the edges in the full tree above the edge of length a; S_B is defined similarly for the edges below the edge of length c. Clearly $|RSMT(N)| = S_A + (a + b + c + e + f) + S_B$. By the inductive hypothesis, $|RMST(A)| \leq \frac{3}{2}S_A$ and $|RMST(B)| \leq \frac{3}{2}S_B$. Therefore

$$|RMST(N)| \leq |RMST(A)| + |RMST(C)| + |RMST(B)| \leq \frac{3}{2}|RSMT(N)|$$

□

It is easy to establish that 2/3 is the optimal bound by considering the case where N is the four terminals each a unit distance from, say, the origin. Clearly $|RSMT(N)| = 4$ and $|RMST(N)| = 6$. Further, by clustering more terminals very near these four terminals, there exist N that are arbitrarily close to the 2/3 bound for all n.

Let $s(n, a, b)$ be the length of the longest RSMT for any n terminals in an $a \times b$ rectangle, and $s(n) = s(n, 1, 1)$, i.e., for the unit square. Chung and Graham [2], extending the results of Few [4], proved $s(n, a, b) \leq \frac{1}{2}(a+b+at+\frac{bn}{t})$ for an arbitrary positive integer t, when $a \leq b$. The proof begins by dividing the rectangle into subrectangles of size $a \times \frac{b}{t}$ with horizontal wires and then slides these wires up and down into convenient positions; details are in [2]. By choosing $t = \sqrt{\frac{bn}{a}} + o(1)$ one gets $s(n, a, b) \leq \sqrt{abn} + \frac{1}{2}(a + b) + o(1)$. Hence $s(n) \leq \sqrt{n} + 1 + o(1)$. Since there are examples formed from subsets of the square lattice with $s(n) \geq \sqrt{n} + O(1)$, the upper bound is very tight. Further, $s(r^2) = r + 1$. Let $Rect(N)$ be the smallest area bounding rectangle of the set N, with horizontal and vertical sides. Let $L_R(N)$ be the *semiperimeter* of $Rect(N)$; $L_R(N) = a + b$ when $Rect(N)$ is an $a \times b$ rectangle. Chung and Hwang [3] defined $\hat{\rho}(N) = \frac{|RSMT(N)|}{L_R(N)}$ and $\hat{\rho}(n) = \max_{|N|=n}\{\hat{\rho}(N)\}$. The semiperimeter has been recommended as a suitable and easily computed estimate of the RSMT. The results for $s(n, a, b)$ imply that $\hat{\rho}(r^2) = \frac{1}{2}(r + 1)$, for a positive integer r. Since $\hat{\rho}(n)$ is monotone nondecreasing, $\frac{1}{2}(\sqrt{n}+1)$ is a reasonable estimate for $\hat{\rho}(n)$. Chung and Hwang [3] determined $\hat{\rho}(n)$ exactly

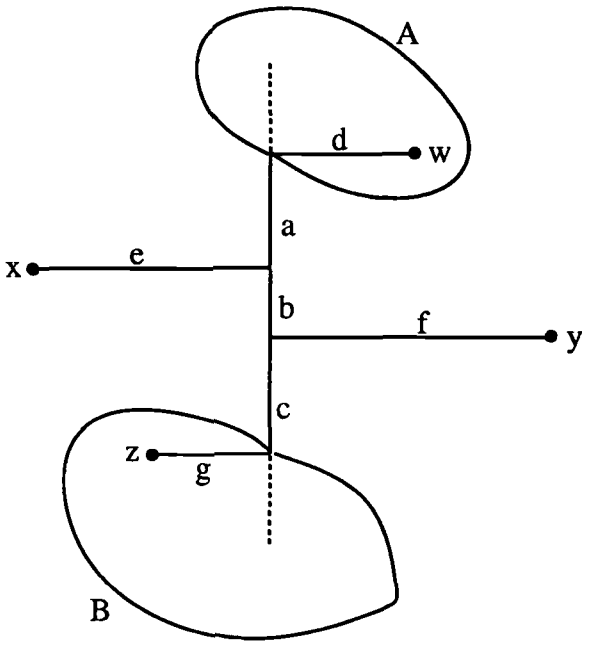


Figure 1.10: A wide subtree within a full subtree

for $n \leq 10$ using fairly involved arguments.

n:	2	3	4	5	6	7	8	9	10	11
$\hat{\rho}(n)$:	1	1	$\frac{3}{2}$	$\frac{3}{2}$	$\frac{5}{3}$	$\frac{7}{4}$	$\frac{11}{16}$	2	2	$\geq \frac{33}{16}$

These are particularly useful since in VLSI applications often $n \leq 10$ and the bounds are relevant.

Suppose N is uniformly distributed over the unit square. Let $L_S(n)$ be the random variable for the length of an RSMT for such a N. Let $\lambda(x)$ be the minimum Euclidean distance from x to another point in N. Komlós and Shing [11] have shown that with probability $1 - o(1)$, $\frac{1}{2}\sum_{x \in N} \lambda(x) = (\frac{1}{4} + o(1))\sqrt{n}$. Since $L_S(N)$ is lower-bounded by the left-hand side, they were able to show $L_S(n) \geq \frac{1}{5}\sqrt{n}$ with probability $1 - o(1)$. An upper bound on $L_S(n)$ is provided by a result by Gilbert [7] for random RMSTs. Let $L_M(n)$ be the random variable for the length of an RMST for n points drawn uniformly from the unit square. Gilbert showed that $L_M(n) < 0.71\sqrt{n}$ asymptotically.

Bern [1] analyzed the expected number of Steiner points in a RSMT drawn from the unit square. Bern's result was for a Poisson distribution with intensity n. Hwang and Yao [10] extended the analysis to the uniform distribution. The expected number of Steiner points in both cases is $\geq 0.035n$, in the limit.

1.6 Computational Complexity

Garey and Johnson [6] have established the NP-completeness of the rectilinear Steiner tree decision problem. In particular, the following problem is NP-complete:

- **GIVEN:** Set N of points in the plane with integer coordinates and a positive integer B.
- **DECIDE:** Is $|RSMT(N)| \leq B$?

Unlike the Euclidean case, it is trivial to show the problem is in NP. This follows from Theorem 1.2. Since the candidate Steiner points can be selected with integer coordinates it follows that if provided with the set of Steiner points used in some RSMT the length can be verified in polynomial time.

The proof that the problem is NP-hard is done by a reduction from the *vertex cover problem* A *vertex cover* of an undirected graph is a subset of the vertices such that each edge is incident to some vertex in that subset. The proof of this reduction is very lengthy and is omitted here.

Recall that Theorem 1.2 has an alternative interpretation: the RSMT corresponds to a Steiner minimal tree of $GG(N)$. Since $GG(N)$ is a planar network it follows readily that the Steiner problem for networks decision problem is NP-complete, even when restricted to planar networks (as was mentioned in Section 1.5 of Part II).

1.7 Exact Algorithms

Any exact algorithm for the network case, discussed in Chapter 3 of Part II, can be used to solve the rectilinear problem. Theorem 1.2 permits the reduction of any rectilinear problem to an analogous problem on $GG(N)$. No systematic exploration of the applicability of the many exact algorithms proposed for the network case has yet been done. It has often been remarked that the dynamic programming approach suggests itself since n is small, relative to the number of vertices in $GG(n)$.

The first optimal algorithm in the literature is due to Yang and Wing [16,17]. It is essentially a variant of the branch-and-bound algorithm for networks. In fact it is a slightly inferior version. The largest problem they report solving had $n = 9$, and the underlying network had 20 vertices and 31 edges (they used the rectilinear convex hull pruning of $GG(N)$). This example took 255 seconds.

Another exact algorithm, Exhaustive Partitioning, is discussed in Section 2.4.4.

Hanan [8] described optimal algorithms for $n \leq 5$. These are useful since they are used within many heuristics, discussed in the next chapter. In particular, for $n = 3$, the single Steiner point, if it exists, is easily located: its x-coordinate is the median of the x-coordinates in N, and the y-coordinate is handled in the same way.

References

[1] M. W. Bern, Two probabilistic results on rectilinear Steiner trees, *Algorithmica* **3** (1988) 191–204.

[2] F. R. K. Chung and R. L. Graham, On Steiner trees for bounded point sets, *Geom. Dedicata* **11** (1981) 353–361.

[3] F. R. K. Chung and F. K. Hwang, The largest minimal rectilinear Steiner trees for a set of n points enclosed in a rectangle with given perimeter, *Networks* **9** (1979) 19–36.

[4] L. Few, The shortest path and shortest road through n points, *Mathematika* **2** (1955) 141–144.

[5] Y. Fu, Application of linear graph theory to printed circuits, *Proc. Asilomar Conf. Systems and Circuits* (1967) 721–728.

[6] M. R. Garey and D. S. Johnson, The rectilinear Steiner tree problem is NP-complete, *SIAM J. Appl. Math.* **32** (1977) 826–834.

[7] E. N. Gilbert, Random minimal trees, *J. SIAM* **13** (1965) 376–387.

[8] M. Hanan, On Steiner's problem with rectilinear distance, *J. SIAM Appl. Math.* **14** (1966) 255–265.

[9] F. K. Hwang, On Steiner minimal trees with rectilinear distance, *SIAM J. Appl. Math.* **30** (1976) 104–114.

[10] F. K. Hwang and Y. C. Yao, Comments on Bern's probabilistic results on rectilinear Steiner trees, *Algorithmica* **5** (1990) 591–598.

[11] J. Komlós and M. T. Shing, Probabilistic partitioning algorithms for the rectilinear Steiner problem, *Networks* **15** (1985) 413–423.

[12] T. M. Nicholl, D. T. Lee, Y. Z. Liao and C. K. Wong, On the X-Y convex hull of a set of X-Y polygons, *BIT* **23** (1983) 456–471, (1983).

[13] D. S. Richards and J. S. Salowe, A simple proof of Hwang's theorem for rectilinear Steiner minimal trees, *Ann. Oper. Res.* **33** (1991) 549-556.

[14] T. L. Snyder, On the exact location of Steiner points in general dimension, Technical Report, Georgetown Univ. (1990).

[15] C. C. De Souza and C. C. Ribeiro, O problema de Steiner na métrica retilínea (in Portugese), *Investigacion Operativa* **1** (1990) 213–249.

[16] Y. Y. Yang and O. Wing, An algorithm for the wiring problem, *Digest of the IEEE Int. Symp. on Electrical Networks* (1971) 14–15.

[17] Y. Y. Yang and O. Wing, Supoptimal algorithm for a wire routing problem, *IEEE Trans. on Circuit Theory* **CT-19** (1972) 508–510.

This Page Intentionally Left Blank

Chapter 2

Heuristic Algorithms

Since the rectilinear Steiner problem is NP-hard, the bulk of applied research has been on heuristic approximation algorithms, as well as tractable special cases (discussed in the next chapter). Many of the algorithms have analogues discussed in Part II. However, there are also many analogues to Euclidean heuristics, primarily due to the pervasive reliance on minimum spanning tree techniques.

This chapter begins with heuristics that make straightforward use of a given RMST. In the next section analogues to algorithms that construct MSTs are discussed. In Section 2.3 heuristics suggested by computational geometry techniques are presented. In the last section less typical approaches are presented.

2.1 Heuristics Using a Given RMST

In the last chapter it was shown that, for any terminal set N, $\frac{2}{3}|RMST(N)| \leq |RSMT(N)| \leq |RMST(N)|$. It is an open question to determine the expected value of $|RSMT(N)|$ in terms of $|RMST(N)|$, for any distribution. (Without a specific application, the common convention is adopted in this chapter that the terminals are drawn uniformly from the unit square; for this case it has been conjectured [6] that $E[|RSMT(N)|] \approx 0.88E[|RMST(N)|]$.)

Since the exact computation of $RSMT(N)$ is intractable for all but the smallest problems, the easily computed $RMST(N)$ is usually used as a yardstick with which to measure the efficacy of a heuristic. Let T_A be the approximate RSMT produced by using heuristic A. The *percentage of improvement*

$$\sigma_A(N) = \frac{|RMST(N)| - |T_A(N)|}{|RMST(N)|}$$

can be calculated for any terminal set N. Typically the average value of σ_A over several random sets of terminals is reported. In this chapter the average value of σ_A will vary from 4% to 12% (with the conjectured upper bound of about 12%, see above).

How fast can the RMST be calculated for a given N? Any standard MST algorithm can be used if one first builds a complete network for N by computing all the $O(n^2)$ inter-terminal distances (as discussed in Part II). Using an implementation of either Prim's or Kruskal's algorithms the RMST can be computed in $O(n^2)$ additional steps. To produce sub-quadratic algorithms it is necessary to use at most a sparse subnetwork of the complete network.

Hwang [18] was the first to do this; see also [26]. Hwang gave an $O(n \log n)$ algorithm for building the rectilinear Voronoi diagram for N, based on an earlier Euclidean algorithm [32]. (A Voronoi diagram partitions the plane into regions each containing one terminal. For any terminal, all points in the plane that are (rectilinearly) closer to that terminal than any other are in that terminal's region; see Fig. 2.1.) It is easy to show that an RMST of N is a subgraph of the dual of the planar Voronoi diagram, known as the Delaunay triangulation. This triangulation is planar and is easily found in $O(n)$ time, given the Voronoi diagram. Since the MST of a planar network can be found in linear time [8] the run-time is dominated by the time to find the Voronoi diagram. (Hence a simpler $O(n \log n)$ time MST algorithm could be used.) Recently, Chew and Fortune have announced (unpublished) an $O(n \log \log n)$ time algorithm to compute the rectilinear Voronoi diagram, when the terminals have been previously sorted by both their x- and y-coordinates; this may be true in some applications. Hence, in this special case, the overall run-time for finding the RMST is improved as well.

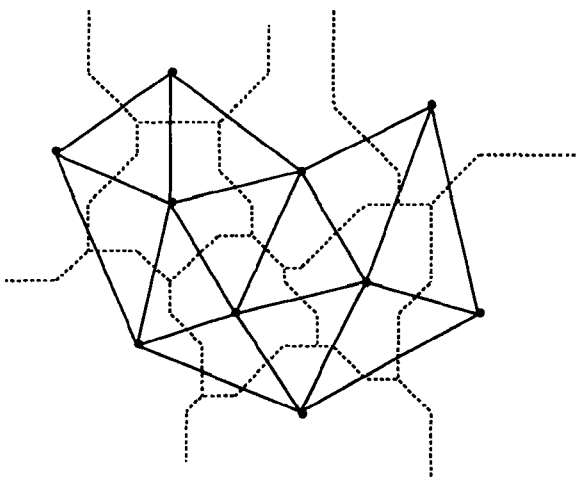


Figure 2.1: Rectilinear Voronoi diagram and its Delaunay triangulation

Computing the Voronoi diagram is a bit tedious, and the rectilinear case is more complex than the well-studied Euclidean case. Fortunately, there are simpler ways to find the RMST in $O(n \log n)$ time. Around each terminal u there are octants, regions separated by lines spaced 45° apart; see Fig. 2.2. Every other point v is located in one of the octants around u (points which happen to fall on the dividing lines are said to be in the next octant, clockwise). The primary observation is:

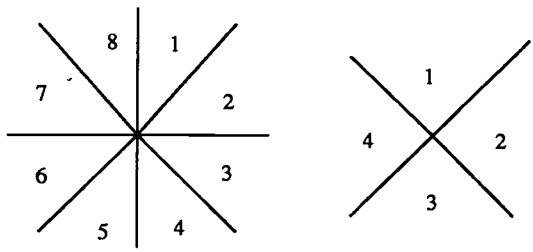


Figure 2.2: The octants and quadrants around a terminal

Lemma 2.1 ([14]) *In an RMST, if terminals u and v are connected then v is the closest point to u in the octant around u that v occupies.*

Proof: Suppose that u was connected to v in an RMST T but w was closer to u than v, where w and v are in the same octant around u. It follows by simple geometric arguments that w is also closer to v than u. Hence replacing (u, v) in T by either (w, v) or (w, u) must produce a shorter spanning tree than T. □

Using similar arguments [17] it can be shown that only the nearest neighbors properly within the four quadrants shown in Fig. 2.2 need to be considered, together with the four nearest neighbors on the four lines separating the quadrants. In both versions, of course, for each terminal not all eight nearest neighbors are necessarily defined. The octant version is used below.

The terminals are preprocessed by finding the nearest neighbor to each terminal in all of its octants. This can be done in $O(n \log n)$ total time [14]. A network $NN(N)$ is built with vertex set N and edges between vertices iff the corresponding terminals are nearest neighbors within their octants. An RMST is a subnetwork of $NN(N)$, which contains at most $8n$ edges. Hence the RMST can be found in $O(n \log n)$ additional time. This method has been called the *geographic nearest neighbor* approach and was originally studied by Yao [42].

How can an RMST be used to approximate an RSMT? Each edge of the RMST can be represented by various rectilinear shortest paths in the plane between the corresponding terminals, unless the terminals are connected by a horizontal or vertical line, in which case only one path exists. Unless otherwise specified, it is assumed each edge is represented by a path with at most one corner-point. In other words, each edge is embedded in the plane by either a straight wire, or one of two *L-shaped* wires. An RMST has up to 2^{n-1} different such embeddings in the plane. A typical embedding has pairs of wires, from different edges, that overlap. The overlap can be removed by introducing Steiner points, as in Fig. 2.3. Depending on how the embedding is chosen, different heuristics for generating an RSMT are specified.

Suppose the embedding of each L-shaped wire is chosen arbitrarily. Empirical results [29], for $n = 500$, show that removal of the overlap produces an average improvement of 3.5%. Unfortunately, when wires are placed arbitrarily, it is possible for two L-shaped wires to properly intersect (while not overlapping). This problem

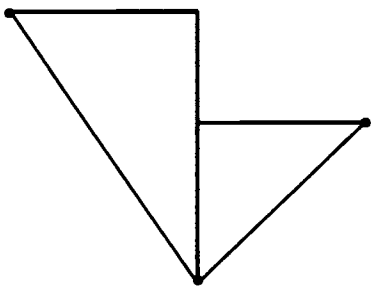


Figure 2.3: Removing the overlap between two wires introduces a Steiner point

does not occur in a similar approach by Bern and de Carvalho [6]. They arbitrarily decided, *a priori*, that each L-shaped wire is embedded so that the horizontal segment was above the vertical segment. They found an average improvement of about 5%. Below heuristics are explored that consider the potential overlap when selecting an embedding.

2.1.1 Simple Embedding (NE, SE, GE)

First a simple heuristic. Pairs of adjacent edges in the RMST are chosen arbitrarily and if they can be embedded with overlap then the corresponding embedding is chosen and fixed. All pairs of adjacent edges are explored with a simple scan. This takes linear time beyond the preprocessing step that constructs the RMST. There can be two distinct ways to embed with overlap (when using the edges (i, j) and (i, k) and j and k are in the same quadrant of i). The *naive embedding heuristic* (NE) when given such a choice, chooses one arbitrarily. The *simple embedding heuristic* (SE) when given such a choice, chooses the one with the most overlap. Since there is essentially no time penalty for the latter, SE is preferred to NE.

A *greedy* variant is the same as SE, except that a simple scan of the pairs of adjacent edges is replaced by a greedy search. In particular, the pair of wires which can be embedded for maximum overlap, over all such pairs, is selected and the corresponding embeddings are fixed.

Richards performed experiments [29] and found, using $n = 500$, $\sigma_{NE} = 6.2\%$, $\sigma_{SE} = 6.5\%$, and $\sigma_{GE} = 8.2\%$. (Unless otherwise stated the averages in this chapter are over at least 20 random instances.) By maintaining the possible overlaps in a priority queue, this can be implemented in $O(n \log n)$ time, after preprocessing.

2.1.2 Iterative Simple Embedding (ISE, IGE)

The SE heuristic arbitrarily chooses two wires incident to a terminal, removes any overlap, and repeats. There is a simple variation on this heuristic [29]. After the overlap is removed between two wires and a Steiner point is introduced, it iterates on the new tree that includes the Steiner point, not the original RMST. In particular, during later stages, overlap can be removed between two wires incident

to a Steiner point. To show that this is effective, consider Fig. 2.4; using only L-shaped wires SE will not do as well as ISE, which finds the optimal solution for this example.

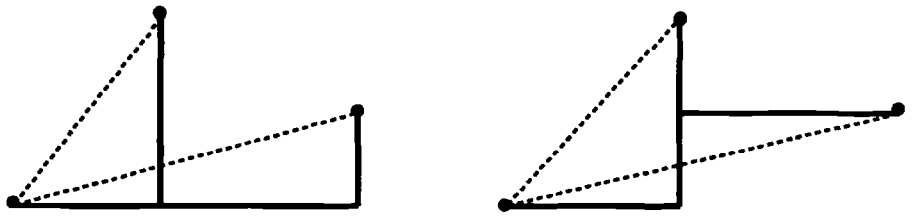


Figure 2.4: The insufficiency of L-shaped wires

The time complexity of this approach is dominated by the run-time of the RMST algorithm. Note that since there are at most $n-2$ Steiner points, and the degree of each terminal and Steiner point is $O(1)$, then the heuristic runs in $O(n)$ time, after preprocessing.

The ISE heuristic can be changed by introducing a greedy choice at each stage, as was done with the GE heuristic. In particular, the overlap that is removed at each iteration is the maximum overlap possible, over all pairs of adjacent wires in the current tree. As argued above, there are only $O(n)$ possible overlaps at any time and these are maintained in a priority queue, ordered by amount of overlap. Further, each iteration will require only a constant number of insertions and deletions to maintain the priority queue. Using any standard priority queues, such as heaps, one can implement IGE in $O(n \log n)$ time, since each query takes $O(\log n)$, after preprocessing. This matches the time complexity of producing the RMST.

Empirical results [29] with $n = 500$ give $\sigma_{ISE} = 8.3\%$ and $\sigma_{IGE} = 9.4\%$. This is a very good percentage considering the simplicity of the heuristic.

2.1.3 Simple Steinerizations (SS, GS, ISS, IGS)

A *Steinerization* replaces a connected subtree of the current tree with three vertices i, j, and k (connected by the wires (i,j) and (i,k)) by an RSMT for the same three vertices. (A more general definition of Steinerization, used in Part I, does not require i, j, and k form a connected subtree.) This is very similar to what the SE heuristic accomplishes. In fact the embeddings SE chooses are Steinerizations, except when j and k are in the same quadrant of i. Even in the latter case, the iterated versions, ISE and IGE, will usually achieve the same effect as a Steinerization, but in two steps.

The *simple Steinerization heuristic* (SS) iteratively performs arbitrary Steinerizations, on a given RMST, as long as possible. The *greedy* version GS, on each iteration, examines all possible Steinerizations and chooses the one with the most overlap. As with SE there are *iterated* versions of these, ISS and IGS, that allow Steinerizations to include newly introduced Steiner points, as well as terminals.

Experiments with $n = 500$ give $\sigma_{SS} = 6.8\%$, $\sigma_{GS} = 8.6\%$, $\sigma_{ISS} = 8.3\%$, and $\sigma_{IGS} = 9.5\%$ [29].

A digression: The choice of input can have an effect on reported performances. The random instances used to generate the σs above (for SE, GE, ISE, and IGE) were 500 terminals chosen uniformly from a 10000×10000 grid. While this is not the same as drawing from the unit square, it does provide an excellent approximation, while permitting integer arithmetic. Such an instance is in *general position* with high probability; i.e., no two terminals will be on the same row, column, or diagonal. Instances in general position are very likely to have many pairs of adjacent wires contributing some overlap. In contrast, consider 100 terminals drawn from a 20×20 grid; the improvement ratio for IGS drops considerably to 6.1%, because there are many aligned terminals and very few opportunities for overlap.

The main point is that the choice of instance can have a dramatic impact on the performance. A small grid may be representative of a typical instance in some application and the results for points in general position may be deceptive. On the other hand, to fairly compare two heuristics all instances should be drawn from large grids (or the unit square).

2.1.4 L-Shaped Optimal Embedding (LOE)

Ho, Vijayan and Wong [17] give a polynomial time algorithm to find the optimal embedding with the assumption that each wire has at most one corner. (An "optimal embedding" is relative to a given RMST and is not necessarily an RSMT.) The key to their algorithm is to *not* begin with an arbitrary RMST, but to choose a specific RMST. If there are edges of equal length there could be many RMSTs.

A *separable* RMST has proper overlap between two edges iff the edges are incident to the same terminal. They show that by adding clever tie-breaking rules, for equal length edges, that a separable RMST always exists. Further, using these rules any MST algorithm can be used to construct a separable RMST (though the effect, if any, of the rules on the Voronoi diagram is not discussed). The advantage of such an RMST is that the effect of an embedding a wire is localized. They also show that in an RMST a terminal is incident to at most six L-shaped wires, using an argument analogous to the proof of Lemma 2.1.

They give a $O(n)$ time dynamic programming algorithm to compute the optimal embedding. They begin by rooting the separable RMST at some leaf, and solve the subtrees bottom-up. The key observation is that, with a separable RMST, the optimal solution for a subtree depends only which of the two embeddings of the L-shaped wire, connecting the root of the subtree to its parent, was chosen. The subtree is solved for both choices. After solving all the subtrees of a vertex, the analyses can be combined in constant time, because it has a constant number of subtrees. Hence $O(n)$ total time is needed for finding the optimal embedding, in addition to the time needed to build the RMST (which dominates the run-time).

Empirical results [17] with $n = 100$ gave $\sigma_{LOE} = 9.7\%$, with less improvement for smaller n.

2.1.5 Z-Shaped Optimal Embedding (ZOE)

As can be seen in Fig. 2.4, L-shaped wires are insufficient to find true optimal embeddings. A *Z-shaped* wire has exactly two corner points. The example in Fig. 2.4 requires at least one Z-shaped wire.

Ho, Vijayan and Wong [17] extended the previous heuristic and gave a polynomial-time algorithm to find the optimal embedding of a separable RMST, when each wire has at most two corners. Further, they prove that there always exists an optimal embedding with at most two corners per wire; i.e., three corners are never required. Hence their algorithm finds a optimal embedding with no restrictions. (It is open whether one could do better starting with a non-separable RMST.)

The approach is similar to LOE. The principal difference is that there are many Z-shaped wires connecting the root of a subtree to its parent, and the subtree must be solved for each of these possible embeddings. However, since a Z-shaped wire can be constrained to the grid graph $GG(N)$, there are only $O(n)$ possible embeddings. Actually there are only k possibilities, where k is the maximum number of edges in $GG(N)$ used by any Z-shaped wire. Since there are at most six such wires connected to a terminal, it can be shown the heuristic runs in $O(nk^6)$ time. Even though this can be $O(n^7)$ in the worst case, this is a very pessimistic analysis. Empirical results show that it runs in $O(n^2)$ time, or better, on random instances. This dominates the preprocessing time.

Empirical results [17] with $n = 100$ gave $\sigma_{ZOE} = 10.2\%$, with less improvement for smaller n. This is about 0.5% better than σ_{LOE}.

Recently Dastghaibyfard, Teo and Tuan [9] proposed an approximation of ZOE. It does not consider all possible embeddings of each Z-shaped wire between terminals i and j. It chooses only those embeddings that use edges in $GG(N)$ that are determined by i and j and the terminals adjacent to i and j in the given separable RMST. It can be shown that there are at most eight such embeddings. Hence this approximation can be implemented in $O(n)$ time. They gave empirical results that showed, for n up to 100, the resulting embedding was identical to that produced by ZOE 99% of the time.

In the sequel, again, wires are restricted to have at most one corner.

2.1.6 Hierarchical Embedding (HE)

Sarrafzadeh and Wong [30] proposed a heuristic that preprocesses a given RMST. Any tree T with bounded degree, such as an RMST, has an edge e whose removal leaves two subtrees T' and T'' of roughly equal size. In particular, $\frac{1}{3}|T| \leq |T'|, |T''| \leq \frac{2}{3}|T|$, where $|T|$ is the number of vertices in T. Similarly T' and T'' can each be partitioned and one can continue recursively. This recursive partitioning can be represented with a binary tree B_T defined recursively: the root of B_T corresponds to e, the left subtree is $B_{T'}$, and the right subtree is $B_{T''}$. Note that the height of B_T is $O(\log n)$. The leaves are singleton trees, which are the terminals.

The heuristic begins with an RMST T. It builds an approximate RSMT for the terminals in each subtree of B_T, bottom-up. Consider an internal node x of B_T associated with the edge between terminals i and j. After building the approximate

RSMTs for the terminals in each of the two subtrees of x (one containing i and the other containing j) it connects these two subtrees with a wire from i to j. It tries both L-shaped wires, removes any overlap and breaks any cycle, and retains the shortest tree found.

They also proposed a parameterized variation of this scheme. For a given k, it begins by considering each subtree of height k (with at most 2^k terminals) and optimally solving the rectilinear Steiner problem for the terminals in those subtrees. The heuristic then continues to solve the other subtrees, bottom-up, as before. Call this heuristic HE_k.

They found, for $n = 100$, that $\sigma_{HE_1} = 8.5\%$, $\sigma_{HE_2} = 10.0\%$, and $\sigma_{HE_3} = 10.3\%$. The runtime is exponential in k, but is $O(n \log n)$, for constant k.

2.1.7 Loop Detection (LD)

Chao and Hsu [7] proposed a heuristic that begins with an RMST. (They modified the rectilinear metric to favor an edge with more "slant" in the case of ties.) Rather than embed each edge they consider the set of "candidate" grid points along the possible L-shaped embeddings. When they add an edge between such a point and terminal (or a corner of an L) they get a loop, which they break by removing the longest edge on the loop. They collect the set S of all such candidates that cause the tree to be shorter as a result of breaking the loop. They augment N by adding a maximal independent subset of S. They form a new RSMT and iterate until no further improvement can be made. Call this heuristic LD.

They also suggested a preprocessing step where all *essential Steinerizations* are done. A Steinerization between i and its neighbors j and k is *essential* if there is no other such j and k for i admitting a Steinerization. Call the heuristic with this added step SLD. These heuristics can be implemented in $O(n^2 \log n)$ time.

They found empirically, for $n = 80$, that $\sigma_{LD} = 10.5\%$ and $\sigma_{SLD} = 10.6\%$, which is very good.

2.1.8 Delaunay Triangulation-Based (DT)

Smith, Lee and Liebman [34] proposed a hybrid scheme that is based on the same principle as IGE. However it is more complicated and produces less improvement, so it is only sketched here.

There are two distinguishing features. First, it restricts itself by examining only pairs of edges in the RMST that form two edges of a "selected" triangle in the Delaunay triangulation, that was used to construct the RMST. Unfortunately, due to degeneracies in the corresponding Voronoi diagram (that do not occur in the Euclidean case), additional *ad hoc* searches are conducted for degenerate triangles.

The second feature is that it also allows a triangle neighboring the selected triangle, at each stage, to be considered. It then asks whether it can do better by making a local improvement using the four connected points in these two triangles, rather than just using the three in the selected triangle.

They give an $O(n \log n)$ time implementation. They report $\sigma_{DT} = 8.6\%$, which is far inferior to IGE.

2.1.9 Four-Point Steinerizations (4PS)

Beasley [2] gave an heuristic related to the IGE heuristic, that uses greedy four-point Steinerizations, rather than just three-point Steinerizations. The approach is much simpler than the similar technique in DT. A *four-point Steinerization* chooses a four point connected subtree (either a three-pointed star or a simple path) of the current tree and determines an RSMT for the same four points. Each such four point RSMT can, of course, be computed in constant time. The Steinerization with the most improvement, over all connected subtrees with at most four vertices, is selected at each stage. Rather than adding the RSMT of the four points itself, he just adds the Steiner points to the terminal set and computes a new MST. (He does not formally prove the procedure converges.)

No time complexity is given but empirical results suggest that Beasley's implementation runs in $O(n^{1.253})$ expected time. An improvement of $\sigma_{4PS} = 9.9\%$ is reported. (He also applied this to the Euclidean metric and it is discussed in Section 4.2 in Part I.)

