Lecture Notes in Computer Science 1596
Edited by G. Goos, J. Hartmanis, and J. van Leeuwen

Berlin
Heidelberg
New York
Barcelona
Hong Kong
London
Milan
Paris
Singapore
Tokyo

Riccardo Poli Hans-Michael Voigt
Stefano Cagnoni David Corne
George D. Smith Terence C. Fogarty (Eds.)

Evolutionary Image Analysis, Signal Processing and Telecommunications

First European Workshops,
EvoIASP'99 and EuroEcTel'99
Göteborg, Sweden, May 26-27, 1999
Proceedings

Volume Editors

Riccardo Poli
Department of Computer Science, University of Birmingham
Edgbaston, Birmingham B15 2TT, UK
E-mail: R.Poli@cs.bham.ac.uk

Hans-Michael Voigt
Centre for Applied Computer Science
Rudower Chaussee 5, D-12484 Berlin, Germany
E-mail: voigt@gfai.de

Stefano Cagnoni
Department of Computer Engineering, University of Parma
Parco delle Scienze 181/a, I-43100 Parma, Italy
E-mail: cagnoni@ce.unipr.it

David Corne
University of Reading
Whiteknights, PO Box 225, Reading RG6 6AY, UK
E-mail: D.W.Corne@reading.ac.uk

George D. Smith
University of East Anglia
Norwich, Norwich NR4 7TJ, UK
E-mail: gds@sys.uea.ac.uk

Terence C. Fogarty
Napier University
219 Colinton Road, Edinburgh EH14 1DJ, UK
E-mail: tcf@dcs.napier.ac.uk

Cataloging-in-Publication Data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme
Evolutionary image analysis, signal processing and telecommunications : first European workshops ; proceedings / EvoIASP '99 and EuroEcTel '99, Göteborg, Sweden, May 26 - 27 1999. Riccardo Poli ... (ed.). - Berlin ; Heidelberg ; New York ; Barcelona ; Hong Kong ; London ; Milan ; Paris ; Singapore ; Tokyo : Springer, 1999
(Lecture notes in computer science ; Vol. 1596)
ISBN 3-540-65837-8

CR Subject Classification (1998): I.4, C.3, I.5.4, H.4.3, F.1

ISSN 0302-9743
ISBN 3-540-65837-8 Springer-Verlag Berlin Heidelberg New York

Typesetting: Camera-ready by author
SPIN: 10704703 06/3142 – 5 4 3 2 1 0 Printed on acid-free paper

Preface

Evolutionary Computation (EC) is a rapidly expanding field of computer science which creates, studies and applies problem solving, optimisation and machine-learning techniques inspired by the theories of genetic inheritance and natural selection. Although the origins of this field can be traced back to the inventors of the computer, Turing and von Neumann, it was only properly founded in the 1960s by people such as Holland, Rechenberg and Fogel, and only became widely known and worked on in the late 1980s.

During this time a number of results were reported in the literature which showed how EC techniques can solve problems in domains such as automatic design, optimisation, pattern recognition and control. Until recently, however, only very rarely could one claim that EC techniques approached the performance of human experts in these domains.

Thanks to the technological improvements as a result of empirical work in EC, advances in EC theory, and the increased power of the computer, EC is now ready for large scale applications in complex engineering domains, such as image analysis, signal processing and telecommunications.

This volume contains the proceedings of EvoIASP'99, the first European Workshop on Evolutionary Computation in Image Analysis and Signal Processing, and of EuroECTel'99, the first European Workshop on Evolutionary Telecommunications, held, respectively, on 28 and 29 May 1999, in Göteborg, Sweden.

EvoIASP'99 was the first event specifically devoted to the application of EC to image analysis and signal processing. The aims of the workshop were, firstly, to give European and non-European researchers in these fields, as well as people from industry, an opportunity to present their latest research and discuss current developments and applications, and, secondly, to foster closer future interaction between members of the three scientific communities involved.

EuroECTel'99 was the first international meeting specifically oriented towards work on the application of EC to the variety of optimisation problems which exist in the field of telecommunications. Some of the very latest work on this topic was presented and discussed at this workshop, including new methods for network optimisation, issues of optimisation in distributed databases, and new EC-based methods for verifying communications protocols.

The workshops were held in conjunction with two other major European events: EvoRobot'99, the second European Workshop on Evolutionary Robotics, held on May 28 and 29, and EuroGP'99, the second European Workshop on Genetic Programming, held on May 26 and 27.

Thirteen papers were accepted for publication in the EvoIASP'99 proceedings and for presentation at the workshop while five papers were accepted for publication in the EuroECTel'99 proceedings and for presentation at the workshop. Many of these papers are written by researchers internationally recognised

in their respective fields and all are of high quality. This has been assured by two international program committees which include the best EC researchers from around the world, as well as experts in image analysis, signal processing, and telecommunications.

May 1999

Riccardo Poli, Hans-Michael Voigt, Stefano Cagnoni,
David Corne, George Smith, and Terence C. Fogarty

Organization

EvoIASP'99: the first European Workshop on Evolutionary Image Analysis and Signal Processing was organized by EvoIASP, the EvoNet Working Group on Image Analysis and Signal Processing

EuroECTel'99: the first European Workshop on Evolutionary Telecommunications was organized by ECTelNet, the EvoNet Working Group on Telecommunications

EvoIASP'99 Organizing Committee

Program co-chair:	Riccardo Poli (University of Birmingham, UK)
Program co-chair:	Hans-Michael Voigt (GFaI Berlin, Germany)
Publication chair:	Terence C. Fogarty (Napier University, UK)
Publicity chair:	Stefano Cagnoni (University of Parma, Italy)
Local chair:	Peter Nordin (Chalmers University of Technology, Sweden)

EvoIASP'99 Program Committee

Giovanni Adorni, University of Parma, Italy
Wolfgang Banzhaf, University of Dortmund, Germany
Alberto Broggi, University of Pavia, Italy
Stefano Cagnoni, University of Parma, Italy
Ela Claridge, University of Birmingham, UK
Dave Cliff, MIT, USA
Jason Daida, University of Michigan, USA
Kalyanmoy Deb, University of Dortmund, Germany
Terence C. Fogarty, Napier University, UK
David Hogg, University of Leeds, UK
Mario Koeppen, FhG IPK, Germany
William B. Langdon, University of Birmingham, UK
Evelyne Lutton, INRIA, France
Peter Nordin, Chalmers University of Technology, Sweden
Riccardo Poli, University of Birmingham, UK
Jim Smith, University of the West of England, UK
Hans-Michael Voigt, GFaI Berlin, Germany

EuroECTel'99 Organizing Committee

Program co-chair:	David Corne (University of Reading, UK)
Program co-chair:	George Smith (University of East Anglia, UK)
Program co-chair:	Martin Oates (BT, UK)
Publication chair:	Terence C. Fogarty (Napier University, UK)
Local chair:	Peter Nordin (Chalmers University of Technology, Sweden)

EuroECTel'99 Program Committee

Thomas Baeck, Informatik Centrum Dortmund, Germany
Brian Carse, University of the West of England, UK
Marco Dorigo, Free University of Brussels, Belgium
Terence C. Fogarty, Napier University, UK
Jin K. Hao, EMA-EERIE, France
Markus Hoehfeld, Siemens, Germany
Andy Keane, Southampton University, UK
Bernard Manderick, Free University of Brussels, Belgium
Jason Mann, Nortel, UK
Masaharu Munetomo, Hokudai University, Japan
Peter Nordin, Chalmers University of Technology, Sweden
Ken Sharman, University of Glasgow, UK
Matteo Sonza Reorda, Politecnico di Torino, Italy
Mark Sinclair, University of Essex, UK
Alice Smith, University of Pittsburgh, USA
John Turner, Nortel, UK
Brian Turton, Cardiff School of Engineering, UK
Athanasios Vasilakos, ICS-FORTH, Greece

Sponsoring Institutions

Chalmers University of Technology and Göteborg University, Sweden
EvoNet: the Network of Excellence in Evolutionary Computing

Table of Contents

Evolutionary Telecommunications Talks

Evolutionary Telecommunications Poster

An Evolutionary Approach to Fitting Constrained Degenerate Second Order Surfaces

C. Robertson, R.B. Fisher, N. Werghi, and A.P. Ashbrook

Division of Informatics, University of Edinburgh,
Edinburgh, EH1 2QL, UK
craigr@dai.ed.ac.uk

Keywords: Evolutionary algorithms, surface fitting, Genocop III.

Abstract. In this work we examine the applicability of an evolutionary strategy to the problem of fitting constrained second-order surfaces to both synthetic and acquired 3D data. In particular we concentrate on the Genocop III algorithm proposed by Michalewicz [8] for the optimization of constrained functions. This is a novel application of this algorithm which has demonstrably good results when applied using parametric models. Example times for convergence are given which compare the approach to standard techniques.

1 Introduction

Shape analysis of objects from range data (captured three dimensional co-ordinates of surface points) is a key problem in computer vision with several important applications in manufacturing, such as assembly, quality control and reverse-engineering. The problem is generally formulated as a nonlinear programming problem (NLP), which tries to optimally fit the data to candidate shape descriptions. The NLP optimises a function subject to several constraining equations and inequalities. Especially with nonlinear constraints, it is notoriously difficult to optimise and there is no known method to guarantee a satisfactory solution. Traditional techniques, such as gradient descent, are unsatisfactory for the solution of NLPs, due to the local nature of their search methods and the reliance on smooth derivatives in the search-space. In previous work [7] we examined the applicability of evolutionary strategies to the problem of fitting lines and surfaces to both synthetic and acquired object range data. In this paper we effectively take the next step, which is to fit degenerate second order surfaces that have *a priori* constraints and geometric relationships. The Genocop III algorithm developed by Michalewicz [8], Ch.7 was used and extended in this paper by adding a complex evaluation function. It is an evolutionary algorithm system which is specialised to handle constrained function optimisation and particular to it is the handling of non-linear constraints. It uses real-valued genes, and includes methods to deal with linear, non-linear, domain and inequality constraints. We have used a specialised fitness function (described in section 2.3.3), applied to the problem of fitting parametric 3-dimensional surface equation chromosomes to range data while simultaneously applying several necessary geometric and domain constraints. The constraints applied are of two typical types : domain, the restriction on the parameter size; relational, relationships between surfaces that are known *a priori*.

R. Poli et al. (Eds.): EvoIASP'99 and EuroEcTel'99, LNCS 1596, pp. 1–16, 1999.

Since this problem has a specific context it is important to illustrate it. Our group is researching the reverse engineering machined parts. These parts are often complex and possess many surfaces which may have know geometrix relationships. Segmentation and parameterisation of the captured 3-dimensional range data is a difficult multi-part task involving the following elements:

1. *Data collection.* This is performed using a moving-bed, orthogonal laser ranger which provides data at up to 0.5mm steps in the X-Y plane. Noise on the data is around $0.15\ mm$.
2. *Data registration.* This is performed using a variation on the **iterated closest point** algorithm [1].
3. *Segmentation.* There are many ways of segmenting the 3d dataset, most are based upon changes in local surface curvature followed by some form of least-squares optimisation, for example [9].
4. *Exploitation of constraints.* Constraints may be applied to exploit knowledge about surface relationships.

The formulation of constraints and the application of constraint-based correction and optimisation of surface fitting has been achieved previously [5] with notable success using the several constraint application strategies. There are, however, some associated problems with this approach: complex formulation of the constraint function; heavy reliance on the global convexity of the solution space; reliance on very accurate initial estimate of solution.

The 'processing pipeline' that is required for this approach also can lead to a build-up of problems that must be solved in the constraint application stage. In order to alleviate some of these problems a more holistic strategy is proposed where segmentation, fitting and constraint management takes place simultaneously. Previously [7] it was demonstrated that simultaneous fitting and constraint management could be achieved in a single evolutionary algorithm with careful chromosome management and good generation of starting conditions.

In this paper, the technique has been explored and improved by:

1. *Making the representation more efficient* by only applying the technique to degenerate quadric surfaces.
2. *Enhancing the evaluation* of chromosomes by applying specialised fitting functions for degenerates.
3. *A simple objective function*, the least-squares error metric, then using the constraints to define the manifold of allowable solutions.
4. *Including a naive segmentation function* as part of the evaluation function.

2 Method

2.1 Data Generation

Free Quadrics

A free quadric is a second order surface of the type:

$$a_1x^2 + a_2y^2 + a_3z^2 + a_4xy + a_5xz + a_6yz + a_7x + a_8y + a_9z + a_{10} = 0 \quad (1)$$

This covers all second order shapes and these fall into several family groups : cylinders, cones, paraboloids, hyperboloids, ellipsoids, planes. Some of these forms are degenerate and there is sufficient variety in the shapes to cover all easily machineable surfaces. Note that higher order free surfaces also exist on machined parts, perhaps as a result of casting, but are not considered here.

Degenerate Quadrics

In this paper, the degenerate forms of these surfaces are used, this is a small subfamily of surfaces of the following types: spheres, cylinders, right circular cones. The machined surfaces that we address rarely contain whole pieces of these shapes and the surfaces are often fragmentary or partial. In order to generate synthetic versions of this subfamily, patches of the given types were generated as shown in figure 1. These fell into the following three groups:

1. *Spherical sections* generated about a given vector (spherical caps) as well as whole spheres
2. *Cylinders* of differing radii, length and wedge angle.
3. *Cones* of differing slope-angle, wedge-angle and length. Truncated cones were also used.

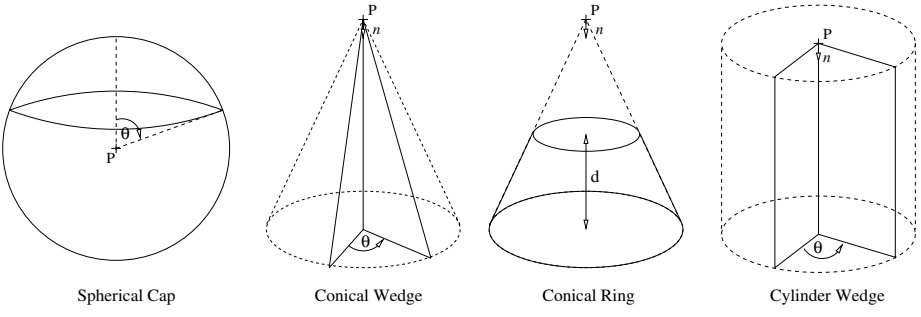


Fig. 1. Degenerate Quadric Patches

2.2 Gene and Chromosome Formulation

In standard GAs binary encoding forms the chromosomes in the solution, however in an evolution program each gene is a floating point number. Genes are then concatenated into a chromosome. In the cases we have previously investigated, typical part-chromosomes were parameter vectors ($A = a_i : i \in \{1, .., 10\}$ representing second order surfaces. In the new parametric representations, the vectors may be any length.

In the case of **planes**, we have used the 4 gene parametric representation : $A :< \hat{n}, d >$ where $\hat{n}$ is the unit normal describing the plane and d is the constant defining its minimum distance from the origin. In the case of **spheres**, we have used the 4 gene parametric representation : $A :< \mathbf{P}, r >$ where $\mathbf{P}$ is the centre point and

r is the sphere radius. In the case of **cylinders**, we have used the 7 gene parametric representation : $A :< \mathbf{P}, \hat{n}, r >$ where $\mathbf{P}$ is defined as the start point of the cylinder, $\hat{n}$ is the axis direction and r is the radius. For **cones**, the 7 gene representation used is : $A :< \mathbf{P}, \hat{n}, \alpha >$,where $\mathbf{P}$ is defined as the start point (or tip) of the cone, $\hat{n}$ is the axis direction and α is the half-angle between the axis and the slope of the cone.

A full chromosome, G, describing a given object, is a set of concatenated part-chromosomes: $G = \{A_j\}$. The parametric representation of a set of degenerate quadrics as a chromosome is much shorter than a set of general quadrics as explored in [7]. This cuts down the complexity of constraint representation and makes it amenable to straighforward manipulation without the need to employ geometrical constraints on the surfaces' form, only on pairs of forms taken as systems.

2.3 Algorithm

Traditional Methods of Generating Constrained Populations. Evolutionary methods have been shown to be useful for solving general *NLP* problems [11][6][7]. There are four main techniques for dealing with chromosomes that contravene constraints on solutions [10]: rejection, which discards infeasible solutions immediately throughout the process; repairing, which depends on methods to repair solutions back to feasible; modifying operators, which means designing crossover, mutation and other operators than only ever produce feasible offspring; penalties, which is the most common technique for optimizing constrained functions. A penalty function is one which punishes chromosomes for straying from the constraints by decreasing their fitness or removing them from the population. There are no good guidelines for designing such penalty functions however [10].

Almost all optimization problems are constrained in some way. What we required is some way of generating solutions that are both iteratively improving as well as satisfying these constraints. Most optimization problems are defined on a search space, D, as follows [1] : $D \subseteq R^q$, where $D = \prod_{k=1}^{q} < l_k, r_k >$ and each x_k is in the interval $< l_k, r_k >$. The set R^q is thus a crucial characteristic of the problem. Significant optimization theory only exists where this set is convex. In GENOCOP, this convexity is assumed, i.e we seek to optimize : $f(x_1, \ldots, x_q) \in R^q$, where $(x_1, \ldots, x_q) \in D \subseteq R^q$. D is convex and is defined by the range of variables $l_k \leq x_k \leq r_k$ for $k = 1, \ldots, q$.

Because D is convex, for each point in the search space, there exists a range where other variables remain fixed. We also assume that this range can be efficiently computed. This property is useful for performing **mutation**. If the variable x_k is to be mutated, it can be moved inside its range so any offspring produced are feasible.

Also, for any two points, $\mathbf{x}_1$ and $\mathbf{x}_2$, in the space D, the linear combination $a\mathbf{x}_1 + (1-a)\mathbf{x}_2$, (where $a \in [0, 1]$) is also in D. This is used for **crossover**.

Genocop II and III Calculus based methods assume that the objective function, $f(\mathbf{x})$, and all constraints are twice continuously differentiable functions of $\mathbf{x}$. The general approach is to transform the non-linear problem (*NLP*) into a sequence of sub-problems

[1] This brief account follows the one given by Michalewicz [8] which the reader is encouraged to read for further details.

and then solve those, requiring an explicit computation of the objective function. Some of these methods become ill-conditioned and fail.

Genocop II uses a sequential quadratic penalty function and is formulated as the optimisation of the function:

$F(\mathbf{x}, r) = f(\mathbf{x}) + \frac{1}{2r}\overline{C}^T\overline{C}$, where $r > 0$ and $\overline{C}$ is the vector of active constraints, $c_1, \ldots, c_l$

Attia has provided solutions to the instability of this approach [12]. The set of all constraints, C, is divided into the linear constraints,L, the non-linear equations,N_e and the non-linear equalities,N_i. A set of active constraints, A is then built from N_e and the violated constraints from N_i (a constraint is said to be violated if it is more than some tolerance δ from its correct value), which are called V. The structure of Genocop II is outlined in [8]. Inside its main loop, Genocop I optimizes the function $F(\mathbf{x}, r) = f(\mathbf{x}) + \frac{1}{2r}\overline{A}^T\overline{A}$. Several mutation operators take an initially identical population and introduce diversity to it. At convergence, the best individual,$\mathbf{x}^*$, is saved and the the value of the penalty parameter is decreased.

Most of the essential elements of Genocop III are the same as those of Genocop II. However, in this algorithm two populations are kept, a reference set $\overline{R}$ and a search set $\overline{S}$. The reference population is a set of fully feasible individuals which satisfy all the constraints whereas the search population may not. At each iteration, the search population are allowed to move around the solution space and are repaired back onto the constraint manifold. If the search point is $\overline{S}$ and the reference point is $\overline{R}$, then a random point $\overline{Z}$ is created from the segment between $\overline{S}$ and $\overline{R}$ by generating a value, $a \in [0, 1]$ then :

$$\overline{Z} = a\overline{S} + (1 - a)\overline{R} \tag{2}$$

Once a feasible $\overline{Z}$ is found, if it is better than $\overline{R}$, then that reference point is replaced with some probability. As the iterations progress, the set of reference points converge to the maximum or minimum on the search space.

Evaluation Function and Point Assignment for the Fitting. In our application of Genocop III the evaluation function is almost certainly more complex than was initially intended for the algorithm. For each point,x_i, the true geometric distance to the theoretical surface is computed and this is used as the least-squares error for that point relative to that surface.

$$e_i = min_p\{dist(\mathbf{x}_i, S_p)\} \tag{3}$$

where e_i is the error for the point $\mathbf{x}_i$, p is the index of M theoretical surfaces, i is the index of N points, S_p is the parameterised surface and *dist* is the distance to that surface.

The evaluation function to be minimised is then the sum of these minima :

$$E = \sum_{i=0}^{i=N} e_i \tag{4}$$

It is possible to use this as a simple segmentation scheme (especially if the chromosome population variations are small and the start conditions are close to the solution).

The point assignment is thus straightforward. In some tests, such as the real object discussed in section 3.3 the data is pre-segmented. If the assignment information is available it should be used in order that the computation time can be reduced.

Starting Conditions and Relational Constraints In virtually all cases, domain constraints on individual genes are used to narrow the search space for that gene. These are represented as one permanent part of the sequential quadratic penalty function matrix [8] used in the evaluation function. A good example of where domain constraints can reduce the search space is in the case of the three parameters describing a unit normal. Each of these parameters can never be outside the range $[-1, +1]$ so these make good domain constraints.

In-chromosome relational constraints are straightforward to formulate when a parametric form is used. For example, consider two planes,$P_1 =< x_1, x_2, x_3, x_4 >$ and $P_2 =< x_5, x_6, x_7, x_8 >$ which are known *a priori* to be orientated orthogonally. In this case, the chromosome would have the form $G = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8\}$ and the orthogonality constraint would then appear as a non-linear inequality of the form: $(x_1 - x_5)^2 + (x_2 - x_6)^2 + (x_3 - x_7)^2 \leq \epsilon$ where ϵ is the constraint tolerance value.

In order to perform the optimisation, Genocop requires a starting position on the constrained manifold. This is the seed for the search points which are then mutated around it. It is also used to produce the set of reference points as described earlier in this section. In our case this means designing a chromosome which is both close to being a concatenation of the individual least-squares results for the part-chromosomes as well as fulfilling the domain and relational constraints. Starting conditions for increasingly complex solutions with increasingly complex constraints have previously been found to be difficult [7] for the general quadric. However, when a parametric representation is used, start conditions become very simple to generate, even when many constraints are used. For example, when a parametric representation for a right, circular cone is used the chromosome consists of seven floating-point genes with one constraint, that of normality for the axis vector. When a general quadric is used, however, the chromosome consists of ten floating-point genes with six constraints to ensure its form. When further constraints are added, a start condition for the reference population becomes difficult to find for the general representation but relatively easy for the parametric one (since the only constraint is normality for 3 genes in the whole sequence). The complexity increases as further quadrics are added. Consider a chromosome with three general quadrics representing three right, circular cones. In a general representation this would be 30 floating-point genes together with 15 shape constraints and three relationship constraints. In a parametric form it would be 21 genes and six constraints in total. This reduction is quite marked but clearly relies on knowing *a priori* the classification of the surfaces in the data.

3 Results

In total 70 single-object experiments were carried out, all of which reached successful convergence, i.e the summed error over all points in the data set stablized. The final values for chromosome parameters are used as the test of value for each of the experiments.

Where there were significant convergence effects these have been noted. All of the data used for the synthetic surfaces has Gaussian noise added with standard deviation 0.1mm. This is comparable with the laser range finder and therefore represents typical data noise.

3.1 Caps, Rings and Wedges

Spherical Caps

Ten spherical cap datasets were generated for decreasing values of θ from 170^o to 30^o to test feasibility of fitting and speed of convergence. Sphere radii were kept at $10\ mm$ with centre position $(0, 0, 15)$. The number of data points per data set was 1000 so the data at 170^o is much more descriptive of the shape than the data at 30^o, illustrated in the convergence rate, fig.2(a).

One important aspect of these tests is that with $\theta = 30^o$ data the variation in position genes is much higher than with $\theta = 170^o$ (figures 2(b) and 2(c)). Intermediate positions show that the rate of these variations and rate of convergence change gradually with angle. Note that our previous fitting tool which utilises Taubin accumulation [4] fairs similarly in the tests, having almost exactly the same average error per point. After $25,000$ evaluations all radii converged to the correct value given experimental noise.

Cylinder Wedges

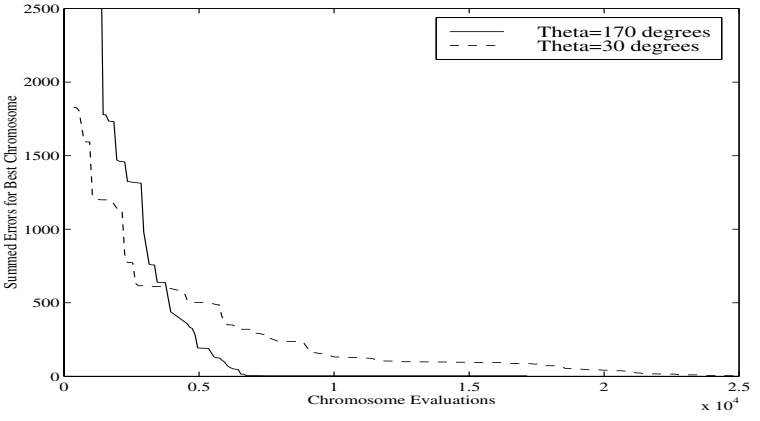
Ten cylindrical wedge datasets were generated for decreasing values of θ from 170^o to 30^o to test feasibility of fitting and speed of convergence. Cylinder radii were kept at 10mm, data was generated around the normal $(0, 0, 1)$ and the starting point was $(15, 10, 10)$. The length of the cylinder in each case was 20mm. Error graphs for the cylinder wedges tell us little about convergence except that it is dependent upon the initial reference population, which the algorithm generates internally. All of the test examples converged within $100,000$ evaluations with the axis correct to within around 0.5^o and radius correct to around 0.05mm, as shown in table 1.