2.1.10 Neighborhood Steinerizations (NS)

Hasan, Vijayan and Wong [16] present another heuristic based on local Steinerizations. It begins with an RSMT T of the terminal set. The *neighborhood* of a vertex v in T, is the subgraph of T induced by v and its adjacent vertices. The weight of a vertex v is the decrease in length when the neighborhood is replaced by an RSMT over the same point set. (Since, by Lemma 2.1, any neighborhood has at most nine vertices, the corresponding RSMT can be computed in constant time.) After computing the weight of each vertex, they find the maximum total weight independent set of vertices in T (i.e., a set of vertices whose neighborhoods share no edges). For every vertex in the independent set, they replace the neighborhood by the corresponding RSMT. They add all the Steiner points to the terminal set to create a larger terminal set and construct a new RMST. They iterate the above process until no improvement is found. (For $n \leq 35$ they found only 4 or 5 iterations are necessary.) There are additional steps for pruning old Steiner points out of the terminal set, in later stages, when they no longer help.

Both computing the weights and finding the maximum weight independent set of a tree (using dynamic programming) can be done $O(n)$ time, which is dominated be the $O(n \log n)$ time for computing the RMST. Hence each iteration of the outer loop takes $O(n \log n)$ time. For $n \leq 35$, they report $\sigma_{NS} = 8.5\%$ which is lower than might be expected.

2.2 Heuristics Based on MST Algorithms

The two best studied algorithms for the MST problem are due to Kruskal and Prim [13] Many heuristics are based on these algorithms; the basic control structure is maintained while varying individual steps. Such heuristics have appeared in the previous parts of this book. Recall, *Kruskal's algorithm* begins with n singleton trees and repeatedly connects the two currently closest trees. *Prim's algorithm*

repeatedly adds to one subtree by connecting it to the closest vertex not currently in that subtree; the initial subtree is an arbitrary vertex.

2.2.1 Prim with Three-Point Steinerizations (P3S)

Lee, Bose and Hwang [27] proposed varying Prim's algorithm. The current subtree is connected to the nearest terminal w not in the subtree. "Nearest" is defined to be the shortest distance to either a terminal or Steiner point in the current subtree; let u be that nearest such point to w. Rather than connecting u to w by a single wire, a three-point Steinerization is performed. A constant number of vertices in the neighborhood of u, in the current subtree, are inspected to choose a vertex v (the *ad hoc* selection process is in [27]). The edge between u and v is replaced by an RSMT on the three points u, v, and w; a Steiner point may be introduced.

The heuristic does not begin with an arbitrary singleton tree. Instead the initial step builds a subtree for three terminals, and the RSMT for those three terminals is the smallest RSMT for any three terminals. A straightforward implementation of this initialization step takes $O(n^3)$ time, which is unfortunate since the remainder of the heuristic takes only $O(n^2)$ time. However, they gave an $O(n^2)$ time initialization procedure. It is analogous to the geographic nearest neighbor approach in Lemma 2.1. Rather than deal explicitly with octants, they show that if only the 20 nearest neighbors to each point were considered then the smallest RSMT on three points could be discovered.

Empirical results were found for small n, $n \leq 35$. Rather than reporting mean improvements, they report only the median value of σ_{P3S} is between 8% and 10%. De Souza and Ribeiro [37] report $\sigma_{P3S} = 8.4\%$, empirically with $n = 100$.

2.2.2 RMST-Driven Prim-Based (RP)

Hwang [19] devised a faster heuristic by using the idea of P3S, but driving the computation with the guidance of a given precomputed RMST T. In particular, the choices of u and w, at each stage, are determined from T, not by a shortest distance computation.

The heuristic begins with a "labeling" phase, which orders the edges of T so that T *could* have been built by adding the edges in that order using Prim's algorithm. (It is not necessary to know how T was actually constructed.) This phase begins by sorting each adjacency list of the vertices of T by increasing length. It then does a breadth-first search (or depth-first search) beginning at some endpoint of the shortest edge. The output is the sequence of edges of T, $(p_2, q_2), (p_3, q_3), \ldots, (p_n, q_n)$, together with the vertex p_1, such that $q_i = p_j, 1 \leq j < i \leq n$. This phase (as well as the computation of T initially) can be done in $O(n \log n)$ time.

Hwang begins to build the approximate RSMT by building an RSMT on p_1, p_2, and p_3. He then connects each p_i, $i > 3$, to the current subtree in the same manner as P3S, with p_i as w and q_i as u. In fact, since each three-point Steinerization takes constant time, this phase takes only $O(n)$ time.

Hwang tried RP out on a handful of examples and found the performance to be almost identical to P3S. De Souza and Ribeiro [37] recently obtained the empirical

improvement $\sigma_{RP} = 9.5\%$, for $n = 100$, which is a clear improvement over σ_{P3S}. These experiments found no real difference in performance between depth-first search and breadth-first search, though they found (without explanation) that the latter ran faster.

2.2.3 Prim-Based with Biased Two-Point Connections (PB2)

De Souza and Ribeiro [37] proposed a Prim-based scheme that connected each new point directly to the current subtree with a single wire. However, it has two features that improve its performance. First, it allows the new wire to connect anywhere in the current subtree, not just to its vertices (i.e., terminals and Steiner points). If it connects to a corner point or the interior of a wire segment then a new Steiner point is created, of course. Second, it biases the choice between the two possible L-shaped wires (when it is not a straight connection) by polling the currently unattached terminals. By bending in the right direction it can shorten later connections. (The final result is often identical to that found using Steinerizations.)

The first feature is very easy to incorporate. Implementations of Prim's algorithm maintain, for each terminal not in the current subtree, the shortest distance to the current tree. It is straightforward to allow the shortest distance to be updated, as each new wire is added, in constant time. This preserves the total $O(n^2)$ total time.

The bias, for the choice between the two L-shaped wires connecting w to u, can be explained by consulting Fig. 2.5. Note, the rectangle defined by the two wires is empty, since w is closest to u, The heuristic finds the closest terminals (closest to the the rectangle) in the six regions labeled 2,3,4,6,7, and 8. It then chooses the wire that minimizes the sum of the distances from those six points to that wire.

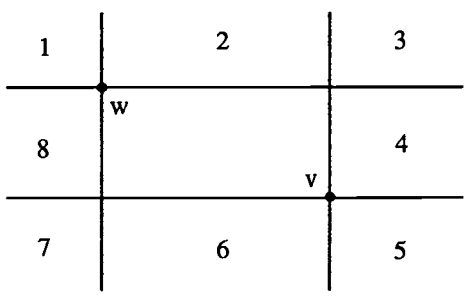


Figure 2.5: Regions defined by the two L-shaped wires connecting w and u

They present an $O(n^2)$ time implementation of this heuristic. They found empirically, for $n \leq 100$, that $\sigma_{PB2} = 9.5\%$, and perhaps a bit better for large n.

2.2.4 Kruskal-Based (KA, KB, KC)

Bern and de Carvalho [6,5] discuss several Kruskal-based heuristics, which they attribute to Clark Thompson. Each heuristic differs in how one defines "closest"

subtrees, and how the chosen subtrees are to be connected.

KA: At each stage find the two subtrees that can be connected with the shortest wire, and use that wire.

KB: At each stage find the two subtrees that can be connected with the shortest wire between two terminals, Steiner points, or corner points, and use that wire.

KC: At each stage find the two subtrees that can be connected with the shortest wire between two terminals; the connecting wire, however, is the shortest wire connecting terminals, Steiner points, or corner points.

In all cases they place L-shaped wires so that the horizontal segment is below the vertical segment. There are examples of terminal sets where each of the three approaches is superior to the other two. Even so, KC is mainly of interest because it is amenable to formal analysis. Bern [5] was able to show that

$$\liminf_{n\to\infty} \frac{E[|RMST| - |T_{KC}|]}{E[|RSMT|]} \geq 0.00098$$

In other words, σ_{KC} is bounded away from 0. Even though the proven bound is very small, this is the only nontrivial analytic bound on the improvement of a heuristic. Bern's results are for terminals drawn from the unit square distributed according to a Poisson process with mean n. Hwang and Yao [20] extended the result to the uniform distribution, for exactly n terminals, and made a small improvement to the constant.

These heuristics can all be implemented in $O(n^2 \log n)$ time [28]. There can only be $O(n)$ wires, terminals, Steiner points, and corner points altogether, at any time, so that the $O(n^2)$ distances between these can be maintained in a priority queue. Bern and de Carvalho [6] presented implementations based on the assumption that the terminals were chosen from an $m \times m$ grid; in particular, they presented $O(mn^2 \log n)$ time implementations. In some applications m is small.

Wee, Chaiken and Ravi [40] presented new implementations of KA and KB. They were able to show that most of the $O(n^2)$ distances, mentioned above, were unnecessary. In fact the only distances required, initially, are those found in $NN(N)$, the geographic nearest neighbor network. The situation is complicated by the fact that as the heuristic progresses one must maintain $NN(N \cup N')$, where N' is the set of Steiner points and corner points introduced so far. They present an efficient algorithm for updating the nearest neighbor network in $O(\log^2 n)$ time. This leads to an $O(n \log^2 n)$ time implementation of KA.

They extend their technique by defining the nearest neighbor network for a set of non-intersecting line segments. While this is more complex, they can still do updates in $O(\log^2 n)$ time, giving an $O(n \log^2 n)$ time implementation of KB.

Bern and de Carvalho [6] present empirical results for $n = 100$, for KB and KC; in both cases the improvement was about 9.3%.

2.2.5 The Relationship to the RMST

Each heuristic in this chapter so far is known to produce an approximate RSMT that is no longer than the RMST. In most cases this is a trivial observation; either the heuristic improves on a given RMST or it mimics a MST algorithm (with further improvement possible at each stage). However, for several years it was open whether this was true for P3S. Recently, De Souza [36] proved it true for P3S. Hence $\sigma_A \geq 0$ for each heuristic A. Alternatively, can each heuristic be as bad as an RMST? In particular, can each heuristic produce as poor a tree as possible, i.e., 50% worse than optimal? De Souza and Ribeiro [38] exhibited a single set of terminals that showed that DT, 4PS, P3S, RP, and PB2 can be that bad.

Shute [33] unified the previous heuristics under the umbrella of "greedy algorithms". A greedy algorithm, in this context, is an algorithm that repeatedly connects a subtree T to another subtree with a wire that is no longer than the shortest connection from any terminal in T to any terminal not in T. All the heuristics so far have been greedy (as well as the SBB heuristic, mentioned later). Shute showed that any greedy heuristic produces a Steiner tree no longer than the MST. He also gave a complicated scheme to construct a terminal set for which any greedy heuristic was as poor as possible (50% worse than optimal). Similar results were given for the Euclidean case.

Hence none of these heuristics can do better than the Steiner ratio; that is, always have $|T_A(N)| \leq \frac{3}{2}|RSMT(N)|$. Recent heuristics can do better, as discussed in the next subsection.

2.2.6 Zelikovsky-Based Heuristics

Zelikovsky's heuristic for networks, described in Section 4.3.6 of Part II, was the first heuristic for networks that was provably better than the Steiner ratio. It has implications for metric spaces such as the Euclidean (see Section 4.8 in Part I) and the rectilinear metrics, when properly generalized. These are discussed in a more general setting in Chapter 1 of Part IV, where it is shown that indeed there are rectilinear heuristics that do better than the Steiner ratio.

Berman and Ramaiyer [3,4] have announced rectilinear results based on their modified upgrading heuristic for networks (see Section 4.3.7 in Part II). Recall that their heuristic has a parameter k. They claim that in the rectilinear case, $|T_{MUH}(N)| \leq \frac{11}{8}|RSMT(N)|$ for $k = 3$, and $|T_{MUH}(N)| \leq \frac{97}{72}|RSMT(N)|$ for $k = 4$. They do not feel that the bound is tight.

2.3 Computational Geometry Paradigms

There are two basic paradigms for algorithms in Computational Geometry literature: plane sweep and divide-and-conquer based on partitioning.

2.3.1 Prim-Based Plane Sweep (PPS)

Hanan's original technical report [15] mentioned a heuristic that is best described as a plane sweep. Conceptually, a *plane sweep* algorithm slides a horizontal bar over the set of terminals, from bottom to top. It processes each terminal as it is encountered and maintains a data structure that represents the decisions and events that have occurred "below" the bar.

Hanan begins by sorting the terminals by increasing y coordinate and considers them in order, hence the "plane sweep". A subtree connecting all terminals already seen is maintained, and each new terminal, when it is encountered, is connected to the current subtree with the shortest possible wire (i.e., it is Prim-based). An L-shaped wire will have its horizontal segment above the vertical segment. Since there are only $O(n)$ wires, terminals, Steiner points, and corner points, the shortest wire at each step can be found in $O(n)$ time. Hence the sweep can be easily implemented in $O(n^2)$ time. Hanan suggests continuing by rotating the set of terminals by 90° three times, each followed by a plane-sweep; the reported tree is the shortest of the four computed trees.

Empirically this heuristic is good for small terminal sets; $\sigma_{PPS} = 9.0\%$ when $n \leq 10$. In fact it is guaranteed to be optimal for $n < 5$. However, it is very poor for large problems [37]; $\sigma_{PPS} = 5.3\%$ when $n = 100$. (If no rotations are done then the expected improvement is even worse, only 3% for $n = 100$.) The poor performance is a result of combining Prim's technique with the plane sweep paradigm. As a result, especially at the beginning of the sweep, decisions to connect distant pairs of terminals are made that should have been postponed until more terminals had been "seen".

The run-time can be improved. Servit [31] proposed speeding up the heuristic by considerably simplifying the basic connection step. In particular, each new terminal is connected by the shortest wire to the wire placed on the previous step! Clearly the heuristic runs in $O(n)$ time in addition to the $O(n \log n)$ time necessary to sort the terminals. The average improvement for this technique is not known.

Richards [28] gave an $O(n \log n)$ time implementation of PPS, which obviates Servit's version. This heuristic uses a data structure to maintain an ordered view of just the relevant wire segments in the current subtree. In particular, portions of the subtree "hidden" below 45° trailing projections of wires (see Fig. 2.6) can be ignored. The remaining relevant portion of the subtree is a sequence, sorted by x coordinate, of points and horizontal wire segments (see Fig. 2.7). This sequence of points and segments can be searched and maintained in $O(\log n)$ time, leading to an overall $O(n \log n)$ time implementation. A simple version of the data structure was shown to have an $O(n^{3/2})$ expected time complexity, for terminals drawn uniformly from the unit square.

Unlike previous heuristics, that are never worse than an RMST, Hanan's heuristic can produce a tree worse than the RMST. For example, a construction with the terminals in a large "X" formation gives the ratio of the length of the approximation to the length of the RMST approaching 5/4. Occasionally a tree that was marginally longer than the RMST was produced during the experiments. However in no randomly-generated case was the best of the trees, over the four rotations, ever inferior to the RMST.

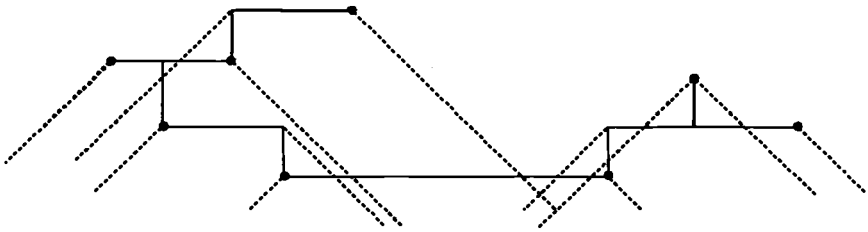


Figure 2.6: Portions of the current tree are hidden behind the upper wires

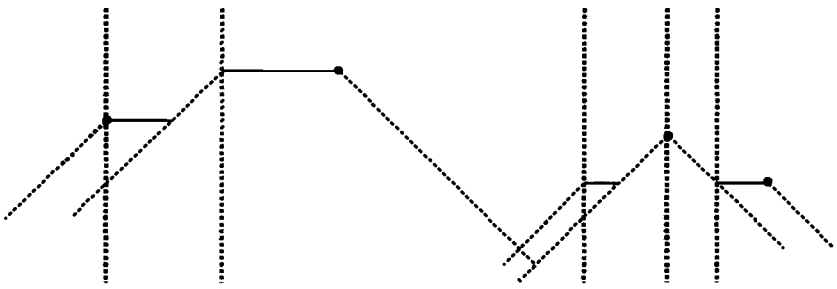


Figure 2.7: The sequence of exposed portions of the current tree

2.3.2 Kruskal-Based Plane Sweep (KPS)

Deneen and Shute [10] implemented a plane sweep loosely based on Kruskal's algorithm. It defers making connections, allowing several subtrees to be built and connected later. The rationale is that it would overcome the principal factor in the poor performance of PPS, namely premature connections.

Their implementation is based on Richards' heuristic for PPS. They maintain a similar data structure for the sequences of relevant horizontal wire segments (again, see Fig. 2.7). However, they do not force all terminals below the scan line to be connected into one tree. Instead they allow those terminals to be connected into several subtrees. A subtree is connected to its nearest subtree only when it becomes "hidden", at a later stage. (Actually the connection is made if some representative segment of the subtree is hidden.) By using sophisticated data structures the subtrees, and nearest neighbor information, can be maintained in $O(\log n)$ time, leading to an overall $O(n \log n)$ time heuristic.

Empirical results, for $n = 100$, indicate that $\sigma_{KPS} = 8.5\%$. This is a clear improvement over PPS, but not as good as KB, another Kruskal-based scheme.

2.3.3 Probabilistic Partitioning (PPA, PPB)

Komlós and Shing [25] present two similar divide-and-conquer heuristics based on a similar approach by Karp [24] for the traveling salesman problem.

The first heuristic, which is designated PPA, has two phases. The first phase partitions the plane up into small rectangles, each containing at most t terminals, where t is a parameter to be given later. The basic step in this phase partitions a rectangle into four subrectangles, each with at most one fourth of the terminals in the original rectangle; an example is in Fig. 2.8

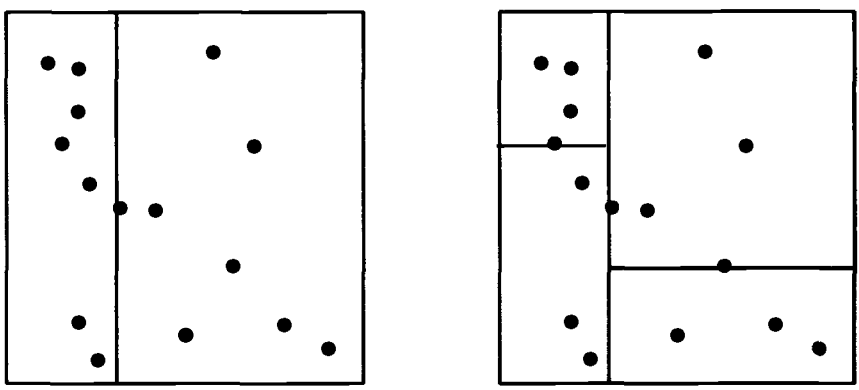


Figure 2.8: Partioning step evenly divides the terminals into four subrectangles

This can be done, in linear time, with three calls to a median-finding routine. The phase begins with a rectangle bounding the entire terminal set, and recursively applies the basic step to every subrectangle until at most t terminals are in each one. This phase can be implemented in overall $O(n \log n)$ time, for any t.

The second phase of the heuristic has two steps. First, the optimal RSMT for the terminals in each subrectangle is computed exactly using, say, the Dreyfus and Wagner algorithm (see Section 3.4 in Part II). Second, Kruskal's standard MST algorithm is run to connect the various subtrees. The final tree is pruned, so that all leaves are terminals.

The total time, using Dreyfus and Wagner's algorithm, is $O(t3^t n + n \log n)$. So, for example, if $t = \log_3 n$ then one gets $O(n^2 \log n)$ time, and $t = \log_3 \log n$ gives $O(n \log n \log \log n)$ time. No empirical results are presented, and no estimate of σ_{PPA} is given. However, it is shown, with a detailed argument, that their approximation is within a factor of $1 + O(\frac{1}{\sqrt{n}})$ of optimal, with probability $1 - o(1)$, when the terminals are drawn uniformly from the unit square.

Komlós and Shing presented another related heuristic in the same paper. It divides the unit square up into small squares "buckets", so that the *expected* number of terminals in each bucket is t. Then two new artificial terminals are added on the boundary of each bucket, so that each new terminal in one bucket is in the same position as a new terminal in a neighboring bucket. The new terminals are positioned in such a way that the union of spanning trees of each bucket will

result in a large spanning tree of all the terminals. Their heuristic, then, builds an approximate RSMT by building an optimal RSMT for the terminals in each bucket.

This heuristic, which is designated PPB, has the same probabilistic performance as PPA. However it depends heavily on the uniformity of the terminal's distribution in the unit square; it can produce a tree that is at least a factor of t worse than optimal. It can be shown that the heuristic runs in $O(ne^{3t})$ time, with probability $1 - o(1)$. Therefore, for each value of t, PPB is faster than PPA. Clearly, for some sets of terminals, some buckets can be overfilled; if that happens it is suggested a heuristic be used instead of an exact algorithm for that bucket.

2.4 Other Heuristics

There are other heuristics that do not easily fit in the previous sections. Some are discussed below, while others are found in Chapter 5.

First two heuristics are mentioned briefly. Jiang and Tang [21] began by collecting statistics for all subproblems using 3 terminals from N. Then, after some additional preprocessing, a Kruskal-based heuristic is given that favors Steiner points and edges suggested by the statistics. Also Few [11] gave a rectilinear heuristic that is described in Section 4.6 of Part I.

2.4.1 Iterated 1-Steiner (I1S)

Kahng and Robins [23,22] presented an heuristic with excellent improvement characteristics. It is related to an earlier, and less effective, heuristic of Smith and Liebman that is discussed below. These heuristics are based on routines that solve the *1-Steiner problem*: What is the optimal Steiner tree if at most one Steiner point is permitted? Alternatively: What is the location of a single point p such that the RMST of $N \cup \{p\}$ is minimized, over all such p?

Georgakopoulos and Papadimitriou [12] gave an $O(n^2)$ time algorithm for the 1-Steiner problem in the Euclidean case, and Kahng and Robins have adapted this to the rectilinear metric in the same time bound. The algorithm partitions the plane into $O(n^2)$ *isodendral regions*. An important property is that for each isodendral region R one can compute the MST for any p in R in constant time, after a $O(n^2)$ time preprocessing step. Further, in constant time one can find a point p that minimizes the length of the RMST of $N \cup \{p\}$ over all points in R.

The iterated 1-Steiner heuristic, I1S, simply solves the 1-Steiner problem. If such a point p does exist that decreases the length if the RMST, then set $N \leftarrow N \cup \{p\}$ and iterate. It is possible for there to be more than $n - 1$ Steiner points; for example, see Fig. 2.9. Hence, they arbitrarily limit the number of iterations to n. Clearly the heuristic runs in $O(n^3)$ time.

Empirical results for $n = 40$ give $\sigma_{I1S} = 10.9\%$, which is the best for any heuristic discussed so far. They claim that their improvement is due to the fact that they do not, implicitly or explicitly, use an RMST. They conjecture that 10% improvement is the ceiling on the performance of those methods. While Berman

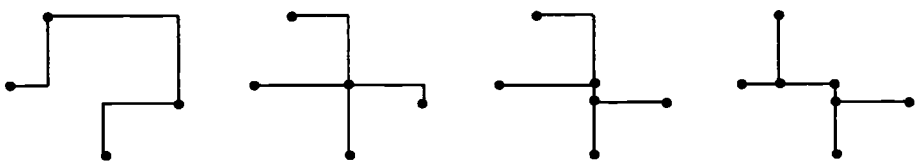


Figure 2.9: Example where I1S uses more than $n - 1$ points

and Ramaiyer do use an RMST there seem to be some unexplored connections between their heuristic and I1S.

2.4.2 Batched 1-Steiner (B1S)

Kahng and Robins [23] presented several variations on the I1S heuristic, including using a random choice of p, rather than an optimal choice. The most promising variation, however, batches many choices of p, without recomputing the isodendral regions. This heuristic scores every one of the $O(n^2)$ candidate Steiner points in $GG(N)$. By preprocessing the isodendral regions, in $O(\log n)$ time each candidate point p can identify the isodendral region it is in; it can then calculate the length of the RMST over $N \cup \{p\}$ in constant additional time (see above). The *score* for each p is the improvement, if any, of the new RMST over the original RMST.

The heuristic then sorts the candidates by decreasing score and considers each candidate in that order. Let $N_0 \leftarrow N$ be the original set of points, which have been preprocessed. For each p, in order, if the RMST for $N_0 \cup \{p\}$ is shorter than the RMST for $N \cup \{p\}$ then set $N \leftarrow N \cup \{p\}$ (This test is meant to insure that the candidate p is chosen only if its effect is "independent" of the other candidates just previously selected.) The maintenance of the RMST as N is augmented, requires $O(\log n)$ per update, with sophisticated techniques. The isodendral regions are not updated during this loop. After all the candidates are considered, the isodendral regions are recomputed and the remaining candidates are rescored. The heuristic iterates while improvement is made. (Empirically, for $n = 40$, an average of 2.05 iterations of the outer loop were performed.) The iteration of the outer loop runs in $O(n^2 \log n)$ time.

Empirical results for $n = 40$ show that $\sigma_{B1S} = 10.9\%$, the same as σ_{I1S}.

Smith and Liebman [35] gave an $O(n^4)$ time heuristic, very similar to B1S, that used several *ad hoc* stages. It began by selecting a subset of the vertices of $GG(N)$ as candidate Steiner points. They found successive (Euclidean) convex hull "layers" and found the bounding rectangle of each one. Each candidate Steiner point was a projection onto the nearest bounding rectangles! The heuristic was constrained to consider only these $O(n)$ candidates. Another difference is that it does not use the "independence" check (see above). Presumably for these reasons, the performance reported for this variant is considerably worse: the improvement is less than 8%.

2.4.3 Suboptimal Branch-and-Bound (SBB)

Yang and Wing [41] gave a heuristic algorithm related to their exact branch-and-bound algorithm, discussed in the previous chapter. It differs in several ways; principally, it places entire wires (with at most one corner) at once, and it greedily chooses the next wire to place. Informally, it is a Prim-based approach with backtracking.

Step 1: Let B be the length of the best approximate Steiner tree seen so far; $B \leftarrow \infty$ initially. Choose some $p \in N$ arbitrarily and let T be the tree with p as its only vertex. Push T on an initially empty stack S. Let V be the vertex set of $GG(N)$.

Step 2: If S is nonempty then pop T off of S, otherwise stop. Let A be the vertex set of T and let L be the length of T. If $L \geq B$ then repeat this step.

Step 3: If $A \subseteq N$ then set $B \leftarrow L$. Let p' and v' be the vertices that minimize $\{dist(p', v') \mid p' \in N \cap (V - A), v' \in A\}$. Let T_1 and T_2 be the two possible trees formed by adding to T a wire from p' to v', with at most one bend. (If the wire is straight then T_2 is superfluous.) Push T_1 and T_2 onto S. Return to step 2.

There is no known polynomial bound on such an approach. However, empirical results show that they can solve problems with $n \leq 225$ in less than three minutes. They report $\sigma_{SBB} = 11.9\%$ for various examples with $100 \leq n \leq 225$, and more improvement with smaller n. This is the best improvement reported for any heuristic.

2.4.4 Exhaustive Partitioning (EP)

The obvious divide-and-conquer approach is to divide the unit square (or bounding rectangle) in half and solve each half independently. In fact Basart and Huguet [1] suggested doing this. They made exactly one vertical (or horizontal) cut and solved each half with Servit's version of PPS. Their improvement dropped quickly to 4% as n increased. Part of the poor performance is due to the inferiority of Servit's PPS. However, it is not expected that a single cut will intersect an RSMT in only one place. In fact many intersections are expected as n increases.

Thomborson, Deneen and Shute [39] suggested using divide-and-conquer but did not assume a "clean cut". In fact, they suggest using k, the maximum number of horizontal lines of the RSMT that a central vertical line cuts. They show that the expected value of k is $O(\sqrt{n})$, with high probability. For each value of $i \leq k$ they enumerate all ways that i horizontal lines of an RSMT could be cut. For each of the enumerated cases, they recursively solve the corresponding right and left subproblems, with i artificial terminals added on the boundaries where the lines were presumed to be cut.

They present two versions of this general scheme. First they actually compute k, by easily checking to see how many horizontal lines can be packed into the cut

subject to separation constraints imposed by nearby terminals. This leads to an *exact* algorithm that runs in $n^{O(\sqrt{n})}$ time, with high probability.

The second variation, which is designated EP, begins by estimating k by arbitrarily setting it to $\lfloor 2\sqrt{n} \rfloor$. This heuristic runs in deterministic $n^{O(\sqrt{n})}$ time, but only finds an optimal tree with high probability. No estimates of the improvement of this heuristic are known, though the results should be strong. They report by setting $k = 1$ (too low, of course) that a 50 terminal example took 20 hours to compute, examining 841 subproblems!

References

[1] J. M. Basart and L. Huguet, An heuristic algorithm for rectilinear Steiner minimal trees, Technical Report, Univ. Autonoma de Barcelona (1987).

[2] J. E. Beasley, A heuristic for the Euclidean and rectilinear Steiner problems, Technical Report, Manag. School., Imperial College, London (1990).

[3] P. Berman and V. Ramaiyer, An approximation algorithm for the Steiner tree problem, Technical Report, Pennsylvania State Univ. (1991).

[4] P. Berman and V. Ramaiyer, Improved approximations for the Steiner tree problem, Technical Report, Pennsylvania State Univ. (1991).

[5] M. W. Bern, Two probabilistic results on rectilinear Steiner trees, *Algorithmica* (1988) 191–204.

[6] M. W. Bern and M. de Carvalho, A greedy heuristic for the rectilinear Steiner tree problem, Technical Report, Computer Science Division, Univ. of California at Berkeley (1985).

[7] T.-H. Chao and Y.-C. Hsu, Rectilinear Steiner tree construction by local and global refinement, *Int. Conf. on Computer-Aided Design* (1990) 432–435.

[8] D. Cheriton and R. E. Tarjan, Finding minimum spanning trees, *SIAM J. Comput.* **5** (1976) 724–742.

[9] G. Dastghaibyfard, K. H. Teo and T. C. Tuan, A localized underlying grid-based approach for the Steiner tree problem, Technical Report, Univ. of Oklahoma (1991).

[10] L. L. Deneen and G. M. Shute, A plane-sweep algorithm for rectilinear Steiner trees with deferred connections, Technical Report, Univ. of Minnesota (1990).

[11] L. Few, The shortest path and shortest road through n points, *Mathematika* **2** (1955) 141–144.

[12] G. Georgakopoulos and C. H. Papadimitriou, A 1-Steiner tree problem, *J. Algorithms* **8** (1987) 122–130.

[13] R. L. Graham and P. Hell, On the history of the minimum spanning tree problem, *Ann. Hist. Comput.* **7** (1985) 43–57.

[14] L. J. Guibas and J. Stolfi, On computing all north-east nearest neighbors in the L_1 metric, *Inf. Process. Let.* **17** (1983) 219–223.

[15] M. Hanan, Net wiring for large scale integrated circuits, Technical Report, IBM, Yorktown Heights (1965).

[16] N. Hasan, G. Vijayan and C. K. Wong, A neighborhood improvement algorithm for rectilinear Steiner trees, *Proc. ISCAS* (1990) 2869–2872.

[17] J.-M. Ho, G. Vijayan and C. K. Wong, New algorithms for the rectilinear Steiner tree problem, *IEEE Trans. on Computer-Aided Design* **9** (1990) 185–193.

[18] F. K. Hwang, An $O(n \log n)$ algorithm for rectilinear minimal spanning trees, *J. Assoc. Comput. Mach.* **26** (1979) 177–182.

[19] F. K. Hwang, An $O(n \log n)$ algorithm for suboptimal rectilinear Steiner trees, *IEEE Trans. on Circuits and Systems* **CAS-26** (1979) 75–77.

[20] F. K. Hwang and Y. C. Yao, Comments on Bern's probabilistic results on rectilinear Steiner trees, *Algorithmica* **5** (1990) 591–598.