θ	Normal Error/^{o}	Radius Error/mm
170	0.5733	0.0287
155	0.4196	0.0012
140	0.0000	0.0027
125	0.5754	0.0040
110	2.5648	0.1464
90	0.5500	0.0032
70	0.5751	0.0191
50	0.5747	0.0033
45	0.0000	0.1213
30	0.0000	0.1851

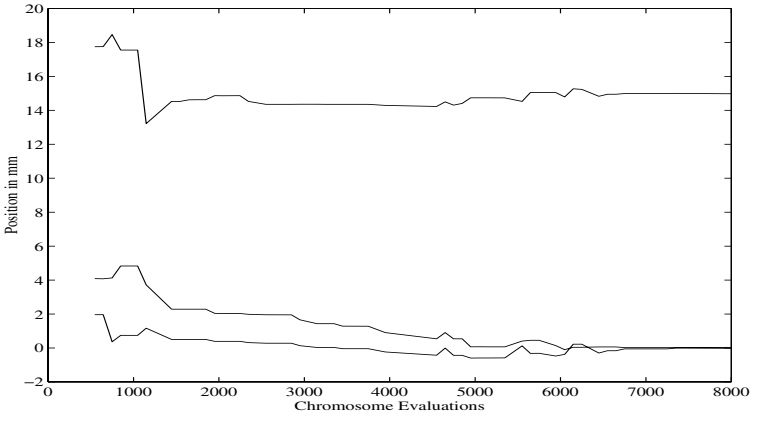
Table 1. Absolute Errors on Cylinder Wedge Parameters

Conical Wedges

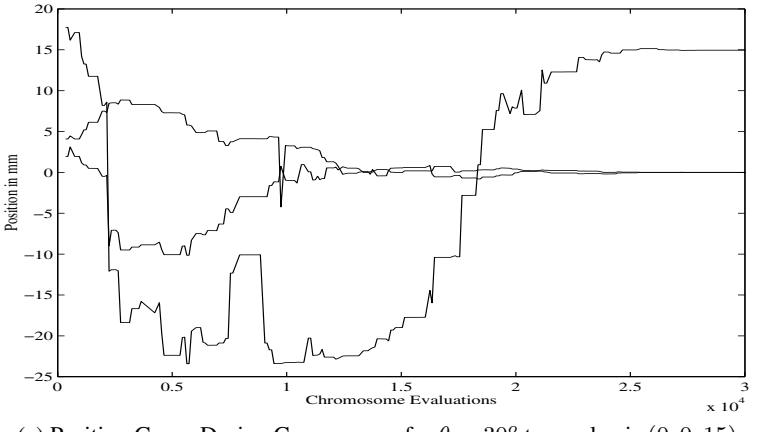
Ten conical wedge datasets were generated for decreasing values of θ from 170^o to 30^o to test feasibility of fitting and speed of convergence. Cone slope half-angle was kept at 30^o, axis was $(0, 1, 0)$, apex position was $(0, 25, 0)$ and cone depth was 20mm.



(a) Error Convergence for $\theta = 170^o$ and $\theta = 30^o$



(b) Position Genes During Convergence for $\theta = 170^o$ true value is $(0, 0, 15)$



(c) Position Genes During Convergence for $\theta = 30^o$ true value is $(0, 0, 15)$

Fig. 2. Error Graphs for Spherical Datasets

1000 data points were used per set, with noise as before. Details of the results are shown in table 2. The error on the cone normal increases as the wedge decreases. At 30^o it is actually outside acceptable error bounds. This caveat may be explained by the error in the estimate of the apex position, which can be seen to drift as the wedge angle decreases.

θ	Normal Error/^{o}	Slope Angle Error/^{o}	Apex position estimate		
170	0.5815	0.0375	-0.002763	25.008604	0.0161751
155	0.5314	0.4210	0.001005	25.012029	0.0150246
140	0.5747	0.0460	0.005348	25.018583	-0.0153982
125	0.5808	0.0204	0.021809	25.007158	-0.0135571
110	0.5727	0.0007	0.002103	25.007488	-0.0285622
90	0.5757	0.0059	-0.014792	24.991615	-0.0065875
70	0.7000	0.3159	-0.009692	25.162830	-0.1308158
50	1.5526	1.2332	0.006790	25.636304	-0.4279015
45	1.1866	0.8757	0.006669	25.655696	-0.4368368
30	3.0208	2.7954	0.000730	26.736854	-1.0613992

Table 2. Absolute Errors on Cone Wedge Parameters and Apex Estimate

Conical Rings

Ten conical ring datasets were generated for decreasing values of length, from 100mm to 10mm from the base, from a cone of total length 110mm measured base to apex. Cone slope angle was kept at 30^o, axis was $(0, 1, 0)$, apex position was $(0, 25, 0)$. A full 180^o spread was also used and 1000 data points were used per set.

Convergence over all of the conical ring datasets was uniform. Errors are detailed in table 3. Average error for the normal axis was around 0.5^o and the error on the cone slope angle was less than 0.01^o. It can also be seen that the apex position estimates are much better that those in the cone wedge case.

Length	Normal Error/^{o}	α Error/mm	Apex Position		
100	0.5730	0.0020	-0.010790	24.981763	0.027289
90	0.2149	0.0023	-0.014435	25.012355	0.012871
80	0.5630	0.0104	-0.000247	25.026578	-0.005360
70	0.4681	0.0042	0.037935	24.979377	0.002949
60	0.5464	0.0012	0.001648	24.981952	-0.002780
50	0.5723	0.0017	0.005134	24.981803	0.000237
40	0.4681	0.0046	-0.031845	25.003873	0.090405
30	0.3621	0.0021	0.006406	24.984144	-0.004444
20	0.5630	0.0430	0.043338	24.811958	-0.014173
10	0.5835	0.0037	-0.025032	24.973985	-0.342472

Table 3. Absolute Errors on Cone Ring Parameters and Apex Estimate

3.2 Constrained Degenerate Quadric Pairs

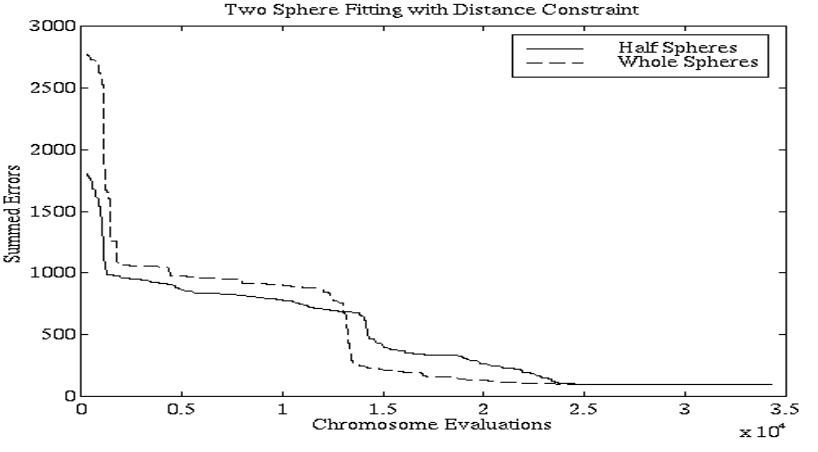
Distance Constraints

Spheres were fitted with the distance constraint applied to their centres. The sphere parameters used were as follows:

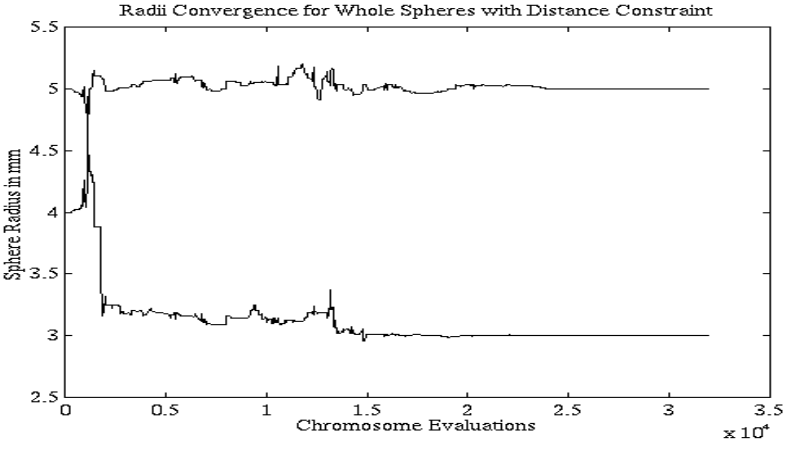
*Sphere 1:*Position$(0, 0, 10)$, Radius 5, complete/half sphere, 1000 data points.

*Sphere 2:*Position$(0, 0, 0)$, Radius 3, complete/half sphere, 1000 data points.

Summed errors for the fitting are shown in fig. 3(a) and the convergence of the radii genes is shown in fig, 3(b). Position for both spheres was found accurately to within 0.001mm and the radii were found to within 0.01mm. Since the results were similar for both half-spheres and spheres, only whole spheres are shown.



(a) Summed Errors for Constrained Sphere Fitting (Whole Spheres)



(b) Radii Convergence for Constrained Sphere Fitting using Whole Spheres with Distance Constraint (true values are 3mm and 5mm)

Fig. 3. Constrained Sphere Fitting Graphs

Convergence for spheres and half-spheres that were overlapping produced similar results although using slightly more chromosome evaluations. It was also noted that the quality of results obtained was as good as without constraints in both cases, i.e radii to within 0.001mm and position parameters to within 0.005mm.

Mixed Constraints

In mixed constraint experiments whole cones and cylinders were used. In the first set of experiments cones were used with four constraints : two unit normal constraints on the axes, a distance constraint between apexes and an orthogonal constraint on the axes. Data for the cones was as follows:

Cone 1: Apex position $(0, 0, 0)$, Axis $(0, 0, 1)$, Length 20, half-angle $\alpha = 30^o$ or 0.5235988 radians.

Cone 2: Apex position $(0, 0, 20)$, Axis $(0, 0, 1)$, Length 20, half-angle $\alpha = 30^o$ or 0.5235988 radians.

Cone 3: Apex position $(20, 0, 0)$, Axis $(1, 0, 0)$, Length 20, half-angle $\alpha = 30^o$ or 0.5235988 radians.

Once again, Gaussian noise is applied with a standard deviation of 0.1mm.

Two Cones with Apex Distance and Same Axis Constraints

In this experiment, cones 1 and 2 were used. The starting vector for the reference population was simple to find since the constraints on the fitting have been much simplified since earlier efforts to perform this test [7]. Previously, for two right circular cones with a distance constraint and an axis constraint the total number of inter-gene constraints would have been 16, now it is only 3. The cones are generated on the same axis but with apexes 20mm away from each other. The vector for the reference populations was as follows:

$\mathbf{P}_1 = (0, 0, 1), \hat{n}_1 = (0, 0.141, 0.9899), \alpha_1 = 0.5$ radians

$\mathbf{P}_2 = (0, 0, 21), \hat{n}_2 = (0, 0.141, 0.9899), \alpha_2 = 0.5$ radians

which is in the same order as the generation data above. Position for each of the two part-chromosomes is 1mm away from the correct position and the slope angle is 0.024 radians (1.35^o) from the true value. The normal axis is also at an angle of around 0.5^o to the correct value. These values were chosen because they are typical of values found after registration using ICP [1]. The result vector was as follows :

$\mathbf{P}_1 = (0.000216, -0.004444, -0.013367), \hat{n}_1 = (-0.000111, 0.000021, 0.999969), \alpha_1 = 0.523469$ rad.

$\mathbf{P}_2 = (0.002930, -0.007804, 19.986713), \hat{n}_2 = (-0.000325, -0.000021, 0.999951), \alpha_2 = 0.523393$ rad.

This shows that the convergence to the correct cone models and constraint satisfaction is remarkably good, to the noise level on the data set.

Two Cones with Apex Distance and Orthogonal Axis Constraint

In this experiment, cones 1 and 3 were used. The cones were generated on the same axis but with apexes 20mm away from each other. The vector for the reference populations was as follows:

$\mathbf{P}_1 = (0, 0, 1), \hat{n}_1 = (0, 0.141, 0.9899), \alpha_1 = 0.5$ radians

$\mathbf{P}_2 = (0, 0, 21), \hat{n}_2 = (0, 0.141, 0.9899), \alpha_2 = 0.5$ radians

The result vector was as follows :

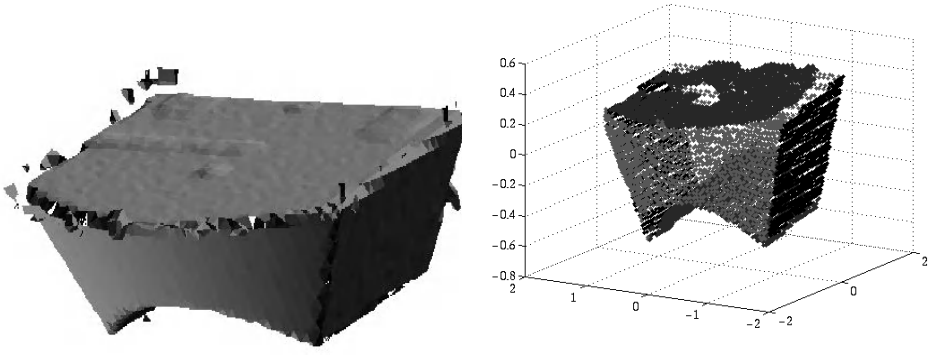
$\mathbf{P}_1 = (-0.006925, -0.001579, -0.024732), \hat{n}_1 = (0.000279, -0.000089, 0.999950)$, $\alpha_1 = 0.523178$ rad.

$\mathbf{P}_2 = (19.993097, -0.012630, -0.000523), \hat{n}_2 = (0.999949, 0.000792, -0.000378)$, $\alpha_2 = 0.523704$ rad.

This shows that the convergence to the correct cone models and constraint satisfaction.

3.3 A Real Object

In order to test the overall application of these technique a reasonably complex real part was examined. This machined part (called the UFO) is an object consisting of six surfaces, four planes and two quadrics as shown in figure 4. It is made to high tolerances but is formed from eight data sets which are then registered together.



UFO Part as Polygons UFO Part after Segmentation

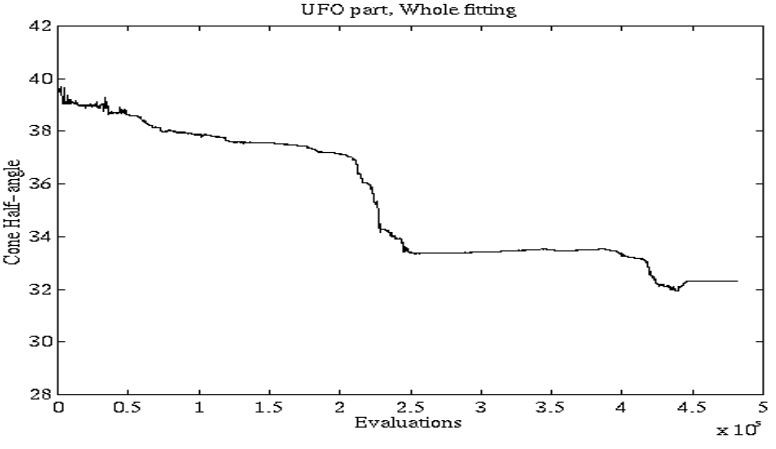
Fig. 4. Real Object Rendered as Polygons and Vertices

The object was first segmented into individual surfaces using a region growing method and Taubin accumulation for the fitting [13]. Each of the different surfaces was saved as a 3d data set. There were 7274 data points in total, representing the polygon centres. This is a valid subsampling since the polygon mesh is statistically representative of the original data set (many tens of thousands of points). The chromosome representing the constrained shape consisted of 30 individual genes as follows: $G = A_1, ..., A_6$, where A_1 is the 7 parameter cylinder, A_2 is the 7 parameter cone and A_3 to A_6 are the 4 parameter planes. A total of 11 constraints were used, the first 6 were unit normal constraints and the final 5 were geometric as follows:

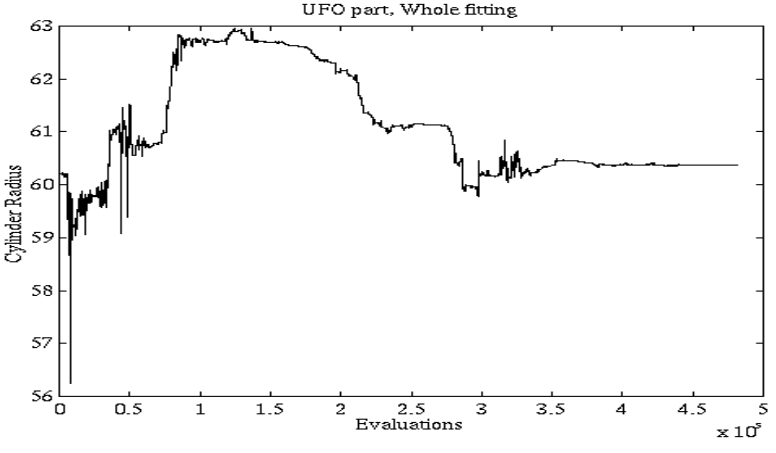
1. Cylinder axis is the same as that of the back plate normal.
2. Cone axis is the same as the bottom plate normal
3. The back plate normal is orthogonal to the bottom plate
4. The back plate normal is orthogonal to the sloping side (side 1)
5. The sloping sides are at 120^o to each other

These five constraints fully constrain the object's shape. The start conditions for this object were simply the least-squares fit for each of the surfaces which were then adjusted to fit the constraints (obviously sub-optimally). The controlling ground-truth used for this test was that the cylinder radius was known to be 60mm and the cone half-angle was known to be 30^o from the normal at its apex.

The graphs in figure 5 show the convergence of the parameters 500,000 evaluations.



(a) Cone Half Angle for Constrained UFO Fitting



(b) Cylinder Radius for Constrained UFO Fitting

Fig. 5. Constrained UFO Fitting Graphs

The full set of estimations for the parameters is given below:

- *Cylinder :* $\mathbf{P}_1 = 154.086, -2.184, -57.711, \hat{n}_1 = 0.995561, 0.005837, 0.088583$, radius = 60.366mm.
- *Cone :*$\mathbf{P}_2 = 45.307, -5.697, -137.595, \hat{n}_2 = -0.044865, 0.039102, 0.998023$, $\alpha = 32.3^o$
- *Planes*
  1. $\hat{n}_1 = 0.995425, 0.003886, 0.091327$, d = 46.62732)
  2. $\hat{n}_2 = 0.046830, -0.885180, -0.461808$, d = 47.79109
  3. $\hat{n}_3 = 0.041421, 0.845461, -0.531490$, d = 47.06203
  4. $\hat{n}_4 = -0.092407, 0.041308, 0.995308$, d = 14.12157

The numbers of points used was as follows: cylinder 1896, cone 1386, back-plate 616, side1 767, side2 839, bottom plate 1770. Time to reach a stable solution was 218.75 minutes when running on a 269MHz Ultrasparc 10 at approx 50% CPU usage. These results have several important aspects. Firstly, the cone half angle is within the same margin of error as the synthetic data when only a 30^o wedge is used. This is explained by the fact that not only is this a 60^o wedge of a cone but it is truncated at only 30mm out of its 140mm height. The cylinder parameters are very good since the cylinder data describes only a 60^o wedge of the original data. It should also be noted that all of the constraints are satisfied to within the tolerance prescribed $(\epsilon = 0.001)$ which represents 0.0573^o error on axis constraints. The summed least-squares error (Euclidean rather than algebraic) was 19.8367 over the whole of the six surfaces. This should also be seen in light of the fact that there were around 1% of outliers in the whole dataset and the registration process is almost certainly imperfect.

4 Conclusions

4.1 Improvements

In previous work [7] we showed that optimal surface fitting under geometric constraints was feasible with an evolutionary algorithm. In this paper we have demonstrated that the method is more accurate when parametric models are used for degenerate surfaces. We have also demonstrated that the geometric constraints for these models are much simpler to formulate. This leads to the quicker formulation of starting reference populations.

We have shown that even a naive point-assignment algorithm can perform simple segmentation when used in conjunction with geometric constraints and we have thus tentatively proposed a full processing system for range-data. The pptimization of least-squares surface fitting is comparable to methods currently employed for second-order surfaces. In some instances (for example where data is sparse) it is actually superior. This is due partially to the fact that data presentation is not ordered as it is in, say, the Taubin accumulation process [3],[4]. No claims are made for the segmentation algorithm other than in circumstances where two sphere datasets overlap if a distance constraint is applied the algorithm still convergenes in a time comparable to no-overlap. Therefore segmentation is a plausible addition to the functionality of the algorithm. The application of geometic and domain constraints has ensured that the convergence to the optimal solution (subject to tolerance) has been achieved. When geometric constraints have

been applied they also have been fulfilled (subject to tolerance). These constraints have been : distance, axis normality and relative axis position. These are the only constraints applicable to the degenerate surfaces we have examined.

4.2 Caveats - New and Old

Of the problems mentioned in the first paper [7], none are now applicable. The formulation of the initial vector for the reference set is easy for parameterized chromosomes. Generating two right cirular cones with orthogonal vector normals, for example, using a parameterized model is straightforward whereas using the previous representation was difficult. When these initialisations were compounded the task was infeasible.

The problem of traversing the space different amounts in different dimensions is somewhat mitigated by this parameterizarion since the units are at least within the same order. The problem of what the slack constraint variables actually *mean* is still different for each constraint. This problem is not so pronounced here as in other methods, for example [5].

One new caveat is that the parametric versions of the objective functions are not so easy to speed up by using off-line calculations. This can mean that for objects involving many (possibly hundreds of) thousands of points the least-squares error calculations will be time-consuming. A simple sub-sampling scheme has been implemented where at each iteration a sample from such a huge dataset is taken and errors computed from this. Although initial tests have been positive, the rates of convergence are not as predictable as those found in this paper - even if the time to convergence is much lower. It is widely held that evolutionary schemes, in fact GAs as a whole, are quick to implement but slow to run. In the main this is true but with a computationally complex evaluation function it is doubly so.

4.3 Further Work

This work is a **proof of concept**, i.e that an evolutionary algorithm could solve the problem of constrained surface fitting, and as such is complete. There are many side issues that should be addressed: speeding-up the chromosome evaluation; including data registration, although how is not clear; outlier removal from the data at run time would provide modest improvements; and weighting the surface errors to skew the fit towards more important surfaces.

Acknowledgements

The work presented in this paper was funded by a UK EPSRC grant GR/H86905. The authors would also like to thank Andrew Tuson and David Corne for advice on several aspects of this work.

References

1. D. Eggert, A. W. Fitzgibbon, R. B. Fisher. "Simultaneous registration of multiple range views for use in reverse engineering", Proc. Int. Conf. on Pat. Recog., pp 243–247, Vienna, Aug. 1996.
2. H. Hoppe, T. DeRose, T. Duchamp, J. McDonald and W. Stuetzle. "Surface Reconstruction from Unorganized Points", Computer Graphics, 26(2), pp 71–78, 1992.
3. G. Taubin. "Estimating the tensor of curvature of a surface from a polyhedral approximation", Proc. 5th Int. Conf. on Computer Vision, pp 902-907, 1995.
4. G. Taubin. "Estimation of planar curves, surfaces and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation", Proc. IEEE PAMI, 13(11), pp1115–1138, 1991.
5. N. Werghi, R. B. Fisher, A. Ashbrook, C. Robertson, "Improving model shape acquisition by incorporating geometric constraints", Proc. British Machine Vision Conference BMVC97, Essex, pp 520–529 September 1997.
6. Z. Michalewicz, D. Dasgupta, R. G. Le Riche and M. Schoenauer, "Evolutionary algorithms for constrained engineering problems", special issue on *Genetic Algorithms and Industrial Engineering*, ed. M. Gen, G.S.Wasserman and A. E. Smith, *International Journal of Computers and Industrial Engineering*, 1996.
7. C. Robertson, D. Corne, R. B. Fisher, N.Werghi, A.Ashbrook, "Investigating Evolutionary Optimisation of Constrained Functions to Capture Shape Descriptions from Range Data", Proc. 3rd On-line World Conference on Soft Computing (WSC3) (see `http://www.cranfield.ac.uk/wsc3/` also in *Advances in Soft Computing - Engineering Design and Manufacturing*, eds. Roy, Furuhashi and Chawdhry, Springer-Verlag, 1998.
8. Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Third Edition, Springer, 1996.
9. A. Hoover, G. Jean-Baptiste, X. Jiang, P. J. Flynn, H. Bunke, D. Goldgof, K. Bowyer, D. Eggert, A. Fitzgibbon, R. Fisher, "An Experimental Comparison of Range Segmentation Algorithms", IEEE Trans. Pat. Anal. and Mach. Intel., Vol 18(7), pp673–689, July 1996
10. M. Gen and R. Cheng, "A Survey of Penalty techniques in Genetic Algorithms", In *Proceedings of the IEEE International Conference on Evolutionary Computation 1996*, 1996.
11. Z. Michalewicz and N. Attia, "Evolutionary Optimization of Constrained Problems", in *Proceedings of the 3rd Annual Conference on Evolutionary Programming*, San Diego, CA, 1994, pages 98-108. World Scientific.
12. N. F. Attia, *New Methods of Constrained Optimization Using Penalty Functions*, Ph.D Thesis, Essex University, United Kingdom, 1985.
13. R. B. Fisher, A. W. Fitzgibbon, D. Eggert, "Extracting Surface Patches from Complete Range Descriptions", Proc. Int. Conf. on Recent Advances in 3-D Digital Imaging and Modeling, Ottawa, Canada, pp 148-155, May 1997.

Evolution of Digital Filters Using a Gate Array Model

Julian F. Miller

School of Computing, Napier University, 219 Colinton Road, Edinburgh, EH14 1DJ, UK
j.miller@dcs.napier.ac.uk

Abstract. The traditional paradigm for digital filter design is based on the concept of a linear difference equation with the output response being a weighted sum of signal samples with usually floating point coefficients. Unfortunately such a model is necessarily expensive in terms of hardware as it requires many large bit additions and multiplications. In this paper it is shown how it is possible to evolve a small rectangular array of logic gates to perform low pass FIR filtering. The circuit is evolved by assessing its response to digitised pure sine waves. The evolved circuit is demonstrated to possess nearly linear properties, which means that it is capable of filtering composite signals which it has never seen before.

1 Introduction

The difference equation is a fundamental concept employed in the construction and analysis of digital filters [8]. Formally this is represented in the following way. The output of the filter at time n, $y(n)$, may be a function of N samples of the signal $x(n-i)$ at earlier times, and may also, if feedback is present, involve earlier outputs $y(n-i)$ given by the following equation:

$$y(n) = \sum_{i=0}^{N-1} a_i x(n-i) + \sum_{i=1}^{M} b_i y(n-i) \qquad \textbf{(1)}$$

where the coefficients a_i and b_i are real valued floating-point numbers. The essential problem of filter design is the choice of $\{a_i\}$, $\{b_i\}$, N, and M, so that the filter has the desired behaviour (i.e. frequency response). In practice the coefficients $\{a_i\}$, $\{b_i\}$ are of finite precision. The practical requirements of implementing such a system in hardware consists of providing a number of shift registers, multipliers, and adders. Large bit multipliers are very costly in hardware terms. Three of the most important factors in the design of digital filters are quality of signal response, size (cost) of hardware implementation, and speed of operation. There are many traditional approaches which have been developed to address these issues [8]. In particular one popular method for reducing the complexity of implementation is to restrict the filter coefficients to integer coefficients, see [4] and references therein. Recently, researchers have started to explore the application of evolutionary algorithms to filter design [1] [2] [3] [5] [6] [15] [17] [18] [19]. The essential idea employed by most of these authors is to use an evolutionary algorithm to optimise the filter coefficients.

R. Poli et al. (Eds.): EvoIASP'99 and EuroEcTel'99, LNCS 1596, pp. 17-30, 1999.

This may be in combination with finite wordlength analysis [1] [6] for IIR filter design, or it may be in an adaptive context [5][18]. Other workers have employed evolutionary algorithms to optimise coefficients together with add and shift operations in so-called multiplier-less designs [15] [17] [19]. In [3] a genetic algorithm was used to design an efficient non-linear filter for signal noise reduction by finding a suitable positive boolean function (PBF). The PBF could be represented as a boolean sum of products, involving AND gates and OR gates.

The main idea of the work presented in this paper is to explore for the first time at a logic gate level whether it is possible to evolve networks of logic gates to carry out filtering tasks. This is an interesting thing to do for two main reasons. Firstly to explore the concept of digital filtering in a space of possibilities which is considerably larger and richer than the traditional human, top-down, difference equation method. Secondly to see how effective a microscopic number of logic gates might be in a filtering task. The pioneering concept of gate-level evolution of digital functions was developed in [7]. In [13] the authors generalised the concept of gate-level evolution to the so-called functional level, and they showed how it was possible to carry out adaptive equalisation on a communications channel with superior bit error rates to the conventional least mean squares method. Their method was not rigidly fixed to be linear in operation, it could be carried out very quickly, and relatively inexpensively in hardware. These authors believed that it would not be possible to achieve real-world performance using a gate-level approach. One of the objectives of the work presented here is to show that that the possibilities afforded by gate-level evolution have been left largely unexplored, and that there remains much fundamental work to be done at this level. An additional motivation for attempting this work is the enormous potential for new knowledge discovery afforded by the simple nature of logic functions. In other words, can new principles be extracted from gate-level evolution which can inspire and contribute to new methodological paradigms? There are of course enormous questions that need to be addressed if such a filtering method is to become practicable. Foremost among these would be the question of linearity. If a gate array is to be trained to carry out a filtering task then can this be done in such a way that composite signals, which can be represented as weighted sums of sine waves, will also be filtered? This would imply that the circuit at least be weakly linear. The findings presented in this paper are encouraging in this regard, as in section 4 it is shown that the evolved gate arrays do appear to be quasi-linear.

The actual method employed here to evolve a gate array (section 2) is developed from earlier work in [9][10][11]12] and has some similarity to a method called Parallel Distributed Genetic Programming (PDGP) [14]. In earlier work [10][11][12], the objective was to synthesise an entire truth table. This becomes increasingly time consuming and difficult as the number of inputs grow. It is obvious that attempting to evolve truth tables of larger sizes will not be feasible. It was argued in [9] that the real applications for gate-array evolution probably lie in real number mapping problems, where the digitised real numbers are presented to a circuit and a digitised real number output is desired. In such a scenario the number of input conditions is determined by the problem and is not necessarily an exponential function of the number of inputs. Such a scenario is ideally furnished by the digital filtering task. In this paper only a simple low pass FIR filter is considered. The details of this are explained in section 3. In section 4 the evolved filtering characteristics of the gate array are examined,

including some results which show the quasi-linear behaviour. These are discussed in section 5, and conclusions are given in section 6.

2 Gate-Level Evolution of Digital Circuits

The chromosome representation used is best explained with a simple example. In Fig. 1 is shown a small gate array consisting of four logic cells. The logic cells in this case have functions XOR, AND, or MUX (multiplexer). The gate array implements the one-bit adder (with carry-in). The circuit in question actually arose in an earlier experiment reported elsewhere [12] and is quite novel in its own right. A, B, and Cin denote the primary inputs. Cout and Sum are the output bits of the adder. Each cell is assumed to possess three input connections. If the cell function does not require inputs then the corresponding genes are ignored. For example the upper right cell (output 5) below has input connections 3, 2, 1. Thus, the first input is connected to the output of the cell with output label 3 (upper left), the second input is connected to the primary input Cin, and the third input is connected to primary input B. The function of each cell is expressed as the fourth gene associated with each cell. The primary outputs of the gate array are also expressed as connections. For example Cout is connected to the output of the cell with output label 6. The gate array is envisaged as being divided into vertical columns of cells and the representation is so constrained that columns of cells may only have their inputs connected to connection points on their left. This ensures the feed-forward nature of the circuit and removes any time dependent behaviour. Actually the connectivity is further constrained by the presence of a parameter denoted *l*, which dictates the number of columns on the left (including the primary inputs at column zero) to which the inputs of cells in column *l* may be connected. The purpose of this is to constrain the fan-out of signals and thereby improve the ease with which the circuit may be routed when it is physically implemented.

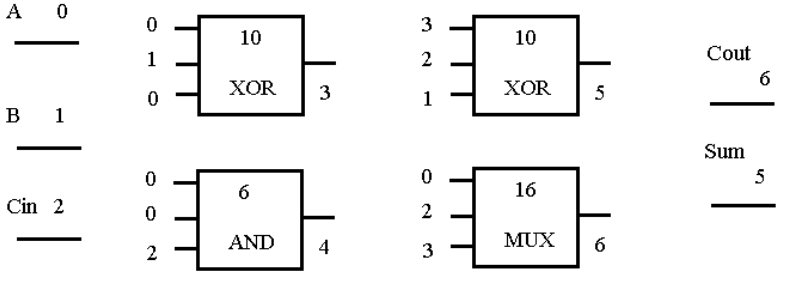


Fig. 1. One-bit adder (with carry-in) implemented as a feed-forward gate array

The chromosome representing the gate array shown in Fig. 1 is given below:

0 1 0 **10** 0 0 2 **6** 3 2 1 **10** 0 2 3 **16** 6 5

where the emboldened integers are the cell functions. The allowed cell functions can be chosen to be any subset of those shown in Table 1, where ab implies a AND b, a indicates NOT a, ^ represents the exclusive-OR operation and | the OR operation.

Table 1. Allowed gate functions

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	1	a	b	a	b	ab	a$\overline{b}$	$\overline{a}$b	$\overline{ab}$	a^b	a^$\overline{b}$	a|b	a|$\overline{b}$	$\overline{a}$|b	$\overline{a\|b}$	a$\overline{c}$|bc	a$\overline{c}$|$\overline{b}$c	$\overline{a\|b}$ a$\overline{c}$|bc	$\overline{a\|b}$ a$\overline{c}$|$\overline{b}$c	a^(bc)

Functions 16-19 describe various multiplexers and 20 describes a Reed-Muller ULM. The last five functions prove to be very effective components in assisting the evolutionary process, this is probably due to their flexibility in that they are all universal logic modules and allow the synthesis of any logic function of one or two variables. The genetic algorithm employed random mutation which respected the feed-forward nature of the circuits and also the different alphabets associated with connections and functions. Uniform crossover was employed with a 50% genetic exchange. Elitism was always used as it is markedly beneficial [11]. A probabilistic tournament selection method (size 2) was used in which the winner of the tournament was selected with a certain probability (between 0.5 and 1.0).

3 Evolving a Filter Response with a Gate Array

The incoming analogue signals which are to be processed by the gate array are sampled at frequency f, with sampling period p. Thus the number of samples used, s, is given by $s=fp$. The samples are digitised and represented by a wordlength of r bits. In a FIR filter of order n one therefore must collect nr bits at each sampling time. These nr bits for the s samples are collected and represent the input conditions to the gate array. For each nr input bits the gate array must produce r output bits. In this way a set of input-output conditions are defined. When s samples have been collected the discrete fast fourier transform (DFFT) is taken. A program, which was freely available in [8] was used to do this. In this way the frequency characteristics of the evolving gate array can be assessed for each input signal. The input signals chosen were pure sine waves with zero phase. They had frequencies which were integral multiples of the fundamental f_1 $(1/p)$ up to the Nyquist frequency, f_n (half sampling frequency) minus 1. The sine waves were translated by the addition of a d.c. component so that they assumed only positive values, this removed the need for two's complement number representation. One can envisage this more clearly by noting that the fundamental corresponds to a single exact sine cycle fitting into the sampling window. The entire arrangement is shown in Fig. 2. In this figure an input sine wave is shown on the left which is digitised to binary numbers with wordlength 4 and filter order 2. An entire history of samples are collected for each sine wave. These are the input conditions presented to the gate array. On the right of the gate array is shown the outputs of wordlength equal to 4 bits. The desired filter response is characterised by a low pass cutoff point f_p. To evaluate the fitness of a chromosome each digitised sine wave with

frequency f is presented to the gate array and the DFFT of the output response is calculated.

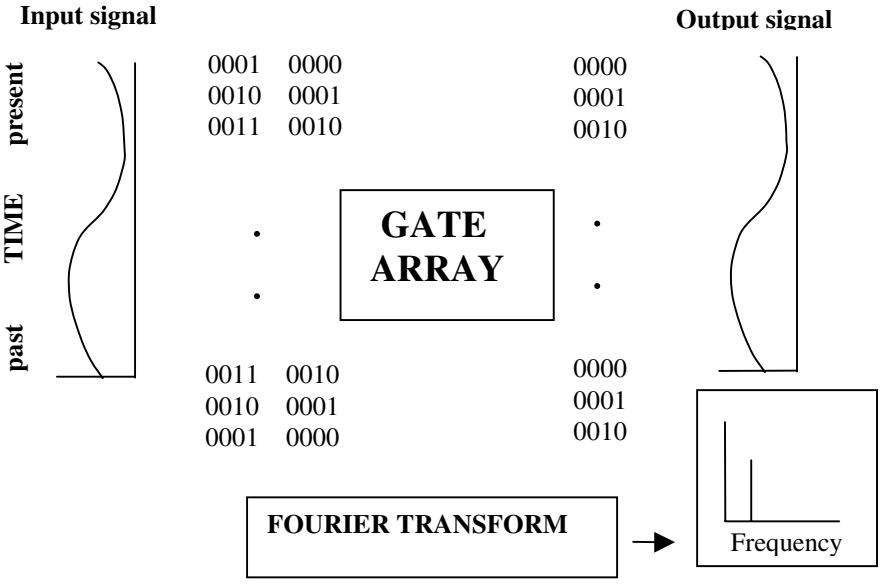


Fig. 2. The training scenario for evolving a gate array with filtering properties

The power in the frequency domain *W(f)*, defined as the modulus of the output response in the complex frequency domain, is normalised by dividing by the maximum power associated with the DFFT of a pure sine wave. The d.c. component of the output is ignored. The fitness x_i of the gate array for a sine wave of frequency f_i is calculated in the following way:

$$x_i = W(f_i) - \max\{W(f_j), \forall j : j \neq i, f_1 \leq f_j \leq f_n - 1\}, \; f \leq f_p \qquad \textbf{(2)}$$

$$x_i = 1.0 - \max\{W(f_j), \forall j : f_1 \leq f_j \leq f_n - 1\}, \; f > f_p$$

The total fitness x associated with a given chromosome is then given by the sum of the components x_i for all frequencies up to f_n*-1*. Thus if the maximum power for a sine wave with frequency greater than the cutoff point is zero, the fitness contribution will be 1, if the maximum power is not zero then the fitness component will be lower. This definition of fitness means that one is trying to suppress sine waves with frequencies above the cutoff point, and trying to enhance only the *pure* frequencies below the cutoff point. Thus the degree to which the actual shape of the outgoing sine wave conforms to a pure sine wave is being rewarded for frequencies below the cutoff point.

4 Results

The experimental parameters for this paper are given below, the nominal sampling period p was chosen to be 1 for convenience. Thus the sampling frequency f equals the number of samples s.

- number of samples s=128, wordlength r =8, filter order = 4,
- normalised passband cutoff = 0.08 (10.24 un-normalised)
- population_size is 10, breeding rate is 100%, mutation probability is 0.005
- num_generations is 5000, number of runs is 2, elitism,
- tournament selection (size 2) acceptance probability is 0.7
- number of rows in gate array is 9, number of columns in gate array is 9
- connectivity parameter $l = 9$. The only gate type allowed was the multiplexer (type 16).

The results shown in this paper are for the best of two runs of the genetic algorithm under the above conditions. Investigation of the most suitable parameter settings lies outside the scope of this paper. A small population size was chosen purely for speed of execution. The frequency response of the evolved filter is shown in Fig. 3.

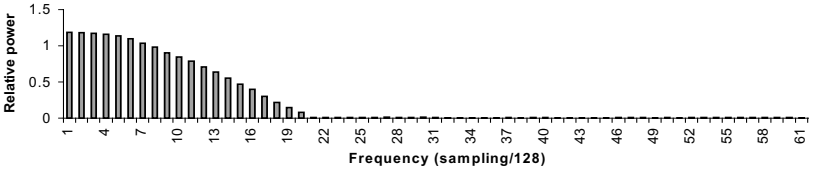


Fig. 3. Frequency response of the evolved filter

4.1 Filter Response to Pure Sine Signals in the Passband

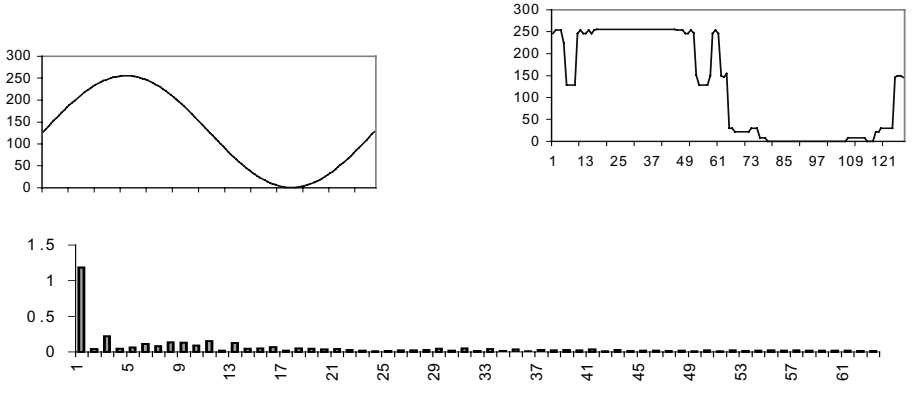


Fig. 4. Incident signal f_1 , output response and frequency response

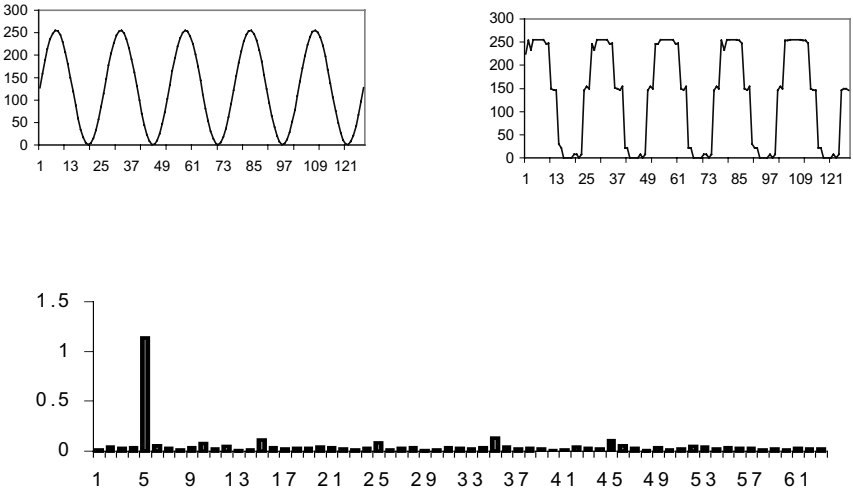


Fig. 5. Incident signal f_5 , output response and frequency response

4.2 Filter Response to Pure Sine Signals in the Stopband

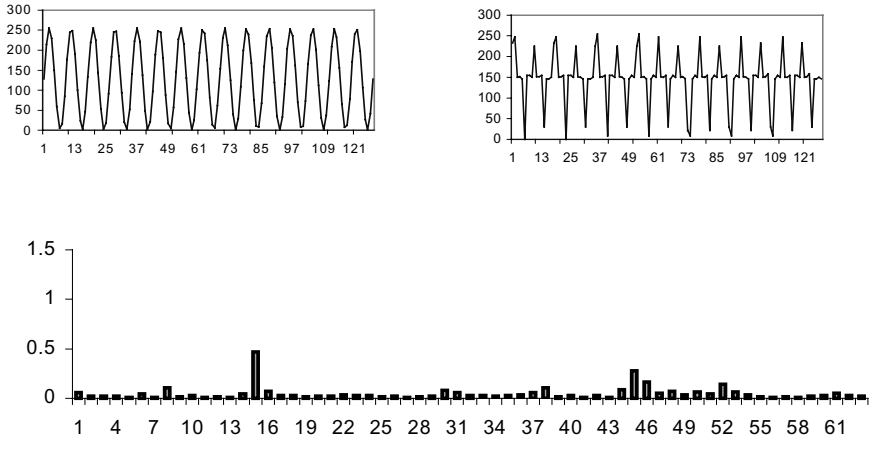


Fig. 6. Incident signal f_{15} , output response and frequency response

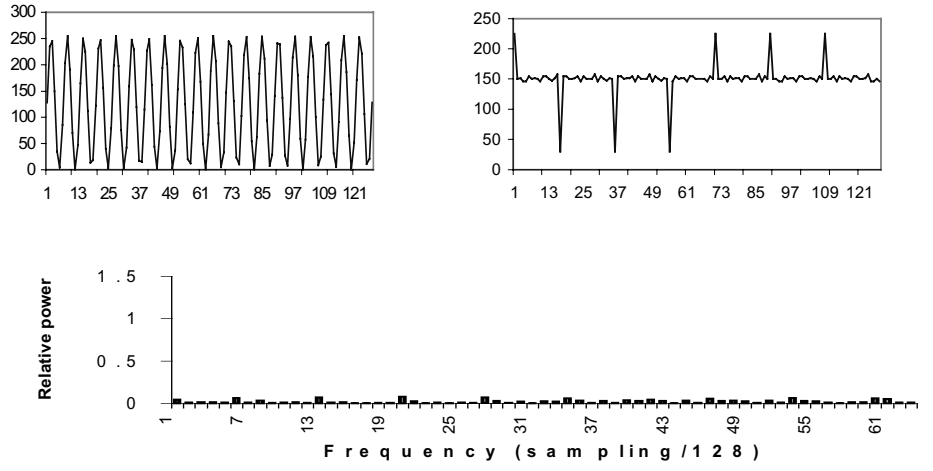


Fig. 7. Incident signal f_{20} , output response and frequency response

4.3 Filter Response to Signals Which are a Sum of Two Sine Waves

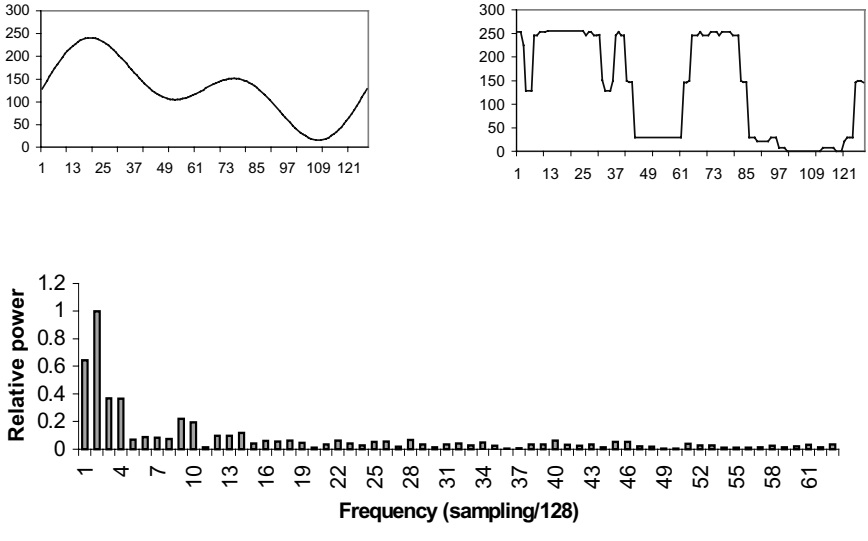


Fig. 8. Incident signal $0.5(f_1 + f_2)$, output response and corresponding frequency response

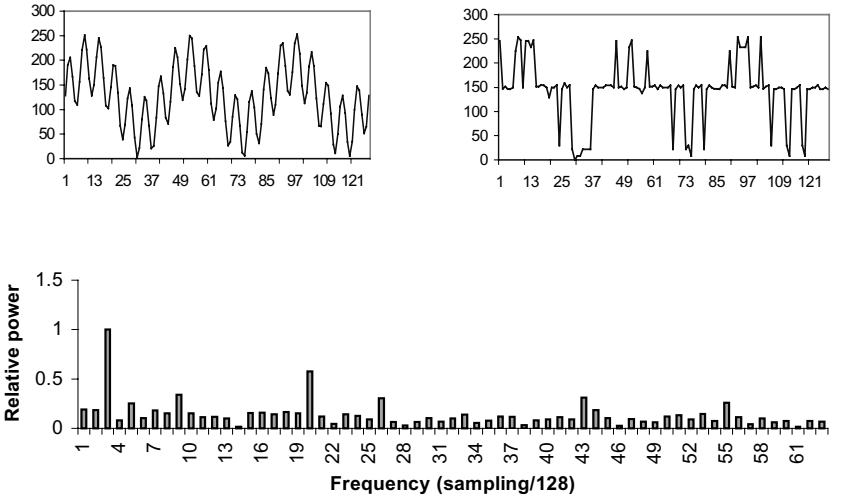


Fig. 9. Incident signal $0.5(f_3 + f_{20})$, output response and corresponding frequency response

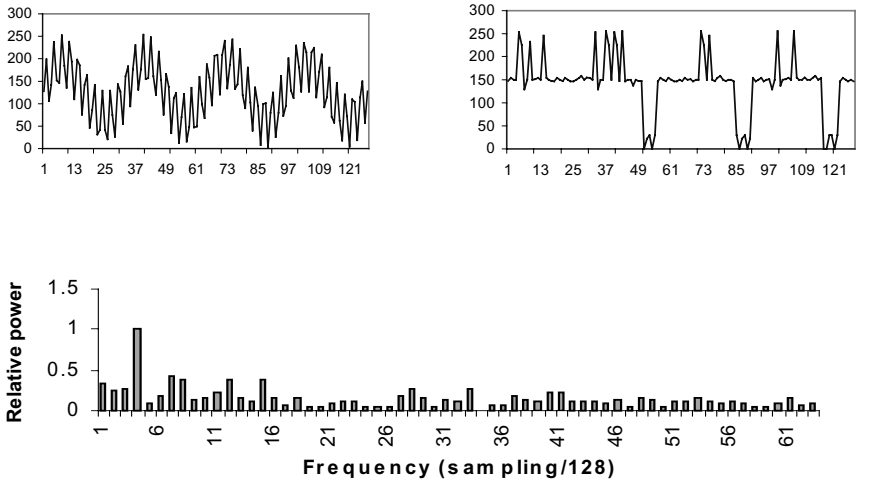


Fig. 10. Incident signal $0.5(f_4 + f_{40})$, output response and corresponding frequency response

4.4 Filter Response to Signals Which are a Sum of Three Sine Waves

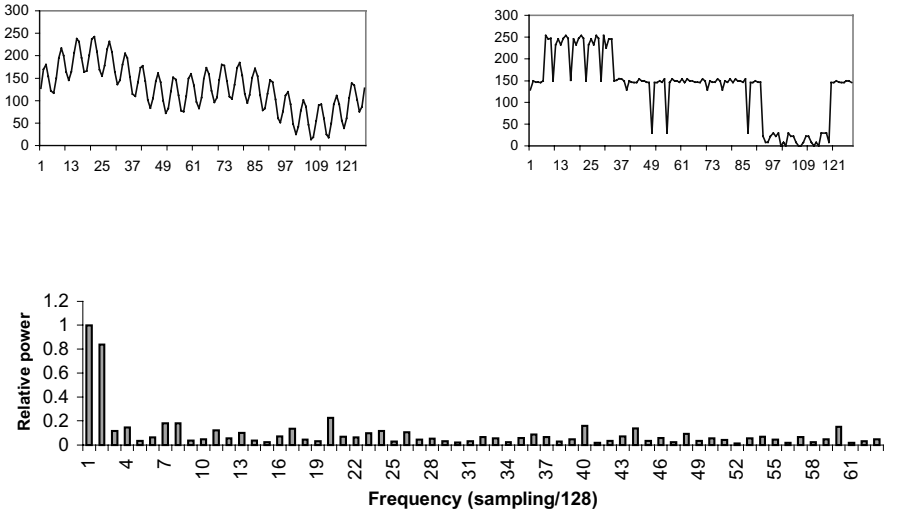


Fig. 11. Incident signal $0.33(f_1 + f_2 + f_{20})$, output response and corresponding frequency response

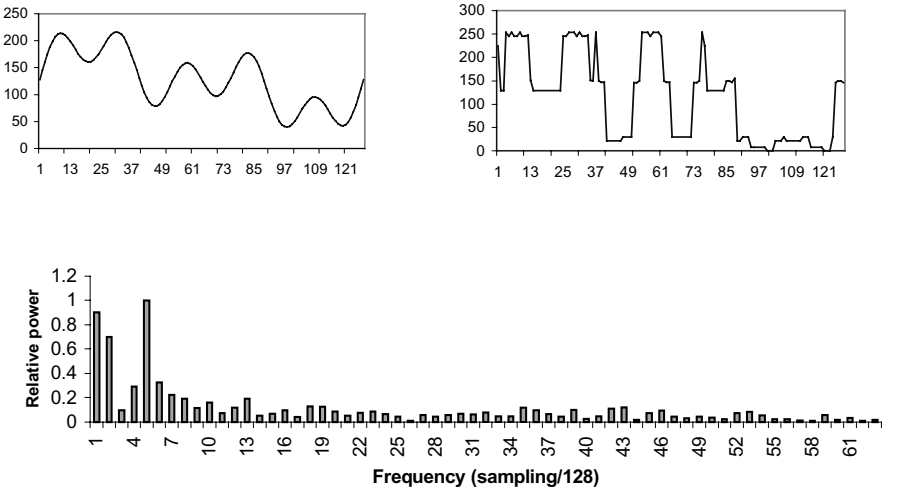


Fig. 12. Incident signal $0.33(f_1 + f_2 + f_5)$, output response and corresponding frequency response

5 Discussion of Results

5.1 Filter Characteristics

In Fig. 3, the filter response is shown, there is still a noticeable tail which extends past the cutoff frequency of 10 f_1 from 11f_1 to about 20 f_1. However it should be noted that the gate array is tiny and the work is still at a preliminary phase. The quality of the frequency response in meeting the specification is encouraging. In section 4.1 are shown the output responses of the filter to incident pure sine signals and also the output response in the frequency domain. Sine waves in the passband are being passed with little attenuation, however it can be seen especially in the case of the lowest frequency sine wave (Fig. 4) that there is the largest distortion of the signal. In Figures 6 and 7 the incident sine signals have frequencies in the stopband so they should be highly attenuated. One can see that there is a marked drop in signal amplitude as the signal is converging to a d.c. component. As the frequency of the incident signals are increased the off d.c. spikes become more and more sparse. Actually there is something a little puzzling here as the fitness function is designed to suppress frequencies in the stopband with uniform probability so that the attenuation of those frequencies should show no frequency dependent behaviour. The reason for this is not currently understood but it may be due to a frequency dependent distortion in the incident sine signals. In Figs. 8-12 are shown the output responses of the filter to various sums of sine waves. All these signals have never been seen by the filter before. In Fig. 8 it can be seen that the filter is exaggerating the changes in amplitude of the incident signal. The frequency response shows the dominant frequencies to be the same as the incident. The filter is displaying a nearly or quasi-linear response. In Figs. 9 and 10 the higher frequency lies in the stopband thus for ideal filter behaviour one would expect the higher frequency component to be highly attenuated. The evolved filter appears to be doing this as it is responding to the slower changes in the signal. This is confirmed by the frequency responses. In Figs. 10 and 11 more complex signals were presented to the filter. These were sums of three sine waves. In the first case (Fig. 11) two components were in the passband. Again the filter is still trying to follow the slower changes and the frequency response is dominated by the lower frequencies. In Fig. 12 all the frequencies lay in the passband, again it is seen that the filter is trying to follow all the changes in the incident signal. However once again it is exaggerating the changes.