[21] J. W. Jiang and P. S. Tang, An algorithm for generating rectilinear Steiner trees (in Chinese), *J. Fudan Univ. (Natural Science)* **25** (1986) 343–349.

[22] A. Kahng and G. Robins, A new class of Steiner tree heuristics with good performance: The iterated 1-Steiner approach, *Int. Conf. on Computer-Aided Design* (1990) 428–431.

[23] A. Kahng and G. Robins, On a new class of iterative Steiner tree heuristics with good performance, Technical Report, Univ. of California - Los Angeles (1990).

[24] R. M. Karp, Probabilistic analysis of partitioning algorithms for the traveling-salesman problem in the plane, *Math. Oper. Res.* **2** (1977) 209–224.

[25] J. Komlós and M. T. Shing, Probabilistic partitioning algorithms for the rectilinear Steiner problem, *Networks* **15** (1985) 413–423.

[26] D. T. Lee and C. K. Wong, Voronoi diagrams in L_1 (L_∞) metrics with 2-dimensional storage applications, *SIAM J. Comput.* **9** (1980) 200–211.

[27] J. H. Lee, N. K. Bose and F. K. Hwang, Use of Steiner's problem in supoptimal routing in rectilinear metric, *IEEE Trans. on Circuits and Systems* **CAS-23** (1976) 470–476.

[28] D. S. Richards, Fast heuristic algorithms for rectilinear Steiner trees, *Algorithmica* **4** (1989) 191–207.

[29] D. S. Richards, On the effectiveness of greedy heuristics for the rectilinear Steiner tree problem, Technical Report, Univ. of Virginia (1991).

[30] M. Sarrafzadeh and C. K. Wong, Hierarchical Steiner tree construction in uniform orientations, Technical Report, IBM (1991).

[31] M. Servit, Heuristic algorithms for rectilinear Steiner trees, *Digital Processes* (1981) **7** 21–32.

[32] M. I. Shamos and D. Hoey, Closest-point problems, *Proc. of the 16-th Ann. Symp. on Foundations of Computer Science* (1975) 151–162.

[33] Shute G, M, Worst-case length ratios for various heuristics for rectilinear and euclidean Steiner minimal trees, Technical Report, Computer Science Dept., Univ. of Minnesota (1990).

[34] J. M. Smith, D. T. Lee and J. S. Liebman, An $O(n \log n)$ heuristic algorithm for the rectilinear Steiner minimal tree problem, *Eng. Opt.* **4** (1980) 179–192.

[35] J. M. Smith and J. S. Liebman, Steiner trees, Steiner circuits and the interference problem in building design, *Eng. Opt.* **4** (1979) 15–36.

[36] C. C. De Souza, O problema de Steiner na métrica retilínea propriedades, novas heurísticas e estudo computacional (in Portugese), Technical Report, Catholic Univ. of Rio de Janeiro (1989).

[37] C. C. De Souza and C. C. Ribeiro, Heuristics for the minimum rectilinear Steiner tree problem: New algorithms and a computational study, *Discrete Appl. Math.*, to appear.

[38] C. C. De Souza and C. C. Ribeiro, A tight worst case bound for the performance ratio of heuristics for the minimum rectilinear Steiner tree problem, *OR Spektrum* **12** (1990) 109–111.

[39] C. D. Thomborson (a.k.a. Thompson), L. L. Deneen and G. M. Shute, Computing a rectilinear Steiner minimal tree in $n^{O(\sqrt{n})}$ time, in Albrecht et al. (eds.) *Parallel Algorithms and Architectures*, Springer-Verlag, Berlin, **LNCS 269** (1987) 176–183.

[40] Y. C. Wee, S. Chaiken and S. S. Ravi, Rectilinear spanning trees using geographic nearest neighbors, *Proc. of the 27-th Allerton Conf. Communication, Control and Computing* (1989) 555–563.

[41] Y. Y. Yang and O. Wing, Optimal and suboptimal solution algorithms for the wiring problem, *Proc. IEEE Int. Symp. Circuit Theory* (1972) 154–158.

[42] A. C.-C. Yao, On constructing minimal spanning trees in k-dimensional spaces and related problems, *SIAM J. Comput.* **11** (1982) 721–736.

Chapter 3

Polynomially Solvable Cases

If the arrangement of the terminals in the plane is restricted then it may be possible to devise polynomial time algorithms to find the RSMT. In this chapter several such restricted problem domains are presented. For example, if all the terminals happen to lie on the boundary of a rectangle then the problem can be solved in linear time. Note that if the terminal set is presented as an unordered set of points then a $\Omega(n \log n)$ lower bound on the time required exists (by a simple reduction from sorting). Hence, in order to even discuss linear time algorithms, it is assumed that the terminal set is given as input in sorted order; either sorted by both coordinates, or sorted cyclically about the boundary of, say, a rectangle.

In this chapter the emphasis will be on the existence of polynomial time algorithms for various cases, rather than on presenting the details of the best known algorithms. Unfortunately, the fastest algorithms rely on detailed case analyses and, therefore, are beyond the scope of this book. Pointers to these results will be given. In the first section it is assumed that the terminals are all on the boundary of a rectangle. In Section 3.2 this is generalized to a convex boundary and, in the final section, to a few layers of such boundaries.

3.1 Terminals on a Rectangular Boundary

In this section it is assumed that all the terminals lie on the boundary of a rectangle. (It is assumed throughout that all lines, including boundary edges, are rectilinear, i.e., having a horizontal or vertical orientation.) First a simple case is discussed, where all the terminals are on two opposite sides of the boundary. Then an algorithm is sketched for the case where terminals fall properly on all four sides.

All the algorithms in this chapter make use of the grid graph $GG(N)$. This network is planar and, therefore, all the results for the planar case in the graphical SMT literature apply. In particular, when all the points are on the boundary of the rectangle, they are also on the boundary of $GG(N)$. Hence, it is the *1-planar* case, and Bern's algorithm [3] for that case indicates the existence of an $O(n^5)$ time algorithm for the rectangle case. It is easy to improve on this for the two-sided

case. By using detailed case analyses even the four-sided case can be solved in linear time.

3.1.1 Terminals on Two Opposite Sides of a Rectangle

The case where all the terminals lie on two opposite sides of a rectangle is known as the "$2 \times n$" case. This name is suggested by the simple structure of $GG(N)$, as seen in Fig. 3.1. The $GG(N)$ for the $2 \times n$ case is a series-parallel network and, hence, there is a linear time algorithm for this case, as discussed in Chapter 5 of Part II.

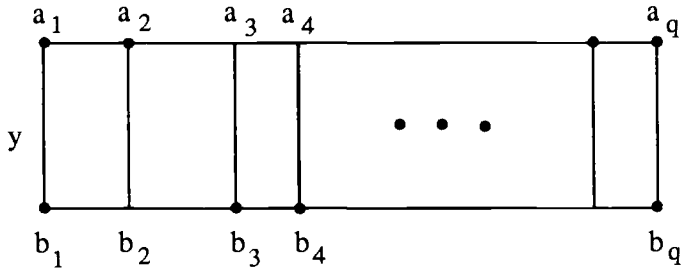


Figure 3.1: The grid graph for the $2 \times n$ case

Aho, Garey and Hwang [2] had earlier presented a linear time algorithm for the $2 \times n$ case that is a simplified version of that series-parallel algorithm (due to the regular structure of $GG(N)$). Let $a_1, a_2, \ldots, a_q$ be the vertices of $GG(N)$ along the top edge, and $b_1, b_2, \ldots, b_q$ be the corresponding bottom vertices; see Fig. 3.1. (Note that one can assume a_1, a_q, b_1, and b_q are all terminals. Otherwise there would be degenerate tabs that can be removed; see Chapter 1.)

The following definitions are analogous to those for the series-parallel algorithm in Part II. Let $N(i)$ be the set of all terminals among $a_1, \ldots, a_i$ and $b_1, \ldots, b_i$.

$B(i)$ = the minimum cost of a subtree connecting $\{a_i, b_i\} \cup N(i)$.

$F(i)$ = the minimum cost of a subtree connecting $\{a_i\} \cup N(i)$.

$L(i)$ = the minimum cost of a subtree connecting $\{b_i\} \cup N(i)$.

Note that $B(1) = F(1) = L(1) = y$, the vertical height of the rectangle. Let x_i be the distance from a_i to a_{i-1}, $2 \leq i \leq n$. Therefore, for $2 \leq i \leq n$:

$$B(i) = \min\{F(i-1) + x_i + y, L(i-1) + x_i + y, B(i-1) + 2x_i\}$$

$$F(i) = \begin{cases} B(i) & \text{if } b_i \in N \\ F(i-1) + x_i & \text{otherwise} \end{cases}$$

$$L(i) = \begin{cases} B(i) & \text{if } a_i \in N \\ L(i-1) + x_i & \text{otherwise} \end{cases}$$

The desired result is $B(q)$, which can easily be found in linear time using these recurrences. The tree itself can be reconstructed by saving back pointers at each stage.

3.1.2 Terminals on Four Sides of a Rectangle

The case where all the terminals happen to lie on the boundary of a rectangle was first solved by Aho, Garey and Hwang [2]; see Fig. 3.2. Their algorithm ran in $O(n^3)$ time (more precisely, it ran in $O(p^2q+pq^2)$ time, where $GG(N)$ is composed of p horizontal lines and q vertical lines). Later, linear time algorithms for this case were given by Cohoon, Richards and Salowe [8] and Agarwal and Shing [1]. Below the first linear time algorithm is sketched; details can be found in [8]. It is assumed there are at least two terminals on each of the four sides of the rectangle; otherwise there is a degeneracy that can be removed.

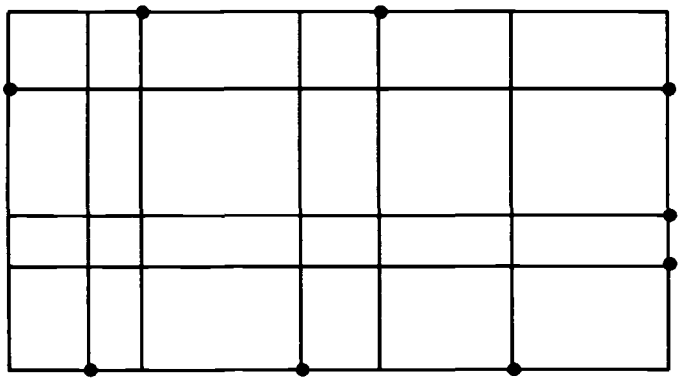


Figure 3.2: An example where all the terminals lie on the boundary of a rectangle

Begin by observing that the topology of such a tree is very restricted. Define an *interior segment* as a segment of a rectilinear tree that is not on the boundary of the rectangle; other terms, like *interior line* and *interior corner*, are defined analogously. A *spanning* interior line has both endpoints on the boundary. Recall that a leg of a complete corner, the body of a T, and a spoke of a cross all end at terminal and, hence, on the boundary. Further, none of these lines can intersect. These observations, with some case analyses and the results in Chapter 1, show that interior lines of an RSMT are configured in the following ways (see Fig. 3.3):

- A cross extending to all sides of the rectangle.
- A spanning interior line with two or more alternating incident interior lines.
- A complete interior corner with one interior line incident to one leg and one or more interior lines incident on the other leg.
- Zero, one, or two occurrences of (a) a spanning interior line with just one incident interior line and/or (b) a complete interior corner with just one

incident line on a leg. These each extend to three sides of the rectangle, including two "opposing" sides; if there are two occurrences they must use the same opposing sides. Further, there can be other nonintersecting spanning interior lines, with no incident interior lines, that connect the same opposing sides.

(The case of a complete corner with two or more interior incident lines on one leg and none on the other leg need not be considered since it is not fulsome; it is always possible to transform such a tree to increase the sum of terminal degrees.)

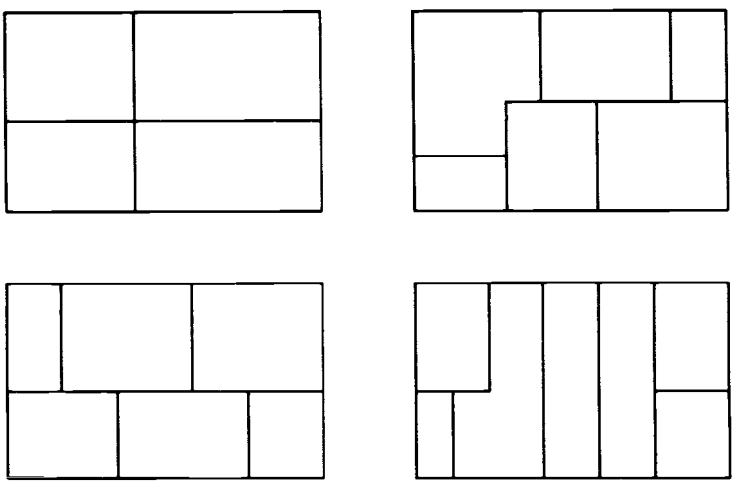


Figure 3.3: The four general arrangements of interior lines possible

Next it is shown how the optimal tree for each of these cases can be found in linear time. Hence, in linear time the RSMT must be found, since the cases are exhaustive. All orientations of these cases are not discussed since the terminal set can be rotated and reflected eight ways to force the tree into the orientation shown. (In practice, of course, one would simply make the routines more sophisticated, but such optimizations are not discussed here.) Note that while the *existence* of an RSMT with one of the above topologies was proven in Chapter 1 by assuming that the tree was canonical and fulsome, the algorithm can make local modifications (e.g., slides) so that the RSMT found may not necessarily be canonical.

The cross case

Regard the cross as being composed of two spanning interior lines. It is assumed that the vertical spanning line is slid as far left as possible (without increasing length) and the horizontal spanning line is slid as far up as possible.

A corner of the boundary is *forbidden*, relative to some specified interior lines, if the (fulsome) RSMT cannot have a terminal at the corner, nor can the Steiner tree extend around the corner along the boundary. Note that all four corner are

forbidden in the cross case. If there was a terminal at a corner a spoke could be slid to increase the sum of terminal degrees. If the tree extended around a corner, from a spoke, then that spoke could be slid to decrease total length.

Hence, all the terminals on the left boundary are connected, along that side of the boundary, to the endpoint of the left spoke; similarly for the other sides. Consider the topmost terminals on the left and right boundaries, and let x be the lower of these two. It can be assumed that the horizontal spanning interior line is incident on x. The vertical line is similarly constrained. Therefore the spokes of the cross can be fixed in position and then the rest of the tree is forced, since the corners are forbidden. (In this presentation, additional simple tests that could eliminate this case – and other cases below – are ignored entirely.)

The spanning line with several incident lines case

Assume that the spanning interior line has a horizontal orientation, with at least one incident interior line above and below it. Assume the line has been slid up, or down, as far as possible.

Suppose there are an odd number of alternating incident lines with, say, more going up than down. In that case the upper left and upper right corners are forbidden and the spanning line can be slid up to the point x, defined above. Similarly, if more lines point down than up then it can be slid down to y, the higher of the two bottommost terminals on the right and left boundaries. Suppose, instead, that there are an even number of incident lines. In that case it can be shown that two opposite corners are forbidden and that the spanning line can be slid to either x or y.

Hence, it can be assumed that the spanning line is incident to either x or y. For each placement of the spanning line the analysis below is done. Consider the example in Fig. 3.4, the other cases are similar. Note the upper left corner is forbidden, but the upper right corner can be used in an RSMT, so there are two cases as shown. This can be solved by working on the upper and bottom halves separately. Consider the top half; the bottom is symmetric. Note that if the left and right boundary terminals are ignored then the top half is essentially a $2 \times n$ case, where the entire bottom boundary is required to be in the tree (which corresponds to the spanning horizontal line in the original setting). (For the bottom half the entire upper boundary is required.)

This constrained case can be easily solved by creating an artificial $2 \times n$ case, as follows. The new top boundary has the same terminal placements as the original top boundary. The new lower boundary has equal number of terminals, each directly below the corresponding terminal of the top. The vertical separation is the same as the separation between the original top and the spanning line.

Lemma 3.1 *Any RSMT for the artificial $2 \times n$ case above can be transformed into an RSMT that includes the entire lower boundary.*

Proof: If any segment of the bottom boundary is absent then the corresponding segment on the top boundary must have been included; otherwise the tree would

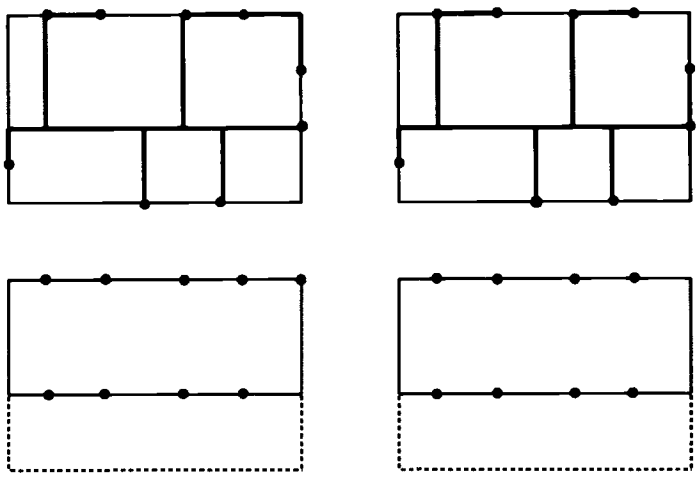


Figure 3.4: Two cases for decomposing the top half when there is a spanning line

be disconnected. If the missing lower segment is inserted and the upper segment is removed then another RSMT is obtained. Repeat this argument until the entire lower boundary is included. □

The two endpoints of the spanning line are included in this artificial $2 \times n$ case. Further, two instances (corresponding to the subcases where the upper right is used and is not used) are created, differing only in whether the upper right corner is artificially added as a terminal or not. After solving the upper and lower $2 \times n$ cases the additional terminals on the left and right boundaries are connected to the tree (by finding the optimal breakpoint between those connected around a corner and those connected to the spanning line) by a single scan.

The algorithm for this case can be summarized. First the two placements of the spanning line are found. For each placement, artificial $2 \times n$ cases for the top and bottom halves are created. For each of the corresponding solutions the remaining terminals on the right and left are connected into the $2 \times n$ solution to form a candidate Steiner tree. The best of all these attempts is reported. Since each of the $2 \times n$ cases are done in linear time, this case is also done in linear time.

The corner with incident lines on both legs case

Assume the corner is oriented as shown in Fig. 3.5. It can be assumed that the RSMT cannot extend leftward from d, along the boundary, as far left as the terminal b. Otherwise, the line beneath d could be slid over b creating a spanning vertical line. If this was the only line incident to the horizontal leg it follows that there is an RSMT with a topology already treated in the previous subsection; hence, there is no need not explore this corner topology any further since a shorter candidate RSMT cannot be discovered. Otherwise, if there were other lines incident to a horizontal leg then the next one, from the left, must point down to a terminal e;

hence, a segment of the horizontal leg can be slid down between b and e, increasing the sum of terminal degrees, contradicting the fact that only fulsome trees need to be considered.

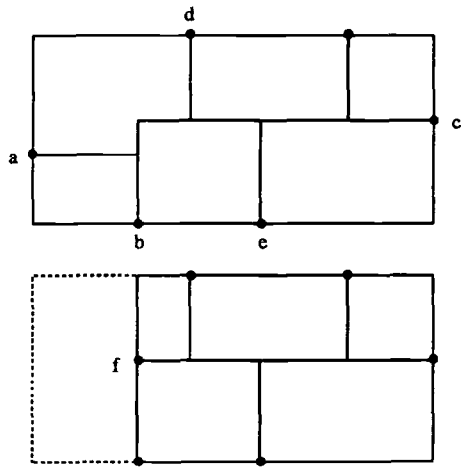


Figure 3.5: Reducing a corner case to a spanning line case on a smaller rectangle

Next, it is argued that it can be assumed a is the uppermost terminal on the left boundary. Any terminal above a must be reached from below, from a, since the connection cannot come around from d. It can be assumed that the interior line incident to a has been slid up as far as possible. If a is not uppermost then a is collinear with c, connected by a spanning horizontal line; again, nothing is gained by considering this case further. Therefore, the only remaining case is where a is uppermost. Further, the upper left corner is forbidden and d can be assumed to be the leftmost terminal on the top boundary.

There are further restrictions that can be imposed. By sliding the vertical leg leftwards it can be assumed that b is the leftmost terminal on the bottom, and that the bottom left corner is forbidden. Further it can be assumed, using similar arguments, that c is the lowest terminal on the right boundary above a.

Hence all that remains to be determined is the placement of the lines incident to the horizontal leg. This is easily done by creating an artificial instance of the previous case, i.e., a spanning line with several incident lines. Cut away the portion of the rectangle to the left of the vertical leg (the only terminals lost are on the left boundary and these must be reached from a — recall both left corners are forbidden). On the new left boundary introduce a new terminal f where the corner point was originally; see Fig. 3.5. If d is on the only line incident to the horizontal leg the corresponding candidate RSMT can easily be computed. Otherwise, there is an instance with a spanning horizontal line with two or more incident lines, which can be solved in linear time. This gives an overall linear time for this case.

The remaining case

A "vertical" orientation is assumed for this case, i.e., all components of interior lines extend from the top to the bottom boundary. Recall, that this case has zero or more vertical spanning interior lines with no incident lines and, perhaps, at either end (or both ends) there is a single interior Steiner point (on a vertical spanning line with just one incident line, or on a complete corner with just one line incident on its legs). This case is solved by exhaustively guessing how the ends are configured, and for each guess solving the remaining $2 \times n$ instance (see below). There are only nine guesses that need to be tried; each end can have a corner, a spanning line, or neither.

Only one of the subcases is discussed, since the others are treated similarly. Assume at the right end that a corner is present as shown is Fig. 3.6. The upper right corner is forbidden, so it can be assumed (by sliding the interior line at a to the right) that a is the rightmost terminal on the top boundary. Further, it can be assumed the vertical leg is slid to the right as far as possible. Hence, it can be assumed c is the rightmost terminal to the left of a. Let b be any terminal on the right boundary; all terminals to the right of a must be connected to b along the boundary. Note that the corner can be flipped to create a spanning vertical line at a. This situation can be reduced to the (right) end of a $2 \times n$ instance. Cut away the portion of the rectangle to the right of a (including all those terminals that b reaches) and introduce a new terminal d on the bottom boundary opposite a. Proceed by guessing how the left end is configured and analogously creating the left end of the $2 \times n$ instance. The remainder of the $2 \times n$ instance uses the remaining top and bottom terminals. After solving the $2 \times n$ case the ends can be easily restored, so the cost of the corresponding candidate RSMT can be calculated. Since a constant number of $2 \times n$ instances will be considered, all guesses can be tried out in linear time.

This completes the outline of a linear time algorithm for the case where all the terminals lie on a rectangular boundary.

3.1.3 Terminals on a Rectangle with Obstacles

Chiang, Sarrafzadeh and Wong [6,7] have generalized the above approach to the case where there are k rectangular obstacles in the interior of the rectangle. Edges of the Steiner tree cannot cross the interior of an obstacle. They solve the problem in linear time if there are a constant number of obstacles.

First they establish an analogue of Hanan's theorem. They show that it suffices to find a Steiner tree on the following grid graph. ¿From each terminal project lines across the interior until reaching the boundary or an obstacle. Similarly project lines across the interior from each corner of each obstacle. The intersections of these lines are the vertices of the grid graph and the segments of the lines are the edges.

They give an algorithm to find the RSMT on this grid graph that runs in $O(F_1(k)n + F_2(k))$ time, where F_1 and F_2 are exponential functions of k. The algorithm is based on the approach outlined above, in conjunction with an exhaustive

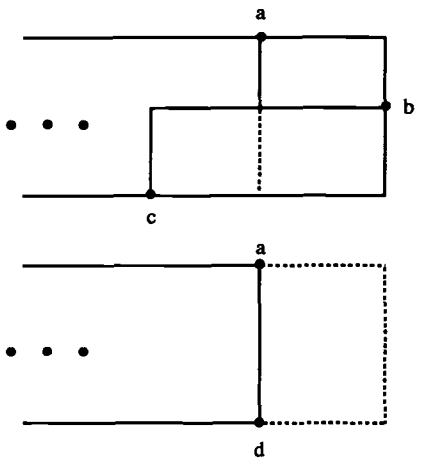


Figure 3.6: Reducing an end with a corner with one incident line to create a $2 \times n$ instance

analysis of how subcases can interact.

3.2 Rectilinearly Convex Boundaries

In Chapter 1 rectilinearly convex boundaries were discussed, together with related definitions. In particular, $Rconv(N)$ is the rectilinear convex hull of the terminal set. Recall that it is assumed, after some preprocessing, that $Rconv(N)$ has no degeneracies, i.e., it has four non-degenerate tabs and no staircases intersect. In this section the case where all terminals lie on $Rconv(N)$ is considered.

The results for 1-planar networks imply the existence of a polynomial time algorithm for this problem, by the following observations. Recall that an RSMT must be found in the subnetwork of $GG(N)$ on, or within, $Rconv(N)$. In the current problem all the terminals are on the infinite face of this planar subnetwork. Provan [10], building on the results of Erickson, Monma and Veinott [9], observed that one could use the known $O(v^2n^2 + \alpha)$ time algorithm for the 1-planar problem (where α is the time complexity of an all-pairs shortest paths calculation). This gives an $O(n^6)$ time algorithm, since $v = O(n^2)$, in $GG(N)$, and $\alpha = O(v^2)$ (since shortest path lengths are trivially calculated in $GG(N)$).

Bern, also building on the dynamic programming approach of Erickson et al., improved the time complexity. Using priority queues to avoid linear time scans for minima, he reduced the time complexity to $O(n^5 \log n)$ [4]. Using a "generalized Voronoi diagram" the time complexity was reduced to $O(n^5)$ [4]. Later, Bern [3] gave an improvement for the 1-planar problem (and other related problems) that translated as an $O(n^5)$ time algorithm for the current problem, that does not rely on geometric considerations (as his previous algorithms had).

Richards and Salowe presented an algorithm with time complexity $O(nk^4)$, where k is the number of corners on the boundary of $Rconv(N)$. Since $k = O(n)$ this can be no worse than the previous algorithms. However, when k is a constant this gives a linear time algorithm. In most applications k is small, compared to n. When all the terminals are on a rectangular boundary, as in the previous section, $k \leq 12$ ($k = 12$ when there are no terminals in the rectangle's corners) yielding get another linear time algorithm for that case. In fact, this algorithm is an outgrowth of the algorithm in the previous section. Unfortunately, it is a very long and detailed approach; a very brief overview is given in the remainder of this section.

A *simple* rectilinear Steiner tree is a Steiner tree which has no interior Steiner points; in other words, all interior lines are spanning lines with no incident lines. A *minimal simple rectilinear Steiner tree* (MSRST) is the shortest such tree (it might not be as short as an RSMT). Recall that in $2 \times n$ case all RSMTs were simple. An algorithm to find a MSRST for terminals in $Rconv(N)$ can be given that, as with the $2 \times n$ case, is based on the algorithm for the series-parallel case. Our network is not a series-parallel network but using the same basic outline it is easy to produce an $O(n^4)$ time algorithm for this case.. This can be reduced to $O(n + k^2)$ time, using bookkeeping and detailed case analyses.

Recall that in the last section one case was solved by decomposing the rectangular region into two smaller regions (which happened to be $2 \times n$ instances) and, by adding some additional artificial terminals, forcing the subsolutions into forms that could be combined to build a solution to the original problem. The algorithm in this section does exactly the same thing. A variety of subregions are defined and for each an RSMT is found (though in some of the regions only a MSRST needs to found). Each subregion is rectilinearly convex. In fact, each is bounded by four tabs that are lines in $GG(N)$, with the remainder of the boundary of the subregion coming from the boundary of $Rconv(N)$; see Fig. 3.7. It follows that there are only $O(n^4)$ such subregions. These subregions are solved in increasing order, by area.

Rather than give all the details only two representative cases are mentioned. First, suppose that there is an RSMT with a spanning interior line with two or more incident lines. For each of the $O(n)$ placements of the spanning line, the region can be decomposed into two subregions (that have already been solved since they have less area), as shown in Fig. 3.8. By using artificial terminals one can force the solution of, say, the left subregion to use its entire rightmost side in any RSMT it reports. Hence the subsolutions can be combined to build a solution to the original problem.

A second case is when an RSMT is assumed to have a complete interior corner with lines incident to both legs. In this case the initial region can be partitioned into four subregions; see Fig. 3.9. Again, the subregions can be combined to give the best RSMT with a corner placed in that position. Note that the position of each of the legs and the two incident lines nearest the corner must be specified. Hence, there are $O(n^4)$ decompositions to be considered, each of which can dealt with in constant time if the subregions have been precomputed.

There are a constant number of other cases that are handled similarly to these two cases. Each of the $O(n^4)$ regions can be solved by considering at most $O(n^4)$

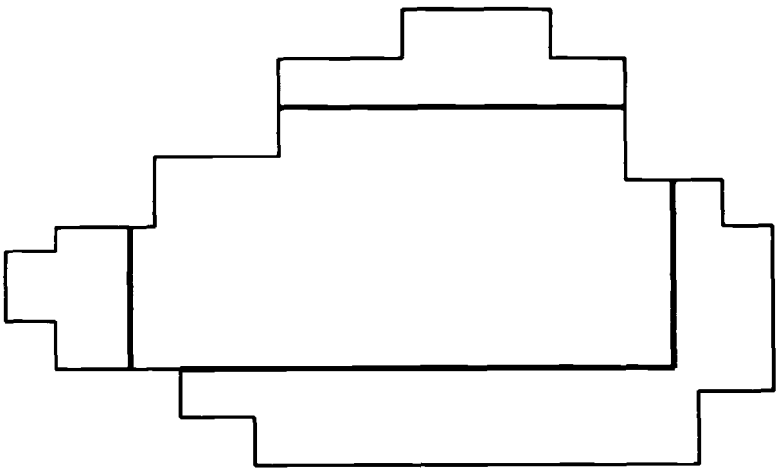


Figure 3.7: An example of a convex subregion delimited by four grid lines

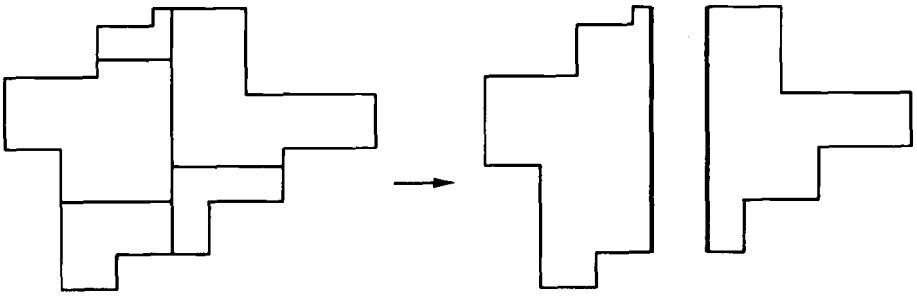


Figure 3.8: Decomposing a spanning line case into smaller subregions

decompositions. Hence there exists an $O(n^8)$ time algorithm for the problem. By sliding arguments all the decompositions can be forced to have their cuts "near" one of the k corners of the boundary of $Rconv(N)$; therefore each subproblem can be solved with $O(k^4)$ decompositions. A corollary of this is that each of the lines in Fig. 3.7 must also be near a corner, so there are only $O(k^4)$ subregions to solve. Considerations such as these lead to an $O(k^8+k^4n)$ time algorithm. By doing very detailed case analyses, this can be improved to $O(k^4n)$ time. For example, those subregions which need only contain solutions to the easier MSRST problem can be identified.