5.2 Hardware Requirements and Speed of Evolved Filter Compared with Conventional

When the evolved filter circuit was analysed it was found to require 29 multiplexers (equivalent to 87 two-input gates). In addition the filter would produce the filtered response very quickly as one only has to wait for the signals to propagate through the gate-array. A conventional filter of order 4 and wordlength 8 would require at least an eight-bit adder and multiplier as well as registers to store the coefficients. A conventional cellular adder and multiplier of this size would require n^2 AND gates and $n(n-1)$ full adders (where $n=8$). Thus it would require 344 two-input gates. The output would be delayed by a number of clock cycles to accumulate the response (see equation 1).

6 Conclusions

In this paper it has been shown that it is possible to evolve filtering characteristics with a gate-array containing very few components. The gate-array filter is produced without many of the conventional assumptions in that it does not employ coefficients or any explicit arithmetic operations. The evolved filter has a quasi-linear response that has emerged naturally. There is currently no mathematical framework for understanding how to design filters at this level. It is felt that the results presented here may encourage some thinking about a mathematical underpinning of this. There is still an enormous amount of further investigation to be undertaken. The work raises almost as many questions as it answers. Why is the evolved filter quasi-linear? Can one evolve it in such a way as to enhance its linearity? Would this require greater gate resources? What would the filtering action of cascades of these smaller filters be like? How would the filter response to changes in phase of the incident sine waves? It is felt that this work once gain demonstrates the enormous capacity of a few gates to display complex behaviours, a fact which has become evident in much work in the field of evolvable hardware [16].

Acknowledgements

Especial thanks to Gary Robertson for many helpful discussions, and also, thanks to George Rae, Mohammed Yaminysharif, and Jay Hoy of the EE&CE Department, Napier University.

References

1. Arslan T., and Horrocks D. H., “A Genetic Algorithm for the Design of Finite Word Length Arbitrary Response Cascaded IIR Digital Filters”, : Proceedings of the First IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA’95), No. 414, IEE, London, pp. 276-281, 1995.
2. Chellapilla K., Fogel D. B., and Rao S. S., “Gaining Insight into Evolutionary Programming Through Landscape Visualization: An Investigation into IIR Filtering”, Evolutionary Programming 97, pp. 407-417, 1997.
3. Delibasis K. K., Undrill P. E., and Cameron G. G., “Genetic algorithm implementation of stack filter design for image restoration”, ”, IEE Proceeedings in Vision, Image and Signal Processing, Vol. 143, No. 3, pp. 177-183, 1996.
4. Dempster A. G., and Macleod M. D., “Use of Minimum-Adder Multiplier Blocks in FIR Digital Filters”, IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing, Vol. 42, No. 9, pp. 569-577, 1995
5. Esparcia Alcazar A. I., and Sharman K. C., “Some Applications of Genetic Programming in Digital Signal Processing”, in Late Breaking Papers at Genetic Programming 96, Stanford, pp. 24-31, 1996.

6. Harris S. P., and Ifeachor E. C., "Automating IIR filter design by genetic algorithm", Proceedings of the First IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA'95), No. 414, IEE, London, pp. 271-275, 1995.
7. Iba H., Iwata M., and Higuchi T., Machine Learning Approach to Gate-Level Evolvable Hardware, in Higuchi T., Iwata M., and Liu W., (Editors), Proceedings of The First International Conference on Evolvable Systems: From Biology to Hardware (ICES96), Lecture Notes in Computer Science, Vol. 1259, Springer-Verlag, Heidelberg, pp. 327 – 343, 1997.
8. Ifeachor E. C., and Jervis B. W., "Digital Signal Processing: A Practical Approach", Addison-Wesley, 1993.
9. Miller J. F., and Thomson P., "Evolving Digital Electronic Circuits for Real-Valued Function Generation using a Genetic Algorithm". Koza, John R. et al, (Editors). Genetic Programming: Proceedings of the Third Annual Conference, July 22-25, 1998, University of Wisconsin, Madison, Wisconsin. San Francisco, CA: Morgan Kaufmann pp. 863-868, 1998.
10. Miller J. F., Thomson P., "Aspects of Digital Evolution: Evolvability and Architecture", in (Editors), Proceedings of The Fifth International Conference on Parallel Problem Solving from Nature (PPSNV), Lecture Notes in Computer Science, Vol. 1498, Springer-Verlag, Heidelberg, pp. 927-936, 1998.
11. Miller J. F., Thomson P., "Aspects of Digital Evolution: Geometry and Learning", in Sipper M., Mange D., and Perez-Uribe A. (Editors), *Proceedings of* The Second International Conference on Evolvable Systems: From Biology to Hardware (ICES98), Lecture Notes in Computer Science, Vol. 1478, Springer-Verlag, Heidelberg, pp. 25-35, 1998.
12. Miller J. F., Thomson P., and Fogarty T. C., "Designing Electronic Circuits Using Evolutionary Algorithms. Arithmetic Circuits: A Case Study", in Genetic Algorithms and Evolution Strategies in Engineering and Computer Science: D. Quagliarella, J. Periaux, C. Poloni and G. Winter (eds), Wiley, 1997.
13. Murakawa M., Yoshizawa S., and Higuchi T., "Adaptive Equalisation of Digital Communication Channels Using Evolvable Hardware", in Higuchi T., Iwata M., and Liu W., (Editors), Proceedings of The First International Conference on Evolvable Systems: From Biology to Hardware (ICES96), Lecture Notes in Computer Science, Vol. 1259, Springer-Verlag, Heidelberg, pp. 379 – 389, 1996.
14. Poli R., "Evolution of graph-like programs with parallel distributed genetic programming", in Bäck T. (Editor), Genetic Algorithms: Proceedings of the Seventh International Conference, Morgan Kaufmann, pp. 346-353, 1997.
15. Redmill D. W., and Bull D. R., "Design of Low Complexity FIR Filters using Genetic Algorithms and Directed Graphs", in Proceedings of the Second IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA'97), No. 446, IEE, London, 1997.
16. Sipper M., Sanchez E., Mange D., Tomassini M., Perez-Uribe A., and Stauffer A., "A Phylogenetic, Ontogenetic, and Epigenetic View of Bio-Inspired Hardware Systems", IEEE Transactions on Evolutionary Computation, Vol. 1, No 1., pp. 83-97, 1997.

17. Sriranganathan S., Bull D. R., and Redmill D. W., “Design of 2-D Multiplierless FIR Filters using Genetic Algorithms”, Proceedings of the First IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA’95), No. 414, IEE, London, pp. 282-286, 1995.
18. Sundaralingam S., and Sharman K. C., “Genetic Evolution of Adaptive Filters”, in Proceedings of DSP, London UK, pp. 47-53, 1997.
19. Wade G., Roberts A., and Williams G., “Multiplier-less FIR filter design using a genetic algorithm”, IEE Proceedings in Vision, Image and Signal Processing, Vol. 141, No. 3, pp. 175-180, 1994.

GA Optimisation of Spatio-Temporal Grey-Scale Soft Morphological Filters with Applications in Archive Film Restoration

Neal R. Harvey[1] and Stephen Marshall[2]

[1] Astrophysice & Radiation Measurements Group, Los Alamos National Laboratory, Los Alamos, NM 87545, USA
[2] Department of Electronic & Electrical Engineering University of Strathclyde, Glasgow, G1 1XW, UK

Abstract. A technique is described for the optimisation of spatio-temporal (3-D) grey-scale soft morphological filters for applications in archive film restoration. The optimisation is undertaken using genetic algorithms. By employing filters which incorporate the temporal dimension, this technique extends and improves upon previously described techniques which were based purely in the spatial (2-D) domain. Examples of applying the technique to real-world film restoration problems are shown.

1 Introduction

There has been a growing interest in recent years in the area of archive film restoration. This has no doubt come about in part due to the emergence of digital television broadcasting and the growth in video sales. In order to satisfy demand, it is becoming more attractive to offer much of the available archive material. However, a great deal of the archive material has suffered some form of corruption and requires restoration in order to be of a sufficient quality for resale or broadcast. This paper addresses the particular problem associated with archive film material, known as *film dirt*. This occurs when particles get caught in the film transport mechanism and damage the film, causing loss of information. This damage manifests as "blotches" of random size, shape and intensity.

Fig. 1 shows an example of an image from a sequence of images corrupted with film dirt.

Here we describe a global filtering strategy for the restoration of image sequences corrupted with film dirt, using 3-D grey-scale soft morphological filters and a method for the optimisation of the filter parameters using a genetic algorithm.

2 Soft Morphological Filters

Soft morphological filters are a relatively recently introduced class of non-linear filters [1,2]. Their original definition was related to the class of (standard/structural) morphological filters (discrete flat morphological filters), but they have

R. Poli et al. (Eds.): EvoIASP'99 and EuroEcTel'99, LNCS 1596, pp. 31–45, 1999.

Fig. 1. Image corrupted with film dirt

since been extended to the grey-scale (function processing) case [3]. The idea behind soft morphological filters is to slightly relax the standard definitions of morphological filters in such a way as to achieve robustness whilst retaining most of the desirable properties of standard morphological filters. Whereas standard morphological operations are based on local maximum and minimum operations, in soft morphological operations these operations are replaced by more general weighted order statistics. The key idea of soft morphological operations is that the structuring element is divided into two parts: the *hard centre* which behaves like the standard structuring element and the *soft boundary*, where maximum and minimum are replaced by other order statistics. This makes the operations behave less rigidly in noisy conditions and makes the operations more tolerant to small variations in the shapes of the objects in the filtered image.

Just as the fundamental standard morphological operations are dilation and erosion, the fundamental soft morphological operations are *soft dilation* and *soft erosion*. In a manner similar to that of standard morphological operations, the secondary soft morphological operations of *soft opening* and *soft closing* and the tertiary soft morphological operations of *soft open-closing* and *soft close-opening* can be defined.

Before proceeding to the definitions of the soft morphological operations, some other concepts need to be defined:

The *Structuring System* $[b, a, r]$ consists of three parameters; functions a and b, having supports A and B, respectively ($A \subset B$) and a natural number, r, satisfying $1 \leq r \leq |B|$, where $|B|$ is the cardinality of B. Function b is called the *structuring function*, a its (hard) *centre* (A the support of its (hard) centre), $b \backslash a$ its (soft) *boundary* ($B \backslash A$, the support of its (soft) boundary) and r the *order index of its centre* or the *repetition parameter*.

2.1 Fundamental Grey-Scale Soft Morphological Operations

Grey-scale *soft dilation* of a signal f by the structuring system $[b, a, r]$ is denoted by $f \oplus [b, a, r]$ and is defined by:

$$\begin{aligned} f \oplus [b, a, r](x) = \text{the } r^{th}\text{largest value of the multiset} \\ \{r \diamond (f(x - \alpha) + a(\alpha))\} \bigcup \{f(x - \beta) + b(\beta)\} \\ \text{where } \alpha \in A, \beta \in B \backslash A \end{aligned} \tag{1}$$

Grey-scale *soft erosion* of a signal f by the structuring system $[b, a, r]$ is denoted by $f \ominus [b, a, r]$ and is defined by:

$$\begin{aligned} f \ominus [b, a, r](x) = \text{the } r^{th}\text{smallest value of the multiset} \\ \{r \diamond (f(x + \alpha) - a(\alpha))\} \bigcup \{f(x + \beta) - b(\beta)\} \\ \text{where } \alpha \in A, \beta \in B \backslash A \end{aligned} \tag{2}$$

As an extreme case, grey-scale soft morphological operations by the structuring system $[b, a, r]$ reduce to the equivalent standard grey-scale morphological operations by the function b if $r = 1$, or, alternatively, if $A = B$. If $r > |B \backslash A|$, grey-scale soft morphological operations by the structuring system $[b, a, r]$ reduce to the equivalent grey-scale standard morphological operations by the structuring function a.

2.2 Secondary Grey-Scale Soft Morphological Operations

Grey-scale *soft opening* by the structuring system $[b, a, r]$ is defined as grey-scale soft erosion by the structuring system $[b, a, r]$ followed by grey-scale soft dilation by the structuring system $[b^s, a^s, r]$ of the soft eroded result.

Grey-scale *soft opening* of f by $[b, a, r]$ is denoted by $f_{[b,a,r]}$ and is defined by:

$$f_{[b,a,r]}(x) = (f \ominus [b, a, r]) \oplus [b^s, a^s, r](x) \tag{3}$$

Grey-scale *soft closing* by the structuring system $[b, a, r]$ is defined as grey-scale soft dilation by the structuring system $[b, a, r]$ followed by grey-scale soft erosion by the structuring system $[b^s, a^s, r]$ of the soft dilated result.

Grey-scale *soft closing* of f by $[b, a, r]$ is denoted by $f^{[b,a,r]}$ and is defined by:

$$f^{[b,a,r]}(x) = (f \oplus [b, a, r]) \ominus [b^s, a^s, r](x) \tag{4}$$

Note that the symmetric function of f, having support F is denoted as f^s and defined as:

$$f^s = \{-f(-x) : x \in F\} \tag{5}$$

i.e. the symmetric function is also known as the reflection of the function, and is accomplished by reflecting the function through the vertical axis and through the horizontal axis. This is equivalent to rotating the graph of the function by 180^o about the origin.

2.3 Tertiary Grey-Scale Soft Morphological Operations

Grey-Scale *soft open-closing* by the structuring system $[b, a, r]$ is defined as grey-scale soft opening by the structuring system $[b, a, r]$, followed by grey-scale soft closing of the soft opened result, using the same structuring system.

Similarly, grey-scale *soft close-opening* by the structuring system $[b, a, r]$ is defined as grey-scale soft closing by the structuring system $[b, a, r]$, followed by grey-cale soft opening of the soft closed result, using the same structuring system.

3 Optimisation of Soft Morphological Filters

Several methods have been described for the optimisation of soft morphological filters. Huttunen et al [4] and Kuosmanen et al [5] describe methods for the optimal choice of (2-D) flat (function-set processing) soft morphological structuring system. These methods do not, however, optimise the choice of soft morphological operation. Harvey [6,7] described a method for the optimisation of (2-D/spatial) grey-scale soft morphological filters which is able to optimise not only the structuring system, but also the choice of soft morphological operation. In [8] this GA optimisation technique was applied to the restoration of film material. Purely spatial (2-D) filtering techniques obviously do not make use of the available temporal information available. The temporal characteristics of the corruption in image sequences containing film dirt (i.e. non-time correlated, temporally impulsive) can provide useful information. In this paper the 2-D (spatial) method of film dirt removal is improved and extended to the 3-D (spatio-temporal) case in order to make use of this valuable temporal information.

3.1 Soft Morphological Filter Parameters

In searching for the optimal choice of soft morphological filter the following parameters have to be considered:

- Size and shape of structuring system's hard centre
- Size and shape of structuring system's soft boundary
- Rank selection parameter
- Choice of soft morphological operations

What follows is a description of how these parameters are incorporated into a genetic algorithm optimisation strategy. The parameters are encoded and mapped to a "chromosome".

Overall Structuring Function. Limits as to the dimensions of the overall structuring function are set (i.e. the spatial, temporal and grey-scale dimensions) and the optimisation process is allowed to search for any size and shape of overall structuring function within this 4-D hypercube "envelope". If the overall spatio-temporal dimensions of the structuring function are fixed, it may be that for a particular structuring function, not all positions within this region are in the actual region of support. In order to take this into account in the GA optimisation, it is necessary for positions outside the structuring function's region of support, but within the overall search envelope, i.e. *don't care* positions, to be distinguishable. A suitable code, therefore, would be one which includes a unique representation for those "null" positions. An example of such an encoding scheme is shown in Table 1. A "*" refers to a position outside the structuring function's region of support. In this example, grey-scale values ranging from 0

Table 1. Example of code for coding positions within and outside the overall structuring function's region of support

Code	Grey-scale value
0	*
1	0
2	1
3	2
4	3
5	4

to 4, as well as positions outside the structuring function's region of support are able to be encoded.

The overall structuring function is divided into two distinct regions: the hard centre and soft boundary. Hence, some method of distinguishing these two regions must be incorporated into the coding.

Hard Centre. A binary string, having a length equivalent to the cardinality of the structuring function's overall support "envelope", is used to flag those positions within the structuring function's support which are in the hard centre. Positions in this string which contain a one are positions within the structuring function which are in the hard centre. After forming each new individual, the hard centre flags are checked against the structuring function portion of the chromosome. If any of the positions within the structuring function portion of the chromosome are coded as being outside its region of support, i.e. "null" positions, then a check is made to ensure that the corresponding position within the hard centre flag string has a zero and is changed as necessary.

Soft Boundary. Any positions within the overall structuring function's support not coded as a null position and not having a one in the corresponding hard centre flag portion of the chromosome are considered to be in the soft boundary of the structuring function.

Rank selection parameter. From the definition of soft morphological operations, we know that for a structuring function, b having a support B, the rank selection parameter, r, has to lie somewhere in the range $1 \leq r \leq |B|$. In other words, we can state that the rank selection parameter is related in some way to the cardinality of the structuring function. So, in order to code the repetition parameter we can have a binary string, the length of which is equal to the overall size of the structuring function (i.e. the pre-set outer limits of the structuring function's support). This binary string is then used to flag whether a position within the structuring function's support contributes to the repetition parameter - a one signifying that it does. To ensure consistency, a check has to be made, after forming each new individual, that those positions flagged as

contributing to the repetition parameter are only those positions coded as being within the structuring function's support. If any positions in the repetition parameter binary string are flagged with a one, but the corresponding positions in the structuring function are coded as being outside the structuring function's support, these flags have to be altered to ensure that they are set to zero. In this way the binary string can only code values lying within the allowable range.

Choice and Sequence of Soft Morphological Operations. When considering the soft morphological operations in the context of design of soft morphological filters, one has to consider the search space within which the GA will operate. Here we will, essentially, be seeking to limit our search to the set of fundamental (primary), secondary and tertiary soft morphological operations, i.e. to the set which includes { soft erosion, soft dilation, soft opening, soft closing, soft open-closing and soft close-opening }. Each member of this set can be defined as some combination of the fundamental soft morphological operations. Therefore, for a coding scheme to be able to encode this set of soft morphological operations, we can state that there are two basic decisions to be made;

- The set of individual soft morphological operators from which to choose.
- The maximum number of soft morphological operations in the sequence.

So, to be able to code the primary, secondary and tertiary soft morphological operations, the set of soft morphological operators necessary is { soft dilation, soft erosion} and the sequence length required is four, i.e. the longest sequence of operations will be for the tertiary operations of soft open-closing and soft close-opening, which can be defined in terms of the fundamental (primary) operations as a sequence of four separate primary operations: soft open-closing can be defined as soft erode, soft dilate, soft dilate, soft erode and soft close-opening as soft dilate, soft erode, soft erode, soft dilate.

In order that the GA should be able to perform optimisation over the entire search space, it is necessary to include the *do-nothing*, or *identity* operation to the set of soft morphological operations. This is due to the fact that the length of the sequence of soft morphological operations is fixed in the genetic algorithm, but it is desirable to include in the search space all the combinations of soft morphological operations from the simple soft erosion and soft dilation, through the soft close and soft open filters, to the soft open-close and soft close-open filters. Hence, if one were to omit the do-nothing (or identity) operation, then the search space would only include those filters which contain exactly four operations, each chosen from the set {soft erode, soft dilate} and the search space would be severely restricted. Each member of the set of soft morphological operations to be considered in the GA optimisation can be coded as a single integer and these integers can then be mapped to appropriate positions in a chromosome. Table 2 shows an example of codings for the set {soft dilation, soft erosion, do-nothing}.

It is necessary to ensure that each possible sequence of operations is unique, i.e. so that no combinations of operations in a sequences can be coded in more

than one way, since some combinations of filter sequences are equivalent, e.g. *soft erode, soft dilate, do-nothing, do-nothing* and *do-nothing, soft erode, do-nothing, soft dilate*. This is accomplished by, after forming each new individual, checking the sequence of operations portion of the chromosome and ensuring that any do-nothing operations are moved to the end of the sequence.

Table 2. Example of a code for soft morphological operations, if choice is from {soft erosion, soft dilation, do-nothing}

Code	Soft Morphological Operator
0	Do-Nothing
1	Soft Erosion
2	Soft Dilation

Combining the coded structuring function's hard centre and soft boundary, sequence of soft morphological operations and rank selection parameter. To form the complete chromosome, the separate strings containing the encoded structuring function, hard centre flags, sequence of soft morphological operations and rank selection parameter are simply concatenated.

The size of the search space is therefore fixed. The overall dimensions of the structuring functions - the maximum size of its region of support (and hence the support of the hard centre and soft boundary and the range of possible rank selection parameters), the maximum grey-level values and the maximum length of soft morphological operations, together with the choice of rank-order morphological operations are all set beforehand. The GA will be capable of searching for any 3-D grey-level soft morphological filter which is a combination of four operations from the set {soft erode, soft dilate, do-nothing}, which will use a structuring function (hard centre and soft boundary) and rank selection parameter chosen from all the possible variations within the overall region of support and maximum grey-level value. This search space encompasses (3-D) spatio-temporal, 2-D (purely spatial) and 1-D (purely temporal) soft morphological filters. In addition, the class of soft morphological filters encompasses several other classes of non-linear filters including standard morphological filters and rank-order filters.

3.2 Fitness Function

In order to provide each individual, representing a particular set of filter parameters with a fitness value, it is necessary to have some method of ascertaining the filter's performance, with respect to some criterion. Criteria commonly used in image processing optimisation problems usually involve some comparison of the filtered image with an "ideal" image, and will include a measure of the mean

absolute and/or mean squared error. In the case of film restoration, however, it is generally not possible to perform a comparison with an ideal image, as such a thing does not exist. After all, if a non-corrupted version of the film exists, why bother trying to restore a corrupted version? It is therefore necessary to base the fitness value on some objective image quality criterion which can be calculated using only the available image sequence. The objective quality criterion used in this technique is based on an objective quality measure, defined by Ramponi et al [9]. This "quality appraiser" is based on the discrimination between background and detail regions, using a measure of the local variance. A threshold is used to discriminate between pixels: those with a variance above a certain threshold are considered as detail pixels and the others are considered as background pixels. The average detail variance and background variance for an image is calculated. The measure of a filter's performance is calculated in terms of the increase in detail variance (and hence increase in sharpness) and decrease in background variance (and hence decrease in noise/corruption) that it effects on the corrupted image. Obviously, in this application, we are more interested in the decrease in background variance (and hence noise/corruption). Any increase in detail variance would be an added bonus.

3.3 Genetic Operators

The actual "genetic algorithm" itself was based upon what is often referred to as a *simple genetic algorithm*, or *SGA*, with minor modifications.

- *Selection*: Roulette wheel selection was used.
- *Crossover*: Uniform crossover was applied, with a probability of 0.75.
- *Mutation*: The mutation operator involved randomly choosing one of the possible values of an allele for a particular locus on the chromosome. Mutation was applied with a probability of 0.03.
- *Population Size*: The population size was set at 30.

4 Application to Real Image Sequences

The GA, as described above, was run, using a representative sub-set of a corrupted image sequence. This sequence consisted of 9 images of 91×132 pixels. The GA was set the task of optimising a soft morphological filter with an overall size of structuring function set at $3 \times 3 \times 3$. The best filter found after 1000 generations was then applied to the full-sized corrupted image sequence. Some examples of this original, corrupted and the corresponding filtered image sequence are shown below.

Figs. 2 to 5 show some examples of regions extracted from a sequence of images corrupted with film dirt, together with the same regions after having been filtered with the grey-scale soft morphological filter found using the GA. The upper example of the pairs is the corrupted version and the lower example is the filtered version.

Fig. 2. Region extracted from image corrupted with film dirt and the same region after filtering with the grey-scale soft morphological filter found using the GA

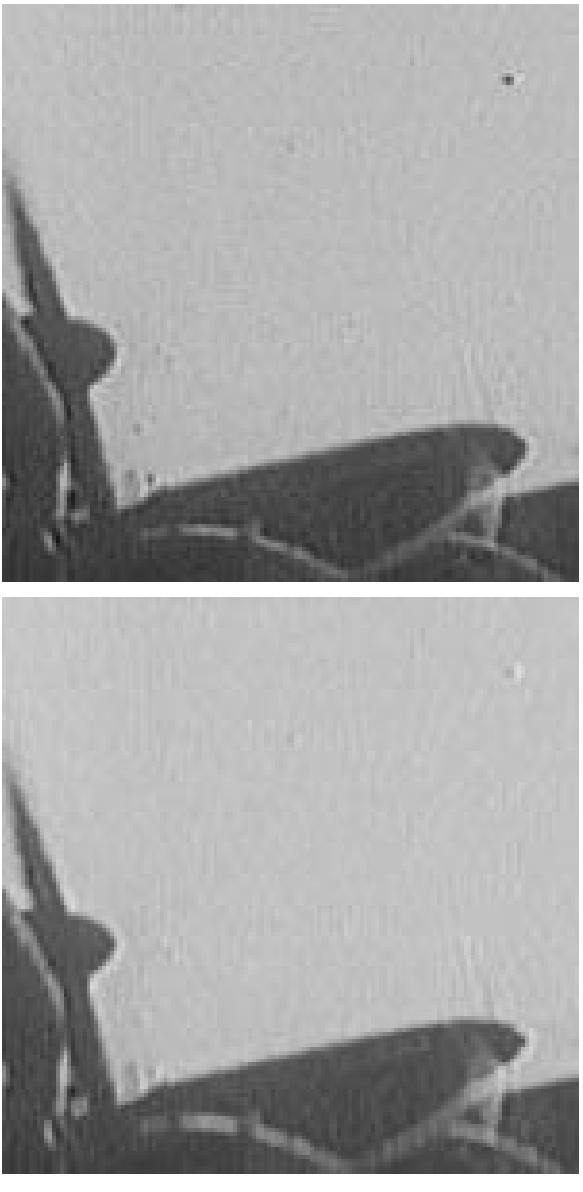


Fig. 3. Region extracted from image corrupted with film dirt and the same region after filtering with the grey-scale soft morphological filter found using the GA

Fig. 4. Region extracted from image corrupted with film dirt and the same region after filtering with the grey-scale soft morphological filter found using the GA

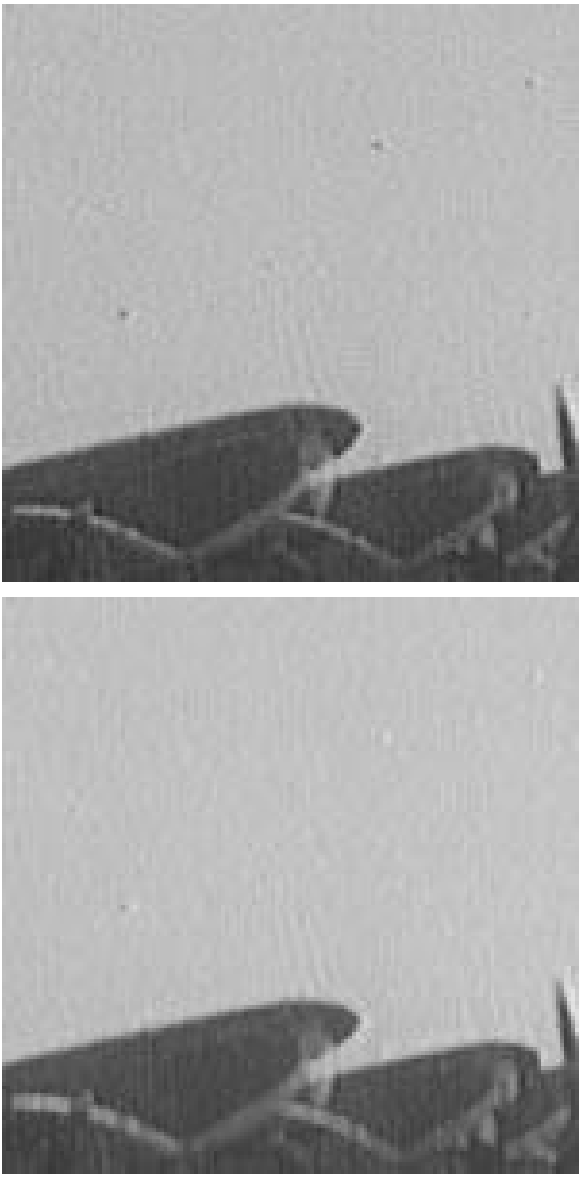


Fig. 5. Region extracted from image corrupted with film dirt and the same region after filtering with the grey-scale soft morphological filter found using the GA

Table 3. Average local background variance (BV) values for the original and filtered image regions shown in Figs. 2 to 5

Fig.	Original BV	Filtered BV
2	15.50	12.38
3	20.13	17.20
4	15.08	12.21
5	20.91	17.52

From Figs. 2 to 5 it can be seen that the filter applied to the corrupted image sequence has been able to subdue significantly the appearance of the corruption within the image without unduly affecting the image detail. This can also be seen, quantitatively, by the decrease in average local background variances, as shown in Table 3. Examples of complete, full-sized images as well as moving image sequences can be viewed at the following web page:

http://www.spd.eee.strath.ac.uk/harve/bbc_epsrc_film_dirt.html

Other restoration methods described in the literature, e.g. [10,11], perform well with resepct to the complete removal of larger film-dirt artefacts. This is due to the fact that these techniques are based upon a detect-and-remove approach where the artefacts are first detected and the entire areas containing these artefacts are completely removed and replaced with some estimate of the original data. However, these methods do not perform well when the film-dirt artefacts are difficult to detect, such as when they are small in size, or the difference between artefact and adjacent non-artefact pixels is small. The global filtering strategy described here is much better at removing these smaller and/or less pronounced artefacts. It is also simple to understand and implement. Another benefit of this method is that it may have applications in other areas of image restoration and enhancement and is not totally restricted to the single task of film dirt removal. The optimisation is carried out with respect to a "local" image quality criterion and this should allow the technique to be applied to image sequences suffering from other types of corruption/noise. In addition, the GA (the fundamental operations), and associated fitness function are not limited to the field of soft morphology and could also be applied to the optimisation of other filters.

5 Conclusion

A method has been described which allows the optimisation of 3-D grey-scale soft morphological filters with respect to an objective quality criterion which is based on a local measure of variance. Soft morphological filters found using this technique show good results in removing film dirt from corrupted image sequences whilst retaining essential image details. The new 3-D technique improve

upon the performance of the existing 2-D method and may have some benefits compared to other non-global strategies.

Acknowledgements I should like to thank the staff at BBC R & D Dept., Kingswood Warren, Surrey, England, for providing archive film material and to the EPSRC for funding this work.

References

1. Koskinen, L., Astola, J., Neuvo, Y.: Soft Morphological Filters. Proc. SPIE Symp. on Image Algebra and Morphological Image Processing II. San Diego, USA (July 1992) 262–270
2. Kuosmanen, P.: Soft Morphological Filtering. Ph.D. Thesis, Dept. of Mathematical Sciences, University of Tampere, Finland (April 1993)
3. Pu, C.C., Shih, F.Y.: Soft Mathematical Morphology: Binary and Grey-Scale. Proc. Int. Workshop on Mathematical Morphology and its Application to Signal Processing. Barcelona, Spain (May 1993) 28–33
4. Huttunen, H., Kuosmanen, P., Koskinen, L., Astola, J.: Optimization of Soft Morphological Filters by Genetic Algorithms. Proc. of Image Algebra and Morphological Image Processing. San Diego, USA (July 1994) 13–24
5. Kuosmanen, P., Koivisto, P., Huttunen, H., Astola, J.: Optimization of Soft Morphological Filters under Shape Preservation Criteria. Proc. Image Algebra and Morphological Image Processing V. San Diego, USA (July 1994)
6. Harvey, N.R., Marshall, S.: Grey Scale Soft Morphological Filter Optimisation by Genetic Algorithms. In: P. Maragos, P. Schafer, R. Butt (eds.): Mathematical Morphology and its Applications to Image and Signal Processing. Kluwer Academic Publishers (1996) 179–18
7. Harvey, N.R.: New Techniques for the Design of Morphological Filters using Genetic Algorithms. Ph.D. Thesis, Dept. of Electonic and Electrical Engineering, University of Strathclyde, UK (September 1997)
8. Harvey, N.R., Marshall, S.: Film Restoration Using Soft Morphological Filters. Proc. 6th Int. Conf. on Image Processing and its Applications (IPIA'97). Dublin, Ireland (July 1997) 279–282.
9. Ramponi G., Strobel N., Mitra S., Yu T-H.: Nonlinear Unsharp Masking Methods for Image Contrast Enhancement. Journal of Electronic Imaging **5(3)** (July 1996) 353–366
10. Kokaram, A.C., Morris, R.D., Fitzgerald, W.J., Rayner, P.J.W.: Detection of Missing Data in Image Sequence. IEEE Trans. on Image Proc. Vol. 4. No. 11 (November 1995) 1496–1508
11. Kokaram, A.C., Morris, R.D., Fitzgerald, W.J., Rayner, P.J.W.: Interpolation of Missing Data in Image Sequence. IEEE Trans. on Image Proc. Vol. 4, No. 11 (November 1995) 1509–1519

Simulation of Evolvable Hardware to Solve Low Level Image Processing Tasks

Gordon Hollingworth, Andy Tyrrell, and Steve Smith

University of York, Heslington, York, England
{gsh100, amt, sls}@ohm.york.ac.uk
http://www.amp.york.ac.uk/external/aseg/evolarch.html

Abstract. The long term goal of the work described in this paper is the development of a bio-inspired system, employing evolvable hardware, that adapts according to the needs of the environment in which it is deployed. The application described here is the design of a novel and highly parallel image processing tool to detect edges within a wide range of conventional grey-scale images. We discuss the simulation of such a system based on a genetic programming paradigm, using a simple binary logic tree to implement the genetic string coding. The results acquired from the simulation are compared with those obtained from the application of a conventional Sobel edge detector, and although rudimentary, show the great potential of such bio-inspired systems.

1 Introduction

Bio-inspired systems have been present in the electronics and computer science communities for many years [21]. It is possible to classify bio-inspired systems into three domains: phylogeny, ontogeny and epigenesis. Each of these is relatively well understood in the world of natural science. However, inspiration is required to bridge the gap between natural sciences and engineering.

Phylogeny embraces the evolution of species through the passing of genes from one generation to the next. Infrequent errors occurring during the copying of genes, known as mutations, originate new traits on the species. The survival of species depends upon these traits and allow the species to adapt better to changes in the environment. The environment is represented by co-evolving populations and the resources needed for the survival of species. The ideas of phylogenetics have been applied for more than three decades with artificial systems. These are generally known as evolutionary algorithms or evolutionary computation, with specific examples being genetic algorithms, evolution strategies, evolutionary programming and genetic programming [10,14,18]. The evolution of hardware systems can be either extrinsic or intrinsic. In the first case a software description of the electronic circuit is evolved using computer simulation, and only the final elite chromosome is downloaded onto the programmable chip. Examples of extrinsic evolution include simple synchronous logic circuits. In the latter case the adaptation is done on-line in real-time.

R. Poli et al. (Eds.): EvoIASP'99 and EuroEcTel'99, LNCS 1596, pp. 46–58, 1999.

Evolutionary design techniques like Genetic Algorithms (GAs), Genetic Programming (GP) and Evolvable Hardware (EHW) have been applied to many simple design applications [18,19,11] and some advanced ones [6,8,16]. It is believed that the application of these techniques can create systems that will react to their inputs in a method akin to the adaptation of natural biological beings to the environment. In the work reported here, high level image processing applications are evolved that have the ability to adapt to changes in the environment that would normally create errors in the system.

This paper first describes the image processing problem of edge detection, followed by a brief introduction to evolutionary algorithms like the GP, and the application of GP to the Image Processing (IP) problem, resulting in a natural parallel architecture. Finally the actual edge detector evolved is discussed and evaluated in its abilities, in comparison to the Sobel edge detector.

2 Evolutionary Design Techniques

Genetic Algorithms were first explored by John Holland [4]; he showed how it is possible to evolve a set of binary strings which described some system to which a measure of fitness is applied. The analogy to normal evolution is that the binary strings are analogous to the DNA sequence (the genotype)carried by all living things, the phenotype (the body) is built through a process known as embryological development, which is a sequence of chemical interactions within each cell which distinguishes various cells and describes their action. The body is then subject to the environment where its fitness for reproduction is assesed, more fit individuals have a higher rate of reproduction. This means that genes within the DNA that code for specific 'good' traits (traits which describe better reproduction abilities) will have a higher probability of existing in future populations.

In artificial evolution a binary string describes the system, this system is either described directly or is described through some form of embryological development. The system is then assesed within the environment to find its fitness relative to the other individuals. This measure is then used to weight the probability of reproduction so that through many generations, good genes win out and bad genes die away.

A simple example is to find the maximum of some function f(x), the genotype-phenotype mapping is simple since the value of x to insert into the function is the binary value of the genotype. The fitness is literally the value of the function, and as generation passes to generation the average fitness increases. Holland showed that this works mathematically when the rate of reproduction is higher for 'more fit' individuals[4].

Genetic Programming (GP) is a simple extension to the Genetic Algorithm, introduced by John Koza[9]. Instead of describing the system using a simple binary string, a tree structure of functions is used. This structure creates a system with inputs at the end of each branch and a simple output at the top. An example tree is shown in figure 5.

A further extension to the domain of evolutionary techniques came after the creation of Field Programmable Gate Array (FPGA)[7] devices and Programmable Logic Devices (PLD), since these can be programmed using a binary string or by coding a binary logic tree. The electronic circuits can then be evaluated either electronically, to compare their output with the required output (intrinsic EHW), or in simulation (extrinsic EHW) to create some measure of the fitness[22].

This has promoted much research on EHW [16,18,19,11,8,6] showing not only the successful evolution of electronic circuits, but also some desirable features, such as fault tolerance [17].

The power of bio-inspired electronics is in its potential as an adaptive hardware which can change its behaviour and improve its performance while executing in a real physical environment (as opposed to simulation). Such on-line adaptation is more difficult to achieve but theoretically gives many advantages over extrinsic systems. At present, work has mostly been concered with off-line adaptation. That is, the hardware is not used in an execution mode while evolving. Problems involved with on-line adaptation include, time to adapt and accuracy of adaptation. However, if these problems can be overcome, the power of bio-inspired electonics offers much to many.

3 Image Processing Operation

Edge detection is used to establish boundaries between regions in an image, based upon the relative gray-levels. Common applications of edge detection might include locating cell-walls, the outline of an aircraft and the pre-processing stage for character recognition. The particular type of edge detector used often depends upon the type of edge detection criteria specified for the image under consideration and may differ depending upon whether the located edges are intended for human interpretation or futher machine manipulation. An ideal edge detection operator would be capable of detecting all types of edges, include simple steps, gradients and changes of texture, regardless of orientation and the quality of the image, which can commonly be distorted due to noise, corruption and poor lighting.

As might be expected, no such edge detection operator currently exists, although a number of different fundamental approches have been developed which include gradient-based, template matching and edge fitting operators and finally statistical detectors [20].

Gradient-based operators work on the principle that edges may be defined between areas of varying image intensity. It is common to represent images digitally as a number of picture elements or pixels, each of which has a value relating to the grey-scale intensity of the image at that point [1]. The gradient value at a pixel $f(x, y)$ is therefore related to the two dimensional differential:

$$\Delta f = \sqrt{\left[\left(\frac{df}{dx}\right)^2 + \left(\frac{df}{dy}\right)^2\right]} \tag{1}$$

The other edge detection operators listed above use highly non-linear methods. Template matching uses a cross-correlation between the image and a set of templates that detect edges in various orientations. Edge fitting uses a mathematical model of a step-edge with a search function to find the best fitting model at each point in the image. Finally, statistical detectors use statistical techniques to segment the image and indicate the edges between the segmented regions.

Gradient operators are more commonly used in IP, since they can be effectively implemented through a simple 2D FIR filter, which is simple to implement on Digital Signal Processor(DSP) technology.

The first step in testing Image Processing (IP) with evolutionary design techniques is to choose a common algorithm to simulate. The algorithm chosen is the process of edge detection, this is because it is a simple and well understood algorithm which is based at a pixel intensive low level. This is useful because we are interested in understanding how well evolutionary design will work with common IP applications.

4 An IP Architecture for EHW

The first problem which must be addressed is that of the reduction of complexity of the IP problem, this is generally due to the high number of input and output pixels that must be processed. Images are usually composed of thousands of individual pixels, each of which is represented by a number of bits, creating a massive network to deal with a function between input and output pixels. Since a genotype must code the function and connections between input and output pixels, a correspondingly large genotype is required. In general, the larger the genotype, the longer the time required for evolution. To minimise the evolution time (and therefore the time to adapt to changes in the environment) the genotype should be reduced in length as far as possible. A further requirement is that the system should be independant of the image size.

This problem can be overcome by exploiting the parallelism of images. Figure 1 shows a diagram of the basic structure of the architecture, each block in the network is pre-loaded with a single pixel of the image which is than output to the local neighbourhood. This architecture allows a rich and varied form of image processing from edge detection to high and low pass filtering.

Figure 2 shows the common processing block for each pixel (as seen in figure 1). Within each block are four main elements, the input pixel value, the output pixel value, the genotype string and the functional block. The genotype string is used to 'program' the functional block to perform some mapping between the local neighbourhood (input pixel and eight neighbour pixels) pixels and the output pixel. One method of doing this, for example, is to use lookup tables and multiplexors. Of course this would create some non-linear function due to the non-linear nature of the processing functions (e.g., and, or, not) It should be noted that this architecture is restricted to working within the local neighbourhood, but this will be addressed in later versions of the system.

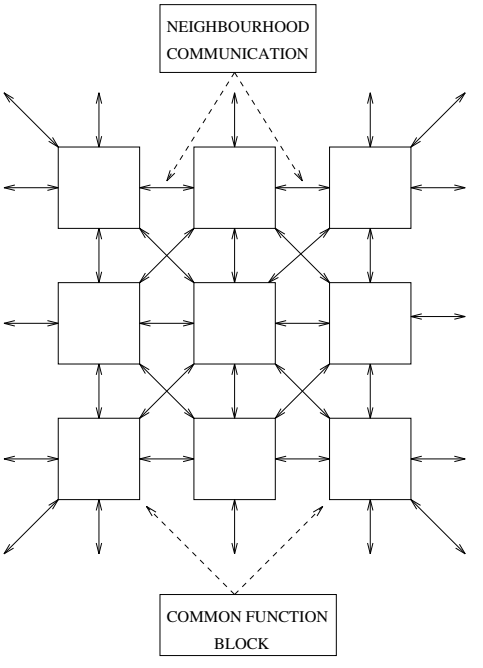


Fig. 1. Evolvable Hardware platform for grayscale local neighbourhood image processing

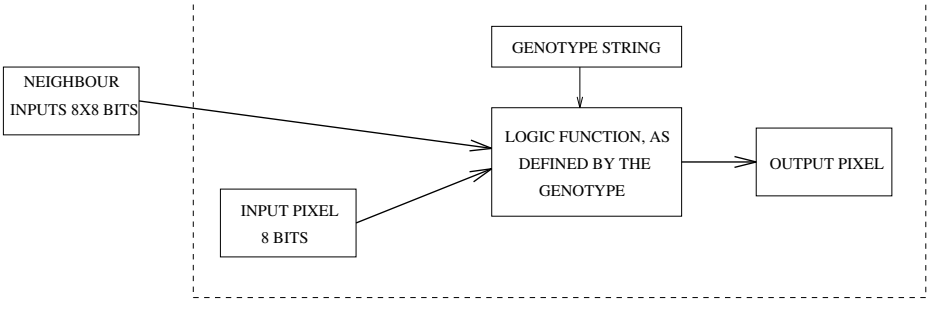


Fig. 2. Single pixel element block diagram

Since the processing block can be programmed using a binary string, this string can then be subjected to the forces of evolution to design an image processing function.

The proposed architecture will have a number of characteristics to help with implementation, these include: high regularity, which simplifies its implementation on silicon; modular in nature, making the actual function of the processing element independant from the function of the remaining blocks within a cell; simple, in terms of the processing elements used, allowing built-in self test logic to provide self diagnosis without excessively increasing the silicon area[13,12].

5 Simulation of the Evolvable Hardware

The application of EHW in the design of Image Processing hardware as described above is entirely novel and unproven. It is therefore prudent to evaluate the system performance as far as possible before committing the design to hardware. This is achieved by simulating the entire system in software, paying special consideration to the following areas:

Desired Edge Detection Operation The type of edge detection operation to be designed by the hardware system.

Fitness Evaluation The method adopted for assessing the performance of the function evolved.

Type of Evolutionary Algorithm The type of Evolutionary Algorithm to be employed (i.e. Genetic Algorithm or Genetic Program)

Genotype-Phenotype mapping The G-P mapping system used to convert between the genotype and the hardware system.

5.1 Desired Edge Detection Operation

As described earlier a number of different approaches to edge detection are available for conventional IP work, the most popular of these are gradient operators. It was therefore decieded that a gradient operator should be used as the operation by which the EHW system should be compared. The specific gradient operator chosen for evaluation of the EHW simulation is that devised by Sobel [15]. The Sobel operator is a simple, but effective neighbourhood processing or mask operator that combines good edge detection with immunity to noise.

Neighbourhood processing is achieved by considering the grey-scale values of the 8 pixels that surround the pixel under investigation. According to the weights specified in the mask, (see figure 3) a new value is calculated for the central pixel. This process is repeated for every pixel in the image.

The Sobel operator utilises two such 3x3 pixel masks which, are shown in figure 3. The first calculates the gradient in the horizontal plane and therefore detects vertical edges while the second calculates the gradient in the vertical plane and detects horizontal edges. In both cases the gradient is calculated by multiplying each pixel by the respective weighting and summing the result.

1	0	-1
2	0	-2
1	0	-1

1	2	1
0	0	0
-1	-2	-1

Fig. 3. Masks to detect vertical and horizontal edges

The vertical and horizontal gradients can be combined, using equation 2, to give a measure of the magnitude of the gradient at each point (i, j) in the image.

$$im[i,j] = \sqrt{(Sobelh[i,j])^2 + (Sobelv[i,j])^2} \tag{2}$$

Where $Sobelh[i,j]$ is the horizontal gradient at (i,j), and $Sobelv[i,j]$ is the vertical gradient.

The above is commonly approximated, using equation 3, to reduce computational complexity whilst maintaining the desired operation:

$$im[i,j] = |Sobelh[i,j]| + |Sobelv[i,j]| \tag{3}$$

The Sobel operator is generally followed by a simple thresholding operation in which each pixel in the image is assigned a value representing either black or white depending on the magnitude of the gradient at that point compared to some global threshold value. This operation is illustrated in figure 4.

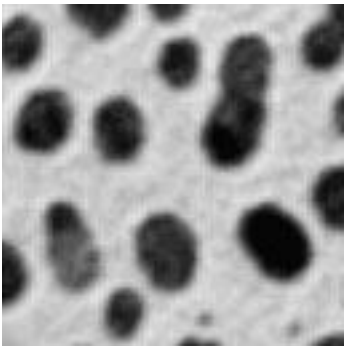


Fig. 4. (a) Original image (b) Sobel Output

5.2 Fitness Evaluation

The fitness of the genetically derived edge detector must be evaluated with respect to the Sobel operator previously described. Since the output image will be one bit (either a pixel is an edge or it is not) there are a number of methods

of comparing binary edge outputs from edge detectors[3,2], although in general, such methods are mathematically complex and too computationally intensive to preform practically in siumulation. An alternative method, described here, is a simplified minimisation of under and over detection of edge pixels. In essence, the results of the application of a Sobel operator and the genetically contrived edge detector to the same original image, are compared on a pixel-by-pixel basis. Two calculations are made based on those edges identified by the Sobel operator but not the genetically derived edge detector and vice-versa.

Underdetection (P_{ef}) is the number of edge pixels not detected by the genetically derived edge detector divided by the total number of edge pixels detected by the Sobel operator.

Overdetection (P_{nf}) is the number of non-edge pixels detected by the genetically derived edge detector divided by the total number of non-edge pixels detected by the Sobel operator.

Both of these values require minimisation simultaneously and is achieved by maximising equation 4:

$$fitness = \frac{1}{1 + P_{ef} + P_{nf}} \tag{4}$$

5.3 Type of Evolutionary Algorithm

Phylogeny embraces the evolution of species through the passing of genes from one generation to the next. The basis of this evolutionary development is that infrequent errors occuring during the copying of genes (mutations) originate new traits upon the species. Occasionally the mutation increases the individual's suitability for a changing environment, meaning that the probability of reproduction is increased. Two classes of Evolutionary Algorithm exist which exploit this method of evolutionary adaptation, Genetic Algorithms and Genetic Programming. The survival of species depends upon these random traits producing 'fitter' individuals and thus allowing the species to adapt better to a changing environment.

A number of tests have been conducted to find the most suitable algorithm for this application. The first was to evolve the convolution kernels (one horizontal and one vertical) to perform the stated tasks. This method used the Genetic Algorithm approach and showed itself to be successful, although, the function unit described above (within the standard processing block) would be limited to a simple linear convolution within the local neighbourhood.

The efficiency of the Genetic Programming paradigm has been investigated for these problems. This method creates a tree of functions, with inputs at the lowest layer of the tree and a single bit output at the top of the tree. In our implementation the inputs are the various bits of the various neighbouring image pixel values and the output equals a one or zero for an edge or non-edge respectively. Figure 5 shows a simple tree structure using the basic logic

functions. Each terminal is shown in bold and are named by the position they are in respect to the current pixel, i.e. NorthWest, SouthEast etc. When a tree is evaluated a fitness value is assigned which is then used in the breeding of the new population. Higher fitness individuals are more likely to reproduce, leading to fitter individuals reproducing more often. This operation tends to create a new individual that is better, more fit, than either of its parents.

5.4 Genetic String Coding

The coding method used with GP is simple, using a selection of the node functions and the terminal values:

{AND OR NOT XOR} Node Functions
{N NE E SE S SW W NW} Neighbouring pixel terminal values.

The set of functions was chosen to mimic the set of functions available within an FPGA. This then gives an indication of how well the application would transport across to current devices.

An example tree is shown in figure 5 using the functions and terminal sets shown.

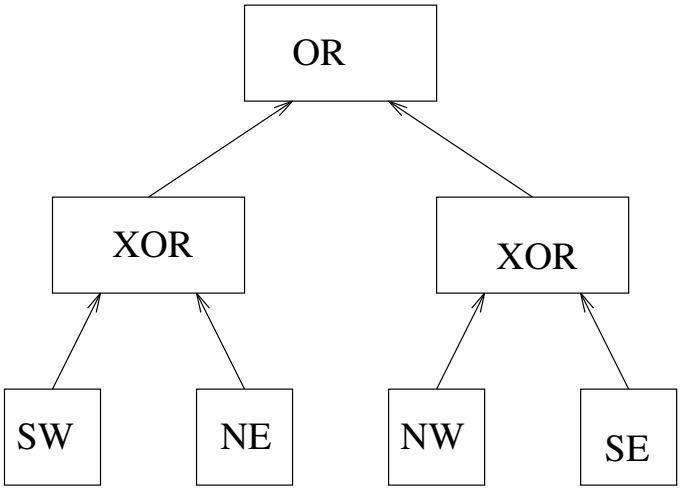


Fig. 5. A simple binary logic tree

The terminals, shown in bold in the figure, are actually only a single bit wide (due to the fact we are using logic gates). This means that with an 8 neighbourhood, 8 bits per pixel, 64 terminals would be required, making the evolution time long. For the purposes of this simulation the task has been simplified to use only 3 bits per pixel.