3.3 Layered Terminal Sets

If all the terminals do not lie on their rectilinear convex hull, there may still be enough structure to admit efficient algorithms. Successive *layers* of terminals can

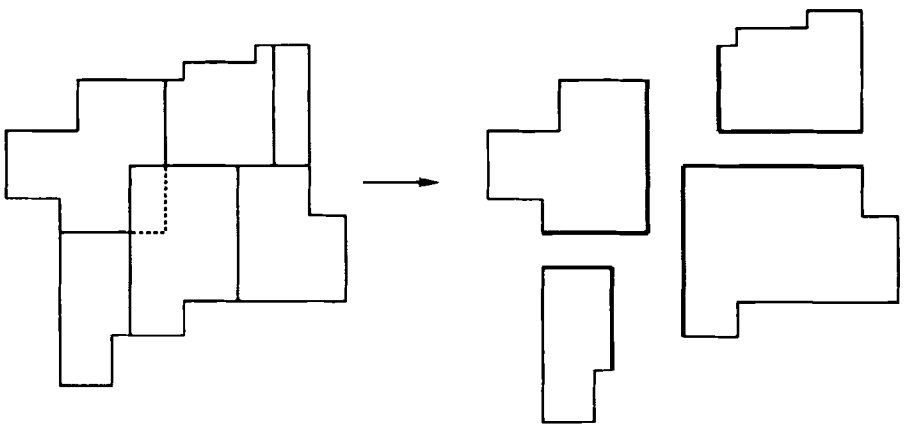


Figure 3.9: Decomposing a corner case into four smaller subregions

be defined. Let L_1 be the set of terminals on the boundary of $Rconv(N)$. Further, L_2 is the set of terminals on the boundary of $Rconv(N \setminus L_1)$, and L_i is the set on $Rconv(N \setminus (L_1 \cup L_2 \cup \cdots \cup L_{i-1}))$. Let $l_i = |L_i|$. If l is the least index for which $N = L_1 \cup L_2 \cup \cdots \cup L_l$ then the terminal set has l layers.

Consider the case where nearly all the terminals lie in L_1, i.e., $n - l_i$ is small. Bern solved the corresponding for general planar networks with most terminals on the infinite face [3]. This approach gives an $O(n^2 l_1^3 3^{n-l_1})$ time algorithm for the rectilinear case. This, of course, is polynomial time only when $n - l_1$ is bounded.

Bern and Bienstock [5] considered the case when $l = 2$. The gave a dynamic programming algorithm that builds small forests on the union of consecutive terminals (along the hulls) from L_1 and L_2, called "double intervals". Unfortunately there are many such cases to solve leading to an $O(n^{11})$ time algorithm. This approach can be extended, giving an $O(n^{6l-1})$ time algorithm, for each $l \geq 2$.

References

[1] P. K. Agarwal and M.-T. Shing, Algorithms for the special cases of rectilinear Steiner trees: i. points on the boundary of a rectilinear rectangle, *Networks* **20** (1990) 453–485.

[2] A. V. Aho, M. R. Garey and F. K. Hwang, Rectilinear Steiner trees: Efficient special-case algorithms, *Networks* **7** (1977) 37–58.

[3] M. W. Bern, Faster exact algorithms for Steiner trees in planar networks, *Networks* **20** (1990) 109–120.

[4] M. W. Bern, Network design problems: Steiner trees and spanning k-trees, Technical Report, Computer Science Division, Univ. of California at Berkeley (1987).

[5] M. W. Bern and D. Bienstock, Further polynomially solvable special cases of the Steiner problem in planar graphs, Technical Report, Xerox Palo Alto Research Center (1989).

[6] C. Chiang, M. Sarrafzadeh and C. K. Wong, An optimal algorithm for rectilinear Steiner trees for channels with obstacles, to appear in *Int. J. Circuit Theory Appl.*.

[7] C. Chiang, M. Sarrafzadeh and C. K. Wong, An optimal algorithm for rectilinear Steiner trees for switchbox with obstacles, Technical Report, IBM (1991).

[8] J. P. Cohoon, D. S. Richards and J. S. Salowe, An optimal Steiner tree routing algorithm for a net whose terminals lie on the perimeter of a rectangle, *IEEE Trans. on Computer-Aided Design* **9** (1990) 398–407.

[9] R. E. Erickson, C. L. Monma and A. F. Veinott, Send-and-split method for minimum-concave-cost network flows, *Math. Oper. Res.* **12** (1987) 634–664.

[10] J. S. Provan, Convexity and the Steiner tree problem, *Networks* **18** (1988) 55–72.

This Page Intentionally Left Blank

Chapter 4

Generalizations

There are several ways in which the basic rectilinear model can be generalized. Several are presented in this chapter, and in the following chapter these ideas are generalized further, in a specific application domain. In both chapters of Part IV other generalization are discussed. This chapter begins with problems defined on the the underlying rectilinear grid graph. In Section 4.2 directed formulations are presented. In the next two sections are discussions of hypercube settings and then other higher dimensional cases.

4.1 Rectangle Trees

Suppose that only a portion of the grid graph $GG(N)$ could be used in constructing a minimal tree. In particular, if $G = (V, E)$ is a subnetwork of $GG(N)$, with $N \subseteq V$, then problem is to find the Steiner minimal tree for N in G. In some circumstances this Steiner minimal tree will be a true RSMT, and in other cases it may be a good approximation.

Farley, Hedetniemi and Mitchell [2] have worked on this problem. Other work on this problem is in the routing literature, discussed in the next chapter. They only give results for "rectangle trees", which they define. A subnetwork of the grid graph is a *rectangle tree* if it is outerplanar (all vertices on the infinite face) with no degenerate edges (edges with both "sides" on the infinite face). An alternative constructive definition of the rectilinear topology of a rectangle tree can be given. Begin with a rectangle on four vertices, and iteratively apply this rule:

- Add a new rectangle by identifying one of its edges with an edge (with neither endpoint of degree 4) of the existing graph.

An example is given in Fig. 4.1.

It is clear from the constructive definition that a rectangle tree is a series-parallel network. Hence there is a linear-time algorithm for finding the optimal Steiner minimal tree in a rectangle tree. In [2] a linear time algorithm specifically for

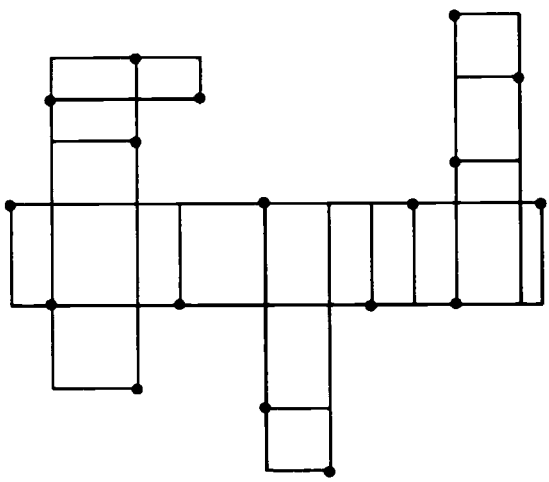


Figure 4.1: An example of a rectangle tree

rectangle trees was presented; however there were restrictions on the height/width ratio of some rectangles.

A rectangle tree G is *convex* if there is a rectilinear shortest path in G between each pair of vertices in G. Clearly $G = (V, E)$ is convex if and only if the boundary of $Rconv(V)$ coincides with the infinite face of G. If there exists a convex rectangle tree for N then the corresponding Steiner minimal tree is an RSMT for N [2]. This complements the results in Chapter 3. This result states that if all the terminals lie on $Rconv(N)$ and the corresponding subnetwork of $GG(N)$ is outerplanar then there is a linear time algorithm to find the RSMT. (Note that any Steiner tree in a rectangle tree is "simple", a MSRST, since there can be no interior Steiner points.)

4.2 Rectilinear Steiner Arborescences

In Part II the Steiner problem for directed networks asked for the shortest arborescence, rooted at a given point, that spans all the terminals. An *arborescence* is a rooted directed tree (sometimes called a *branching*). There is a natural analogue in the rectilinear metric.

Point a *covers* point b if there is a rectilinear shortest path from a to the origin that passes through b. Formally, a covers b if $|x_a| \geq |x_b|$ and $|y_a| \geq |y_b|$, and they are in the same quadrant. A *rectilinear arborescence* is a rectilinear tree that includes the origin in which each edge connects two points if and only if one covers the other. Clearly, an edge can be regarded as being "directed" from b out to a if a covers b, and the tree can be regarded as being rooted at the origin. For a given set N of terminals a *rectilinear Steiner minimal arborescence* (RSMA) is a minimum length rectilinear arborescence that spans N. When edges are only allowed between terminals (that cover one another) this defines the *rectilinear minimum spanning*

arborescence (RMSA) problem, analogous to the MST problem.

Only the case where the tree is entirely in the first quadrant will be discussed. The general RSMA problem can be effectively decomposed into four problems, one in each quadrant, because of the limited interaction between the quadrants. For example, the first quadrant and the second quadrant only interact along the positive y-axis. It may be mutually beneficial to use a longer tree edge along that axis than an optimal solution to either quadrant alone would require. By adding artificial terminals further along the y-axis one can force each quadrant to independently discover optimal solutions that can be glued together to get a tree optimal for both quadrants. Similar comments apply to the interactions between the other quadrants. Below, it is shown that only $O(n)$ such artificial points need to be tried. Hence each quadrant, with artificial terminals on each axis, needs to be solved at most $O(n^2)$ times. After solving all these problems, the optimal way to combine them, to solve the entire problem, can be found in $O(n^3)$ additional time [13]. Hence, below, it is assumed that all terminals are in the first quadrant; see Fig. 4.2.

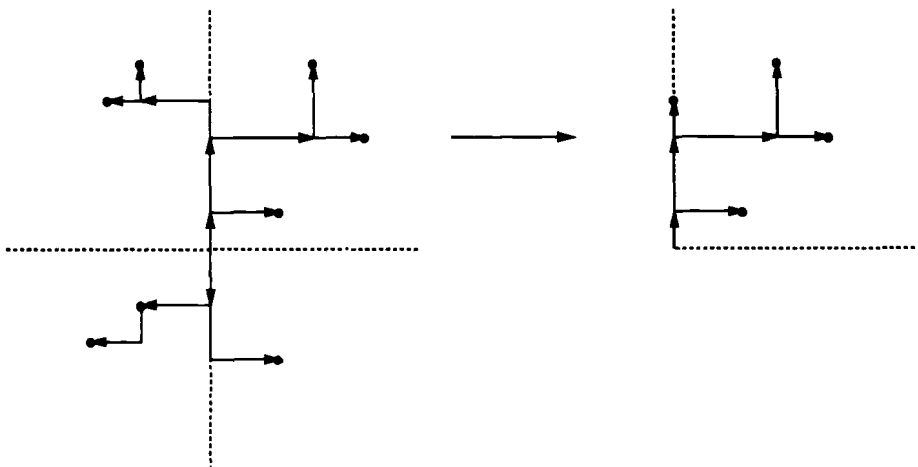


Figure 4.2: The first quadrant of an optimal solution is found by introducing an artificial terminal on an axis

Nastansky, Selkow and Stewart [12] proposed an integer programming formulation, which has exponential time complexity. Ladeira de Matos [9] proposed an exponential time dynamic programming algorithm to solve this problem. Such time complexities are familiar for Steiner problems. However the RSMA problem has not yet been shown to be NP-hard. Hence it seems possible that a polynomial time algorithm might exist.

Trubin [21] announced a polynomial time algorithm. The paper is difficult to read (and perhaps poorly translated). Central to Trubin's approach, which is based on an integer programming formulation, is the claim that the linear programming relaxation (of the dual) contains integral solutions. A counter-example to this claim

has been exhibited [13]. Therefore the complexity of the problem is still open.

Rao, Sadayappan, Hwang and Shor [13] have explored exact algorithms for special cases and heuristic algorithms, which are discussed below. Define $m(a,b)$ to be the point with coordinates $(\min\{x_a, x_b\}, \min\{y_a, y_b\})$. Note that, in an RSMA, if c has "children" a and b then c must be $m(a,b)$. Also note that every vertex in a rectilinear arborescence has indegree 1. They begin by proving a strong analogue of Hanan's theorem.

Theorem 4.1 *In an RSMA in the first quadrant if c is a Steiner point then there are two terminals a and b that cover c such that $x_a = x_c$ and $y_b = y_c$.*

Proof: Suppose there is a counterexample with Steiner points that do not satisfy the conditions of the theorem. Let c be such a point with maximal distance from the origin, with directed edges to a and b, which may or may not be terminals. Recall that c is $m(a,b)$. By hypothesis, if a and b are not terminals then they satisfy the theorem. In either case the theorem follows. □

As with Hanan's theorem, this result is best understood in terms of a grid graph, with horizontal and vertical lines through every terminal; in this case all edges of the grid graph are directed, either east or north.

A *slide* is a set of points such that no two cover each other.

Theorem 4.2 ([13]) *If N is a slide then there is an $O(n^3)$ time algorithm to find the RSMA.*

Proof: Clearly every terminal in such an N is a leaf in any Steiner arborescence. Since every Steiner point has outdegree 2 the RSMA must be a rooted binary tree. The terminals are naturally ordered left-to-right in the slide. A simple dynamic programming algorithm can be presented that forms an RSMA for each consecutive (in the ordering) group of i terminals, for $i = 2, 3, \ldots, n$. When working on a group of i terminals the position of the root is fixed; in $O(i)$ time one can find the optimal sizes of the two subtrees, since all the subtrees have been previously solved. This leads to an $O(n^3)$ overall time. □

For a special slide the length of the RSMA can be analytically derived. Let D_k be the set of terminals that have integer coordinates such that $x + y = k$. The following result can be proven by showing that the above algorithm is forced to build a balanced binary tree (see [13] for the proof).

Theorem 4.3 *An RSMA for $N = D_k$ has length $k\alpha + 2\beta$, where $k = 2^\alpha + \beta$, $0 \leq \beta \leq 2^\alpha$.*

A consequence of this theorem is that there is no constant bound on the "Steiner ratio" for arborescences,

$$\rho_A(N) = \frac{|RMSA(N)|}{|RSMA(N)|}$$

In fact, for $N = D_k$ it follows that $|RMSA(N)|$ is $k(k+1)$, since $n = k+1$. Hence $\rho_A(N) = \frac{k(k+1)}{k\alpha+2\beta}$, which grows like $\frac{n}{\log n}$ as $n \to \infty$.

Rao et al. [13] also looked at the ratio between the length of the RSMA and a "normal" RSMT (without regard as to whether points cover each other) for the same terminal set N. They showed the ratio is $\Theta(\log n)$. Further, they showed the length of the longest RSMA for n terminals in the unit square is bounded above by $2 + \sqrt{2n}$.

Rao et al. [13] also presented a fast and effective heuristic. The basic step is: choose a pair of terminals a and b such that $c = m(a, b)$ is as far from the origin as possible, record connections from c to a and b, and replace a and b in N by a new terminal at c (unless c was already a terminal). Repeat the basic step until N is reduced to the origin. This algorithm can be implemented by a (diagonal) line sweep approach that maintains a priority queue of the distances from the origin to each $m(a, b)$, where a and b are adjacent (with respect to the sweep line). This can be implemented in $O(n \log n)$ time. They show that the heuristic produces a rectilinear Steiner arborescence that is at most twice optimal.

Nastansky, Selkow and Stewart [12] extended this problem to higher dimensions where the definition of "cover" extends in the obvious way. In fact, their original exponential time algorithm was for d dimensions, $d \geq 2$.

4.3 Steiner Trees in Hypercubes

The next section discusses the generalization to d dimensions, using the rectilinear metric, where the distance from $x = (x_1, \ldots, x_d)$ to $y = (y_1, \ldots, y_d)$ is $|x_1 - y_1| + |x_2 - y_2| + \cdots + |x_d - y_d|$. This section treats the simplest d-dimensional problem, where each coordinate is 0 or 1. It is convenient, below, to discuss this problem as a Steiner problem for networks and as a d-dimensional rectilinear problem, interchangeably.

The d-dimensional hypercube, $Q(d)$, is represented by a d-dimensional rectilinear network. Each vertex is identified with a point in d-dimensional space, with each of the d coordinates being 0 or 1. Two vertices are connected by an edge if the rectilinear distance between the corresponding points is 1 (i.e., they differ in only one coordinate). Hence there are 2^d vertices, each connected to exactly one vertex along each of the d dimensions.

The rectilinear distance from u to v, in $Q(d)$, is simply the *Hamming distance*: the number of coordinate positions where u and v have different coordinates. A *Hamming path* from u to v is any rectilinear shortest path. If the Hamming distance between two vertices is k then clearly there are 2^k Hamming paths between them. The *weight* of a vertex is the number of its coordinates that are equal to 1; the weight of a vertex is its Hamming distance from the origin.

The Steiner problem specifies a subset N of the vertices of $Q(d)$ and asks for the SMT for N in $Q(d)$, or, equivalently, the RSMT for the points corresponding to the vertices in N. Miller and Perkel [10] have several results for this problem, including exact results for all N with $n \leq 5$. Clearly, when $n = 2$ the tree is just a Hamming path.

Let $L_d(N)$ be the length of the $RSMT(N)$ and

$$L_d(k) = \max_{|N|=k} \{L_d(N)\}$$

A *centroid* of the points in a hypercube, corresponding to N is the following point: the ith coordinate of the centroid is a mode of the ith coordinates of the n points. When n is odd the centroid is unique.

Theorem 4.4 ([10]) *Let $n = 3$ and let r be the number of coordinate positions where all three points of N share the same coordinate. Then $L_d(N) = d - r$ and $L_d(3) = d$.*

Proof: Let c be the centroid of $N = \{v_1, v_2, v_3\}$. In r of the coordinate positions c's coordinate agrees with the same coordinate for each $v \in N$. In the $d - r$ remaining positions c's coordinate disagrees with exactly one $v \in N$. The sum of the lengths of the Hamming paths from c to v_1, v_2, and v_3 therefore is $d - r$. Hence $L_d(N) \leq d - r$.

Let T be any RSMT for N. Suppose T has a Steiner vertex c with degree 3. In at least $d - r$ coordinate position c's coordinate must disagree with at least one $v \in N$. Hence the length of T is $\geq d - r$. If there is no such Steiner vertex then T must be a path , and the argument can be repeated with c being an endpoint of the path. Hence $L_d(N) \geq d - r$.

Finally, to show $L_d(3) = d$, conside these three points: $(0, 0, \ldots, 0)$, $(1, 1, \ldots, 1)$, and any other point. □

Miller and Perkel refined the centroid argument and used detailed case analyses to prove similar results. By using more complex analogues of r they determined exactly $L_d(N)$, when $n \leq 5$. Further, they found worst case terminals sets to show, for $d \geq 3$, $L_d(4) = \lfloor \frac{5}{3} d \rfloor$, and

$$L_d(5) = 2d - \left\lceil \frac{d}{10} \right\rceil - \left\lceil \frac{d-4}{10} \right\rceil$$

Miller and Pritikin [11] considered the problem where the terminal set from $Q(d)$ is W_k, the set of all points with weight k. Note that $|W_k| = \binom{d}{k}$. It is easy to show that the subgraph of $Q(d)$ induced by $W_{k-1} \cup W_k$ is connected. Hence W_{k-1} can act as a set of Steiner points for spanning W_k. Since any spanning tree of the connected induced subgraph has $\binom{d}{k} + \binom{d}{k-1} - 1$ edges, it follows that the same quantity is an upper bound on the length of an RSMT for W_k. They strengthened this observation by using only a subset of W_{k-1}, to obtain

$$L_d(W_k) \leq \binom{d}{k} + (1 + o(1)) \frac{\log(k-1)}{k-1} \binom{d}{k-1}$$

as $k \to \infty$. They used a generalization of Turan's theorem, due to Frankl and Rödl.

There is little hope of finding polynomial time algorithms, since the hypercube Steiner tree problem has been shown to be NP-hard [4,6,10]. In fact, the problem remains NP-hard even if the weight of every terminal is at most 2.

4.4 Higher Dimensions

There have been relatively few results for the rectilinear Steiner problem in d dimensions, using the generalized rectilinear metric. It is possible that trees in higher dimension are more complex, in some sense. Many of the results in Chapter 1 do not seem to generalize. For example, there is no know characterization of fulsome canonical trees, and the result about trombone wires cannot be extended to three dimensions. Below are results that do extend.

Recently Snyder [20] generalized Hanan's theorem to d dimensions.

Theorem 4.5 *There exists an RSMT, for any N, such that if $x = (x_1, \ldots, x_d)$ is a Steiner point then, for each i, $x_i = y_i$, for some $y = (y_1, \ldots, y_n) \in N$.*

The proof that Snyder gave is very long. It shows that while (generalized) trombone wires do exist they can be slid to satisfy the theorem. Du and Hwang [1] gave a shorter and more general argument, which is discussed in Section 1.1. of Part IV. As with $d = 2$, there is a geometric interpretation of this theorem. Pass d orthogonal $(d-1)$-dimensional hyperplanes, orthogonal to each of the d axes, through each of the terminals. The Steiner points can be located where d orthogonal hyperplanes intersect.

Snyder [20] observed that the problem can be mapped to a d-dimensional rectilinear network with $O(n^d)$ vertices (defined analogously to the two-dimensional $GG(N)$). Hence exact and heuristic algorithms for the Steiner problem in networks can be used. Snyder proposed exact solutions using the spanning tree enumeration algorithm of Hakimi (Section 3.1 of Part II) and the dynamic programming algorithm of Dreyfus and Wagner (Section 3.4 of Part II). Snyder also remarked that the 1-Steiner problem can be solved in polynomial time in d dimensions, for fixed d.

Sankoff and Rousseau [16] proposed a polynomial time algorithm to solve the RSMT relative to a given topology. That is, the tree and the position of each terminal in the tree are specified. Each coordinate of the Steiner points that minimize the length of the given tree is all that is to be determined. Clearly one can independently consider each full subtree (with all the terminals at the leaves), hence it is assumed that the algorithm is working with a full tree. The important simplifying observation is that, in the rectilinear metric, the choice of coordinates in one dimension has no effect on the contribution to the total length made by coordinates in other dimensions. In other words, the algorithm can solve d independent subproblems: solve for the i-th coordinate of each Steiner point in a given full subtree.

The algorithm to solve an instance of the latter problem (that is, for a single coordinate) begins by rooting the given tree at an arbitrary Steiner point. Recall, only the leaves are initially specified. A dynamic programming approach is used. It follows from the generalization of Hanan's theorem that each coordinate must take on one of $O(n)$ values. The *Steiner minimal subtree at v relative to x* is the shortest possible subtree rooted at v given that the coordinate at v is x.

Bottom-up, each subtree determines (and saves) the $O(n)$ shortest lengths of the Steiner minimal subtrees, relative to each possible coordinate value x. Suppose the algorithm is processing the subtree rooted at v and that it previously solved v's subtrees, rooted at $v_1, v_2, \ldots, v_k$, where $k \leq 2d$. Each of the $O(n)$ calculations at v can be done in $O(n^k)$ time, by considering all possible coordinate values at its children. The algorithm ultimately returns the shortest length calculated at the root. Since there are $O(n)$ internal Steiner points, the algorithm runs in $O(dn^{2d+2})$ time, which is polynomial time for fixed d. When the average degree in the tree is low the performance is much better. (When every node has an odd number of children the calculation is simplified; each value is simply the median of values of its childen [16].)

This algorithm is, of course, limited since it assumes a given tree. An exact algorithms can be derived by trying all possible trees. Unfortunately there are $\sum_{i=0}^{n-2}(n+i)^{n+i-2}$ trees to try. Greedy heuristics can be attempted that successively modify the topology. In some applications an intelligent guess can be made regarding the topology. These points are addressed further in Chapter 2 of Part IV.

It is still open for $d > 2$ to determine a tight lower bound of the Steiner ratio, ρ_d, for all N. Gilbert and Pollak [5] essentially proposed the following constructive upper bound. Let C_d be the set of $2d$ points with coordinates $(\pm 1, 0, 0, \ldots, 0)$, $(0, \pm 1, 0, \ldots, 0), \ldots (0, 0, 0, \ldots, \pm 1)$. (These points are the corners of "d-crosspolytope", of radius 1, containing all points within distance 1 of the origin.) By using a Steiner point at the origin it follows that $|RSMT(C_d)| = 2d$, whereas $|RMST(C_d)| = 2(2d-1)$. Therefore $\rho_d \leq \frac{d}{2d-1}$. It is widely conjectured that this bound is tight (e.g., [14,8]). Kahng and Robins [8] have shown that several heuristics based on Kruskal's and Prim's algorithms achieve, for some inputs, the $\frac{d}{2d-1}$ bound (the input they use is just many d-crosspolytopes chained together).

Moore (in [5]) showed that $\rho_d > \frac{1}{2}$. His argument was very general and was based on the observation that $2|RSMT(N)| \geq TSP(N) > |RMST(N)|$, where $TSP(N)$ is the length of the shortest Hamiltonian circuit for N — the traveling salesman problem. Note that as $d \to \infty$ the upper and lower bound meet; see also [3].

Let $M_d(n)$ be the length of the longest RSMT for any n points in the unit d-dimensional hypercube. Snyder [19] proved that $M_d(n) \sim \beta_d n^{\frac{d-1}{d}}$, where β_d is a constant for each dimension $d \geq 2$. He showed that $1 \leq \beta_d \leq d4^{\frac{1-d}{d}}$ [19,18]. Recently Smith [17] has improved these bounds to

$$\frac{d}{4e} < \beta_d \leq \frac{d}{2e}$$

The following derivation of the lower bound is due to Salowe [15]; for technical lemmas consult [7]. The volume of the d-crosspolytope with radius r is $\frac{2^d r^d}{d!}$. The Minkowski-Hlawka sphere packing theorem states the greatest packing density Δ of translates of the d-crosspolytope satisfy $\Delta \geq \frac{1}{2^{d-1}}$. So, for n sufficiently large, $n\left(\frac{2^d r^d}{d!}\right) \geq \Delta$ implies that n d-crosspolytopes can be packed into a unit hypercube

with $r \geq \frac{1}{4}\left(\frac{d!}{n}\right)^{1/d}$ (or $r \geq \frac{d}{4e}n^{-1/d}$). Note that a MST of N, the points at the centers of the d-crosspolytopes, must have length $\geq 2r(n-1)$. Using Moore's result, mentioned above, it follows that

$$|RSMT(N)| \geq \frac{1}{2}|RMST(N)| \geq \frac{1}{2}2r(n-1) \geq \frac{d}{4e}n^{\frac{d-1}{d}} - o(n^{\frac{d-1}{d}})$$

References

[1] D. Z. Du and F. K. Hwang, Reducing the Steiner problem in a normed space, *SIAM J. Comput.*, to appear.

[2] A. M. Farley, S. T. Hedetniemi and S. L. Mitchell, Rectilinear Steiner trees in rectangle trees, *SIAM J. Alg. Disc. Meth.* **1** (1980) 70–81.

[3] L. R. Foulds, Maximum savings in the Steiner problem in phylogeny, *J. Theor. Biology* **107** (1984) 471–474.

[4] L. R. Foulds and R. L. Graham, The Steiner problem in phylogeny is NP-complete, *Adv. Appl. Math.* **3** (1982) 43–49.

[5] E. N. Gilbert and H. O. Pollak, Steiner minimal trees, *SIAM J. Appl. Math.* **16** (1968) 1–29.

[6] R. L. Graham and L. R. Foulds, Unlikelihood that minimal phylogenies for a realistic biological study can be constructed in reasonable computation time, *Math. Biosci.* **60** (1982) 133–142.

[7] P. M. Gruber and C. G. Lekkerkerker, *Geometry of Numbers*, North-Holland (1987).

[8] A. Kahng and G. Robins, On performance bounds for two rectilinear Steiner tree heuristics in arbitrary dimension, Technical Report, Univ. of California - Los Angeles (1990).

[9] R. R. Ladeira de Matos, Rectilinear arboresence and rectilinear Steiner tree problems, Technical Report, Univ. of Alabama, Birmingham (1980).

[10] Z. Miller and M. Perkel, The Steiner problem in the hypercube, Technical Report, Dept. Math, Miami Univ., Oxford, Ohio (1990).

[11] Z. Miller and D. Pritikin, Applying a result of Frankl and Rödl to the construction of Steiner trees in the hypercube, Technical Report, Dept. Math, Miami Univ., Oxford, Ohio (1990).

[12] L. Nastansky, S. M. Selkow and N. F. Stewart, Cost-minimal trees in directed acyclic graphs, *Z. Oper. Res.* **18** (1974) 59–67.

[13] S. K. Rao, P. Sadayappan, F. K. Hwang and P. W. Shor, The rectilinear Steiner arborescence problem, *Algorithmica* **7** (1992) 277-288.

[14] M. H. Rudowski, Hypothesis on the length of minimal Steiner tree in multidimensional spaces with a rectilinear metric (in Polish), *Arch. Autom. Telemech.* **29** (1984) 353–358.

[15] J. S. Salowe, A note on lower bounds for rectilinear Steiner trees, Technical Report, Univ. of Virginia (1991).

[16] D. Sankoff and P. Rousseau, Locating the vertices of a Steiner tree in an arbitrary metric space, *Math. Program.* **9** (1975) 240–246.

[17] W. D. Smith, Personal communication (1990).

[18] T. L. Snyder, Lower bounds for rectilinear Steiner trees in bounded space, *Inf. Process. Lett.* **37** (1991) 71–74.

[19] T. L. Snyder, On minimal rectilinear Steiner trees in all dimensions, *Proc. of the 6-th ACM Symp. Computational Geometry* (1990) 311–320.

[20] T. L. Snyder, On the exact location of Steiner points in general dimension, Technical Report, Georgetown Univ. (1990).

[21] V. A. Trubin, Subclass of the Steiner problems on a plane with rectilinear metric, *Cybernetics* **21** (1985) 320–324.

Chapter 5

Routing

The automatic routing of wires in a circuit has been the premier application of the rectilinear Steiner tree problem, since Hanan's original paper. There have been other applications, such as heating systems in large buildings [20]. However, from the early work on printed circuit boards (PCB) to today's active research programs on very-large scale integrated circuits (VLSI), the rectilinear Steiner problem is frequently addressed. For a variety of reasons, principally due to manufacturing technologies, the wires are usually assumed to be composed of rectilinear segments.

The first section discusses the nature of the application and its relationship to various Steiner problems. Section 5.2 discusses the simpler case of individual wiring problems, and Section 5.3 presents techniques for handling many wiring problems simultaneously. The final section discusses the uses of non-planar routing domains.

5.1 Introduction

A lot has changed during the transition from PCB to VLSI applications. The PCB problem could be effectively mapped into a planar instance (even though wiring can be done on more than one layer). However, as the number of layers available to VLSI designers has increased from 2, to 3, to 4, and beyond, it has become harder to map corresponding routing problems into the plane in a meaningful or effective way.

Another change in the VLSI domain, of course, is that the sheer size of the overall problem has increased by several orders of magnitude, and it will continue to increase. There are two ramifications of this. First, the overall design process has become automated, from beginning to end, and can be viewed as a pipeline of different design steps. There are many possible ways to partition the process into a series of functions. Typically, the initial steps specify a series of modules, each with as set of *pins* or *terminals* (used as interfaces to other modules). A set of *nets* are also specified, where each net is a set of terminals, from various modules, that need to be made electrically common, i.e., connected by wires. The connection for a two-terminal net is a simple path, but for multiterminal nets the connection is

a tree. Such a connection is a Steiner tree since it can have vertices that are not terminals (though the connection is not necessarily a Steiner minimal tree).

The second ramification of the sheer size of VLSI problems is that the wire routing process has to be decomposed into, at least, two steps: global routing and detailed routing. This is not because the nets are becoming larger, but because there are so many nets that routing each one separately would lead to intolerable congestion. The *global routing* stage is a planning stage that decides what regions of the layout are going to be crossed by the wire segments of each tree. The *detailed routing* stage actually maps the logical wire segments, crossing each region, to rectilinear segments in the circuit layout.