6 Results of Evolution

The results of the test runs are described as a set of images corresponding to various stages through a single run of the GP. These are compared with the output achieved by processing the same original image using the Sobel edge operator described earlier (figure 6a)

Figures 6b,7a and 7b shows the results of a single run of the GP using the logic functions on 3 bits of image data. Figure 6b is the result of a random logic tree, figure 7a is the output after 300 generations, where the trees have become very similar (almost a species of logic trees!) Figure 7b is the result after running the system for 791 generations. It is interesting to note that although the trees have been converged for some time, regular improvements are still being made in the edge detectors operations.

Since the edge detector evolved was only tested on a single image, it was thought useful to compare its performance on a different image. The results of this are seen in figure 8a and b. It is obvious that the evolved edge detector is in fact very specific to the properties of edges in the original image. Future runs of the algorithm will address this issue and alter the fitness measurement to include other varieties of edge types.

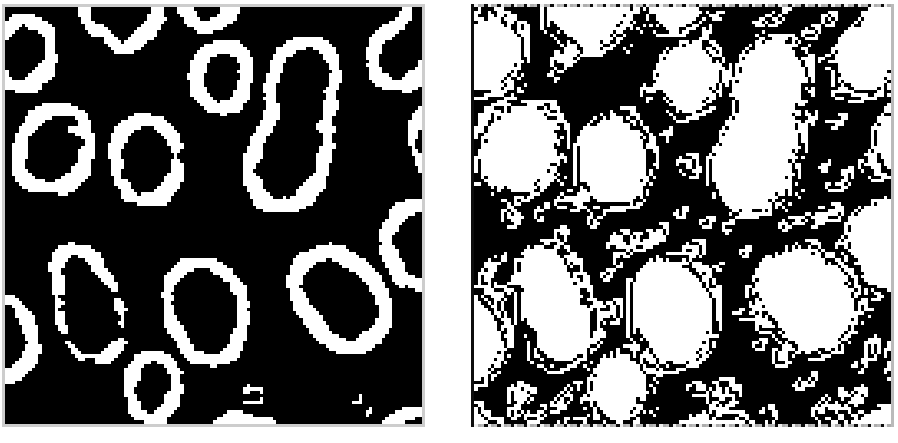


Fig. 6. (a) Sobel output using full 8 bits of input image (b) Best output from the first random generation

7 Discussion

The images considered above, resulting from a simulation of the proposed EHW architecture for image processing, indicate that an edge detection operator is being evolved. The final image shown is still far from perfect when compared

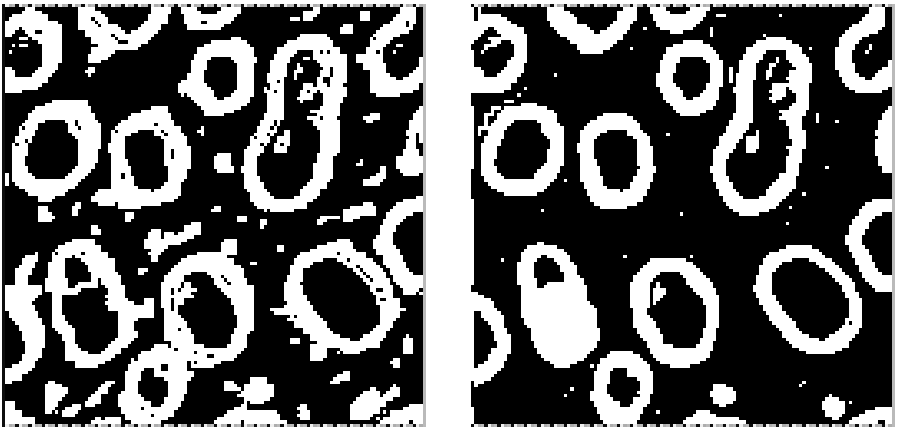


Fig. 7. (a) Results after 300 generations (b) Results after 791 generations

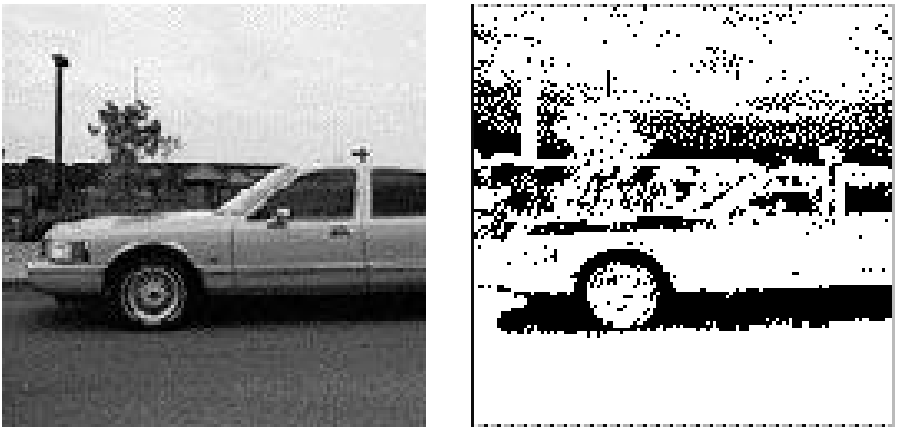


Fig. 8. (a) Original Image (b) Result after being passed through the same edge detector as figure 7b

with the output of the Sobel edge detector, however, it should be noted that a form of edge detection has been evolved from a zero starting point, showing that using the EHW architecture is a valid method of simple low-level image processing. The main importance of the derived edge detector is in its speed, since it is only a simple logic tree, there is only a small propogation delay from output to input. This would be of the order of ten times faster than current DSP devices if implemented in an ASIC.

The main problem with this method of evolution is the time required to evaluate the population, with only a single image being tested, the system took 24 hours to reach its current capability, however, these results look promising and work continues to refine and improve the evolution strategies.

8 Conclusion

The method described in this paper is an image processing operation that can be acheived using evolutionary algorithms. It is believed that the results presented demonstrate that this has been acheived through the simulation of a new architecture designed to exploit the parallism of images for the implementation of Evolvable Hardware.

Although the evolved system has some measure of sucess, it is important to note that the detector is not as good as a Sobel edge detection operator. Obviously the evolved system cannot produce something which is better than the Sobel, since, a Sobel would be perfect (in terms of its measured fitness). Instead the important result from this work is the EHWs ability to detect edges in some degree, this means that low- level image processing operators are able to be evolved through this method. This leads to the authors' belief that higher level IP operators can be evolved, where, it would be possible to test the system using more abstract descriptions of the behaviour of the image processing systems. Finally it is hoped that the work would give rise to novel solutions to well understood problems.

References

1. R.C. Gonzalez and R.E. Woods. *Digital Image Processing*. Addison-Wesley, 1993.
2. R.M. Haralick. Digital step edges from zero crossing of second directional derivatives. *IEEE Trans., Pattern Anal. Machine Intell.*, PAMI-6:58–68, Jan. 1984.
3. M. Heath, S. Sarkar, T. Sanocki, and K.W. Bowyer. A robust visual method for assessing the realtive performance of edge-detection algorithms. *IEEE Transactions on Pattern and Machine Intelligence*, 19(12):pp 1338–59, 1996.
4. J.H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.
5. G.S. Hollingworth, S.L. Smith, and A.M. Tyrrell. Design of highly parallel edge detection nodes using evolutionary techniques. In *Proceedings of 7th Euromicro Workshop on Parallel and Distributed Processing*. IEEE Press, 1999.
6. H. Iba, M. Iwata, and T. Higuchi. *Gate-level Evolvable Hardware: Empirical study and application*, pages 259–279. Springer-Verlag, 1997.
7. Xilinx inc. Xc6200 field programmable gate array data book, 1995. http://www.xilinx.com/partinfo/6200.pdf.
8. M. Iwata, I. Kajitani, H. Yamada, H. Iba, and T. Higuchi. A pattern recognition system using evolvable hardware. In *International Conference on Evolutionary Computation: The 4th Conference on Parallel Problem Solving from Nature*, pages 761–770. Springer, 1996.
9. J.R. Koza. *Genetic Programming*. MIT Press, 1992.
10. M. Murakawa, S. Yoshizawa, and T. Higuchi. Adaptive equalisation of digital communication channels using evolvable hardware. In Higuchi et al., editor, *Proceedings of 1st International Conference on Evolvable Systems: From Biology to Hardware*, volume 1259 of *LNCS*, pages 379–389. Springer, 1997.
11. M. Murakawa, S. Yoshizawa, I. Kajitani, T. Furuya, M. Iwata, and T. Higuchi. Hardware evolution at functional level. In *International conference on Evolutionary Computation: The 4th Conference on Parallel Problem Solving from Nature*, pages 62–71, 1996.

12. C. Ortega and A.M. Tyrrell. Biologically inspired real-time reconfiguration technique for processor arrays. In *Proceedings of 5th IFAC Workshop on Algorithms and Architectures for Real-Time Control*, 1998.
13. C. Ortega and A.M. Tyrrell. Design of a basic cell to construct embryonic arrays, 1998.
14. M. Sipper. Designing evolware by cellular programming. In Higuchi et al., editor, *Proceedings of 1st International Conference on Evolvable Systems: From Biology to Hardware*, volume 1259 of *LNCS*, pages 81–95. Springer, 1997.
15. I.E. Sobel. Camera models and machine perception (phd thesis), 1970.
16. A. Thompson. *Evolving Electronic Robot Controllers that exploit hardware resources.*, pages 640–656. Springer-Verlag, 1995.
17. A. Thompson. Evolutionary techniques for fault tolerance. *UKACC International Conference on Control*, pages 693–698, 1996.
18. A. Thompson. An evolved circuit, intrinsic in silicon, entwined with physics. In *Procedures of the 1st international conference on Evolvable systems (ICES96)*. Springer, 1996.
19. A. Thompson. Silicon evolution. In J.R. et al. (Eds) Koza, editor, *Proceedings of Genetic Programming 1996 (GP96)*, pages 444–452. MIT Press, 1996.
20. D. Vernon. *Machine Vision: Automated Visual Inspection and Robot Vision*. Prentice Hall, 1991.
21. J. Von Neumann. *Theory of Self Reproducing Automata*. University of Illinois Press, 1966.
22. X. Yao and T. Higuchi. Promises and challenges of evolvable hardware. In *International Conference on Evolvable Systems: From Biology to Hardware*. Springer, 1996.

Genetic Snakes for Medical Images Segmentation

Lucia Ballerini

Department of Electronic Engineering, University of Florence
Via S.Marta, 3 - 50139 Firenze - Italy
lucia@asp.die.unifi.it

Abstract. In this paper an approach is described for segmenting medical images. We use active contour model, also known as snakes, and we propose an energy minimization procedure based on Genetic Algorithms (GA). The widely recognized power of deformable models stems from their ability to segment anatomic structures by exploiting constraints derived from the image data together with a priori knowledge about the location, size, and shape of these structures. The application of snakes to extract region of interest is, however, not without limitations. As is well known, there may be a number of problems associated with this approach such as initialization, existence of multiple minima, and the selection of elasticity parameters. We propose the use of GA to overcome these limits. GAs offer a global search procedure that has shown its robustness in many tasks, and they are not limited by restrictive assumptions as derivatives of the goal function. GAs operate on a coding of the parameters (the positions of the snake) and their fitness function is the total snake energy. We employ a modified version of the image energy which consider both the magnitude and the direction of the gradient and the Laplacian of Gaussian. Experimental results on synthetic images as well as on medical images are reported. Images used in this work are ocular fundus images, snakes result very useful in the segmentation of the Foveal Avascular Zone (FAZ). The experiments performed with ocular fundus images show that the proposed method is promising in the early detection of the diabetic retinopathy.

1 Introduction

The study of the retinal vessels plays a crucial role in many clinically relevant diseases such as systemic hypertension, arteriosclerosis and diabetes. In particular, diabetic retinopathy is the leading cause of new adult blindness. Thought diabetes can affect the eye in a number of ways, the fine network of blood vessels in the retina is usually involved - hence the term diabetic retinopathy. One way to early detect diabetic retinopathy is the study of the Foveal Avascular Zone (FAZ). In fact, retinal capillary occlusion produces a FAZ enlargement. Moreover, the FAZ is characterized by qualitative changes showing an irregular contour with notchings and indentations [1]. The detection (segmentation) of FAZ boundary in usually considered the starting point for this kind of analysis.

We propose an automatic segmentation procedure to correctly identify the FAZ boundary. The observation of the particular anatomy of the FAZ prompted

R. Poli et al. (Eds.): EvoIASP'99 and EuroEcTel'99, LNCS 1596, pp. 59–73, 1999.

us to use a robust global segmentation method combining constraints derived from the image data with *a priori* knowledge about the position, size, and shape of this structure. The method was derived from the theory of active contours [2], along with Genetic Optimization [3]. The widely recognized power of deformable models stems from their ability to segment anatomic structures by exploiting constraints derived from the image data along with *a priori* knowledge about the location, size, and shape of such structures. However, the application of snakes to extract region of interest suffers from some limitations. In fact, there may be a number of problems associated with this approach such as algorithm initialization, existence of local minima, and the selection of model parameters.

We propose the use of GA to overcome some of these limits. GAs offer a global search procedure that has shown its robustness in many tasks, and they are not limited by restrictive assumptions on the objective function, such as the existence of derivatives. GAs operate on a coding of the free variables (the positions of the snake) and their fitness function is the total snake energy. We have employed a modified version of the image energy which accounts for both the magnitude and the direction of the gradient and the Laplacian of Gaussian. Genetic algorithms have several advantages over traditional methods: they operate on codings of the parameters rather than on the parameters themselves, they explore a population of points rather than a single point, they take advantage of information on the objective function and do not need other auxiliary knowledge, they use probabilistic rather than deterministic rules.

Snake optimization through genetic algorithms proved particularly useful in order to overcome problems related with initialization, parameter selection and local minima. In the following the proposed snake model will be referred to as a *Genetic Snake*.

Compared to current methods for segmenting the FAZ (manual selection or threshold methods), the proposed method offers high quantitative accuracy for the measurement of area and perimeter and we expect it will prove sufficiently robust in the aid to ophthalmological diagnosis.

The organization of the paper is as follow: in Section 2 we discuss active contours, the basic notions, their limitations and some improvements proposed in literature. In Section 3 we present the genetic optimization procedure. Experimental results on synthetic and medical images are reported in Section 4, in particular for retinal images we proposed an energy functional based on FAZ properties, which will allow snakes to give accurate FAZ boundary localization. We also discuss the choice of the model coefficients for this kind of images.

2 Active Contours

The mathematical foundations of deformable models represent the confluence of geometry, physics, and approximation theory. Geometry serves to represent object shape, physics imposes constraints on how the shape may vary over space and time, and optimal approximation theory provides the formal underpinnings of mechanisms for fitting the models to measured data. Deformable curves, sur-

faces, and solid models gained popularity after they were proposed for use in computer vision and computer graphics [4]. Seemingly, *snakes* are the most popular deformable model [2].

Snakes are planar deformable contours that are useful in several image analysis tasks. They are often used to approximate the locations and shapes of object boundaries on the basis of the reasonable assumption that boundaries are piecewise continuous or smooth.

Representing the position of a *snake* parametrically by $\mathbf{v}(s) = (x(s), y(s))$ with $s \in [0, 1]$, its energy functionals can be written as

$$E_{snake} = \int_0^1 E_{int}\left[\mathbf{v}(s)\right] ds + \int_0^1 E_{ext}\left[\mathbf{v}(s)\right] ds \tag{1}$$

where

- E_{int} represents the internal energy of the spline due to bending and it is associated with *a priori* constraints
- E_{ext} is an external potential energy which depends on the image and accounts for *a posteriori* information.

The final shape of the contour corresponds to the minimum of this energy.

In the original technique of Kass et al. [2] the internal spline energy is defined as

$$E_{int}\left[\mathbf{v}(s)\right] = \frac{1}{2}\left[\alpha(s)\left|\frac{\partial \mathbf{v}(s)}{\partial s}\right|^2 + \beta(s)\left|\frac{\partial^2 \mathbf{v}(s)}{\partial s^2}\right|^2\right]. \tag{2}$$

The spline energy is composed of a first order term controlled by $\alpha(s)$ and a second order term controlled by $\beta(s)$. The two parameters $\alpha(s)$ and $\beta(s)$ dictate the simulated physical characteristics of the contour: $\alpha(s)$ controls the *tension* of the contour while $\beta(s)$ controls its *rigidity*. The values of the non negatives functions $\alpha(s)$ and $\beta(s)$ determine the extent to which the snake can stretch or bend at any point s on the snake. For example, increasing the magnitude of $\alpha(s)$ increase the *tension* and tends to eliminate extraneous loops and ripples by reducing the length of the snake. Increasing $\beta(s)$ increases the bending *rigidity* of the snake and tends to make the snake smoother and less flexible. Setting the value of one or both of these functions to zero at a point s permits discontinuities in the contour at s.

The external energy couples the snake to the image. It is defined as a scalar potential function whose local minima coincide with intensity extrema, edges, and other image features of interest.

The external energy, which is commonly used, is defined as

$$E_{ext}\left[\mathbf{v}(s)\right] = -\gamma|\nabla I(x, y)|^2 \tag{3}$$

where I(x,y) is the image intensity, ∇ the gradient operator, and γ a weight associated with image energies. In this case the snake will be attracted to intensity edges.

According to Marr-Hidreth's theory [5] of edge-detection, Kass et al. [2] experimented also a slightly different edge functional

$$E_{ext}\left[\mathbf{v}(s)\right] = -\gamma|\nabla G_\sigma * I(x,y)|^2 \tag{4}$$

where $G_\sigma * I(x,y)$ denotes the image convolved by a Gaussian filter with a standard deviation σ. This edge functional is used by many researchers.

Although originally developed for application to problems in computer vision and computer graphics, the potential of deformable models for use in medical image analysis has been quickly realized. Deformable models are capable of accommodating the significant variability of biological structures over time and across different individuals. Deformable models have been successful applied to problems of fundamental importance in medical analysis including segmentation, shape representation, matching, and motion tracking [6].

The application of snakes and other similar deformable contour models to extract regions of interest is, however, not without limitations. For example, snakes were designed as interactive models. In non-interactive applications, they must be initialized close to the structure of interest to guarantee good performance. The internal energy constraints of snakes can limit their geometric flexibility and prevent a snake from representing long tube-like shapes or shapes with significant protrusions or bifurcations. Furthermore, the topology of the structure of interest must be known in advance since classical deformable contour models are parametric and are incapable of topological transformations without additional machinery.

Various methods have been proposed to improve and further automate the deformable contour segmentation process.

Cohen [7,8] used an internal (*inflating force*) to expand a snake model past spurious edges towards the real edges of the structure, making the snake less sensitive to initial conditions. This model, called *balloon*, reduces the sensibility to initialization, but increases the numbers of parameters.

McInerney and Terzopoulos [9,10] have been developing topology independent shape modeling schemes that allow a deformable contour or surface model to not only represent long tube-like shapes or shapes with bifurcations, but also to dynamically sense and change its topology. Their model is known as *topologically adaptable snake*.

Gunn and Nixon [11,12,13,14] used a *dual active contour*, which is combined with a local shape model to improve the parameterization. One contour expands from inside the target feature, the other contracts from the outside. The two contours are interlinked to provide a balanced technique with an ability to reject weak local energy minima.

Neuenschwander et al. [15,16,17] propose a snake-based approach that lets a user specify only the distant endpoints of the curve he wishes to delineate without having to supply an almost complete polygonal approximation. They simplify the initialization process and achieve much better convergence properties than those of traditional snakes by using the image information around these end points to provide boundary conditions and by introducing an optimization schedule

that allows a snake to take image information into account first only near its extremities and then, progressively, toward its center. In effect, the snakes are clamped onto the image contour in a manner reminiscent of a zip-lock being closed.

Lai and Chin [18,19,20,21] present an integrated approach to modeling, extracting, detecting and classifying deformable contours directly from noisy images. They begin by conducting a case study on regularization, formulation and initialization of the active contour models. Using the minimax principle they derive a regularization criterion whereby the values can be automatically and implicitly determined along the contour.

Yezzi et al. [22,23] formulate a geometric active contour model (*geometric snake*) based on defining feature based metrics on given images which in turn leads to a novel snake paradigm in which the feature of interest may be considered to lie at the bottom of a potential well.

3 Genetic Optimization of Snakes

We propose an energy minimization procedure based on Genetic Algorithms. This helps to overcome the difficulties related to initialization and local minima. In addition, we have observed a noticeable improvement of the segmentation with respect to standard snake algorithm.

Applying a Genetic Algorithm to any practical problem requires the definition of the following items:

1. a structural (chromosomal) representation of solutions to the problem;
2. an evaluation (objective function) of individuals in terms of their "fitness";
3. a method to initialize the population of candidate solutions;
4. values of parameters used by the algorithm (e.g., population size, crossover, etc.);
5. genetic operators which produce new sets of individuals;
6. a termination criterion for the Genetic Algorithm.

Some of this components (e.g. representation, evaluation and initialization) are entirely domain-dependent and will be discussed in the following, whereas others are implemented independently of the application domain.

The parameters that undergo genetic optimization are the positions of the snake in the image plane $\mathbf{v}(s) = (x(s), y(s))$. The coordinates x and y are codified in the chromosomes using a Gray-code [24,25]. To simplify the implementation we used polar coordinates.

The fitness function is the total snake energy as previously defined in Equation (1), where E_{int} and E_{ext} are defined in Eqs. (2) and (4). The sigma scaling option is used [3].

The genetic optimization requires the definition of a region of interest (see Fig. 1), given by r and R (the minimum and the maximum magnitude allowed for each $\mathbf{v}(s)$. The initial population is randomly chosen in such region, and each solution lies in this region (r and R are user defined). This replaces the

original initialization with a region-based version, enabling a robust solution to be found by searching the region for a global solution. This overcomes the problems associated with sensitivity to initialization which was a crucial problem of "hill climbing" techniques. As a result, the new optimization criterion is better at extracting non-convex shapes compared to conventional snakes.

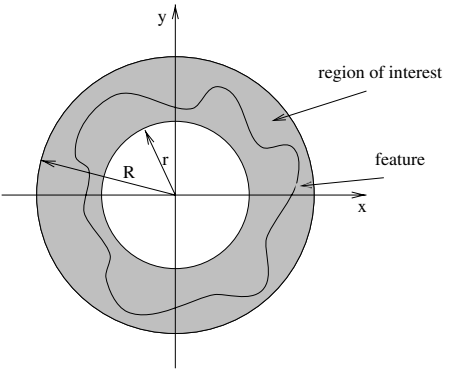


Fig. 1. Genetic Snakes Initialization.

The population size was computed according to the length of genome, as suggested by Goldberg [3]. We have used the standard two-point crossover. The crossover rate and mutation rate are set respectively to 0.6 and 0.000006.

4 Results

The most adequate set of parameters or our genetic snakes depends on several things: the characteristics of noise, the digitization parameters, the tortuosity of vessel which determine the FAZ boundary. Therefore, given a particular application some experimentation is required for choosing the best parameters. To show how this can be done, in the following section we report on some experiments performed on synthetic images with different patterns with additive noise of different variance. Then we present some results obtained with real ocular fundus images.

4.1 Experiments on Synthetic Images

The experiment uses synthetic images containing boundary of circles and squares as shown in Figure 2. The intensity images are generated by setting the pixel value to 255 if it belongs to the boundary, and 100 otherwise. We smooth the boundary by convolving images with a 3×3 window who acts as a low pass filter. The two kinds of images (circles and squares) are constructed to study the snake ability to capture corners as well as smoothed boundary. A zero mean,

white Gaussian noise was added to the images. Three different noise levels (corresponding to the standard deviation values: 20, 40, 60) were considered. This allowed us to study the robustness of our segmentation technique with respect to noise variance and to determine an adequate set of weights. Figure 2 shows some examples of simulated images.

On these images we perform experiment using snakes having 50 points, varying the energy weighting coefficients ($\alpha = 0.5, 0.8, 1, 1.2, 1.5$, $\beta = 0.5, 0.8, 1, 1.2$, 1.5 and $\gamma = 1$) running the GA for 2300000 iteration each time. Figure 3 reports some of the results obtained. We observed that snakes with larger values of α and β have better noise rejection capabilities, but snakes having too large values of α and β tend to shrink on itself.

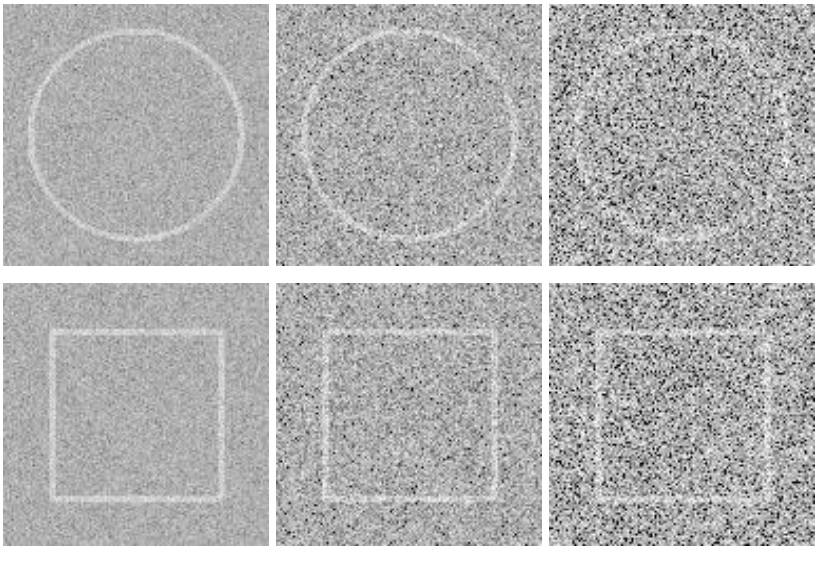


Fig. 2. Examples of synthetic test images with different values of Gaussian noise (from left to right: σ_{noise} =20, 40, 60)

4.2 Experiments on Medical Images

Genetic snakes are then applied to retinal images in order to segment the Foveal Avascular Zone (FAZ). Retinal images were taken by a SLO, with a frequency of 25 frames per second following the injection of a bolus of fluorescein. These images were digitized into 512 × 512 pixel matrices with 256 gray levels per pixel. The region of interest (ROI) i.e. the FAZ is approximately in the center of these images. For simplicity, the origin of the coordinates was located at the center of the FAZ. Its position can be chosen approximately be the user. The energy functionals are chosen according to FAZ properties, we employ a modified

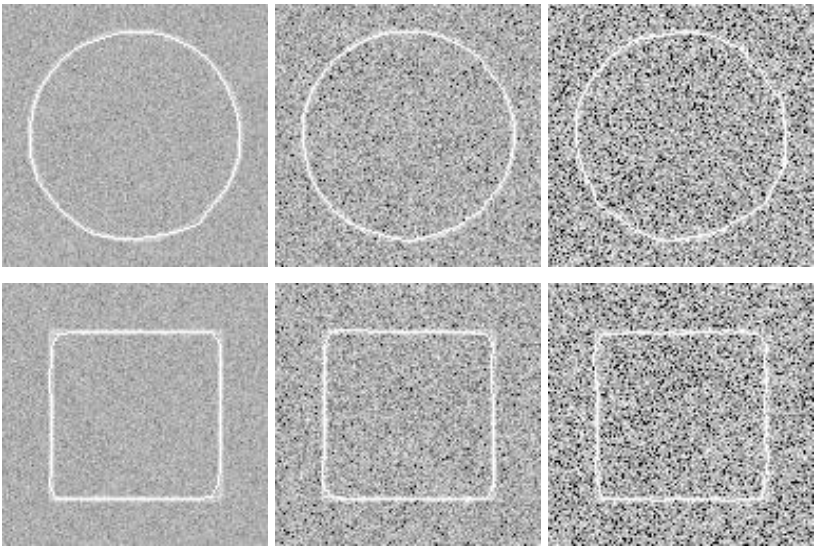


Fig. 3. Simulation results on synthetic test images with different shape, different values of Gaussian noise and with the best set of snake coefficients

version of the image energy which consider both the magnitude and the direction of the gradient and the Laplacian of Gaussian.

Proposed Energy Functional The internal energy term, E_{int}, controls the properties of the snake and it is expressed according to Kass [2] by

$$E_{int}\left[\mathbf{v}(s)\right] = \frac{1}{2}\left[\alpha(s)\left|\frac{\partial \mathbf{v}(s)}{\partial s}\right|^2 + \beta(s)\left|\frac{\partial^2 \mathbf{v}(s)}{\partial s^2}\right|^2\right]. \tag{5}$$

The values of the parameters $\alpha(s)$ and $\beta(s)$ are chosen empirically. The internal energy provides an efficient interpolation mechanism for recovering missing data.

The external energy E_{ext} are the image functionals. It is chosen according to FAZ properties. The image functionals are designed to produce minima corresponding to interesting objects in the image. It is shown that the choice of the image functional can effect the performance of the optimization technique used. For this reason various forms were experimented.

The classical optimization techniques impose different restrictions on the type of image functional that can be employed (for example the existence of derivatives); the use of genetic algorithms give us more freedom on the choice of such functional.

First, we consider a functional which localizes bright lines since FAZ boundaries are ultimately bright lines (i.e. capillaries) with an intensity maximum at their center. A simple external energy functional that attracts a snake to lines

could be the image intensity

$$E_{ext}\left[\mathbf{v}(s)\right] = \gamma I(x,y) \tag{6}$$

where γ is a weight factor whose sign determines whether the snake is attracted by dark or bright lines. For the case of a negative γ, the snake is attracted to local minima of E_{ext}, which corresponds to local maxima of intensity, i.e. bright lines. This functional (see Fig. 4(a)) can detect roof edges. It would be tempting to implement this functional for our purposes, since (6) would localize the medial axis of the capillaries. However, the achievable performances are partially satisfactory, which is due to the adjacency of the snake to the bigger vessels exhibiting a strong maximum; moreover, die leakage introduces a light haze with consequent artifacts on the image function.

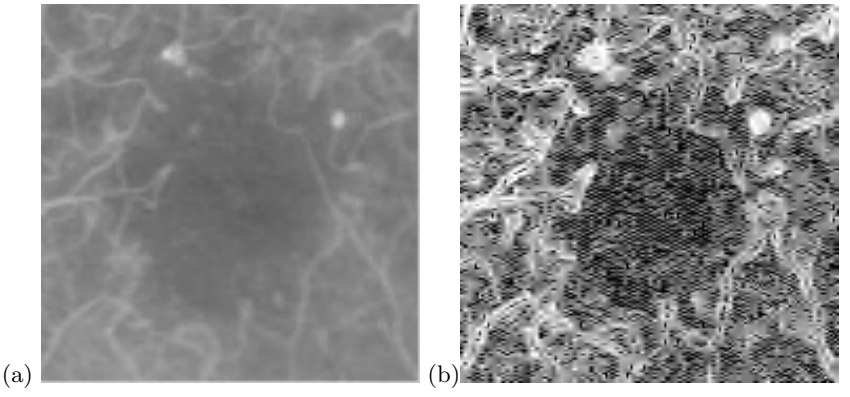


Fig. 4. (a) Image intensity and (b) image convolved by gradient of Gaussian ($\sigma = 2$).

Then, we consider a functional which attracts the snake to image edges, since we would like the snake to detect vessel boundaries. In this case, if edges are of interest, E_{ext} would be defined as

$$E_{ext}\left[\mathbf{v}(s)\right] = -|\nabla I(x,y)|^2 \tag{7}$$

where $\nabla I(x,y)$ is the gradient of the image. An easy implementation of this functional can be obtained by computing the Gradient of Gaussian (GoG) of the image intensity

$$E_{ext}\left[\mathbf{v}(s)\right] = -|\nabla G_\sigma * I(x,y)|^2. \tag{8}$$

The resulting functional image is shown in Fig. 4(b). The weight in this case is negative so that local minima of E_{ext} correspond to maxima of the gradient, i.e. strong edges. Simple implementation of this functional for FAZ boundary extraction also does not give fully satisfactory performance. The fact that it is the edge of the vessels that is localized and not the point of maximum intensity provides a basis for uncertainty.

This suggested to us to consider both the magnitude and the direction of the image gradient. A suitable functional may be obtained by constructing the dot product of the contour tangent with the normalized gradient vector

$$E_{ext}\left[\mathbf{v}(s)\right] = \left| \frac{\partial \mathbf{v}}{\partial s} \cdot \frac{\nabla I(x,y)}{|\nabla I(x,y)|} \right|. \tag{9}$$

The weight of this factor is positive, so that orientation inconsistencies tend be penalized. In this way, edge points whose orientation disagrees with that of the overlaying snake may also yield minimal values of the external energy. Hence the snake is able to discriminate against phantom lines, while allowing for the presence of corners. The two components of the gradient are shown in Fig. 5.

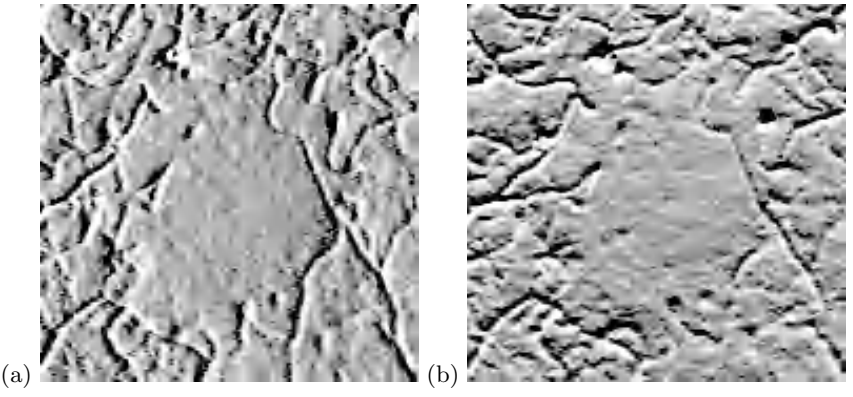


Fig. 5. The x and y components of the gradient of the image. The intensity of each pixel is proportional to the gradient component in that point.

In order to increase the locus of attraction of a minimum, we have experimented with a slightly different edge functional (also proposed by Kass [2]):

$$E_{ext}\left[\mathbf{v}(s)\right] = -|\nabla^2 G_\sigma * I(x,y)|^2. \tag{10}$$

Minima of this functional lie on zero-crossings of $\nabla^2 G_\sigma * I(x,y)$ which define edges in the Marr-Hildreth theory [5]. Adding this term to a snake means the snake is attracted by zero-crossing, but it is still constrained by its own smoothness. This image functional is shown in Fig. 6.

In addition, since image gradient and Laplacian of Gaussian (LoG) produce random edges in the background region where some noise is present, we can improve FAZ boundary localization by including a Gaussianly smoothed version of the image intensity (with large σ).

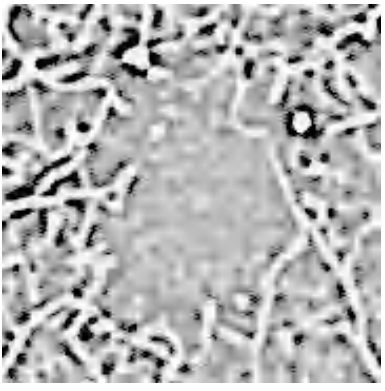


Fig. 6. Image convolved by Laplacian of Gaussian ($\sigma = 2$). (The original image is shown in Fig. 4(a)).

Thus, the proposed energy functional is composed of four terms and can be expressed as

$$E_{ext}[\mathbf{v}(s)] = -\gamma_1 G_\sigma * I(x,y) - \gamma_2 |\nabla G_\sigma * I(x,y)|^2 + \gamma_3 \left(\mathbf{n} \cdot \frac{\partial \mathbf{v}}{\partial s} \right) - \gamma_4 |\nabla^2 G_\sigma * I(x,y)|^2 \tag{11}$$

where $\mathbf{n} = \frac{\nabla G_\sigma * I(x,y)}{|\nabla G_\sigma * I(x,y)|}$.

In a few cases, additional knowledge on the image has been integrated within the snake by adding a constraint energy term E_{con} to Equation (4). In order to specify a particular image feature, located in the interval $[x_1, x_2][y_1, y_2]$, this functional can be defined as

$$E_{con}\left[\mathbf{v}(s)\right] = \begin{cases} 0 \text{ if } (x,y) \in [x_1, x_2] \times [y_1, y_2] \\ 1 \text{ otherwise} \end{cases} \tag{12}$$

This is a soft constraint, which can help in the case that a microaneurysm or a large vessel attracts the snake more than the FAZ boundary we are looking for.

Snake-Model Coefficients The choice of the weights controls the type of solution the active contour produces: large values of the weights associated with image functionals tend to move the snake boundary towards the FAZ contour, while the values of α and β control the smoothness and continuity.

The signal to noise ratio (SNR) can affect the choice of weights: in low SNR images, or where there are missing and/or false edges, an increased contribution from continuity and smoothness terms to the energy functional is usually desirable. Large values for the continuity and curvature weights will discourage convergence to a "busy" contour. On the other hand, small weights may allow the contour to be trapped into false edges or leak out through gaps in the boundary.

The internal energy weights are normally kept constant while image energy weights are varied to find a good balance between the four terms. We set $\alpha(s) = \alpha$

and $\beta(s) = \beta$, where α and β are constant values. In this way different segments of the snake cannot have different elastic behavior. We have observed that values close to 1 give good results. The values of the weights associated with image functionals are chosen in the range $[0.5, 0.8]$.

In Fig. 7 we can see some examples of original images and the corresponding FAZ outlines segmented by our snake model.

5 Conclusion

FAZ segmentation is achieved by using active contour models (snakes). The widely recognized power of deformable models stems from their ability to segment anatomic structures by exploiting constraints derived from the image data along with *a priori* knowledge about the location, size, and shape of such structures, as discussed previously.

Deformable models are capable of accommodating the often significant variability of biological structures over time and across different individuals.

The modified version of the image energy we proposed (which accounts for both the magnitude and the direction of the gradient and the Laplacian of Gaussian) exhibits interesting properties in the localization of FAZ boundary.

The energy minimization procedure based on Genetic Algorithms overcomes the problems associated with sensitivity to initialization and local minima, which was a crucial problem of classical techniques.

In a first stage we have implemented the standard snake algorithm [2]. However the choice of the related parameters resulted very critical in our case. This difficulty was significantly reduced in the proposed model.

In this work we applied GAs to the positions of the snake. The management of the weight controls of the energy function is an open important problem. Further work on this technique could be the evolution of the parameters governing the snake behaviour.

Compared to current methods for segmenting the FAZ (manual selection or threshold methods), a snake-based approach is expected to provide significant improvements. This method offers high quantitative accuracy for the measurement of area and perimeter, which is important for diabetic studies [26]. Thus, we expect these methods will prove sufficiently robust in the aid to ophthalmological diagnosis.

References

1. G. H. Bresnick, R. Condit, S. Syrjala, M. Palta, A. Groo, and K. Korth, "Abnormalities of the foveal avascular zone in diabetic retinopathy," *Arch. Ophthalmol.* **102**, pp. 1286–1293, september 1984.
2. M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *International Journal of Computer Vision* **1**(4), pp. 321–331, 1988.
3. D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, MA, 1989.

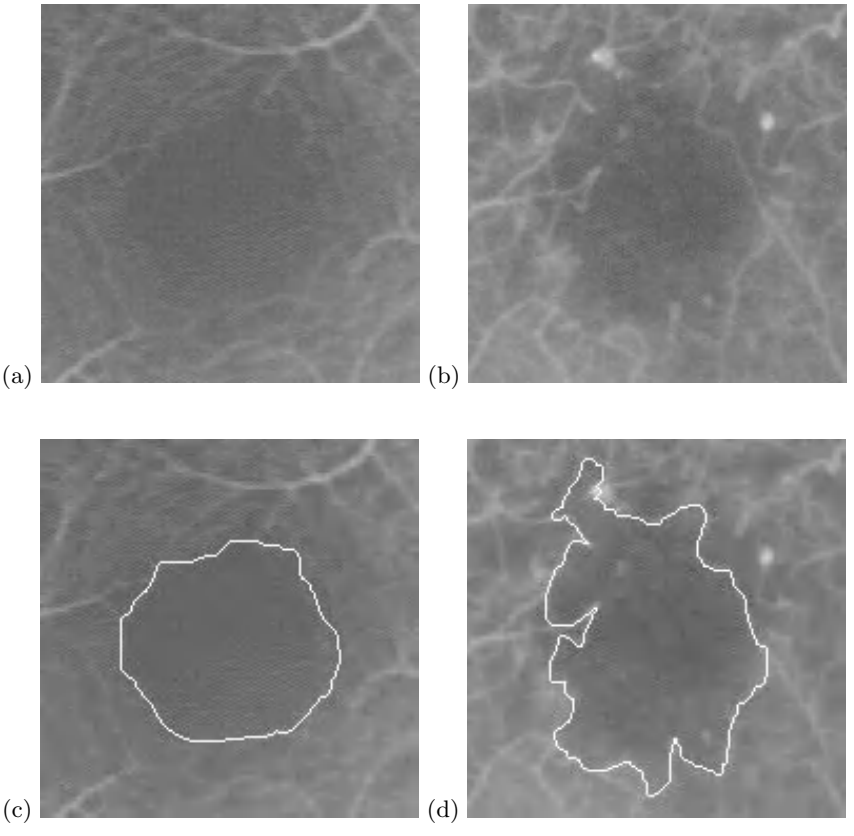


Fig. 7. Images of normal (a) and diabetic (b) FAZ, and the same images with superimposed snakes (c) and (d).

4. D. Terzopoulos and K. Fleischer, "Deformable models," *The Visual Computer* **4**(6), pp. 306–331, 1988.
5. D. Marr, *Vision*, New York: W. H. Freeman, 1980.
6. T. McInerney and D. Terzopoulos, "Deformable models in medical image analysis: A survey," *Medical Image Analysis* **1**(2), pp. 91–108, 1996.
7. L. D. Cohen and I. Cohen, "Finite element methods for active contour models and balloons for 2D and 3D images," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **15**, pp. 1131–1147, November 1993.
8. L. D. Cohen, "On active contour models and balloons," *Computer Vision, Graphics, and Image Processing: Image Understanding* **53**, pp. 211–218, March 1991.
9. T. McInerney and D. Terzopoulos, "Topologically adaptable snakes," in *Proc. Fifth International Conf. on Computer Vision (ICCV'95), Cambridge, MA*, pp. 840–845, IEEE Computer Society Press, (Los Alamitos, CA), 1995.
10. T. McInerney and D. Terzopoulos, "Medical image segmentation using topologically adaptable surfaces," in *Proc. First Joint Conference of Computer Vision, Virtual Reality, and Robotics in Medicine and Medical Robotics and Computer-Assisted Surgery (CVRMed-MRCAS'97), Grenoble, France, March, 1997*, pp. 23–32, Springer-Verlag, (Berlin), 1997.
11. S. R. Gunn and M. S. Nixon, "Robust snake implementation; a dual active contour," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **19**, pp. 63–68, January 1997.
12. S. R. Gunn, *Dual Active Contour Models for Image Feature Extraction*. PhD thesis, University of Southampton, Faculty of Engineering and Applied Science, 1996.
13. S. R. Gunn and M. S. Nixon, "A dual active contour including parameteric shape," tech. rep., University of Southampton, Department of Electronics and Computer Science, 1994. 1994 Research Journal.
14. S. R. Gunn and M. S. Nixon, "A dual active contour for improved snake performance," tech. rep., University of Southampton, Department of Electronics and Computer Science, 1995. 1995/6 Research Journal.
15. O. Henricsson and W. Neuenschwander, "Controlling Growing Snakes by Using Key-Points," in *Proceedings 12th IAPR International Conference on Pattern Recognition*, pp. 68–73, (Jerusalem), 1994.
16. W. Neuenschwander, P. Fua, G. Székely, and O. Kübler, "Initializing Snakes," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 658–663, June 1994.
17. W. Neuenschwander, P. Fua, G. Székely, and O. Kübler, "Making Snakes Converge from Minimal Initialization," in *ARPA Image Understanding Workshop*, pp. 1627–1636, (Monterey, CA), Nov. 1994.
18. K. F. Lai and R. T. Chin, "Deformable contours: Modeling and extraction," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **PAMI-17**(11), pp. 1084–1090, 1995.
19. K. F. Lai and R. T. Chin, "Deformable contours: Modeling and extraction," in *Proc. Computer Vision and Pattern Recognition Conf.*, 1994.
20. R. T. Chin and K. F. Lai, "On regularization, formulation and initialization of active contour models (Snakes)," in *Proc. 1st Asian Conf. on Computer Vision*, pp. 542–545, 1993.
21. R. T. Chin and K. F. Lai, "On classifying deformable contours using the generalized active contour model," in *Proc. Int. Conf. Automation, Robotics and Computer Vision*, 1994.

22. A. Yezzi, S. Kichenassamy, A. Kumar, P. Olver, and A. Tannebaum, "A geometric snake model for segmentation of medical imagery," *IEEE Transactions on Medical Imaging* **16**, pp. 199–209, April 1997.
23. S. Kichenassamy, A. Kumar, P. Olver, A. Tannebaum, and A. Yezzi, "Gradient flow and geometric active contour," in *Proceedings of International Conf. on Computer Vision*, June 1995.
24. D. Beasley, D. R. Bull, and R. R. Martin, "An overview of genetic algoritms: Part 1, fundamentals," *University Computing* **15**(2), pp. 58–69, 1993.
25. D. Beasley, D. R. Bull, and R. R. Martin, "An overview of genetic algoritms: Part 2, research topics," *University Computing* **15**(4), pp. 170–181, 1993.
26. L. Ballerini, *Computer Aided Diagnosis in Ocular Fundus Images.* PhD thesis, Università di Firenze, Italy, 1998.

Evolving a Task Specific Image Operator

Marc Ebner and Andreas Zell

Eberhard-Karls-Universität Tübingen, Wilhelm-Schickard-Institut für Informatik
Arbeitsbereich Rechnerarchitektur, Köstlinstraße 6, 72074 Tübingen, Germany
{ebner,zell}@informatik.uni-tuebingen.de

Abstract. Image processing is usually done by chaining a series of well known image processing operators. Using evolutionary methods this process may be automated. In this paper we address the problem of evolving task specific image processing operators. In general, the quality of the operator depends on the task and the current environment. Using genetic programming we evolved an interest operator which is used to calculate sparse optical flow. To evolve the interest operator we define a series of criteria which need to be optimized. The different criteria are combined into an overall fitness function. Finally, we present experimental results on the evolution of the interest operator.

1 Motivation

A large number of standard image processing operators are available to solve a particular problem. In general, the required operators depend on the current task and environmental conditions. In our work we are trying to evolve image processing operators which perform optimal for the task and the given environmental conditions. To evolve the image operators we have chosen genetic programming [14,15,2] because it allows the evolution of hierarchical structures which are often required to solve image processing tasks. The sample problem which we address here is the evolution of an interest operator which is used to compute sparse optical flow. We show how an interest operator can be evolved which is optimal according to multiple criteria which are specific to the application. Before we present our experimental results we briefly discuss related work of using evolutionary methods for image processing tasks.

2 Background

A number of researchers have used evolutionary algorithms for image processing tasks. The methods used range from evolutionary programming [3], structure evolution [18] a variant of an evolution strategy, to genetic algorithms [25,26,13, 4]. A growing number of researchers are using genetic programming.

Tackett [30] evolved a symbolic expression for image classification based on image features. Koza [15] and Andre [1] evolved character detectors using genetic programming. Johnson et al. [11] evolved Ullman's visual routines [32] using genetic programming to locate the left and right hand in an image showing the

R. Poli et al. (Eds.): EvoIASP'99 and EuroEcTel'99, LNCS 1596, pp. 74–89, 1999.

silhouette of a person. Poli [21] applied genetic programming to the task of image segmentation. Daida et al. [5] used genetic programming to extract pressure-ridges from satellite images of arctic sea ice. Harris and Buxton [9] used genetic programming to evolve one-dimensional edge detectors. Poli and Cagnoni [22] evolved algorithms for image enhancement using interactive program evolution. Winkeler and Manjunath [33] used genetic programming for face detection.

Considerable work has been done in the area of feature extraction and tracking. A match between interesting points extracted from an image sequence or from a pair of stereo images can be established easily [36,34]. Knowledge about point correspondences may be used to establish a three-dimensional model of the world. A number of different methods have been developed to extract interesting points from an image. Moravec [20] developed an interest operator which extracts points with a high variance of pixel values in four directions: horizontal, vertical and both diagonals. Smith [29] developed a corner finder which extracts points where the size of the region belonging to the current pixel in a small neighborhood is a local minimum. Other methods range from using the determinant of the Hesse matrix to find regions of high curvature [27,19], corner detection [27], difference of Gabor filters [36], detection of symmetry [24,35] to the use of entropy [12]. Shi and Tomasi [28] argue that good features are those for which the tracker works best. Extracted features (textured regions) are monitored by calculating a dissimilarity measure computed from an affine model of image motion. Features with a high dissimilarity measure should be abandoned. Lew et al. [17] developed an adaptive method for feature selection. From a set of features they select a subset which maximizes the error distance between the correct match and other possible matches.

Genetic programming has so far been rarely used for the construction of image processing operators. Ebner [6] used genetic programming to evolve operators to extract edges from digitized images and evolved an approximation to the Moravec interest operator [7]. In difference to the previous work no existing operator is used for computing the fitness of an evolved individual. In this paper, we only specify the desired properties of the operator and integrate them into a measure of the individual's fitness.

3 Evolving an Interest Operator

As a sample application we have chosen to evolve an interest operator. The points extracted by the operator are used to calculate sparse optical flow. Optical flow is calculated by establishing corresponding points between the previous image and the current image. It is assumed here that the optical flow can be quite large. This might occur if the camera moves very fast or, equivalently, if delays between subsequent images are long. In this case the calculation of optical flow is simplified by focusing the search only on interesting points in the image. Correspondences are established by comparing the pixels in a small area around the interesting points. The goal is to extract only those points which can be localized accurately in the next image. To achieve this goal we are trying to

optimize a number of different properties of the operator. We start by describing the different properties qualitatively which are formalized later. The following properties were used here.

a) The number of established matches should be large. If only a single point is extracted for every image, localizing the point is easy. However, obtaining a dense flow field is usually desirable.
b) The quality of the match should be good. If some error measure describes the difference between pixel values in a small area surrounding the two matched points then this measure should be small.
c) A threshold is used to determine when a match can be considered adequate. Thus a match is not found for every extracted point. Therefore the ratio between matched points and number of extracted points should be high, that is, a match should be established for most of the extracted points. Otherwise it would be possible to extract almost every point and let the search procedure weed out the unnecessary points. However, this is precisely the task the operator should perform.
d) The matches should be unambiguous. For each point all other points are considered as a possible match. Therefore the difference between the error measure of the best match and the second-best match should be large indicating clearly which of the possible matches is the correct one.
e) The optical flow field should be smooth with only a few discontinuities. That is, nearby flow vectors should have approximately the same direction.
f) The density of the flow field may be regulated by introducing a term that tries to achieve a flow field with a maximum density that is distributed over the image.