In a typical VLSI application the average number of terminals per net is between 2.5 and 3.5 [7]. For example, Table 5.1 gives the number of nets for a recent VLSI chip. In addition to these small nets, there are some very large global connections: power lines, ground lines, clock lines, and bus lines. However, these either are routed on their own layer or are fixed at an early stage, and will not be discussed further here.

net size	2	3	4	5	6	7	8	9	10	11	12	13
no. of nets	6010	1822	808	503	353	248	129	91	66	61	28	21
net size	14	15	16	17	18	19	20	21	22	23	24	25
nr. of nets	12	7	1	1	4	11	7	4	7	1	0	1

Table 5.1: Distribution of net size in a VLSI CPU circuit

5.1.1 The Nature of VLSI Steiner Problems

Over the past 25 years it is common to hear of RSMT routing applications. However, the considerations above have accumulated to define a set of Steiner tree problems that are distinct from the classic RSMT formulation. For a more detailed treatment than presented here, consult the survey work of Lengauer [13], Korte, Prömel and Steger [8], or Kuh and Marek-Sadowski [9].

Recall that placement of modules precedes the wire routing. A "module" can be as small as a gate from a standard library or (in hierarchical design systems) as large as a precomputed monolithic circuit block. There are many placement regimes — such as full-custom, gate array, and sea-of-gates — that define routing "channels" in different ways. In all cases a rectilinear network is the appropriate model for the routing problem. A *rectilinear network* is a network with vertices, identified with points in the plane, connected by edges that are single vertical or horizontal segments. Note that the distance metric in a rectilinear network is *not* necessarily the rectilinear metric, since there can be "holes" in the network. A hole corresponds to a portion of the chip that is not available for routing; Fig. 5.1 illustrates the sparser network that can result. There are also arbitrary edge weights, discussed below. Since many routing problems are posed as Steiner tree problems on rectilinear networks, they are often mistakenly labeled rectilinear Steiner tree problems.

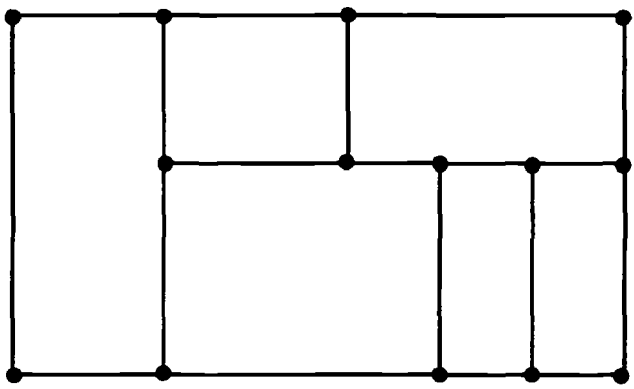


Figure 5.1: A rectilinear network defining the channels for routing

Another characteristic of VLSI problems that distinguishes them, is that length is not the only criterion for judging a Steiner tree. Length, of course, is important for several reasons: to decrease the area consumed by wires, to decrease fault-susceptibility, and to decrease the cycle time of the circuit. However, other issues are important. A bend in the wire requires (for many technologies) a change of layer using a *via*, which consumes area and increases fault-susceptibility. In general, as the number of nets keeps increasing, "routability" (the existence of any solution for a set of nets) is more important than length. This is addressed by global routers.

A *global router* begins by formulating a coarse version of the problem. Sometimes this is done by collapsing a channel to a point (and coalescing all the terminals in that channel). An analogous second formulation is used below. The routing region (or the original large rectilinear network) is reduced to a small rectilinear network, called the *reduced routing network*. The routing region is sliced up into subregions by vertical and horizontal cuts. Each subregion corresponds to a vertex in the reduced routing network, and associated with each such vertex is a set of nets with at least one terminal in that subregion. In Fig. 5.2 a very simple reduced routing network with three nets is shown. The task of a global router is to find Steiner trees in the reduced routing network for each net.

Each edge in the routing network corresponds to a *channel*, through which many different nets can be routed. Each channel has a *capacity*, a limit on the number of nets that can route through it. Hence, the global router may not be able to use the shortest Steiner trees without violating channel capacity constraints. For example, in Fig. 5.2, if the constraint of the left vertical channel is 1 then either the b net or the c net must be routed by a connection of length 3, rather than length 1.

A simple and popular strategy for routing all nets in the global routing phase is to sequentially route the nets, one at a time. To avoid exceeding channel capacities a cost, or penalty, can be associated with each channel so that channels which are currently underfilled will be more likely to be used when the next net is routed. (The penalties change after each net is routed; there are many *ad hoc* methods for calculating the penalties.) Hence, the global router can be viewed as solving a

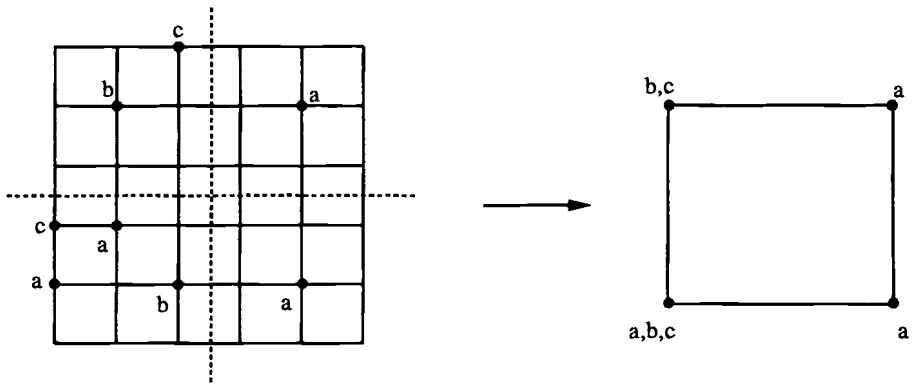


Figure 5.2: A simple 2 × 2 reduced routing network

sequence of Steiner tree problems on a edge-weighted rectilinear network.

The subsequent detailed routing stage can be, and has been, viewed as an instance of the global routing problem with unit channel capacities. However, the most important issue in detailed routing has been routability (especially in tightly packed routing problems); the existence of the Steiner trees has been more important than their lengths. Therefore the large literature on detailed routing will not be discussed further.

The main conclusion to the above discussion is that in most VLSI applications the Steiner tree problems are for terminal sets on weighted rectilinear networks. Therefore most of the results on the rectilinear Steiner problem have limited applicability.

The literature on routing is vast, reflecting its economic importance. Below a handful of papers that make interesting contributions to the Steiner tree literature are reviewed. This section has been a broad overview. It should be remembered that researchers are constantly changing their problem statements as technology changes.

5.2 Heuristics for Single Nets

Despite the comments in the introduction, there are several instances where VLSI systems use RSMT heuristics for routing. Often the routing region does not have holes and some of the nets need to be routed with near minimal length. In fact, many of the heuristics in Chapter 2 are from the VLSI literature. In addition, heuristics from Part II are also used, and some of the contributions to that literature, for rectilinear networks, have been made by VLSI researchers. Typically no performance measures are reported for these heuristics.

Reich and Widmayer [17] wrote such a paper (which was already described in Section 6.6 of Part II). They pointed out that when a module is "placed" it can often still be rotated and reflected so that the terminals are not necessarily fixed,

but that each terminal can be in a small set of possible positions. They gave two algorithms for finding Steiner trees which span at least one of the positions allowed for each terminal.

5.2.1 Prim-Based Heuristic using Common Edges

Xiong [26] proposed a heuristic for rectilinear networks without arbitrary weights (the length of an edge is just its rectilinear distance). The control structure of the heuristic has two phases, each phase is essentially a Prim-based heuristic for Steiner minimal trees (similar to [21]). Each step of each phase finds the nearest terminal not in the current tree and connects it to the tree with a shortest path.

In the first phase, each "path" that is selected can be placed in many ways, using different rectilinear paths of the same length. The "left-side path" and the "right-side path" are both found and the edges on these paths are marked. (These two paths are analogous to the two orientations of L-shaped wires; see Fig. 5.3.) If an edge is marked twice it is a *common edge*. Since neither of the paths are actually selected in the first phase, only terminal-to-terminal connections are found at each step.

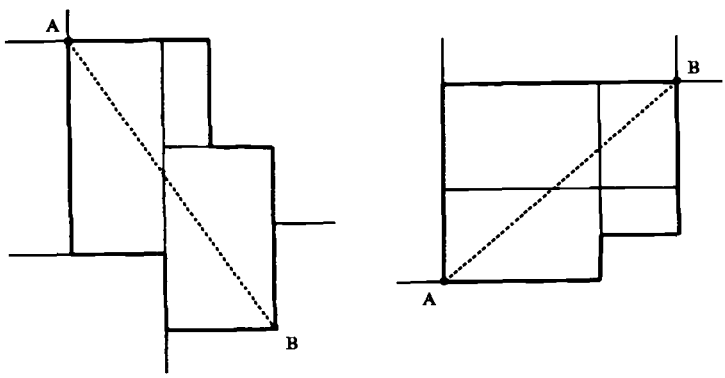


Figure 5.3: Examples of left-hand and right-hand paths between two terminals

In the second stage, the Prim-based heuristic decides on an embedding for each path. Hence, it finds the shortest path from a terminal to any point in the current tree. The embedding chosen is the one that favors the use of common edges. (This allows the introduction of Steiner points and tends to produce shorter trees.) The second stage is run several times with different starting vertices. (There is a backtracking mechanism, for resolving ties, that is unclear.)

5.2.2 Prim-Based Heuristic with Magnetic Attractions

Hsu, Pan and Kubitz [6] proposed a Prim-based heuristic which uses a novel method for selecting each successive path. They modified the internal details of the basic shortest path computation so that (of the many possibilities) the shortest path

that is selected will be one that bends towards other terminals in the net that have not yet been included. The path is bent due to the "magnetic attractions" of those other terminals. (They also present a Kruskal-based variation of this heuristic.)

The modified shortest path heuristic works on any rectilinear network. It finds a path from a source s to a sink t. It is related to the A^* approach (see [19] for a survey of these and related techniques). This approach maintains the set of *visited* verticesindexvisited vertex (initially empty), the set of *fringe* vertices (initially $\{s\}$) which can be "seen" from visited vertices, and the remaining *unseen* vertices. Each fringe vertex x has a priority $f(x) = g(x) + h(x)$, where $g(x)$ is the shortest path length from s to x using visited vertices, and $h(x)$ is (typically) the rectilinear distance from x to t. The $h(x)$ term "guides" the search for a shortest path towards t. The fringe vertex with smallest priority is selected and marked as "visited"; it also ensures that its neighbors are fringe vertices with correct priorities. The new heuristic has an additional feature.

The priority is changed to $f(x) = g(x) + h(x) - i(x)$, where $i(x)$ is determined, generally, by how close other terminals are to x. Consider the example in Fig. 5.4, where y has just been visited, causing x and z to become fringe vertices, each with a "backpointer" to y. To compute $i(x)$ consider the unseen terminals in x's halfplane opposite of y (to the right of x), namely $\{b, c, d\}$; for $i(z)$ consider the halfplane (opposite from y) below z and use $\{a, d\}$. The algorithm computes

$$i(x) = i(y) + \max_{v \in \{b,c,d\}} \{F(x, v) \cos \theta_v\}$$

where $F(x, v)$ is the reciprocal of the rectilinear distance from x to v, and θ_v is the angle of the line xv to the vertical halfplane boundary. (If x was already a fringe vertex then $f(x)$ is already defined; it changes its pointer to y only if it can decrease $f(x)$.) The definition of $i(x)$ is somewhat *ad hoc*. Even so, using the new $f(x)$ will clearly encourage the shortest path computation to "expand" preferentially in the direction that takes it close to nearby terminals.

5.2.3 Steinerization Heuristic for Row-Based Layouts

Whenever one goes from general problems to a specific application, inevitably there are idiosyncratic changes in the heuristics. This subsection gives such an example. In a *row-based* layout the modules are arranged in tightly packed rows, separated by channels. Both gate arrays (where the channels have fixed height) and "standard cell" layouts (where the channel height is to be determined) can be cast this way. (Even sea-of-gate layouts can be regarded as a row-based layout, with zero-height channels.) At various points in a row a wire can pass through slots called *feed-throughs*; see Fig. 5.5. While minimizing congestion and wire length within a channel is important, conserving the relatively scarce feed-throughs is also important.

Lee and Sechen [12] presented a heuristic that reflected these constraints. First, they modified the underlying metric. Rather than using the rectilinear metric, they used a weighted rectilinear metric, where the vertical separation could be weighted

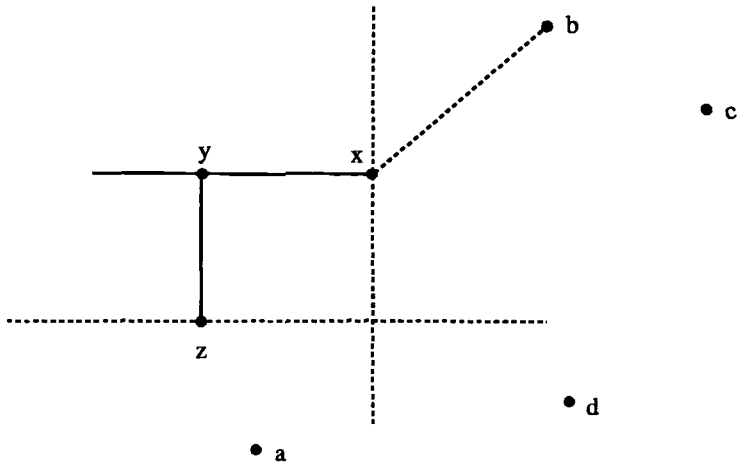


Figure 5.4: The effect of nearby terminals on the expansion from y

more heavily than the horizontal separation. The vertical separation weight is chosen to reflect the availability of feed-throughs.

Second, they gave a variation of the old Steinerization approach. They began with a MST T. The basic step was to take a terminal x and a direction, such as north, and to call *Steinerize(x,north)*. This procedure would find the set of terminals, *North(x)*, adjacent to x in T that were north of x. If there were exactly two terminals in *North(x)* then these three terminals (including x) would be Steinerized in the normal fashion; i.e., connected to a Steiner point determined by the medians of the x- and y-coordinates. If *North(x)* contains three or more terminals then a simple Steiner tree is built to connect all these to x: a vertical line is constructed above x to which each terminal in *North(x)* is connected by a horizontal wire segment. In all cases T is immediatedly updated to reflect the new connections.

The heuristic loops through each terminal x, and for each x it makes these calls: *Steinerize(x,north)*, *Steinerize(x,south)*, *Steinerize(x,west)*, *Steinerize(x,east)*. An example is in Fig. 5.6, where the terminals are numbered in the order they are considered. The relatively high cost of feed-throughs is reflected in two subtle ways. First, the set *North(x)* is connected to a single spine. Second, the north and south directions are done before the east and west are tried. For both reasons there is a preference for Steiner trees that reach more terminals through fewer vertical segments.

5.2.4 Parallel Heuristics

There have been virtually no parallel algorithms proposed for any Steiner tree problem. This is peculiar, as there is a large literature for parallel graph algorithms. In this section two are briefly mentioned, that make a strong assumption about the

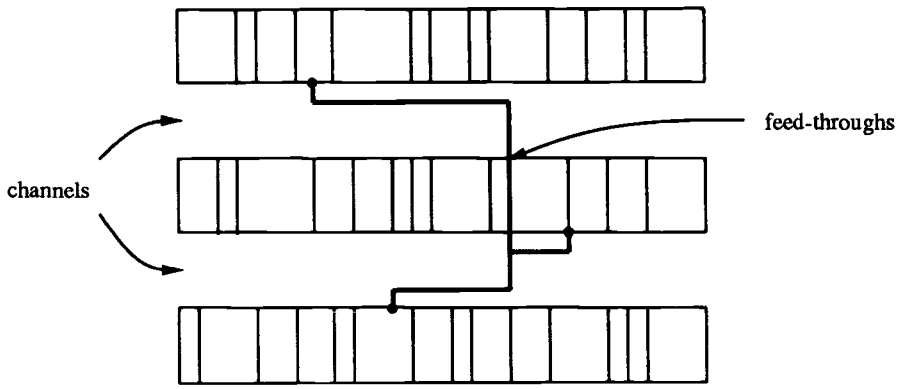


Figure 5.5: Row-based layout

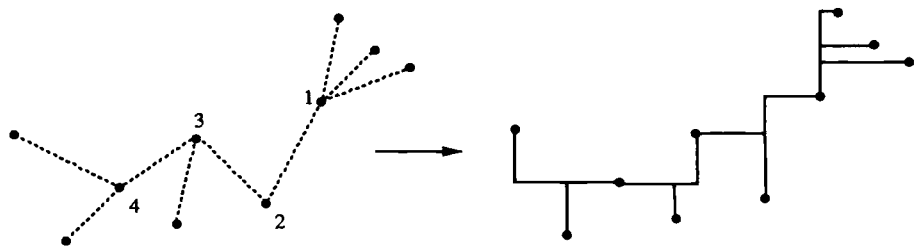


Figure 5.6: Steinerizations with a preference for vertical segments

problem domain.

The assumption is that the plane, or the usable *routing region*, is partitioned into many unit squares or *cells*. Each terminal occupies a cell. A wire is a rookwise connected sequence of cells. For 30 years there have been algorithms proposed that have data structures with an entry for each cell. Such a data structure has space complexity proportional to the size of the routing region, which is perhaps far larger than any economical problem encoding. The same is true of the time complexity of algorithms that scan such a data structure. For example, a simple rectangular routing region may be composed of 10000 cells but contain only 50 terminals. The assumption that the routing region is not too large was valid for PCBs but is less tenable for VLSI circuits as the cell size continues to become smaller.

The classic algorithm for this model is the Lee maze router [10] for computing rectilinear shortest paths through an array of cells. There have been many variations proposed [13] but the central idea is the propogation of a "wavefront" from the source towards the sink. An illustrative example is in Fig. 5.7. Since there are many independent cell calculations made to make as the "wave" moves outwards it is easy to find ways to parallelize this approach; the extreme case is when every cell has its own processor.

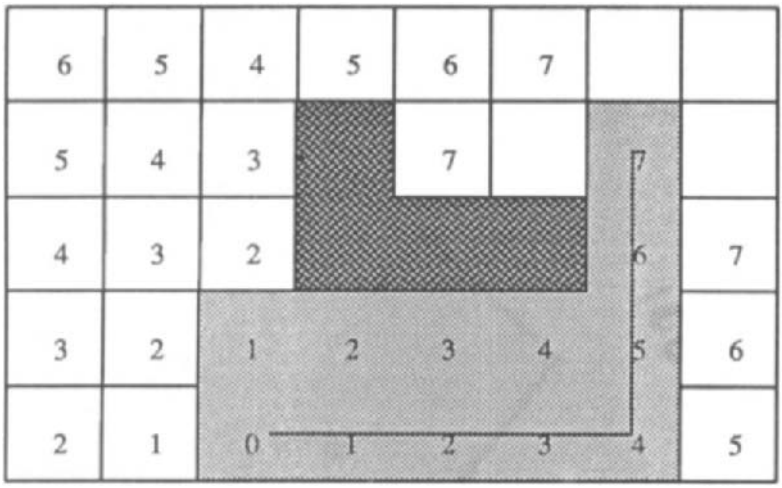


Figure 5.7: Wiring region resulting from using a wavefront shortest path algorithm on an array of cells, with an obstacle

Watanabe and Sugiyama [24] have proposed a Prim-based heuristic based on the Lee shortest path approach. The novel feature of their heuristic is that they do not decide on a particular shortest connecting path at each step. Instead they find the *wiring region* through which any shortest path must pass; see Fig. 5.7. (This region is easy to find by doing an expansion backwards from the sink to the source.) At each stage of their heuristic, they find the unused terminal that is nearest to any terminal or wiring region. Suppose a wiring region is closer than any terminal. Note that this will imply that the wire in the nearest wiring now must be bent to meet the new wire, effectively reducing the size of the wiring region; see Fig. 5.8.

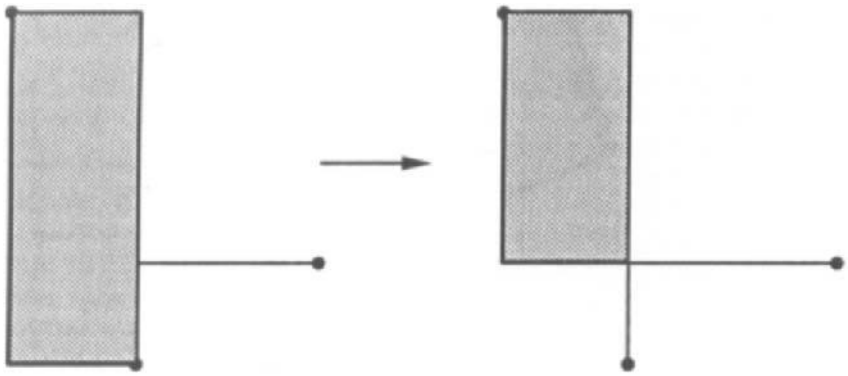


Figure 5.8: The effect on a wiring region when a new wire is attached

To analyze their heuristic assume there is one processor per cell. Let the routing region be contained in an $r \times r$ square array of cells. Lee's algorithm runs in $O(r)$ time in parallel (and $O(r^2)$ time sequentially). There are $n-1$ applications of this leading to a total parallel time complexity of $O(nr)$.

Another parallel heuristic, that also assumes a cell array, was given by Hambrusch and TeWinkel [5]. They use one processor per cell and have no obstacles in the routing region. (Their primary application was image data, not VLSI layouts.) Their heuristic was not based on the Lee algorithm, but, instead, on the literature for image analyses with a mesh of processors. The basic approach is based on Sollin's MST algorithm [22], which begins with n singleton trees, and repeatedly performs this step in parallel: connect each subtree to its nearest subtree. The difficulty with this approach is that cycles can arise. They actually present two heuristics: one finds and breaks cycles and the other one prevents cycles from occurring. Both heuristics are fairly detailed, but run in $O(r \log n)$ parallel time.

5.3 Heuristics for Multiple Nets

First some heuristics for routing all nets at once are presented. Then some representative results on global routing are given. Finally, a recent heuristic that routes all nets incrementally is discussed.

There are hundreds of papers on the problem of routing multiple nets. Most mention Steiner trees without adding to our knowledge of Steiner trees. Here, a few of these papers are mentioned that do make interesting contributions.

Ng, Raghavan and Thompson [15] argue that a single Steiner tree is of little use since there may be many unforeseen constraints. Instead, during routing, they want to have several Steiner trees to choose from. They give a "language" for coding procedures to generate many Steiner trees. They have successfully inserted such code into their routing software.

A popular technique is to allow channels to overfill and then, at a later stage, perform "rip-up" operations to move wires to less congested channels. Lee, Bose and Hwang [11] give an example of such a technique. It begins by perturbing just horizontal segments and then it works on vertical segments. If a channel is overfilled it tries to move one or more wire segments to "nearby" channels (that are not overfilled). The process iterates until channel capacities are acceptable or no improvement can be made (and the process fails). (Rip-up operations are also used to give space to nets that cannot be routed.)

Luk, Sipala, Tamminen, Tang, Woo and Wong [14] discuss the relationship between global routing and floorplanning. *Floorplanning* is the phase when the chip area is subdivided into regions small enough for individual modules (or larger "macros", depending on the scale used). A popular technique is top-down floorplanning, which repeatedly cuts rectangular regions into smaller rectangles. The adjacency graph of these rectangles is a series-parallel graph and global routing can be done on this graph. Hence, linear time algorithms are available.

Zhang, Pillage and Rohrer [28] discuss the possibility of combining aspects of the module placement phase and routing with Steiner trees. If the module placement

is given, Cong [3] discusses the possibility of combining aspects of final terminal placement with Steiner trees.

5.3.1 Packing Trees

The situation with the most *a priori* information is when the routing region is represented by a rectilinear network, with edges for every track a wire could occupy. Call this the (detailed) *routing network*. Each net corresponds to a subset of the vertices. The problem is to find a set of trees, where each net is spanned by a tree.

Typically it is required that either the trees are vertex-disjoint or edge-disjoint. If the routing is on a "single-layer" then the trees must be vertex-disjoint so that wires do not touch. However, most routing is done on at least two layers. Therefore, the wires are allowed to cross at vertices (as one wire passes under another, or they pass "knock-kneed" at right angles) but not to share an edge. (With enough layers, edges could be shared, but it is better to augment the routing network; see Section 5.4.)

The decision complexity of this type of tree packing problem has been discussed in Section 6.7 of Part II. It was shown that, even for two-terminal nets, the problems are NP-complete. Here some algorithmic results for restricted cases are mentioned.

Becker and Mehlhorn [1] considered the case of two-terminal nets with all terminals on the infinite face. The results are for any planar network, with even degrees for every internal vertex (which is true of many routing networks). They gave an $O(v^2)$ time algorithm, where v is the number of vertices in the routing network $G = (V, E)$. They gave a faster algorithm when the routing network satisfied certain stronger connectivity constraints.

They gave related algorithms, with the same time bounds, for multiterminal nets, but in an even more restrictive setting. Again all terminals are on the infinite face of a planar routing network, and all internal degrees are even. Further, $2d(X) < c(X)$, for all $X \subseteq V$, where X is a cut, $d(X)$ is the "density" of the cut, and $c(X)$ is the "capacity" of the cut. Specifically, $d(X)$ is the number of nets having one but not all of its terminals in X, and $c(X)$ is the number of edges in E from X to $V \setminus X$.

5.3.2 Branch-and-Bound

Yang and Wing [27] proposed a branch-and-bound algorithm for simultaneously routing all nets. This was a natural outgrowth of their similar algorithms for single Steiner trees. The algorithm makes several assumptions that are really only valid for PCBs but, in principle, the technique can be extended to VLSI circuits.

They begin by formulating an objective function for a set of trees spanning the nets:

$$\text{(the sum of the tree lengths)} + \alpha \times \text{(the number of intersections at an edge)}$$

$$+\beta \times \text{(the number of intersections at a vertex)}.$$

The rationale for this function is that (in PCBs) two trees can share an edge (i.e., be routed on the same track) if they are "insulated" at a cost of α. Similarly, at a cost of β a vertex can be "split" if two wires touch at that vertex.

The algorithm associates a 0-1 indicator variable with each edge for each net: the variable is a 1 if and only if that edge is in the Steiner tree for that net. The algorithm begins with a singleton tree for each net. The algorithm progresses by adding an edge to some tree (setting the indicator variable to 1). If a feasible solution is found (all nets are spanned) then the objective function is evaluated. As is typical in branch-and-bound algorithms, when the bounding constraints fail then the process backtracks (setting the indicators to 0). This is an exact algorithm for this model.

Such a procedure is exponential time, of course. They proposed, as in their earlier work, a much faster suboptimal scheme, that is still exponential time. Instead of adding a single edge of the routing network to some tree at each step, the suboptimal algorithm connects a terminal to its tree with an L-shaped wire, at each step. The optimal and suboptimal algorithms produced identical results for small problems.

5.3.3 An LP Relaxation Heuristic

Raghavan and Thompson [16] presented a LP-relaxation approach to a simplified problem. They are given the (reduced) routing network $G = (V, E)$ and a set of nets. The trees of the routing will not necessarily be edge-disjoint. The *width* of a set of trees is the maximum number of trees that route across a single edge. The sole objective is to minimize the width of the routing.

Their paper only addresses nets with exactly three terminals. The technique can be extended to larger nets. Unfortunately the number of LP indicator variables grows exponentially in the size of the nets (but *not* in the size of the network or the number of nets). Hence it is only practical for small nets.

For each three-terminal net the principal problem is to decide where the single Steiner point is to be placed. They begin by posing the entire problem as a multicommodity flow problem cast as a 0-1 integer program. They use an indicator variable s_{ij}, for each vertex j and net i, which indicates whether that vertex j is the Steiner vertex for net i. First they solve, in polynomial time, an LP relaxation of this program. Let W be the width of the relaxed solution. Note that each s_{ij} may not be integral, but they require $\sum_j s_{ij} = 1$, for each net i. These represent *fractional* Steiner points.

Their "rounding" heuristic fixes the Steiner point for one net at a time. For net i, they must decide which vertex j will have $s_{ij} = 1$. They evaluate a potential function, defined for each intermediate flow, for each choice of j. They choose the j that minimizes their potential function. They are able to prove upper bounds on the width of the solution, found by the rounding heuristic, in terms of W and $|E|$. For example, if $W > \ln |E|$ then they derive this bound:

$$W + (e - 1)\sqrt{W \ln |E|}$$

where e is the base of the natural logarithm.

5.3.4 Bottleneck Steiner Trees

Chiang, Sarrafzadeh and Wong [2] have suggested another solution to the global routing problem. They sequentially place a Steiner tree for each net. However, each edge in the (reduced) routing network has a weight which reflects its congestion, relative to its capacity. Rather than seeking a Steiner tree with least total weight, they use a tree which minimizes its most costly edge.

The *bottleneck Steiner tree* (or *Steiner min-max tree* [2]) is a tree which spans a net in which the weight of the heaviest edge is minimized. Surprisingly this problem can be easily solved [2,18]).

Theorem 5.1 *Construct a MST of the routing network and prune away every subtree containing no terminals. The resulting tree is a bottleneck Steiner tree for those terminals.*

Proof: For simplicity, the weights are assumed to be distinct, so the MST is unique. It is easy to show that when Kruskal's algorithm adds an edge of weight w, that if two vertices can be connected by path with no edge heavier than w then those vertices are already connected by a subtree. Hence Kruskal's algorithm will automatically link up all the terminals in the net into one subtree with the least bottleneck, as soon as possible. Any additional heavier edges added by the MST algorithm, after the terminals are in one subtree, will be pruned away. □

They actually proposed a hybrid global router that combined bottleneck Steiner trees and approximate RSMTs, found with a Prim-based heuristic.

5.3.5 Global Routing with a Distance Network Heuristic

Korte, Prömel and Steger [8] have proposed a global router based on the distance network heuristic of Wu, Widmayer and Wong [25], denoted WWW here (see Section 4.2.3 in Part II). The input is a weighted (reduced) routing network, where the weight reflects the capacity of a channel. They have priorities associated with each net, which reflects the relative importance of each net. For example, the length of a particular net may have an impact on the cycle time of the clock. (They also have a via penalty for multiple layers; see the next section.)

Recall that a typical strategy for global routing is to route each net, one at a time, with dynamically changing edge weights to reflect congestion accumulating in some channels. Often the nets are processed in an arbitrary order. It has been shown that "shortest MST first" order is effective, where one begins by finding the MST for each net separately. In this section, the heuristic orders the nets by the given priorities.

The novel feature of this heuristic is that it does not completely route each net. Instead it considers the nets, in order, and partially constructs a solution using only inexpensive edges. Only after all the nets have had a chance are the more expensive edges used.

Recall the WWW heuristic, for a single net, simultaneously grows shortest path trees from each terminal, coalescing trees when they touch each other. When two

trees are linked there is a corresponding path between the roots of these trees; call such a path a *discovered path*. They proposed a perturbation of the WWW heuristic: when the number of subtrees, after coalescing has decreased to three, then connect these three trees optimally. The latter procedure, 3EXACT, tries each possible location for the remaining Steiner point, and does a shortest path computation from each such point. This procedure can be implemented in $O(v^2 \log v)$ time. (A corresponding 4EXACT was tried with little measurable effect.)

The global routing heuristic has s stages. There are s given thresholds $\alpha_1, \alpha_2, \ldots, \alpha_s$, where α_s is large. During the ith stage the WWW heuristic is run on each net, in order, until the length of a discovered path exceeds α_i. Before proceeding to the next stage, nets are set aside when their respective computations have been reduced to 3 or fewer subtrees. The WWW heuristic, for each net, picks up in stage $i+1$ where it left off during stage i. After all s stages are finished, the nets are connected using 3EXACT. The total time is $O(nv^2 \log v)$.

There is a post-processing step (that could be applied by any heuristic). In each Steiner tree, each wire is removed, disconnecting it into two subtrees, which are reconnected by the shortest possible wire. Further, it is possible that channel capacities will be exceeded; they use an additional rip-up phase to move low priority nets.

Korte, Prömel and Steger provide some interesting performance data (which is unusual). They used a (CPU) chip with 19318 nets (ranging from 2 to 25 terminals, for a total of 69709 terminals). The chip was partitioned into subregions resulting in a global routing problem with 10196 nets (the remaining 9122 nets were each contained entirely within a single region of the chip and are only considered during the later detailed routing stage.) The reduced routing network was a 39×33 grid graph.

Only the global routing phase is considered. Since the largest net was small they were able to solve each net optimally, with dynamic programming (in a "few days of computing time"). The total length of all the optimal RSMTs, without regard to channel capacities, was 79437. The corresponding total length of all the MSTs was 81476. Next they considered the situation when overfilled channels could not be used. Using a straight WWW heuristic ($s = 1$) the total was 81519, but 16 nets could not be routed at all. Finally, they used their multistage version, with post-processing. They had three different cost functions, for determining edge weights, which produced routings with total lengths 79912, 79877, and 79509, respectively. The computation times varied from 1 to 3 hours.

5.4 Multiple Layers

While VLSI routing is done on several layers, the vast majority of the literature has been for planar routing networks. As discussed above, this has been possible by allowing pairs of nets to be routed that are not vertex-disjoint and/or edge-disjoint. This is especially true during global routing. During detailed routing the penalties for overlapping edges and vertices are substantially increased, so that the resulting routing can be realized with multiple layers, with minimal perturbations

later. (There are many subsequent phases in the circuit design process, notably compaction stages.)

However, as the number of layers increases these *ad hoc* techniques will need to be replaced. The planar routing network is naturally replaced by a routing networks analogous to 3-dimensional grid networks. It is tempting to speculate that the work on the multidimensional rectilinear Steiner tree problem would be relevant. However, as with the planar case, the 3-dimensional routing networks have both missing edges and edge weights, resulting in a situation more similar to the case of a Steiner problem in networks.

Two examples of the possible restrictions on a 3-dimensional routing network are given. An edge connecting vertices on different layers corresponds to a via. Vias have a cost associated both with their area (on both layers) and their effect on reliability. These costs are reflected in weights on those edges, which are dependent on manufacturing technologies. Another example is when the technology dictates that routing on one layer is always horizontal and always vertical on another layer (with bends corresponding to vias). Such a two-layer chip would have the routing network in Fig. 5.9, where all the degrees are 3 or less.

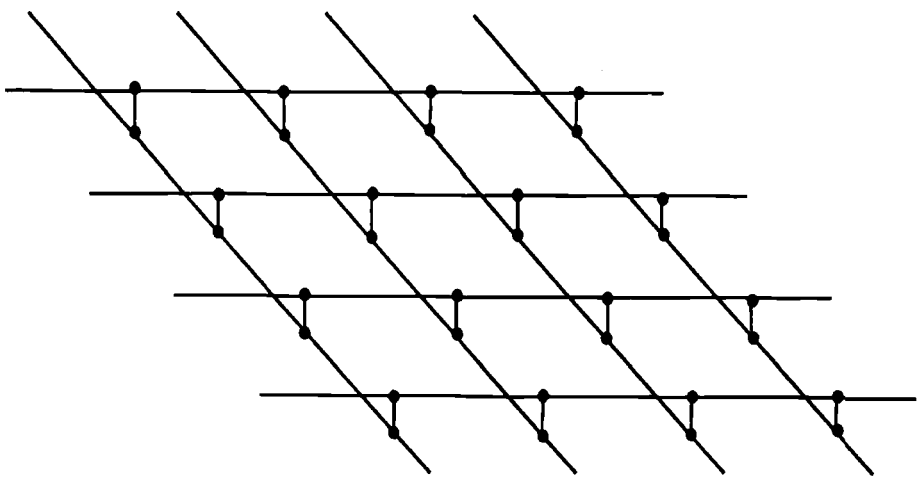


Figure 5.9: A 3-dimensional routing network

There are few Steiner heuristics for 3-dimensional routing networks. Hachtel and Morrison [4] present a technique that reduces the problem, by projection, to a "thinner" network. That thin network is solved, by dynamic programming, and the rest of the Steiner tree is formed by reversing the projections. Tsai, Chen and Feng [23] present a heuristic for multiple layers that assumes the input is a sparse routing network (that is the result of an earlier routing phase). Because of the sparseness, they propose a trivial linear time heuristic: do a breadth-first search and remove all subtrees without terminals.

References

[1] M. Becker and K. Mehlhorn, Algorithms for routing in planar graphs, *Acta Inf.* **23** (1986) 163–176.

[2] C. Chiang, M. Sarrafzadeh and C. K. Wong, A powerful global router: based on Steiner min-max trees, *IEEE Int. Conf. on Computer-Aided Design* (1989) 2–5.

[3] J. Cong, Pin assignment with global routing, *Proc. Int. Conf. on Computer-Aided Design* (1989) 302–305.

[4] G. D. Hachtel and C. R. Morrison, Linear complexity algorithms for hierachical routing, *IEEE Trans. Computer-Aided Design* **8** (1989) 64–79.

[5] S. Hambrusch and L. TeWinkel, Parallel heuristics for the Steiner tree problem in images without sorting or routing, Technical Report, Purdue Univ. (1989).

[6] Y. C. Hsu, Y. Pan and W. J. Kubitz, A path selection global router, *Proc. of the 24-th ACM/IEEE Design Automation Conf.* (1987) 641–644.

[7] T. C. Hu and M. T. Shing, A decomposition algorithm for circuit routing, In T. C. Hu and E. S. Kuh (eds.) *VLSI Layout: Theory and Design*, IEEE Press (1985) 144–152.

[8] B. Korte, H. J. Prömel and A. Steger, Steiner trees in VLSI-layout, in B. Korte, L. Lovasz, H.J. Prömel and A. Schrijver (eds.) *Paths, Flows, and VLSI-Layout* Springer-Verlag, Berlin (1989) 185-214.

[9] E. S. Kuh and M. Marek-Sadowski, Global routing, in T. Ohtsuki (ed.), *Layout Design and Verification*, North-Holland (1986) 169-198.

[10] C. Y. Lee, An algorithm for path connection and its applications, *IRE Trans. on Electronic Computers* **EC-10** (1961) 346–365.

[11] J. H. Lee, N. K. Bose and F. K. Hwang, Use of Steiner's problem in supoptimal routing in rectilinear metric, *IEEE Trans. Circuits Syst.* **CAS-23** (1976) 470–476.

[12] K.-W. Lee and C. Sechen, A new global router for row-based layout, *IEEE Int. Conf. on Computer-Aided Design* (1988) 180–183.

[13] T. Lengauer, *Combinatorial Algorithms for Integrated Circuit Layout*, John Wiley & Sons, Chichester, England (1990).

[14] W. K. Luk, P. Sipala, M. Tamminen, D. Tang, L. S. Woo and C. K. Wong, A hierarchical global wiring algorithm for custom chip design, *IEEE Trans. Computer-Aided Design* **CAD-6** (1987) 518–533.

[15] A. P.-C. Ng, P. Raghavan and C. D. Thompson, A language for describing rectilinear Steiner tree configurations, *Proc. of the 23-rd ACM/IEEE Design Automation Conf.* (1986) 659–662.

[16] P. Raghavan and C. D. Thompson, Mutiterminal global routing: a deterministic approximation sceme, *Algorithmica* **6** (1991) 73–82.

[17] G. Reich and P. Widmayer, Beyond Steiner's problem: a vlsi oriented generalization, in M. Nagl (ed.), *Graph-Theoretic Concepts in Computer Science* **LNCS 411** Springer-Verlag (1989) 196-210.

[18] M. Sarrafzadeh and C. K. Wong, Geometric Steiner min-max trees, Technical Report, IBM (1989).

[19] R. Sedgewick and J. S. Vitter, Shortest paths in Euclidean graphs, *Algorithmica* **1** (1986) 31–48.

[20] J. M. Smith and J. S. Liebman, Steiner trees, Steiner circuits and the interference problem in building design, *Eng. Opt.* **4** (1979) 15–36.

[21] H. Takahashi and A. Matsuyama, An approximate solution for the Steiner problem in graphs, *Math. Jap.* **24** (1980) 573–577.

[22] R. E. Tarjan, *Data Structures and Network Algorithms*, SIAM (1983).

[23] C.-C. Tsai, S.-J. Chen and W.-S. Feng, Generalized terminal connectivity problem for multilayer layout scheme, *Comput.-Aided Des.* **22** (1990) 423–433.

[24] T. Watanabe and Y. Sugiyama, A new routing algorithm and its hardware implementation, *Proc. of the 23-rd ACM/IEEE Design Automation Conf.* (1986) 574–580.

[25] Y. F. Wu, P. Widmayer and C. K. Wong, A faster approximation algorithm for the Steiner problem in graphs, *Acta Inf.* **23** (1986) 223–229.

[26] J. G. Xiong, Algorithms for global routing, *Proc. of the 23-rd ACM/IEEE Design Automation Conf.* (1986) 824–830.

[27] Y. Y. Yang and O. Wing, On a multinet wiring problem, *IEEE Trans. Circuit Theory* **CT-20** (1973) 250–252.

[28] X. Zhang, L. T. Pillage and R. A. Rohrer, Efficient final placement based on nets-as-points, *Proc. of the 26-th ACM/IEEE Design Automation Conf.* (1989) 578–581.

This Page Intentionally Left Blank

Part IV

Other Steiner Problems

This Page Intentionally Left Blank

Chapter 1

Steiner Trees in Other Metric Spaces

Although STs in Minkowski spaces were studied by Cockayne as early as in 1967, the bulk of literature has appeared only very recently. There are three general types of results. The first is to derive some basic properties of an ST, typically, the degree of a Steiner point and the number of sets of parallel edges in an ST. The second is to prove the finiteness of the problem, typically, by showing that there exists an SMT whose Steiner points are all vertices of a given graph. The third is to bound the Steiner ratio. What has been missing in the literature is the invention of efficient algorithms to construct a full SMT for a given full topology, like what the Melzak FST algorithm does for the Euclidean plane. Sankoff and Rousseau [10] gave a dynamic programming solution to this problem. But the approach is computationally feasible only in some special spaces like the rectilinear space (see Part III, Section 4.4) and the space of qualitative characters (see Subsection 2.4.1).

It is surprising that despite the lack of efficient algorithms or heuristics for local minimum trees, there exists an approach to construct a heuristic SMT which has a better performance ratio than an MST for any metric space. However, the efficiency of such a heuristic depends on the existence of an efficient algorithm to construct SMTs for small numbers of terminals, in some case as small as three. Therefore, finding these efficient algorithms for small numbers of terminals becomes a priority task to attack the Steiner tree problem for general metric spaces.

1.1 Minkowski Spaces

A normed space is also known as a Minkowski space. By the Minkowski theorem, given a convex, compact, and center-symmetric body in a linear space, one can define a norm in the space such that the unit ball of the normed space is identical

to the convex body, and vice versa. Cockayne [4] first studied the Steiner problem for the Minkowski space and obtained some results for the 3-terminal SMT.

Suppose the unit ball is a d-dimensional polytope with $2m$ extreme points. Such a space will be denoted by M_d. Since it is center-symmetric, there exist m diagonals each connecting a pair of extreme points and going through the center. The orientations of the diagonals are called *diagonal directions* (see Fig. 1.1). Du and Hwang [7] proved

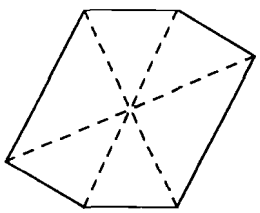


Figure 1.1: The unit disk and three diagonal directions of a Minkowski plane

Theorem 1.1 *For N in M_d there exists an SMT whose edges are all in diagonal directions.*

Proof: Follows directly from the fact that any polytope is the convex hull of its extreme points. □

Call a hyperplane *diagonal* if it is parallel to the span of $d-1$ diagonal directions. Let $GG_1(N)$ be the grid formed by the $\binom{m}{d-1}$ diagonal hyperplanes through each terminal in N. Define a *grid point* to be a point run through by d of these diagonal hyperplanes. The following analogue of Hanan's theorem was earlier established by Snyder for rectilinear d-space (see Part III, Section 4.4). But the proof of Du and Hwang for the general M_d case is much shorter.

Theorem 1.2 *Suppose that N is in M_d but not in any M_{d-1}, and the unit ball has $2d$ extreme points. Then there exists an SMT for N whose Steiner points are all grid points of $GG_1(N)$.*

Proof: Call a coordinate *listed* if it appears as a coordinate of a terminal. Then a grid point of $GG_1(N)$ has only listed coordinates. By Theorem 1.1 there exists an SMT T using only edges in diagonal directions. Let U be the set of unlisted coordinates occurring in Steiner points of T. Suppose that U is nonempty. It will be shown that the cardinality of U can always be reduced by one. This will complete the proof.

Let s be a Steiner point with an unlisted coordinate x_i in the i^{th} diagonal direction. Consider the diagonal hyperplane parallel to the span of the order $d-1$ diagonal directions at s. Clearly, there exists no terminal on this hyperplane or x_i would be listed. Since the d diagonal directions are linearly independent, this hyperplane can be moved along the i^{th} diagonal direction (either side) until it hits

a terminal. Note that firstly, such a move is permissible since no terminal has been moved. Secondly, the number of segments of the i^{th} diagonal direction touching the hyperplane on the two sides must be equal or the hyperplane can be moved towards one side to reduce the total length; a contradiction to the assumption that T is an SMT. Thus moving the hyperplane does not change the length of T. Finally, when the hyperplane hits a terminal, the new x_i is listed but no new unlisted coordinate has been created. □

Theorem 1.2 reduces the SMT problem to a finite problem and opens the door for using SMT algorithm on networks to attack the SMT problem on M_d.

Du and Hwang conjectured that $m = d$ is also a necessary condition for Theorem 1.2. In fact, they also conjectured something stronger. For $i = 2, 3, \ldots$ define $GG_i(N)$ as the grid formed by running the $\binom{m}{d-1}$ diagonal hyperplanes through each grid point of $GG_{i-1}(N)$, and the grid points of $GG_i(N)$ are the intersections of d distinct diagonal hyperplanes.

Conjecture. *If $m > d$, then for every fixed i there exists a set N in M_d such that every SMT(N) contains a Steiner points not in $GG_i(N)$.*

Du and Hwang gave partial support to the conjecture by showing that for every d there exists a space S_d and an SMT(N) as described. For $d = 2$ S_d is the hexagonal metric with 120° angles and the SMT(N)s are the splitting tree (see Part I, Section 5.4). A modification of these splitting trees into higher spaces provides examples for general d.

1.2 Minkowski Planes and l_p Metrics

In this section, only Minkowski spaces of dimension two are considered. An important special case is the l_p metric. In the l_p metric, the distance between two points $a_1(x_1, y_1)$ and $a_2(x_2, y_2)$, where (x, y) are the Cartesian coordinates, is

$$||a_1, a_2||_p = (||x_1 - x_2||^p + ||y_1 - y_2||^p)^{1/p}$$

Note that $p = 1$ yields the rectilinear metric, $p = 2$ the Euclidean metric and $p = \infty$ the maximum metric which is dual to the rectilinear metric in the sense that the two unit disks are 45° rotations of each other.

Consider a Minkowski plane with a unit disc D. The following theorem and a proof was suggested by Chakerian and Ghandehari [3], with details given in [1].

Theorem 1.3 *Suppose that D is differentiable. Then every Steiner point in an SMT has degree three. Furthermore, if a, b and c are three distinct points on the boundary of D, and Δ is the triangle induced by the three tangent lines to D at a, b and c, respectively, then $[o, a]$, $[o, b]$, $[o, c]$ form an SMT for $\{a, b, c\}$ if and only if the origin o is the centroid of Δ.*

Du, Graham and Liu [6] used Theorem 1.3 to prove

Theorem 1.4 *Suppose that D is differentiable and strictly convex. Then every full SMT consists of at most three sets of parallel edges.*

Proof: (sketch) Let T be an SMT and $[s, s']$ an edge of T between two Steiner points s_1 and s_2. Let a be a point on $[s_1, s_2]$ and close to s_1. It can be shown from Theorem 1.3 and strict convexity that there exists a unique pair of points b and c on the boundary of D, equidistant from s as a, such that $[s, a]$, $[s, b]$, $[s, c]$ is an SMT. Furthermore, this construction depends only on the slope of $[s, a]$. This implies that the slopes of the other two edges at s are uniquely determined by the slope of $[s, s']$. Similarly, the slopes of the other two edges at s' are also uniquely determined by the same functions of the slope of $[s, s']$. Hence the two edges at s are parallel to the two edges at s'. Iteratively using the same argument on other edges of T between two Steiner points, Theorem 1.4 is proved. □

The set of three directions of the edges of a full SMT will be called a *consistent triple* of D.
Since the l_p metric is differentiable and strictly convex for $1 < p < \infty$, it follows:

Corollary 1.1 *For N in l_p, $1 < p < \infty$, each Steiner point in an SMT(N) has degree three and every full SMT(N) consists of at most three set of parallel edges.*

Some of the proof techniques used by Du and Hwang in proving the Steiner ratio conjecture for the Euclidean plane are also valid for Minkowski planes. In particular, the following is true.

Lemma 1.1 *Suppose that D is differentiable and strictly convex and C is a consistent triple of D. Let $\mathcal{T}$ denote the set of full SMTs with edges in the directions of C and with at most n terminals. Then the minimum value of $||T(N)||/||MST(N)||$ over all $T \in \mathcal{T}$ is achieved by trees whose terminals are the vertices of a union U of at most $n-2$-congruent equilateral triangles (with edges in the directions of C) so that all terminals are on the boundary of U.*

Call the Minkowski plane *a hexagonal plane*, and denoted by H, if D is a hexagon. Note that a hexagonal plane has a unique consistent triple which is the set of diagonal directions. Du, Graham and Liu proved

Theorem 1.5 *$\rho(D) \geq 2/3$ holds for an arbitrary Minkowski plane D if and only if it holds for every hexagonal plane H and every vertex set N of a union U of at most $n-2$-congruent equilateral triangles such that all terminals of N are on the boundary of U, and an FST(N) exists with all edges in the diagonal directions of H.*

Proof: The "only if" part is trivial. To prove the "if" part, it suffices to consider FSTs only. Suppose D is differentiable and strictly convex. Let T be an FST. By Theorem 1.4, T consists of three sets of parallel edges. Therefore there exists a hexagonal plane H such that $||T||$ is invariant in D or in H. By Lemma 1.1 $||T(N)|| \geq (2/3)||\mathrm{MST}(N)||$ in H. Since $||\mathrm{MST}(N)||$ is not smaller in H than in D,

the above inequality also holds in D. □

Du, Graham and Liu also showed

$$\rho(M_2) \leq \frac{1+\sqrt{19}}{6} = 0.8931\ldots$$

by constructing a set N achieving that bound. They conjectured for an arbitrary Minkowski plane D

$$\frac{2}{3} \leq \rho(D) \leq \frac{\sqrt{3}}{2}$$

For the l_p metric Liu and Du [9] obtained better provable bounds. The following is an account of their results.

It is easily verified that $\|a,b\|_p$ is decreasing in p, but $[(a^p+b^p)/2]^{1/p}$ is increasing in p for $p \geq 1$. Therefore for $1 \leq p \leq q \leq \infty$

$$\|a,b\|_q \leq \|a,b\|_p \leq 2^{\frac{1}{p}+\frac{1}{q}}\|a,b\|_q$$

Using these inequalities it can be shown:

Lemma 1.2 *Suppose that* $1 \leq r \leq p \leq q \leq \infty$. *Then*

$$2^{\frac{1}{q}-\frac{1}{r}}\rho_q\rho_r \leq \rho_p^2 \leq 2^{\frac{1}{r}-\frac{1}{q}}\rho_q\rho_r$$

An immediate consequence is

Theorem 1.6 *For* $1 \leq p \leq \infty$, $\rho_p \geq \sqrt{\frac{1}{\sqrt{2}}\rho_1\rho_2}$.

Proof: If $1 \leq p \leq 2$, choose $r = 1$ and $q = 2$ in Lemma 1.2. If $2 \leq p \leq \infty$, choose $r = 2$ and $q = \infty$. □

Since it is known that $\rho_1 = 2/3$ and $\rho_2 = \sqrt{3}/2$, it follows

Corollary 1.2 *For* $1 \leq p \leq \infty$, $\rho_p \geq \sqrt{\frac{1}{\sqrt{6}}}$.

Finally, by some fine analysis, Liu and Du were able to give an upper bound of ρ_p.

Theorem 1.7 $\rho_p \leq \rho_2 = \sqrt{3}/2$ *for* $1 \leq p \leq \infty$.

Du [5] also conjectured that $\rho_p > \rho_1 = 2/3$ for $1 < p < \infty$.

1.3 λ-Geometry and Hexagonal Plane

Theorem 1.5 suggests the importance of studying hexagonal plane. A λ-geometry is a more general notion which includes the hexagonal plane as a special case, and was first introduced by Sarrafzadeh and Wong [11]. In a λ-geometry only edges

with angles $i\pi/\lambda$ for all i are allowed, where $\lambda \geq 2$ is an integer either dividing π or divisible by π (see Fig. 1.2).

If the unit disc of a Minkowski plane is a regular 2λ-gon with the x-axis being a diagonal direction, then by Theorem 1.1 there exists an SMT in λ-geometry. Therefore the 2-geometry corresponds to the rectilinear plane, the 3-geometry to the hexagonal plane where the axes are 120° apart, and the ∞-geometry to the Euclidean plane.

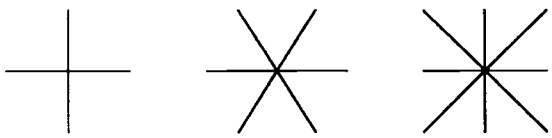


Figure 1.2: λ-geometry

Widmayer, Wu and Wong [12] proved that

Lemma 1.3 *There exists a shortest path between two points in the λ-geometry, $\lambda < \infty$, which uses only two line segments.*

In fact, they showed that if l and l' are the two lines making minimum angles with line pp', where p and p' are the two given points, and l intersects l' at p'', then pp'' and $p'p''$ are the two segments. An edge consisting of two segments is called a *nonstraight edge.*

For a fixed N let MST_λ and SMT_λ denote the corresponding trees in λ-geometry and define $\rho(\lambda)$ to be the Steiner ratio in the λ-geometry. Sarrafzadeh and Wong proved

Theorem 1.8 $\rho(\lambda) \geq \frac{\sqrt{3}}{2} \cos \frac{\pi}{2\lambda}$.

Proof:

$$||\mathrm{SMT}_\lambda|| \geq ||\mathrm{SMT}_\infty|| \geq \frac{\sqrt{3}}{2}||\mathrm{MST}_\infty|| \geq \frac{\sqrt{3}}{2} \cos \frac{\pi}{2\lambda}||\mathrm{MST}_\lambda||$$

where the second inequality is from the Euclidean Steiner ratio theorem and the third inequality an immediate consequence of Lemma 1.3. □

Corollary 1.3 $\rho(3) = 3/4$.

Proof: From Theorem 1.8, $\rho(3) \geq 3/4$. Let N be the set of three corners of an equilateral triangle. It is easily verified that $||\mathrm{SMT}(N)|| = 3$ and $||\mathrm{MST}(N)|| = 4$. □

Du, Graham and Liu studied the Steiner ratio for an arbitrary hexagonal plane H. Let u, v and w be the three unit vectors in the diagonal directions satisfying

$$au + bv + cw = 0$$

Since H is characterized by a, b, c, it will be denoted by $H(a, b, c)$. The Steiner ratio of $H(a, b, c)$ will be denoted by $\rho(a, b, c)$.

Lemma 1.4 *$H(a, b, c)$ is convex if and only if a, b and c satisfy the triangle inequality.*

Proof: Note that

$$\begin{aligned} u &= \frac{b}{a}v + \frac{c}{a}w \\ v &= \frac{c}{b}w + \frac{a}{b}u \\ w &= \frac{a}{c}u + \frac{b}{c}v \end{aligned}$$

Thus $H(a, b, c)$ is convex if and only if $b/a+c/a \geq 1$, $c/b+a/b \geq 1$ and $a/c+b/c \geq 1$. □

Theorem 1.9 *If $H(a, b, c)$ is convex then $\rho(a, b, c) \leq 3/4$.*

Proof: Let ox, oy and oz be the three unit vectors and let $N = \{x, y, z\}$. Then $||\text{SMT}(N)|| = ||ox|| + ||oy|| + ||oz|| = 3$ while $||\text{MST}(N)|| = ||x, y|| + ||y, z|| = 4$. □

Du, Graham and Liu also conjectured

Conjecture. *If $H(a, b, c)$ is convex, then $\rho(a, b, c) \geq 2/3$.*

Note that if this conjecture is true, then Theorem 1.5 implies that $\rho(D) \geq 2/3$ for an arbitrary Minkowski plane D.

Recall that Hanan (Part III, Theorem 1.2) proved that for the 2-geometry there exists an SMT(N) such that every Steiner point is a grid point of $GG_1(N)$. Fig. 1.3 illustrates a set N in the 3-geometry such that one of its SMT contains a Steiner point s not in $GG_1(N)$ (see Fig. 1.3).

However, the above s is still in $GG_2(N)$. In fact, Du and Hwang [7] proved

Theorem 1.10 *For the 3-geometry there always exists an SMT(N) such that all of its Steiner points are grid points of $GG_{n-2}(N)$.*

The proof depends crucially on the following lemma.

Lemma 1.5 *For the 3-geometry there always exists an SMT(N) with at most one nonstraight edge in each full component.*

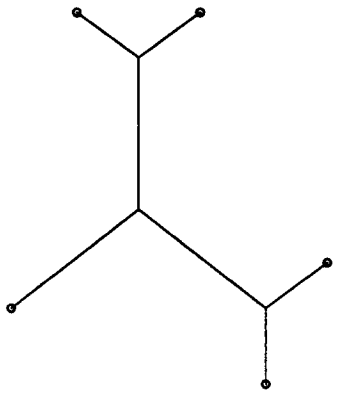


Figure 1.3: s not in $GG_1(N)$

Proof of Theorem 1.10: Theorem 1.10 is trivially true for $n = 3$. The general n case is proved by induction. It suffices to prove for a full SMT.

Note that a Steiner point adjacent to two terminals by straight edges is a grid point of $GG_1(N)$. In general, a Steiner point adjacent to two grid points of $GG_i(N)$ is a grid point of $GG_{i+1}(N)$. By Lemma 1.4 there exists a full SMT T with at most one nonstraight edge. If T contains a nonstraight edge, then every Steiner point s must have two branches connected by straight edges each containing at most $n-1$ terminals (counting s as a terminal). By the induction hypothesis, s is adjacent to two grid points of $GG_{n-3}(N)$, hence s is a grid point of $GG_{n-2}(N)$. (If T contains no nonstraight edge, then it can be verified that every Steiner point is in $GG_{\lfloor (n-1)/2 \rfloor}(N)$.) □

Corollary 1.4 *Suppose that N is in an arbitrary hexagonal plane. Then there always exists an SMT(N) whose Steiner point are all grid points of $GG_{n-2}(N)$.*

Proof: It follows by using a linear transformation from the 3-geometry to the hexagonal plane. □

Unfortunately, Lemma 1.5 and Theorem 1.10 cannot be extended to the λ-geometry for $\lambda \geq 4$.

Theorem 1.11 *For any λ-geometry with $\lambda \geq 4$, there exists a set N such that every SMT(N) contains a Steiner point not in $GG_i(N)$ for any i.*

Proof: An example of N for the 4-geometry is illustrated in Fig. 1.4. Similar constructions can be given for other λ-geometry with $\lambda > 4$. □

Theorem 1.10 shows that the 3-geometry ST problem is finite and any algorithm for the ST problem in networks applies to 3-geometry.

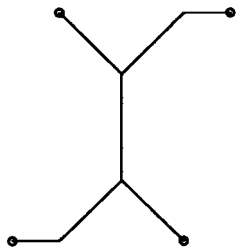


Figure 1.4: An SMT with two nonstraight edges

1.4 Better Heuristics for Arbitrary Metric Spaces

Over the years numerous heuristics for SMTs have been proposed for terminals in various metric spaces and networks. Often their superiority over MSTs were claimed and demonstrated by simulations. But no mathematical proof of superiority was ever given. One exception is that a proof of superiority in average performance by Bern (see Part III, Section 2.2.4) in the rectilinear plane. But still there was no proof that any heuristic has a better performance ratio than the Steiner ratio.

In 1990, Zelikovsky [13] finally had a breakthrough to this problem by giving a better heuristic for the network problem. Unfortunately, his results were published in a not easily accessible conference proceedings, and his writing was very obscure. Berman and Ramaiyer [2] came to the rescue by restating Zelikovsky's results in a much more comprehensible way, giving some improvements for the network problem and also giving a better heuristic for the rectilinear space. Du, Zhang and Feng [8] made further improvements and gave a better heuristic for an arbitrary metric space, including the Euclidean plane as a special case. In the next two sections we will mostly report the general results of Du, Zhang and Feng, and refer to the more specific heuristics of Zelikovsky and Berman and Ramaiyer' only when they are different from the general case.

Recall that a k-size quasi-ST is a tree with a Steiner topology and each full component contains at most k terminals. No polynomial algorithm is known for the construction of a minimum k-size quasi-ST Q_k for $k \geq 3$. However, a heuristic tree Z_k for Q_k exists such that $||Z_k||/||Q_k||$ is bounded.

As mentioned in Part I, Section 6.3, the concept of MST can be extended to the case where some of the terminals are point-sets. For a given set N of n terminals and subsets $N_i \subset N$, $|N_i| > 1$, $i = 1, \ldots, s$, an MST $T(N; N_1, \ldots, N_s)$ is a set of edges interconnecting the s point-sets and $N - |\cup_i N_i|$, where a point-set is considered connected if any of its points is connected. The ordinary MST $T(N)$ can be viewed as the special case $s = 0$. To assure that the connected graph is a tree, it is required that no set and no terminal is contained in another set, $|N_i \cap N_j| \leq 1$ and that there does not exist a cyclic sequence of subscripts such that $|N_i \cap N_j| = 1$ for all i and j consecutive in that cyclic sequence. Such a set

$\{N_i\}$ is called a *cycle-free set.*

Let S_i denote a full SMT for N_i, if one exists. When each S_i, $i = 1, \ldots, s$, exists, define $W(N; N_1, \ldots, N_s) = \|T(N)\| - \|T(N; N_1, \ldots, N_s)\| - \sum_{i=1}^{s} \|S_i\|$. Then W is the saving of length over an MST by using some SMTs for subsets of N.

The k-size quasi-ST Z_k is constructed through the following greedy algorithm:

- **Step 1.** For each $i = 2, \ldots, k$, construct an i-terminal SMT for each $\binom{n}{i}$ subsets of i terminals. Store the SMT in the list L if it is an FST. Set $j = 0$.
- **Step 2.** Set $j = j + 1$. Select S_j from L such that $\{S_1, \ldots, S_j\}$ is cycle-free and $W(N; N_1, \ldots, N_j)$ maximizes $W(N; N_1, \ldots, N_{j-1}, N_j^*)$ over all $S_j^* \in L$. Delete S_j^* from L if $\{N_1, \ldots, N_{j-1}, N_j^*\}$ is not cycle-free.
- **Step 3.** Stop when L is empty. Otherwise go to Step 2.

Since each edge is in L, the output Z_k is always a tree interconnecting N. However, the efficiency of the algorithm depends on the existence of an efficient construction of a k-terminal SMT in M_d. Given that there exists such a construction which requires $f(k)$ time, then step 1 takes $O(kn^k f(k))$ time to generate the $O(kn^k)$ SMTs and step 2 takes $O(n \log n)$ time to generate the MST $T(N; N_1, \ldots, N_{j-1}, N_j^*)$ for each S_j^* in L. Since there are $O(kn^k)$ such S_j^* and step 2 is repeated $O(n)$ time, the Z_k algorithm requires $O(kn^k(n^2 \log n + f(k)))$ time.

For $\{N_1, \ldots, N_s\}$ a cyclic-free set, define

$$\begin{aligned} W(N; N_{t+1}, \ldots, N_s | N_1, \ldots, N_t) &= W(N; N_1, \ldots, N_s) - W(N; N_1, \ldots, N_t) \\ &= \|T(N; N_1, \ldots, N_t)\| - \|T(N; N_1, \ldots, N_s)\| \end{aligned}$$

Theorem 1.12 $\|Z_k(N)\| \leq \frac{k-2}{k-1}\|Q_2(N)\| + \frac{1}{k-1}\|Q_k(N)\|$ *for* $k \geq 3$.

Proof: Let $Q_k = \cup_{i=1}^{t} S_i^*$ where each S_i^* is a full SMT, and let $\{S_1, \ldots, S_s\}$ be the full SMTs chosen by the algorithm. Subtracting from $\|Q_2(N)\| = \|Z_2(N)\|$ both sides of the inequality in Theorem 1.12, then

$$W(N; N_1, \ldots, N_s) \geq \frac{1}{k-1} W(N; N_1^*, \ldots, N_t^*)$$

This inequality is proved by induction on s. If $s = 0$, then each S_i^* must be an edge for otherwise $W(N; N_1) \geq W(N, N_i^*) > 0$ and $s \geq 1$. Therefore $W(N; N_1, \ldots, N_s) = 0 = W(N; N_1^*, \ldots, N_t^*)$ and the inequality is trivially true. So, assume $s \geq 1$. Suppose that S_1 contains x terminals. Then $S_1 \cup Q_k$ contains $x - 1$ cycles since S_1 contributes $x - 2$ new vertices and $2x - 3$ new edges. Let $S_1^*, \ldots, S_y^*$, $1 \leq y \leq x - 1$, be the FSTs whose deletions from Q_k break all the $x - 1$ cycles. Then from the choice of S_1 in the algorithm,

$$(k-1)W(N; N_1) \geq (x-1)W(N; N_1) \geq \sum_{i=1}^{y} W(N; N_i^*) \geq W(N; N_1^*, \ldots, N_y^*)$$

Since

$$W(N; N_1, \ldots, N_s) = W(N; N_1) + W(N; N_2, \ldots, N_s | N_1)$$

and

$$W(N; N_1^*, \ldots, N_t^*) = W(N; N_1^*, \ldots, N_y^*) + W(N; N_{y+1}^*, \ldots, N_t^* | N_1^*, \ldots, N_y^*)$$

it suffices to prove

$$(k-1)W(N; N_2, \ldots, N_s | N_1) \geq W(N; N_{y+1}^*, \ldots, N_t^* | N_1^*, \ldots, N_y^*)$$

Let $T(N; N_2', \ldots, N_r' | N_1)$ denote a minimum k-size quasi-ST given that N_1 is chosen. Then

$$(k-1)W(N; N_2, \ldots, N_s | N_1) \geq W(N; N_1', \ldots, N_r' | N_1) \geq W(N; N_{y+1}^*, \ldots, N_t^* | N_1)$$

where the first inequality is due to the inductive assumption, and the second due to the minimality of $T(N; N_1', \ldots, N_r' | N_1)$.

Let N_{1j}, $j = 1, \ldots, J$, denote the pairs of adjacent vertices in S_1. Then

$$\begin{aligned} W(N; N_{y+1}^*, \ldots, N_t^* | N_1) &= W(N; N_{y+1}^*, \ldots, N_t^* | N_{11}, \ldots, N_{1J}) \\ &\geq W(N; N_{y+1}^*, \ldots, N_t^* | N_1^*, \ldots, N_y^*) \end{aligned}$$

since each N_{1j} is contained in some N_i^*, $i = 1, \ldots, y$. Therefore

$$(k-1)W(N; N_2, \ldots, N_s | N_1) \geq W(N; N_{y+1}^*, \ldots, N_t^* | N_1^*, \ldots, N_y^*) \qquad \square$$

For a tree T interconnecting N in M_d, define

$$\rho(T) = \inf_{N \in M_d} \frac{\|\mathrm{SMT}(N)\|}{\|T(N)\|}$$

Corollary 1.5 $\rho^{-1}(Z_k) \leq \frac{k-2}{k-1}\rho^{-1}(Q_2) + \frac{1}{k-1}\rho^{-1}(Q_k)$ *for* $k \geq 3$.

Therefore, if $\rho(Q_2) > \rho(Q_k)$ for some $k \geq 3$, then Z_k is a better heuristic. Proving this inequality will be the goal of the next section.

Berman and Ramaiyer [2] claimed the following stronger result.

Theorem 1.13 $\rho^{-1}(Z_k) \leq \rho^{-1}(Q_2) - \sum_{i=3}^{k} \frac{\rho^{-1}(Q_{i-1}) - \rho^{-1}(Q_i)}{i-1}$.

1.5 Bounds for Performance Ratios of Quasi-STs

Consider a full SMT T. Turn T into a rooted binary tree T_r by choosing an arbitrary edge and setting its middle point r as the root for T_r. The root is considered at the first level of T_r, and a vertex is at the i^{th} level if the path from the root to the vertex contains i vertices (including both the root and the vertex).

Lemma 1.6 *For any rooted binary tree T, there exists a one-to-one mapping f from internal nodes to leaves such that for u an internal node*

(i) *$f(u)$ is a descendant of u,*

(ii) *all paths from u to $f(u)$ for all u are edge disjoint.*

Proof. Define $f(u)$ as the leftmost leaf of the right subtree of u. Lemma 1.6 is easily verified. □

Theorem 1.14 $\rho(Q_k) \geq \frac{K}{K+1}$ *where* $K = \lfloor \log_2 k \rfloor$.

Proof: The theorem is trivially true for $n \leq k$. The general $n > k$ case is proved by induction on n. It suffices to prove the theorem for a full SMT T. Let T_r be the rooted binary tree obtained from T with the one-to-one mapping f. The length of an edge is its length in M_d.

Let I_l be the set of internal nodes at the l^{th} level. Define $U_j = \cup_{i \equiv j (\text{mod} K)} I_i$. Then U_j for $d = 1, \ldots, K$ are disjoint. Let $p(u)$ denote the path from an internal node u to $f(u)$. Define

$$L_j = \sum_{u \in U_j} ||p(u)||$$

By Lemma 1.1 (ii),

$$||T_r|| \geq \sum_{j=1}^{K} L_j$$

Therefore there exists a j such that $L_j \leq ||T_r||/K$. Consider U_j for such a j. Decomposing T_r at nodes of U_j yields a collection of rooted binary trees each rooted at a node of U_j and having leaves either in U_j or in the leaf set of T_r. Denote such a tree by T_x if it is rooted at x. Clearly, each T_x has at most k leaves.

For each leaf u of T_x which is an internal node of T_r, connect u to $f(u)$ with an edge. Thus T_x is transformed to a quasi-ST $Q(x)$ on at most k terminals. Clearly, the union of $Q(r)$ and $Q(x)$, $x \in U_j$ is connected, hence it is a k-size quasi-ST. Thus

$$Q_k \leq ||Q(r)|| + \sum_{x \in U_j} ||Q(x)|| \leq ||T_r|| + L_j \leq \frac{K+1}{K} ||T|| \qquad \square$$

The case $k = 4$ and 8 were first claimed by Berman and Ramaiyer [2]. Since $\rho(Q_k) \to 1$ as $k \to \infty$,

Corollary 1.6 *For every M_d such that $||SMT(N)|| \neq ||MST(N)||$ for all $N \in M_d$, there exists a k large enough such that Z_k is a better heuristic.*

For $k = 3$ Zelikovsky gave a better result.

Theorem 1.15 $\rho(Q_3) > \frac{3}{5}$.

Proof: Let T_r be defined as in Theorem 1.14 and let T_u be a subtree of T_r rooted at u. Let I_u denote the set of terminal nodes of T_u. For each $u \in I_r$ let $g(u)$ denote a leaf closest to u, and let $p(u)$ denote the path from u to $g(u)$. Then $g(u)$ can be chosen to equal $g(v)$ where v is a child node of u. Suppose that v and w are the two children nodes of u. Let $a(u)$ be the unique simple path connecting $g(v)$ and $g(u)$ in T_r. Define $A_u = \cup_{i \in I_u} a(i)$. Then A_u is a tree interconnecting I_u. It can be easily verified that $\|A_r\| \leq 2\|T_r\| - 2\|p(r)\|$. Define $\Delta(u) = \|A_u\| - \|T_u\|$. Then

$$\Delta(u) = \begin{cases} 0 & \text{if } u \text{ is a leaf} \\ \Delta(v) + p(v) + \Delta(w) + p(w) & \text{otherwise} \end{cases}$$

It is easily solved that $\Delta(u) = \sum_{i \in I_u - \{u\}} p(i)$.

Let $e = [u, v]$ where v is a child node of u, and v has two children nodes x and y. Let N_e denote the three terminals $g(w)$, $g(x)$ and $g(y)$, and let $S(e)$ be an SMT on N_e. Since $p(u) \cap p(v) = p(v)$,

$$\|S_e\| \leq \|a(u)\| + \|a(v)\| - \|p(v)\|$$

Let $A(T|N_{e_1}, \ldots, N_{e_t})$ denote a tree obtained from A_r by substituting $S(e_i)$ for $a(u_i)$ and $a(v_i)$, $i = 1, \ldots, t$. Then

$$\|A_r(N_e)\| \leq \|A_r\| - \|p(v)\|$$

Furthermore, if $\{N_{e_1}, \ldots, N_{e_t}\}$ is a cycle-free set, then

$$\|A_r(N_{e_1}, \ldots, N_{e_t})\| \leq \|A_r\| - \sum_{i=1}^{t} \|p(v_i)\|$$

It can be verified that $\{N_{e_1}, \ldots, N_{e_t}\}$ is a cycle-free set if and only if $e_i \cap e_j = \emptyset$ for all $1 \leq i < j \leq t$. Since the edge set E of a rooted binary tree can be easily partitioned into three disjoint subsets where edges in each subset do not intersect, one subset, say, E_1 must satisfy

$$\sum_{e_i \in E_1} \|p(v_i)\| \geq \frac{1}{3} \sum_{e_i \in E} p(v_i)$$

Thus

$$\begin{aligned} \|A_r(\{N_{e_i} : e_i \in E_1\})\| &\leq \|A_r\| - \frac{1}{3} \sum_{e_i \in E} p(v_i) \\ &\leq \|A_r\| - \frac{1}{3}\Delta(r) \\ &\leq \|A_r\| - \frac{1}{3}\{\|A_r\| - \|T_r\|\} \\ &= \frac{2}{3}\|A_r\| + \frac{1}{3}\|T_r\| \\ &\leq \frac{5}{3}\|T_r\| - \frac{4}{3}\|p(r)\| \end{aligned}$$

It follows immediately,

$$\rho(Q_3) \geq \frac{\|T_r\|}{\|A_r(\{N_{e_i} : e_i \in E_1\})} > \frac{3}{5}$$

□

References

[1] M. Alfaro, M. Conger, K. Hodges, R. Kochar, L. Kuklinski, Z. Mahmood and K. von Haam, The structure of singularities of ϕ-minimizing networks in R^2, *Pac. J. Math.* **149** (1991) 201–210.

[2] P. Berman and V. Ramaiyer, An approximation algorithm for the Steiner tree problem, extended abstract (1991).

[3] G. D. Chakerian and M. A. Ghandehari, The Fermat problem in Minkowski space, *Geom. Dedicata* **17** (1985) 227–238.

[4] E. J. Cockayne, On the Steiner problem, *Canad. Math. Bull.* **10** (1967) 431–450.

[5] D. Z. Du, On Steiner ratio conjecture, *Ann. Oper. Res.* **33** (1991) 437-449.

[6] D. Z. Du, R. L. Graham and Z. C. Liu, Minimum Steiner trees in normed planes, Preprint.

[7] D. Z. Du and F. K. Hwang, Reducing the Steiner problem in a normed space, *SIAM J. Comput.*, to appear.

[8] D. Z. Du, Y. J. Zhang and Q. Feng, On better heuristic for Euclidean Steiner minimum trees, *Proc. of the 32-nd Ann. Symp. on Foundations of Computer Science* (1991) 431-439.

[9] Z. C. Liu and D. Z. Du, On Steiner minimal trees with L_p distance, *Algorithmica* **7** (1992) 179-191.

[10] D. Sankoff and P. Rousseau, Locating the vertices of a Steiner tree in an arbitrary metric space, *Math. Program.* **9** (1975) 240–246.

[11] M. Sarrafzadeh and C. K. Wong, Hierarchical Steiner tree construction in uniform orientations, Preprint (1991).

[12] P. Widmayer, Y. F. Wu and C. K. Wong, On some distance problems in fixed orientations, *SIAM J. Comput.* **16** (1987) 728–746.

[13] A. Z. Zelikovsky, The 11/6-approximation algorithm for the Steiner problem on networks, *Inf. Comput.*, to appear.

Chapter 2

Phylogenetic Trees

This chapter discusses a Steiner tree problem in biology. As with the chapter on routing, these results are often described as rectilinear Steiner problems, while, at the same time, introducing problem dependent constraints that render much of the literature in Part III inapplicable.

For more than 100 years biologists interested in "systematics" have attempted to infer the evolutionary trees that have present-day species at the leaves. Only in the last 30 years have the mathematical and algorithmic aspects of tree construction been investigated. There are several reasons for constructing these trees. Often only the topology is of interest; this answers questions of basic classification. Sometimes one wants to know when divergence occurred and how long the edges of the tree are; this question is sometimes answered by explicitly deriving a description of the inferred (extinct) species.

These questions can be regarded as Steiner tree problems. The input is a set of species, or more generally *taxa*, together with information about each taxon and/or relationships between the taxa. The output is a tree that best fits (according to some criterion) this information. The taxa are at the leaves of the tree and correspond to the notion of terminals (except that no taxon can be an internal node). The internal nodes are inferred ancestral taxa and correspond to Steiner points; there is no standard name for these nodes so they are simply referred to as *internal nodes*. In some cases, when there is a notion of a coordinate space, the location of the internal nodes can be specified. Often only the lengths of the edges of the tree are desired.

This chapter begins by reviewing the various approaches to codifying this problem, followed by discussions of various algorithms for these approaches. It is best to separate what is to be computed from the algorithm that performs the computation, as it easy to confuse the two. Unfortunately, there are several types of phylogenetic trees that can only be described as "the tree this algorithm builds". Computational complexity results are also discussed. In later sections algorithms are presented for the various formulations. Arguably the formulations in the last two sections are not properly Steiner problems, but are included to demonstrate the range of techniques that have been attempted.

2.1 Definitions

It is impossible to give a definitive problem statement. There is no agreement on how to define a phylogenetic tree mathematically. There are several camps and there are definitions with outright conflicting goals. Very often the belief in the quality or reliability of the input determines what kind of tree a researcher is willing to accept. Some of the terminology is not neutral for the various camps — but these issues will be ignored since biological considerations are beyond the scope of this book. For further discussion of the biological motivations for the methods below, see the excellent surveys of Felsenstein [21,22] and Li and Graur [34].

A phylogenetic tree is an evolutionary tree for a given set N of taxa (sometimes called "operational taxonomic units" or OTUs). Such a tree is also called a "phylogeny", "dendrogram", or "cladogram". It is assumed to be a "bifurcating" tree, which means that it is a binary tree. This follows from the assumption that evolution is driven by bifurcating events. (In practice trees are allowed to be multifurcating when the bifurcations are sufficiently close together that known techniques cannot distinguish the correct order of bifurcations.)

2.1.1 Rooted and Unrooted Trees

While a phylogenetic tree is inherently rooted, due to the historical nature of evolution, it is common to define unrooted phylogenetic trees, which specify the branching topology but do not suggest where in the tree the root is found (due to a lack of evidence). It is also convenient to work with unrooted trees when using an approach that exhausts all topologies, since there are fewer of them.

Since most methods produce unrooted trees how does one find a rooted version? Consider the unrooted tree in Fig. 2.1. Recall that only bifurcating trees are used, so every unrooted tree has every internal node with degree 3. To root such a tree a root must be positioned in the interior of some edge of the unrooted tree. Hence the three possible rooted trees shown.

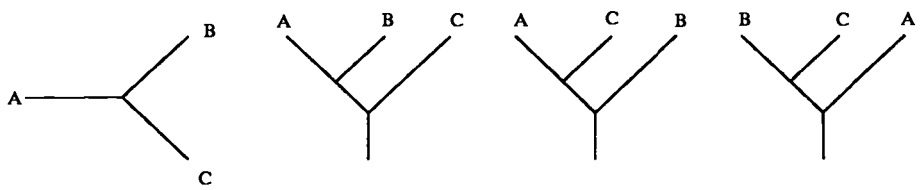


Figure 2.1: Unrooted and rooted phylogenetic trees

There are three popular ways to position the root. First, it can be put on the longest edge. Second, it can be put in the middle of the longest path between two taxa. This is used when evolution is assumed to proceed at a fixed "clock" rate. Third, an "outlier" can be added to the taxa, where the outlier is determined *a priori* on biological grounds. For example, if the taxa are all mammals then a bird

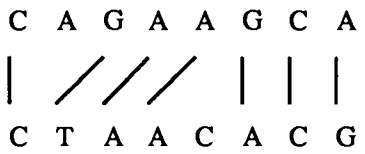
could be added, and the root could be placed on the edge connecting the bird to the subtree containing all mammals (such an edge is presumed to exist in a tree produced by a reasonable algorithm).

2.1.2 Distances

It is fundamental to any Steiner problem, including phylogenetic trees, that there is a notion of distance. When the points are in some metric space the distance are well-defined. However it is not clear whether taxa can be effectively identified with points in some metric space, and if so, which space should be used. Some such spaces are described below. However, there are some other approaches, notably "distance methods", that finesse the notion of an embedding.

Each taxon can be described in terms of a sequence of m values, called *characters*. These characters were originally morphological (e.g., femur length) but today much of the research uses molecular characters. Molecular data, which has become increasingly available since the mid-1960s, comes as either DNA sequences (composed of nucleotides) or protein sequences (composed of amino acids).

Central to the notion of distance is the concept of an *alignment* of sequences. Given two sequences of characters from some "alphabet" (say, nucleotides) an alignment is a partial mapping from characters in one sequence to another, that preserves the left-to-right ordering. As seen in Fig. 2.2, such an alignment can be represented by a diagram with aligned characters above each other, and unaligned characters placed opposite "gaps". The gaps correspond to the insertion or deletion of characters (depending on which sequence is considered primary). Also different characters have been aligned; these are "substitutions" or "mutations". For each pair of characters i and j a *substitution cost* c_{ij} can be defined. By convention, c_{ij} is defined even if i or j is a gap, and c_{ii} is defined to be small or zero. There are quadratic time algorithms to find the *minimum cost alignment*, where the cost of an alignment is the sum of the substitution costs (e.g., see [43]).



C A G A A - G C A

C - T A A C A C G

Figure 2.2: Two representations of a sequence alignment

Evolution, as reflected at the molecular level, proceeds by a series of insertions, deletions, and substitutions (as well as a few other rarer mechanisms which are ignored here). The *alignment distance* between sequences is the cost of the minimum cost alignment. Suppose the same protein is found in taxa A, B, and C. If the alignment distance from A to B is much less than that from A to C, that is taken as strong evidence that A and B have a closer ancestor than A and C do. While this method is a good method for generating a distance between pairs of taxa, it has serious drawbacks. If taxon A is aligned with B, and A is aligned with C,

the two alignments are not necessarily consistent with a minimum cost alignment between B and C. Further it is even more difficult to find sequences with which to label the internal nodes, that are consistent with these alignments. (There is an expensive method, discussed below, that addresses these problems.)

Instead, what is commonly done is to find a *multiple alignment* of all the taxa. This is a difficult and poorly solved problem. However, often all that is ultimately used is a small portion of the alignment that is "highly conserved", i.e., largely unchanged during evolution. In particular, there might be m amino acids (m may be as small as 10) from each taxon that are aligned — usually there are very few, if any, gaps allowed. So essentially the original long sequences are forgotten and each taxon is described in terms of m characters; such characters are often called *sites*, reflecting that they are associated with a site on an original sequence. It is implicit that the same highly conserved sites are also present in the internal nodes.

The notion of alignment distance extends naturally to the set of characters resulting from a multiple alignment. (Since the alignment is already done, there are no further insertion or deletions.) The *distance* between two taxa, given their set of sites, is the sum of the m substitution costs. Unless otherwise stated, this is the distance measure used below. The situation here is simpler than above, of course, because the alignment is fixed *a priori*. Often the *Hamming distance* is used: the number of characters which differ. This corresponds to the substitution cost assignment $c_{ij} = 1$ if $i \neq j$, and $c_{ii} = 0$.

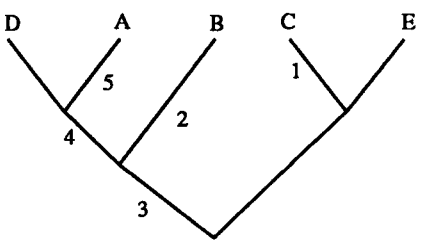
It some cases the situation is simplified even further by assuming each character is *binary*, 0 or 1. If the data is not binary to begin with it can usually be easily encoded as a greater number of binary characters.

2.2 Compatibility Methods

It is important to distinguish between a *method*, a mathematical approach to defining an optimal tree, and the *algorithm* that computes (or approximates) the optimal tree for that method. Sometimes it is impossible to distinguish these terms when no formal criterion is presented.

2.2.1 Hennig Trees

Hennig [33], as early as 1950, proposed a method that took as input n taxa, each with m binary characters. Further, he assumed that if a character was 0 it was known to be ancestral and that a taxon with a 1 character evolved from another ancestral taxon with the same character being 0. The desired tree, a *Hennig tree*, is explicitly rooted, and the root node is assumed to be labeled $(0, 0, 0, \ldots, 0)$, i.e., all m characters are ancestral. The ith character, for each i, is associated with an edge of the tree and a taxon has the ith character being 1 if and only if it is in the subtree reached by that edge. For example, consider the set of 5 characters (ignoring character 6) and the corresponding tree in Fig. 2.3. However if character 6 is also given as input then it is impossible to build the required Hennig tree.



	1	2	3	4	5	6
A	0	0	1	1	1	0
B	0	1	1	0	0	0
C	1	0	0	0	0	1
D	0	0	1	1	0	1
E	0	0	0	0	0	0

Figure 2.3: A character matrix and a Hennig tree for the first five characters

There is a simple characterization of inputs for which Hennig trees exist. Let $N^{(i)}$ be the set of taxa with the ith character being 1. The following result [14] is known as the "Pairwise Compatibility Theorem".

Theorem 2.1 *A Hennig tree exists if and only if for each i and j either $N^{(i)}$ and $N^{(j)}$ are disjoint or one contains the other.*

Gusfield [30] showed that this test can be applied in $O(nm)$ time. He begins by permuting the entries in the $n \times m$ binary matrix of characters. He regards each column as a binary integer and sorts the columns in decreasing order; this can be done in $O(nm)$ time using a radix sort. It is easy at this point to see if the theorem holds. If a Hennig tree exists the tree itself can be found in the same time bounds [30]. To do this sort the matrix a second time, but sort by rows this time. It can be shown that now the matrix has the *consecutive ones property*: all the 1's in each column are in consecutive rows (where row n is adjacent to row 1). It is easy to construct the tree from a matrix in this form. Gusfield showed a $\Omega(nm)$ lower bound on determining if a Hennig tree exists.

Two Hennig trees T_1 and T_2 are *compatible trees* if they are both refinements of another tree T_3. Call T_1 a *refinement* of T_3 if T_1 can be obtained by a series of edge contractions from T_3. Gusfield [30] gave a linear time algorithm to determine if two trees are compatible trees.

2.2.2 Free Hennig Trees

Suppose the binary character input is given but it is unknown whether 0 or 1 represents the ancestral value for each character. This is the *free Hennig tree* problem. The root corresponds to one of the 2^m possible sets of characters that could be the m ancestral values. By trying each one, and relabeling the characters throughout so that the root seems to be $(0, 0, \ldots, 0)$, this case can be reduced to the previous situation. McMorris [35] showed that only one of these 2^m possibilities needs to be tried; if a free Hennig tree exists then a regular Hennig tree exists relative to that choice. In particular, that choice has the ith character of that root being 1 if and only if the majority of the taxa has a 1 for that character.

2.2.3 Maximum Compatibility

What happens when the input does not correspond to a Hennig tree? One approach, known as "maximum parsimony", is discussed below. Le Quesne [38] proposed a more direct approach. Select the largest subset of characters which are *compatible*, in the sense that they satisfy the Pairwise Compatibility Theorem, and return the Hennig tree corresponding to those selected characters. (Of course, this method can be biased by the way the original set of characters was sampled; see [21] for variations.)

Unfortunately, Day and Sankoff [9] have proven that this maximum compatibility method is NP-hard, as well as related variations. (In this chapter each NP-hard optimization problem has a corresponding NP-complete decision problem formulation; see Chapter 1 in Part I.) There are no known heuristics that are not straightforward attacks on the underlying maximum clique problem. In practice, systematists will use any large clique as advisory but not decisive; the characters that are not selected are completely ignored by this method, which may be inappropriate. Often for actual biological data finding the maximum clique has not proven intractable.

2.3 Maximum Parsimony Methods

It is inherent in the compatibility methods that each character is "uniquely derived", inasmuch as a tree is found where there is a single transition from an ancestral to a derived state for each character. *Maximum parsimony methods* relax this assumption by finding a tree which minimizes the number of transitions; these are also known as "minimum evolution methods". In biology there are examples of "reversion" (a return to a ancestral state) and "convergent evolution" (parallel evolution of the same derived state), which are rare but cannot be ignored. The adoption of this method depends on the nature of the characters. For example, reversion of molecular characters is much more common than morphological characters.

This method was originally proposed for continuous character data by Edwards and Cavalli-Sforza [13]. However discrete character domains will be discussed first. In the next subsections such methods are presented. The discussion of the maximum parsimony approach is continued in the following sections as well.

2.3.1 Camin-Sokal Method

Camin and Sokal [2] kept the assumption that characters are binary, ancestral characters are correctly labeled 0, and no reversions (from 1 to 0) are allowed . However they do allow convergent evolution, so that a transition from 0 to 1 can occur several times, for any character. For example, the data in Fig. 2.3 gives the tree in Fig. 2.4, where the transition for character 6 occurs twice. Note that this method produces a rooted tree.

Day, Johnson and Sankoff [8] have shown that the optimal Camin-Sokal tree

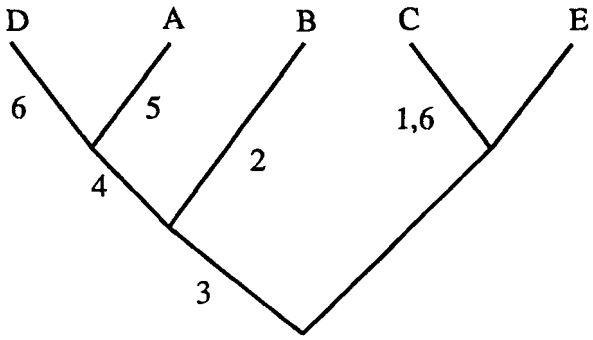


Figure 2.4: A Camin-Sokal tree

problem is NP-hard. For a given tree, with taxa assigned to the leaves, it is simple to calculate the minimum number of transitions (from 0 to 1) needed by this method. For each character, simply use one transition for each maximal subtree of taxa having that character being 1. However there is no guarantee that the given tree is a good topology.

This method can be recast as a minimum Steiner arborescence problem in m dimensions, as discussed in Chapter 4 of Part III. The root is identified with the origin and each edge is directed away from the origin. Each taxon is a corner of the unit m-dimensional hypercube.

2.3.2 Chromosome Inversion Method

This method avoids the need for independent 0 to 1 transitions [17]. It assumes that 0 is ancestral but the transition is not to the 1 state, but to a "polymorphic" (or "heterozygous") 01 state. There is only one transition to the 01 state, for each character, but there can be several later transitions from 01 to 1 and from 01 to 0. The method seeks to minimize the total number of transitions.

Note that the problem can be stated as a Steiner problem in directed networks. A digraph can be derived from the m-dimensional grid graph induced by the point set $\{0, 0.5, 1\}^m$, where the polymorphic state is associated with the value 0.5. The origin, identified with the root, is included with original set of taxa. There are directed edges corresponding to the permitted single transitions: 0 to 0.5, 0.5 to 1, and 0.5 to 0.

Day and Sankoff [10] have shown that this Steiner problem is NP-hard. It is easy to determine the minimum number of transistions for a given tree and can be handled similarly to the Camin-Sokal method.

2.3.3 Dollo Method

This method [19] also assumes a rooted tree is found for binary characters with known ancestral states. However it assumes no convergent evolution but allows

reversions. In particular, for each character, there is only one 0 to 1 transition, but as many 1 to 0 transitions as needed. It is also NP-hard [8] and can be handled similarly to the Camin-Sokal method. It can be cast as a Steiner problem for directed networks, analogously to the previous method. The point set for the graph is $\{0,1\}^m$ and the allowed transitions dictate which directed edges to include.

Other maximum parsimony methods are discussed in the next two sections.

2.4 Wagner Parsimony Method

This is the most studied maximum parsimony method, so an entire section is devoted to it. It is analogous to the rectilinear Steiner problem. In this method there is no assumption about character states being ancestral and there are no restrictions on reversions or convergent evolution. Essentially the method can be cast as the following task: labeling the edges of a tree with characters, such that the taxa at the endpoints of an edge differ exactly in those characters that label that edge (this includes the inferred taxa at the internal nodes). The basic *Wagner tree* problem asks for the tree, with the given taxa at the leaves, that needs the fewest edge labels. It is important to note the tree found by this method is inherently unrooted, since there is no preferred direction across any edge. For example, for the problem in Fig. 2.3 one gets the tree in Fig. 2.5 (which just happens to be the same as the tree in Fig. 2.4 without the putative root).

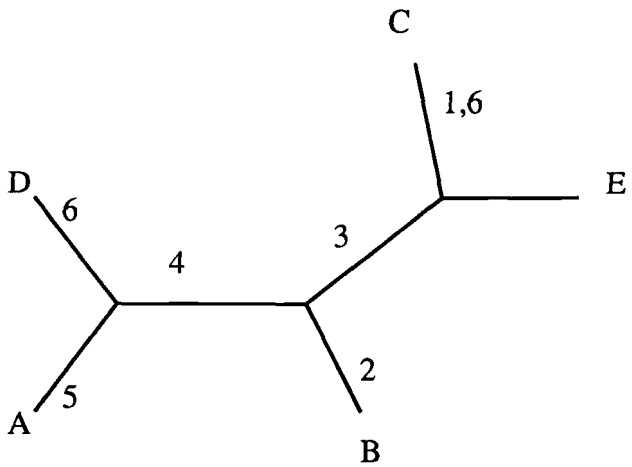


Figure 2.5: A Wagner tree

So far the emphasis has been on binary characters. If binary characters are assumed then for Wagner trees the state of each character for an internal (Steiner) node can easily be inferred if the edge-labeled tree is given. Do a breadth-first search, from any leaf, toggling each character labeling an edge that is crossed. Clearly the Wagner method with binary characters can be stated as an RSMT

problem for terminals on the m-dimensional hypercube, a problem discussed in Chapter 4 of Part III. Recall Graham and Foulds [27,29] have shown that this Steiner problem is NP-hard.

In this section the concentration is on general discrete characters where the ith character takes on a value from the set C_i. For example, when the ith character is a nucleotide site then $C_i = \{A, T, G, C\}$, where these correspond to the four nucleic acids that can be present at a site. In this situation, unlike the binary case, it is not trivial to infer the character states of the internal nodes, since while it is known that the state changes across an edge it is not known what the new state is, without further calculation.

This general Wagner problem can be mapped to a Steiner problem in m-dimensional space, if some coordinate value is associated with each state in each C_i. Formally, each taxa is represented by a point in $C_1 \times C_2 \times \cdots \times C_m$. Each internal node must also be such a point (by definition, so there is no need to establish an analogue of Hanan's theorem).

The Wagner problem can now be described as a Steiner problem. In rare cases (such as those in the previous section) it is meaningful to invoke a true rectilinear metric. However, in practice the distances are more general, so it is natural to pose it as a network Steiner problem on the multidimensional rectilinear graph induced by $C_1 \times C_2 \times \cdots \times C_m$. Note that this rectilinear graph $G = (V, E)$ is not properly a "grid graph" since the transition from one state is not necessarily to an "adjacent" state (hence the vertices along a "row" are connected by a clique, rather than a path). In section 1 the various distance measures were discussed that can be used between two sets of m characters $(a_1, a_2, \ldots, a_m)$ and $(b_1, b_2, \ldots, b_m)$. Here $\sum_{i=1}^{m} d(a_i, b_i)$ is used, where $d(a_i, b_i)$ is the "distance" between the states a_i and b_i, of character i. The length of any edge in E from $(a_1, \ldots, a_{i-1}, a_i, a_{i+1}, \ldots, a_m)$ to $(a_1, \ldots, a_{i-1}, b_i, a_{i+1}, \ldots, a_m)$ has length $d(a_i, b_i)$. Formally, the *Wagner method* asks for a given set of taxa $N \subseteq V = C_1 \times C_2 \times \cdots \times C_m$ to find the Steiner minimal tree in G.

Eck and Dayhoff [12,11] were the first to discuss the discrete Wagner method. The characters used in their study were amino acid sites from protein sequence data. Protein sequences are molecular data like DNA sequences, except that each C_i contains 20 amino acids, instead of just 4 nucleic acids. Some later Wagner methods used morphological characters, but much of the research has used molecular data. Since the character's contributions to the cost of the tree are independent, each character can be treated independently.

2.4.1 Using a Given Tree

Suppose that one is given a specific tree T with its leaves labeled with the given taxa. The problem is to label the internal nodes so that sums of the lengths of the edges is minimized (i.e., it is an optimal Wagner tree relative to T).

This problem was essentially solved in Section 4.4 in Part III, when d-dimensional RSMTs were discussed. The same bottom-up dynamic programming algorithm can be used. There are two reasons the situation is simpler here: the degree of each node is bounded by the constant 3, and the number of possible values at each in-

ternal node, for each character, at each internal node is also bounded by a constant $c = \max_i\{|C_i|\}$. The runtime for this approach is $O(n)$, since each node can be processed in constant time. For molecular data the constants are small.

Many variations of this basic approach have been proposed. Algorithms have been given by Fitch [25,24], Hartigan [31], Moore, Barnabas and Goodman [36], Sankoff and Rousseau [44], and others.

Since there are an exponential number of possible topologies for T, the fast algorithm for a fixed T is of limited utility. Exhaustive analysis is very limited, so in general T is selected by a heuristic or with biological expertise.

2.4.2 Informative Characters

As noted in the first section, the set of characters used in this analysis was created during a preprocessing step that selected those sites that were part of a multiple alignment. In fact it is possible that many of these aligned characters are not informative and are useless. A character for a given set of taxa is *informative* if that character actually can be used to distinguish between two possible phylogenetic trees.

Consider the five characters for the four taxa given in Fig. 2.6, where the characters are nucleotides. Note that there are only three possible trees for the four taxa, and recall that each character can be considered independently. The first character is not informative since every tree has no transitions. The second character has every tree requiring one transition. The third always require two transitions, and the fourth always requires three. The fifth character favors one tree (which only requires one transition) over the other two (which require two transitions) as shown, and is the only informative character in this example.

This can be generalized to show that a character is informative if and only if it has at least two taxa with one value and two other taxa with another value. Unfortunately molecular characters are often chosen because they are highly conserved (relative to some alignment). Hence, there are situations where most characters are not informative. Li and Graur [34] cite a study comparing human, chimpanzee, gorilla, and orangutan sequences, where there were only 27 informative sites found, while the original DNA sequences were 10,200 nucleotides long (before alignment).

2.4.3 Choosing a Tree

For 4 taxa only 3 trees need to be tried. For 8 taxa there are 10,395 trees and for 10 taxa the number grows to 2,027,025. Sometimes the optimal tree is not desired (or it may not be entirely trusted if computed) but instead a handful of candidate trees, suggested by other biological considerations, are all that need to be considered. However, in general, for large n, a heuristic is needed for choosing a tree. Usually these heuristics proceed by modifying the current tree in a greedy fashion. The initial tree can be random or constructed by a distance method, discussed later.

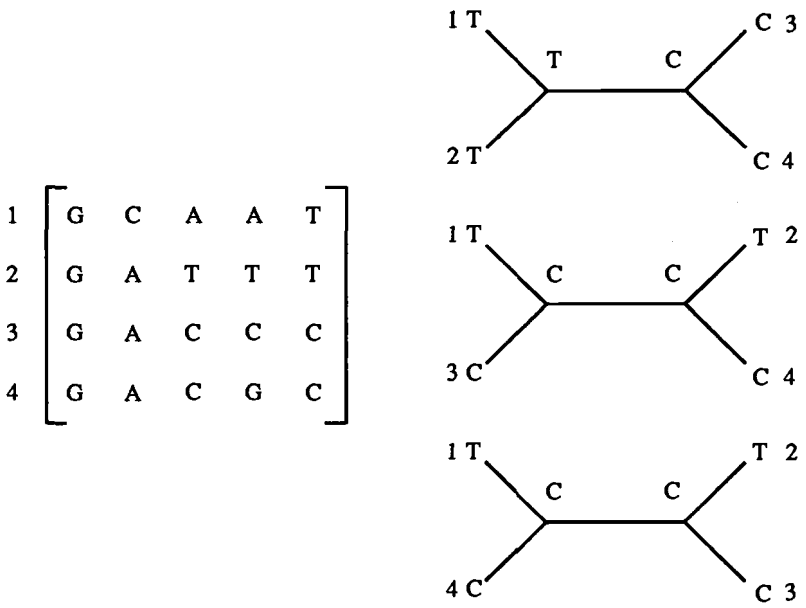


Figure 2.6: A character matrix and trees showing that the fifth character is informative

Cutting and Grafting Dayhoff [11] suggested a cutting and grafting approach. Consider the tree in Fig. 2.7. Pick a single edge and cut the tree into subtrees A and B. ¿From A construct the trees A' (formed by condensing B to a vertex a) and A'' (formed by removing the edge to B and the vertex in A incident on that edge); B' and B'' are analogous. A *graft* of B' onto A'' is formed by placing the vertex b, from B', in the middle of an edge of A''; two examples are shown. The algorithm tries all possible grafts of B' into A'' and of A' into B'' (ignoring the original tree, there are only the two possibilities shown in the figure, for this example). Further the algorithm repeats this process for every possible cut into an A and B. All $O(n^2)$ possible candidate trees are scored by the Wagner criterion and the tree of shortest length is selected for the next iteration of the algorithm.

Local Changes Another technique for generating a list of candidate trees during each iteration has been proposed that only makes local changes (see Cedergren, Sankoff, LaRue and Grosjean [4]). Pick an edge in the current tree that is not incident to a taxon, as shown in Fig. 2.8. Then two new tree topologies can be generated, by interchanging subtrees, as shown. Both new subtrees are generated for every such edge. All of these trees are scored and the shortest is used in the next iteration.

There is another way to describe this algorithm. Replace each connected subtree with four leaves with every other possible subtree on the same leaves (recalling

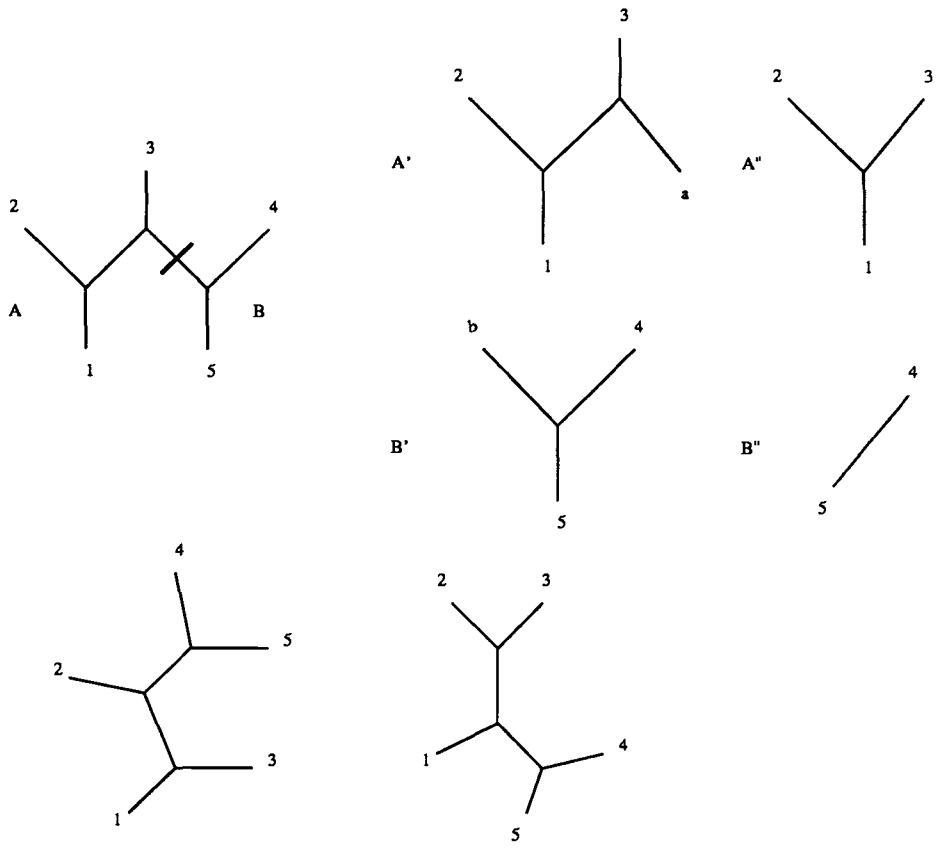


Figure 2.7: A tree that has been cut and below the two new trees that can formed by grafting

that all internal degrees are 3). Czelusniak, Goodman, Moncrief and Kehoe [5] have suggested extending this technique by considering all connected subtrees of 8 leaves. The search is obviously more extensive and perhaps converges faster. The number of cases to consider is still $O(n)$ but the constant of proportionality is very large. Extending beyond 8 leaves probably could not be tolerated.

MST-Based Several researchers have suggested building the tree using agglomerative clustering techniques (e.g., [12,18,24,26]). These techniques are related to the Prim and Kruskal techniques for building up a MST. Just one is described, due to Foulds, Hendy and Penny [28].

They use a Steinerization operation (called "coalescing") that takes a tree and adds a new vertex between three vertices (two connected to a common vertex) such that the total length of the tree is reduced. The algorithm is always greedy, doing the Steinerization with the most improvement first.

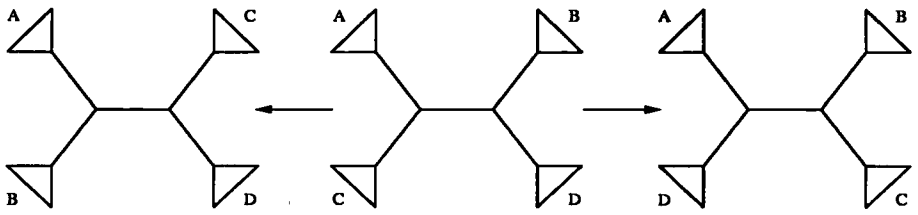


Figure 2.8: Changing a topology by rearranging subtrees around the middle edge

Their algorithm is Kruskal based. It begins with n singleton trees corresponding to the taxa. Each iteration begins by connecting two vertices (either the original taxa or created by Steinerizations) with the shortest such connection between two trees. Afterwards Steinerizations are performed until there is no improvement. Then the process iterates, until a single tree spans all the taxa. (Actually, when there are ties they allow all the equal-length edges to be added, perhaps creating cycles that are broken later with biological reasoning.)

Branch-and-Bound Two branch-and-bound approaches have been presented. Hendy and Penny [32] proposed a technique where taxa are added one at a time. They solved a real example with $n = 11$ in a few minutes. Penny and Hendy [37] used an approach where the characters are added one at a time. They solved an example with $n = 15$ and $m = 33$ in 5 minutes on a personal computer. They conjecture the algorithm performs reasonably well on real biological data, though inputs leading to exponential time exist.

2.5 Other Maximum Parsimony Methods

Two other approaches are presented that fall under the rubric of maximum parsimony methods. The first unifies the alignment process with the evaluation of total tree length. The second explicitly introduces a Euclidean metric.

2.5.1 Sankoff Method

When using molecular data the Wagner method relies on a preprocessing step to align the raw sequences. However many of the sites that cannot be aligned amongst all the taxa are not used as characters. Sankoff has suggested using the basic notion of the Wagner method, maximum parsimony, but to include the alignment in an essential way. A clear advantage of this approach is that more of the raw data is used in the computation. (Another advantage, beyond the scope of this book, is that the resulting multiple alignment is constructed relative to a tree — other non-tree methods create poorly justified alignments.) The disadvantage is that the algorithm is slower. To score a given tree takes $O(nm^n)$ time, where the n input sequences are of length m. This is exponential in n but not in m.

Several versions of the algorithm, for a given tree, have been presented, culminating in the paper by Sankoff and Cedergren [41]. The algorithm is not discussed here since it would involve a digression into the theory of alignments. Given the technique for a fixed topology, various topologies can be explored, as with the Wagner method.

Since there is a practical bound on n that the above algorithm can handle, Sankoff, Cedergren and LaPalme [42] proposed a heuristic for large fixed topologies. Recall, the problem is easily solved for $n = 3$, three taxa surrounding one internal node. They begin with some labeling of the internal nodes of the large tree. Then they go to each internal node and temporarily regard its three neighbors as fixed taxa (even if they are just internal nodes), and then use the algorithm to rescore that central internal vertex. (This could be regarded as a Steinerization operation.) This is done successively to all the internal nodes until no appreciable reduction in the total tree length occurs.

2.5.2 Continuous Methods

The Wagner method has been described as a maximum parsimony method for discrete characters. However, the original work in this area, by Cavalli-Sforza and Edwards [13,3] used continuous characters. For example, in population genetics, one compares the "gene frequencies" of various species, which are real numbers. They chose to compute the distance between taxa as the Euclidean distance, regarding their continuous characters as points in m-dimensional real space.

They recognized the relationship to the Euclidean Steiner problem but proposed no new approaches. Thompson [48] gave a Euclidean approximation algorithm based on repeated Steinerizations. Rogers [39] presents other Euclidean methods that are familiar from Part I.

2.6 Maximum Likelihood Method

This method, proposed by Felsenstein [20], tries to apply a more sophisticated statistical criterion than simple parsimony. The method explicitly associates a length of time with each edge. (He begins by assuming a fixed rate of change along all edges, but the results can be interpreted with varying the rates of change.)

There is a preprocessing step, before reading the input. It is assumed that each character is a nucleotide, though it could be easily generalized to other discrete character domains. He calculates $P_{ab}(t)$, which is the probability that if one begins with a character being a that it will be b after t time units. A reversible Markov process is used.

Suppose a tree is given with all the taxa associated with the leaves. With this method each character is assumed to be independent, so the algorithm will deal with only one character at a time. Suppose further that each internal node has a character (tentatively) associated with it, and that each edge (u, v) has a time span t_{uv} associated with it. Let $\mathbf{a} = (a_1, \ldots, a_{n-2})$, where a_i is the value of the character at internal node i; leaf j ($j > n - 2$ by convention) is labeled with the

taxon's character a_j. The likelihood of such a tree T is

$$P_T(\mathbf{a}) = P_{a_1} \prod_{(u,v)\in T} P_{a_u a_v}(t_{uv})$$

where P_{a_1} is the prior probability of the root being a_1, and each edge (u, v) is directed towards a leaf. However, when no character information about the internal nodes is given then the *likelihood* is $\sum_{\mathbf{a}} P_T(\mathbf{a})$. By reordering the summations this quantity can be calculated, bottom-up, in linear time.

Felsenstein proves that the problem can be regarded as essentially unrooted (using the fact that the underlying Markov process is reversible). He gives a heuristic for choosing the topology of T and for choosing each t_{uv}. He builds T in a Prim-based fashion, grafting one taxon at a time onto a growing tree. A taxon is grafted by adding an edge from it to the middle of an existing edge. The edge that is selected is the edge that gives the best likelihood, over all edges in the (partial) tree. The t_{uv} are selected greedily by optimally setting each relative to those that are already set; details in [20].

While each (fully specified) tree can be evaluated in linear time, each of the above stages involves many such evaluations. This is the slowest heuristic in this chapter. It is felt by some to be the procedure that gives the best results.

The methods so far — compatibility, maximum parsimony, and maximum likelihood — are based on character states and can be cast as Steiner problems. The next section will briefly look at alternative methods that are only indirectly related to Steiner problems.

2.7 Distance Methods

Distance methods begin with an $n \times n$ matrix $D = [d_{ij}]$, where d_{ij} is a measure of the "distance" from taxon i to taxon j; D is called a *dissimilarity matrix*. The goal is to create a tree spanning the taxa that is consistent with the input. Typically the length of each tree edge is also calculated. However, there is not necessarily any notion of the "position" of each taxon (in terms of characters or sites) so it makes no sense to ask for the position of each internal node. In this way this approach is unlike a Steiner problem.

An advantage of this approach is that distances are pairwise and, hence, do not depend on multiple alignments (which are hard to compute and result in a lot of raw data being ignored). Typically for molecular data the distances are alignment distances, but are affected by the lengths of the molecular sequences not just their similarity. It is a basic assumption for the distance methods that evolution goes at a constant rate. There are several ways to correct for this assumption [21].

Ideally the tree T, with known edge lengths, found will satisfy the *additivity condition*: $P_T(i, j) = d_{ij}$, where $P_T(i, j)$ is the length of the path connecting i and j in T. Such a tree is called an *additive tree*. In practice a tree is desired in which the additivity condition holds approximately. Often a relaxed requirement is used in place of the additivity condition: $P_T(i, j) \geq d_{ij}$. Day [7] showed that

determining the minimum total length tree satisfying even this relaxed condition is NP-hard.

Another way to judge how nearly a tree T fits the input D is to use a goodness-of-fit measure,

$$F_\alpha(D,T) = \sum_{1 \le i < j \le n} |P_T(i,j) - d_{ij}|^\alpha$$

where α is either 1 or 2. This has been suggested be several researchers. Day [6] has shown that finding a tree T that minimizes $F_\alpha(D,T)$, for either α, is NP-hard.

2.7.1 Pair Group Method

This is a clustering method. It differs from the MST-based methods for the Wagner method in that the position of each new point is not considered. An early algorithm, given by Sokal and Michener [46], is known as UPGMA, the unweighted pair group method with arithmetic mean.

The UPGMA is a Kruskal based method, that forms larger and larger clusters of taxa. It begins with n singleton clusters. Consider two (disjoint) clusters of taxa $A = \{a_i, a_2, \ldots, a_p\}$ and $B = \{b_i, b_2, \ldots, b_q\}$. The distance between these clusters is defined as

$$d(A,B) = \frac{1}{pq} \sum_{a \in A, b \in B} d_{ab}.$$

The algorithm, on each iteration, merges the two closest clusters and connects the "roots" of the two clusters. The root of a singleton cluster is the vertex itself, otherwise it is a vertex in the middle of the (most-recent) edge used to form the cluster.

2.7.2 Distance Wagner Methods

Farris [16] has dubbed his approach the "distance Wagner procedure", even though it has little to do with the maximum parsimony approach. It is a Prim-based approach. He begins by connecting the closest pair. Thereafter he picks some taxon c not in the current tree (where c minimizes some function on such nodes). He grafts c onto the current tree by connecting it into the middle of some edge, from a to b, where the edge was chosen to minimize $d_{ac} + d_{bc} - d_{ab}$. A new node d connected to a, b, and c is added, replacing the edge from a to b. The distances from d to the other nodes, including the taxa not yet in the tree are computed, and the algorithm iterates with a new c.

The method by which the distances are recomputed was improved by Tateno, Nei and Tajima [47]. Faith [15] surveys these techniques and proposes another method to recompute the distances.

2.7.3 Four-Point Condition Methods

Suppose an additive tree for 4 taxa is given, as shown in Fig. 2.9. Note that $d_{12} + d_{34} \le d_{13} + d_{24} = d_{14} + d_{23}$; where the inequality is strict unless the central

edge is zero length. Two taxa are *neighbors* in a topology if they are both incident to the same vertex. If, for a given dissimilarity matrix, there were an additive tree then the above sums can be computed and it is easily determine that 1 and 2 (and 3 and 4) are neighbors. In that case the tree is completely determined.

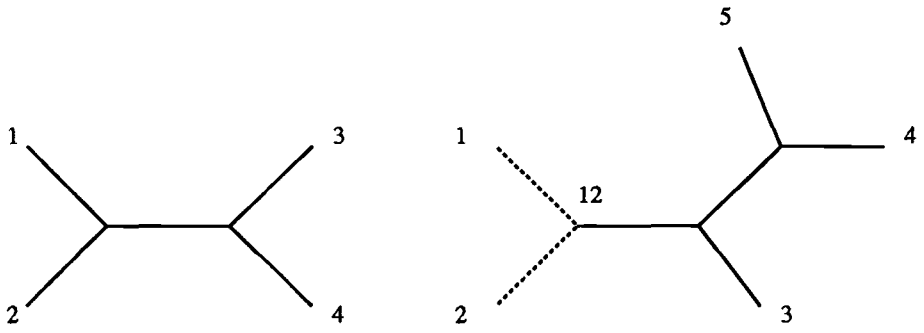


Figure 2.9: A tree with four taxa and tree formed after coalescing neighboring taxa

However, it is unlikely that raw data will correspond to an additive tree, since the distances are always approximations. However one can expect, if the data is well-behaved, to find, say, $d_{12} + d_{34} \leq d_{13} + d_{24} \leq d_{14} + d_{23}$. This is considered to be strong evidence that 1 and 2 (and 3 and 4) are neighbors. In general, if

$$d_{ij} + d_{kl} \leq \max\{d_{ik} + d_{jl}, d_{il} + d_{jk}\}$$

for all taxa i, j, k,and l, then D specifies a *tree metric.* This inequality is know as the *four-point condition.* Bandelt [1] surveys the literature on tree metrics (and related "ultrametrics") and gives fast algorithms for recognizing such D.

Sattath and Tversky [45] have presented an algorithm, based on the above observations, for $n \geq 4$, assuming they have a tree metric. They try every subset of four taxa $\{a_1, a_2, a_3, a_4\}$ and, for each subset, they determine i, j, k,and l, such that $d_{a_i a_j} + d_{a_k a_l}$ corresponds to the left-hand side of the four-point condition. Then the pairs of taxa (a_i, a_j) and (a_k, a_l) are each awarded 1 point. After exhausting all sets of 4 taxa, pick the pair (a, b) that received the most points. This pair is now forced to be neighbors in the final tree. The algorithm proceeds by coalescing a and b into a pseudotaxon "ab" and a new dissimilarity matrix is computed (using the UPGMA technique). The algorithm them iterates until n is reduced to 4. See Fig. 2.9.

Fitch [23] proposed a related algorithm based on the four-point condition. It emphasized the calculation of the length of the internal edges

2.7.4 Neighbor Joining Method

Saitou and Nei [40] presented an algorithm related to those in the previous subsection, in that it builds the tree by successively joining neighbors.

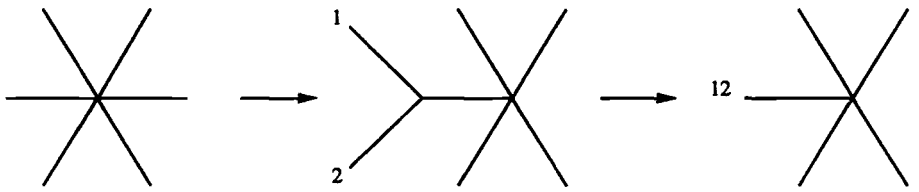


Figure 2.10: A symmetric intermediate tree and a tree formed forcing neighbors and tree formed by coalescing the neighbors

To explain their method consider the trees in Fig. 2.10. The star tree represents a symmetric situation where no topology has emerged, and the second tree represents a decision to have two taxa be neighbors in the final tree. The algorithm computes the sum S_{12} of the lengths of all the edges in the second tree, where 1 and 2 are neighbors:

$$S_{12} = \frac{1}{2(n-2)} \sum_{3 \le k \le n} (d_{1k} + d_{2k}) + \frac{1}{2} d_{12} + \frac{1}{n-2} \sum_{3 \le i < j} d_{ij}$$

assuming the additivity condition. They compute each S_{ij} and choose to make a and b neighbors when S_{ab} is minimum. Then a and b are coalesced (and the distances are recomputed as in UPGMA) and the algorithm iterates on the smaller instance.

References

[1] H. J. Bandelt, Recognition of tree metrics, *SIAM J. Discrete Math.* (1991) **3** 1–6.

[2] J. H. Camin and R. R. Sokal, A method for deducing branching sequences in phylogeny, *Evolution* **19** (1965) 311–326.

[3] L. L. Cavalli-Sforza and A. W. F. Edwards, Phylogenetic analysis: models and estimation procedures, *Am. J. Human Genetics* **19** (1967) 233–257.

[4] R. J. Cedergren, D. Sankoff, B. LaRue and H. Grosjean, The evolving tRNA molecule, *Critical Reviews in Biochemistry* **11** (June 1981) 35–104.

[5] J. Czelusniak, M. Goodman, N. D. Moncrief and S. M. Kehoe, Maximum parsimony approach to construction of evolutionary trees from aligned homologous sequences, in R. F. Doolittle (ed.), *Methods in Enzymology*, Academic Press (1990) 601–615.

[6] W. H. E. Day, Computational complexity of inferring phylogenies from dissimilarity matrices, *Bull. Math. Biology* **49** (1987) 461–467.

[7] W. H. E. Day, Computationally difficult parsimony problems in phylogenetic systematics, *J. Theor. Biology* **103** (1983) 429–438.

[8] W. H. E. Day, D. S. Johnson and D. Sankoff, The computational complexity of inferring rooted phylogenies, *Math. Biosci.* **81** (1986) 33–42.

[9] W. H. E. Day and D. Sankoff, Computational complexity of inferring phylogenies by compatibility, *Systematic Zoology* **35** (1986) 224–229.

[10] W. H. E. Day and D. Sankoff, Computational complexity of inferring phylogenies from chromosome inversion data, *J. Theor. Biology* **124** (1987) 213–218.

[11] M. O. Dayoff, Computer analysis of protein evolution, *Scientific American*, (July 1969) 87–95.

[12] R. V. Eck and M. O. Dayhoff, *Atlas of Protein Sequence and Structure*, National Biomedical Research Foundation (1966).

[13] A. W. F. Edwards and L. L. Cavalli-Sforza, The reconstruction of evolution, *Ann. Human Genetics* **27** (1963) 105.

[14] G. F. Estabrook and F. R. McMorris, When is one estimate of evolutionary relationships a refinement of another? *J. Math. Biol.* **10** (1980) 367–373.

[15] D. P. Faith, Distance methods and the approximation of most-parsimonious trees, *Systematic Zoology* **34** (1985) 312–325.

[16] J. S. Farris, Estimating phylogentic trees from distance matrices, *American Naturalist* **106** (1972) 645–668.

[17] J. S. Farris, Inferring phylogenetic trees from chromosome inversion data, *Systematic Zoology* **27** (1987) 275–284.

[18] J. S. Farris, Methods for computing Wagner trees, *Systematic Zoology* **19** (1970) 83–92.

[19] J. S. Farris, Phylogenetic analysis under Dollos's Law, *Systematic Zoology* **26** (1977) 77–88.

[20] J. Felsenstein, Evolutionary trees from DNA sequences: A maximum likelihood approach, *J. Molecular Evolution* **17** (1981) 368–376.

[21] J. Felsenstein, Numerical methods for inferrring evolutionary trees, *Q. Review of Biology* **57** (1982) 379–404.

[22] J. Felsenstein, Phylogenies from molecular sequences: inference and reliability, *Ann. Review Genetics* **22** (1988) 521–565.

[23] W. M. Fitch, A non-sequential method for constructing trees and hierarchical classifications, *J. Molecular Evolution* **18** (1981) 30–37.

[24] W. M. Fitch, On the problem of discovering the most parsimonious tree, *The American Naturalist* **111** (1977) 223–257.

[25] W. M. Fitch, Towards defining a course of evolution: minimum change for a specific tree topology, *Systematic Zoology* **20** (1971) 406–416.

[26] W. M. Fitch and E. Margoliash, Construction of phylogenetic trees, *Science* **155** (1967) 279–284.

[27] L. R. Foulds and R. L. Graham, The Steiner problem in phylogeny is NP-complete, *Adv. Appl. Math.* **3** (1982) 43–49.

[28] L. R. Foulds, M. D. Hendy and D. Penny, A graph theoretic approach to the development of minimal phylogenetic trees, *J. Molecular Evolution* **13** (1979) 127–149.

[29] R. L. Graham and L. R. Foulds, Unlikelihood that minimal phylogenies for a realistic biological study can be constructed in reasonable computation time, *Math. Biosci.* **60** (1982) 133–142.

[30] D. Gusfield, Efficient algorithms for inferring evolutionary trees, *Networks* **21** (1991) 19–28.

[31] J. A. Hartigan, Minimum mutations fits to a given tree, *Biometrics* **29** (1973) 53–65.

[32] M. D. Hendy and D. Penny, Branch and bound algorithms to determine minimal evolutionary trees, *Math. Biosci.* **59** (1982) 277–290.

[33] W. Hennig, Phylogenetic systematics, *Ann. Review Entomolgy* **10** (1965) 97–116.

[34] W.-H. Li and D. Graur, *Fundamentals of Molecular Evolution*, Sinauer (1991).

[35] F. R. McMorris, On the compatibility of binary qualitative taxonomic characters, *Bull. Math. Biol.* **39** (1977) 133–138.

[36] G. W. Moore, J. Barnabas and M. Goodman, A method for constructing maximum parsimony ancestral amino acid sequences on a given network, *J. Theor. Biol* **38** (1973) 459–485.

[37] D. Penny and M. D. Hendy, Turbotree: a fast algorithm for minimal trees, *CABIOS* **3** (1987) 183–187.

[38] W. J. Le Quesne, A method of selection of characters in numerical taxonomy, *Systematic Zoology* **18** (1969) 201–205.

[39] J. S. Rogers, Deriving phylogenetic trees from allele frequencies, *Systematic Zoology* **33** (1984) 52–63.

[40] N. Saitou and M. Nei, The neighbor-joining method: A new method for reconstructing phylogenetic trees, *Molecular Biological Evolution* **4** (1987) 406–425.

[41] D. Sankoff and R. J. Cedergren, Simultaneous comparison of three or more sequences related by a tree, in D. Sankoff and J. B. Kruskal (eds.), *Time Warps, String Edits and Macromolecules: The Theory and Practice of Sequence Comparison*, Addison-Wesley (1983) 253–263.

[42] D. Sankoff, R. J. Cedergren and G. LaPalme, Frequency of insertion-deletion, transversion, and transition in the evolution of 5s ribosomal RNA, *J. Molecular Evolution* **7** (1976) 133–149.

[43] D. Sankoff and J. B. Kruskal, *Time Warps, String Edits and Macromolecules: The Theory and Practice of Sequence Comparison*, Addison-Wesley (1983).

[44] D. Sankoff and P. Rousseau, Locating the vertices of a Steiner tree in an arbitrary metric space, *Math. Program.* **9** (1975) 240–246.

[45] S. Sattath and A. Tversky, Additive similarity trees, *Psychmetrika* **42** (1977) 319–345.

[46] R. R. Sokal and C. D. Michener, A statistical method for evaluating systematic relationships, *Univ. Kansas Science Bulletin* **28** (1958) 1409–1438.

[47] Y. Tateno, M. Nei and F. Tajima, Accuracy of estimated phylogenetic trees from molecular data I: Distantly related species, *J. Molecular Evolution* **18** (1982) 387–404.

[48] E. A. Thompson, The method of minimum evolution, *Ann. Human Genetics, London* **36** (1973) 333–340.

This Page Intentionally Left Blank

Subject Index

Author Index

This Page Intentionally Left Blank