We now formalize the different optimization criteria. Let $I(t)$ be the image taken at time t. First, the evolved operator is applied to this image. Non-local maxima are suppressed and all points where the pixel value is larger than a threshold ϵ_1 are extracted. Let $F(t)$ be the extracted interesting points of image $I(t)$. Given two images $I(t_1)$ and $I(t_2)$ taken at times t_1 and t_2, respectively, a correspondence between the points in $F(t_1)$ and $F(t_2)$ is established. Given a point $(x_1, y_1) \in F(t_1)$ we calculate the following error measure e for every point $(x_2, y_2) \in F(t_2)$ that is within a specified distance of the original point.

$$e(x_1, y_1, x_2, y_2) = \sqrt{\frac{1}{wh} \sum_{-\frac{w}{2} \le i < \frac{w}{2}} \sum_{-\frac{h}{2} \le j < \frac{h}{2}} \left(\tilde{I}(x_1 + i, y_1 + j) - \tilde{I}(x_2 + i, y_2 + j) \right)^2} \tag{1}$$

where $\tilde{I}(t)$ is obtained by smoothing image $I(t)$ with a Gaussian filter and w and h specify the width and height of the patch which is used to calculate the error measure. The point (x_2, y_2) for which the error measure is minimal is chosen as the corresponding point. In addition a threshold is used to reject bad matches. Therefore a match is only established provided that e is less then a threshold ϵ_2. Let $F_m(t)$ be the points for which a match could be established. Let n_p be the

number of points in $F(t_1)$ and let n_m be the number of points for which a match could be established. Then the following measures of operator quality were used for our experiments.

a) Number of matches:

$$m_1 = n_m \tag{2}$$

b) Quality of matches:

$$m_2 = \frac{1}{n_m} \sum_{(x,y)\in F_m(t_1)} \frac{1}{1 + e_{\min}(x,y)} \tag{3}$$

where $e_{\min}(x_1, y_1) = \min_{(x,y)\in F(t_2)} e(x_1, y_1, x, y)$ is the minimum of the error measure e. The measure m_2 is analogous to Pratt's figure of merit which is used to judge the performance of edge detectors [10].

c) Match percentage:

$$m_3 = \frac{n_m}{n_p} \tag{4}$$

d) Match ambiguity:

$$m_4 = \frac{1}{n_m} \sum_{(x,y)\in F_m(t_1)} \frac{e_{\text{next}}(x,y) - e_{\min}(x,y)}{e_{\max}(x,y) - e_{\min}(x,y)} \tag{5}$$

where $e_{\max}(x_1, y_1) = \max_{(x,y)\in F(t_2)} e(x_1, y_1, x, y)$ is the maximum of the error measure e. Let (x_m, y_m) be the point for which the error measure is minimal. Then the value of the error measure for the second-best match is defined as $e_{\text{next}}(x_1, y_1) = \min_{(x,y)\in F(t_2)\setminus(x_m,y_m)} e(x_1, y_1, x, y)$.

e) Flow smoothness:

$$m_5 = \frac{1}{n_p} \sum_{(x,y)\in F(t_1)} s(x,y) \tag{6}$$

where s is a smoothness measure calculated for a small neighborhood around the point. Let $F_{N(x,y)}$ be the points inside the neighborhood of point (x, y).

$$F_{N(x,y)}(t) = \{(x', y') \in F(t) | \sqrt{(x'-x)^2 + (y'-y)^2} < \epsilon_3\} \tag{7}$$

Then the smoothness measure is calculated as

$$s(x,y) = \frac{1}{2|F_{N(x,y)}(t_1)|} \sum_{(x',y')\in F_{N(x,y)}(t_1)} 1 + \frac{\Delta x \Delta x' + \Delta y \Delta y'}{\sqrt{\Delta x^2 + \Delta y^2}\sqrt{\Delta x'^2 + \Delta y'^2}} \tag{8}$$

where $(\Delta x, \Delta y)$ is the computed optical flow of point (x, y).

f) Maximum flow field density:

$$m_6 = \frac{1}{n_p} \sum_{(x,y)\in F(t_1)} \min\left\{\frac{d_{\min}(x,y)}{d_{\text{des}}}, 1.0\right\} \tag{9}$$

where $d_{\min}(x, y)$ is the distance in pixels between point (x, y) and its nearest point and d_{des} is the desired minimum distance between the extracted points.

4 Using Genetic Programming to Evolve Image Operators

Genetic programming is especially suited to combine simple elementary functions into a complex hierarchical image processing operator. To apply genetic programming to the evolution of an image processing operator we have to define the set of terminal symbols, the set of primitive functions and a suitable fitness measure. We now describe each of these in turn.

4.1 Terminal Symbols

The input image I was our only terminal symbol. The pixel values were normalized to the range $[0, 1]$.

4.2 Primitive Functions

As primitive functions we used the following set of unary and binary functions. Let I_R be the image that results from the application of a primitive function to an input image I in the case of an unary function and two input images I_1 and I_2 in the case of a binary function. Image coordinates are denoted with x and y. Unary primitive functions:

- Square root (`Sqrt`): $I_R(x,y) = \sqrt{|I(x,y)|}$
- Square (`Square`): $I_R(x,y) = I(x,y) \cdot I(x,y)$
- Gabor filters (`Gabor0`,...,`Gabor7`):
  $I_R(x,y) = |\int \Psi(x',y',f,\theta_j) I(x-x',y-y') dx'dy'|$
  with $\Psi(x,y,f,\theta) = \exp(i(fx\cos\theta + fy\sin\theta) - \frac{f^2(x^2+y^2)}{2\sigma^2})$,
  $\sigma = \pi$, $f = \frac{\pi}{2}$ and $\theta_j = \frac{\pi j}{8}$ with $j \in \{0,...,7\}$ (as defined in [16]).
- Average (`Avg3x3`): $I_R(x,y) = \frac{1}{9}\sum_{-1\leq i,j\leq 1} I(x+i,y+j)$
- Median filter (`Median3x3`): $I_R(x,y) = \text{Median}\{I(x+i,y+j) | -1 \leq i,j \leq 1\}$
- Gaussian filter (`Gauss`):
  $I_R(x,y) = \int e^{\frac{x'^2+y'^2}{2\sigma^2}} I(x-x',x-y') dx'dy'$ with $\sigma = 1.0$.
- Derivative of Gaussian in x direction (`GaussDx`):
  $I_R(x,y) = \frac{1}{\sqrt{2\pi}\sigma^3} \int x e^{-\frac{1}{2\sigma^2}(x'^2+y'^2)} I(x-x',y-y') dx'dy'$ with $\sigma = 1.0$.
- Derivative of Gaussian in y direction (`GaussDy`):
  $I_R(x,y) = \frac{1}{\sqrt{2\pi}\sigma^3} \int y e^{-\frac{1}{2\sigma^2}(x'^2+y'^2)} I(x-x',y-y') dx'dy'$ with $\sigma = 1.0$.

Binary primitive functions:

- Addition (`+`): $I_R(x,y) = I_1(x,y) + I_2(x,y)$
- Subtraction (`-`): $I_R(x,y) = I_1(x,y) - I_2(x,y)$
- Multiplication (`*`): $I_R(x,y) = I_1(x,y) \cdot I_2(x,y)$
- Protected division (`/`): $I_R(x,y) = \begin{cases} 1 & \text{if} \quad I_2(x,y) = 0 \\ I_1(x,y)/I_2(x,y) & \text{otherwise} \end{cases}$

Figure 1 shows how some of the primitive functions could be used to build an operator which calculates the determinant of the Hesse matrix.

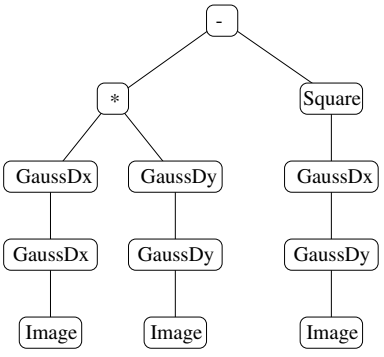


Fig. 1. Example of an existing operator which was manually constructed from the set of primitive functions.

4.3 Fitness Measure

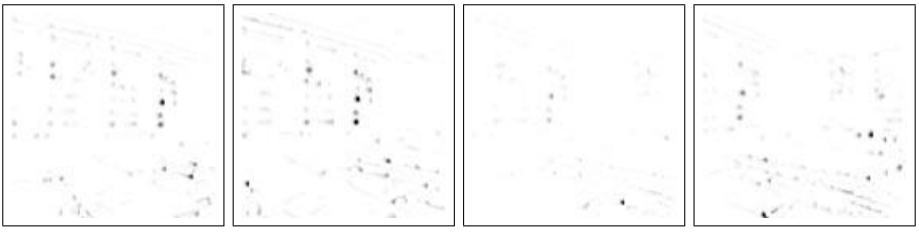
The different criteria have to be integrated into one fitness measure. We have to do multi-objective optimization to evolve a detector which is optimal according to all of the criteria. An overview about multi-objective optimization is given by Fonseca and Fleming [8]. To integrate the different measures into one we calculate the average of each measure over all fitness cases. Let $\bar{m}(i)$ be the average of the measure m for the individual i. Next, we normalize them across all individuals in the population. This gives us a selection probability $p_c(i) = \frac{\bar{m}_c(i)}{\sum_j \bar{m}_c(j)}$ for each criterion c and individual i. The selection probabilities were combined into a single fitness function $\mathrm{f} = \prod_i p_i$. The combined fitness reaches its maximum value only if all of the different selection probabilities have a large value. The normalization is not necessary for the multiplicative contribution of the different measures. We normalize them, because in other experiments an additive contribution was used.

Table 1. Comparison between different interest operators and the evolved interest operator. Absolute fitness is computed as fitness $= \Pi_i \bar{m}_i$ which is used as an absolute measure to compare the different operators.

Name of operator	$\bar{m}_1$	$\bar{m}_2$	$\bar{m}_3$	$\bar{m}_4$	$\bar{m}_5$	$\bar{m}_6$	Absolute fitness
Kitchen-Rosenfeld [27]	54.33	0.9850	0.4669	0.6710	0.9294	0.6154	9.592
Det(H_I) [27,19]	51.67	0.9853	0.5386	0.7396	0.9635	0.6568	12.83
Moravec [20]	49.33	0.9848	0.5242	0.6912	0.9359	0.7293	12.01
SUSAN [29]	77.00	0.9867	0.5426	0.5318	0.9195	0.6042	12.18
Diff. of Gabor filters[36]	64.00	0.9865	0.5967	0.5803	0.9348	0.6891	14.08
Evolved	131.0	0.9871	0.7814	0.5504	0.9170	0.6620	33.77

Fig. 2. Image sequence which was used during evolution.

Response of the operator:



Extracted points:

Sparse optical flow:

Fig. 3. Best individual from generation 50. The first row shows the response of the evolved operator. The second row shows the extracted interesting points. The third row shows the computed sparse optical flow.

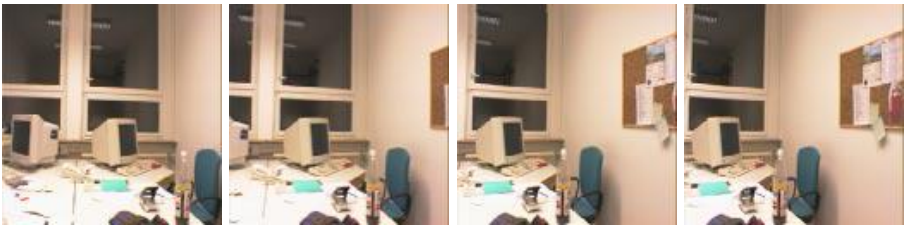
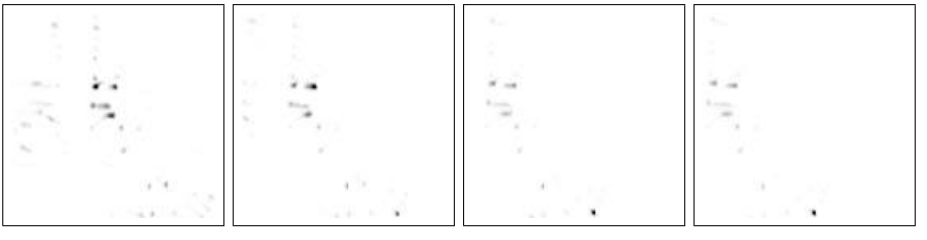
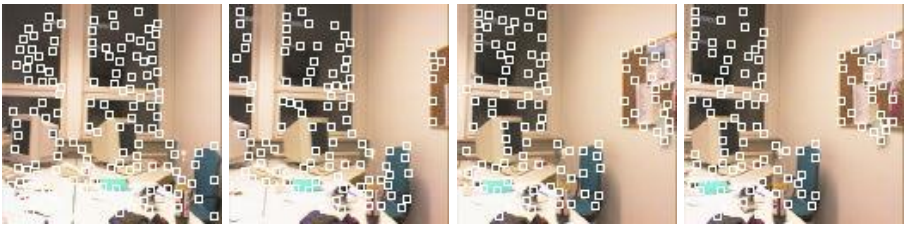


Fig. 4. Image sequence which was used to test the evolved operator.

Response of the operator:



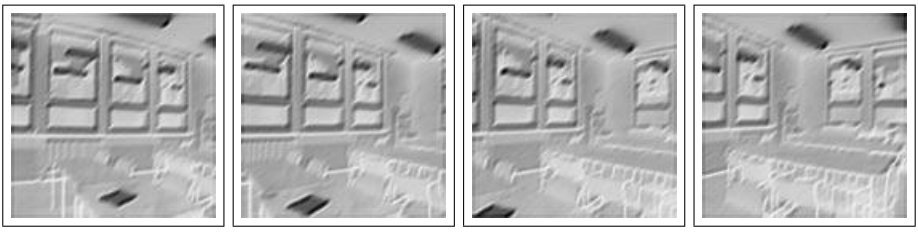
Extracted points:



Sparse optical flow:

Fig. 5. Results of the evolved operator on a test sequence.

Response of the operator:



Extracted points:

Sparse optical flow:

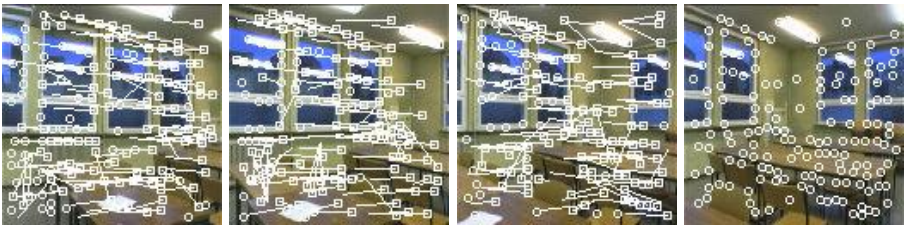
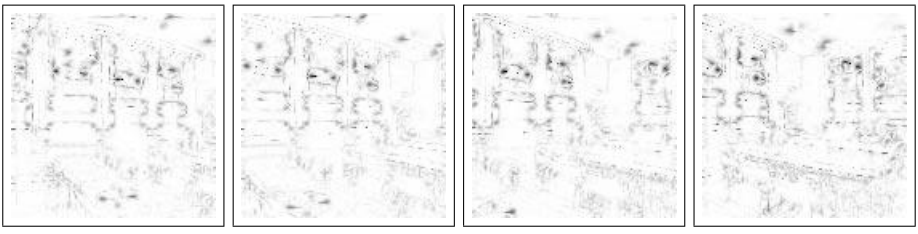


Fig. 6. Best individual from the first generation.

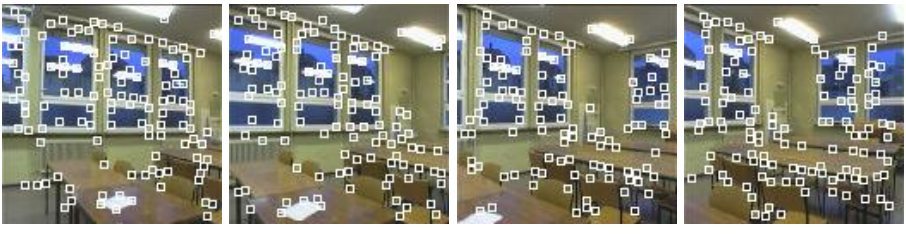
4.4 Results

With the above representation we evolved an interest operator. We used a sequence of 4 images with sizes 128×128 shown in Figure 2. The major parameters of the run were as follows. We used a population size of 500 individuals. The experiment was run for 50 generations. A limit of 1000 nodes and a maximum possible depth of the trees of 17 was used. Tournament selection with size 7 was used and crossover, reproduction and mutation probabilities were set to 85%, 10% and 5%, respectively. The results of the experiment are displayed in Figure 3. The first row shows the response of the best evolved operator from generation 50. The second row shows the extracted interesting points and the third row shows the computed sparse optical flow. The evolved operator was tested on an additional image sequence shown in Figure 4. The results achieved with the evolved operator on the test sequence is shown in Figure 5. The response of the best operator which was found in the first generation of the experiment applies the `Gabor2` operator twice. It extracts edges which are oriented in direction $\frac{\pi}{4}$ (Fi-

Response of the operator:



Extracted points:



Sparse optical flow:

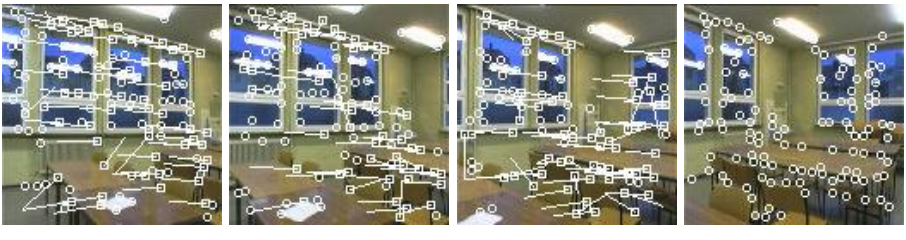
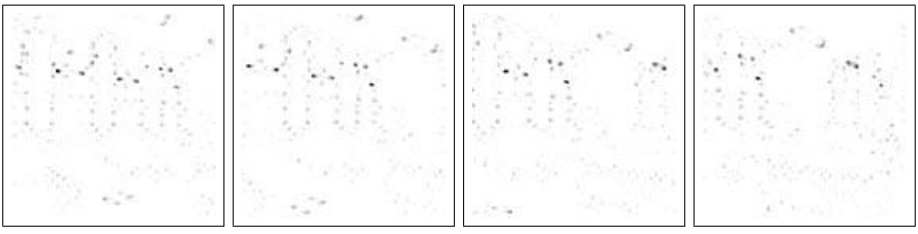


Fig. 7. Results achieved with the Kitchen-Rosenfeld corner detector [27].

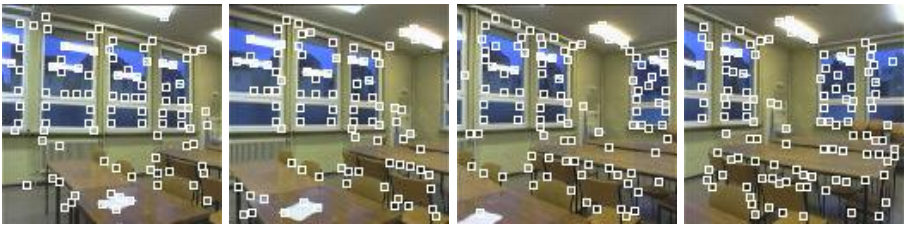
gure 6). Table 1 shows the performance of the evolved operator in comparison to the Kitchen-Rosenfeld corner detector [27], the determinant of the Hesse matrix [27,19], the Moravec operator [20], the SUSAN operator [29], and the difference of Gabor filters [36]. The results of these operators are shown in Figure 7, Figure 8, Figure 9, Figure 10 and Figure 11 respectively. For some quality measures the evolved operator performed better than the other operators whereas for others it performed worse. Selection, however, is done according to the overall fitness. The evolved operator clearly outperformed the existing operators in terms of the overall fitness.

As can be seen the evolved operator highlights regions in the image that are of particular interest for the calculation of sparse optical flow. Some wrong matches are also produced. This is due to the fact that the operator combines different possibly contradicting measures. For instance the number of points extracted should be high and at the same time the matches should be unambiguous. Fitness statistics for the experiment can be found in Figure 12. Except for the Gaussian filter all of the available functions occurred in the evolved individual. The division

Response of the operator:



Extracted points:



Sparse optical flow:

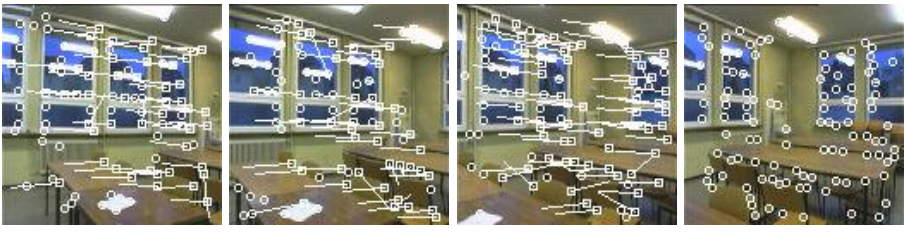


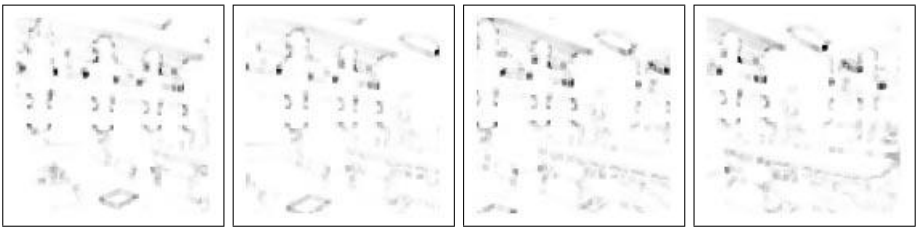
Fig. 8. Results achieved with the determinant of the Hesse matrix [27,19].

operation, derivative of the Gaussian in y direction, average, Gabor filters, the square root and the square function were used several times.

5 Conclusion

We have shown that task specific image operators may be evolved using genetic programming. Different criteria are used to evolve operators which are optimal for the task at hand. As a sample task we evolved an interest operator for the computation of sparse optical flow. The following criteria were used to evolve the interest operator. a) A large number of matches should be produced. b) The quality of the matches should be good. c) The relation of matched points to unmatched points should be high. d) Matches should be unambiguous, e) the flow field should be rather smooth and f) have a maximum density. These criteria led to the evolution of an operator which can be used to extract interesting points from an image.

Response of the operator:



Extracted points:

Sparse optical flow:

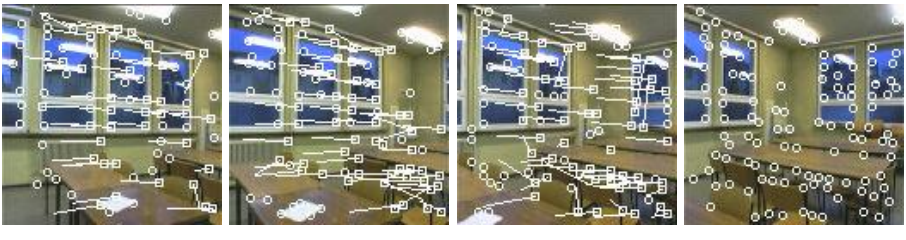


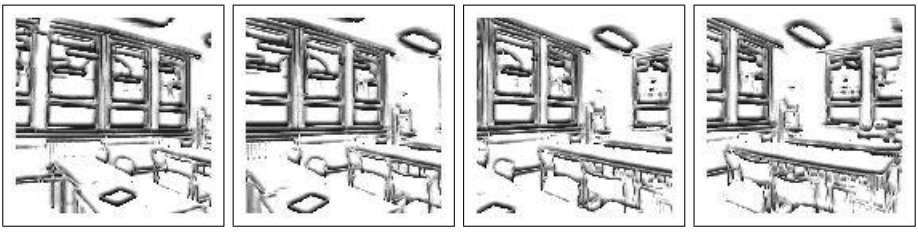
Fig. 9. Results achieved with the Moravec operator [20].

Provided that the fitness evaluation can be done fast enough it might be possible to construct adaptive vision systems which are able to adapt themselves to changing environmental conditions. Just as the pupil's diameter adapts to changing brightness conditions [31] an artificial visual system might evolve optimal or near optimal image processing operators on the fly. At present, however, the evolution is performed offline and evolution of an operator from scratch takes several days to complete on a single PC.

6 Acknowledgements

This work was supported in part by a scholarship to the first author according to the Landesgraduiertenförderungsgesetz. For our experiments we used the lilgp Programming System [37]. For image processing we used the Vista software environment [23].

Response of the operator:



Extracted points:

Sparse optical flow:

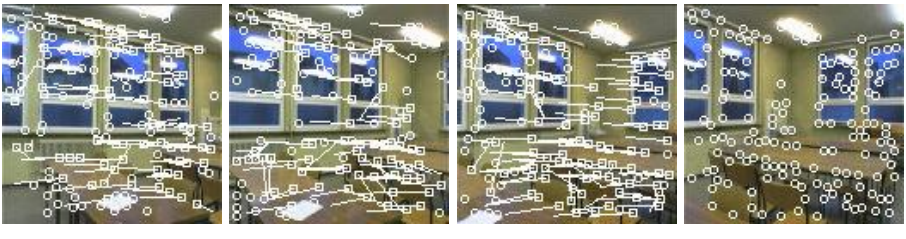
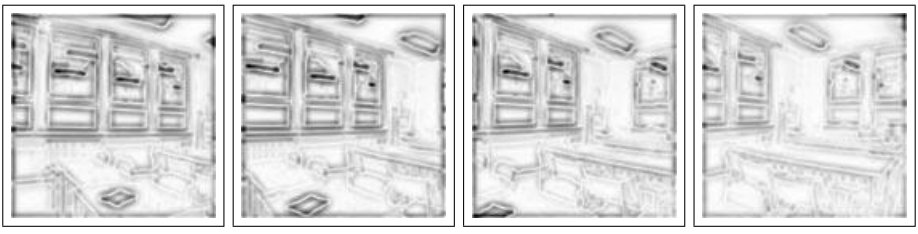


Fig. 10. Results achieved with the SUSAN operator [29].

References

1. D. Andre. Automatically defined features: The simultaneous evolution of 2-dimensional feature detectors and an algorithm for using them. In K. E. Kinnear Jr., editor, *Advances in Genetic Programming*, pp. 477–494, Cambridge, Massachusetts, 1994. The MIT Press.
2. W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone. *Genetic Programming - An Introduction: On The Automatic Evolution of Computer Programs and Its Applications*. Morgan Kaufmann Publishers, San Francisco, California, 1998.
3. A. K. Bhattacharjya and B. Roysam. Joint solution of low, intermediate, and high-level vision tasks by evolutionary optimization: Application to computer vision at low SNR. *IEEE Transactions on Neural Networks*, 5(1):83–95, January 1994.
4. R. R. Brooks, S. S. Iyengar, and J. Chen. Automatic correlation and calibration of noisy sensor readings using elite genetic algorithms. *Artificial Intelligence*, 84:339–354, 1996.

Response of the operator:



Extracted points:

Sparse optical flow:

Fig. 11. Results achieved with the difference of Gabor filters [36].

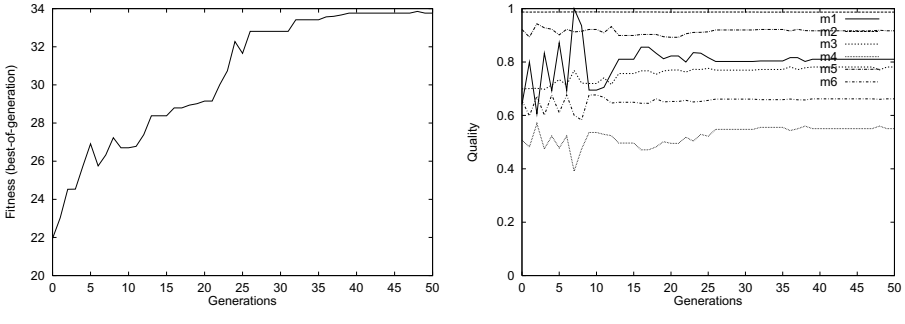


Fig. 12. The absolute fitness (best of generation) is shown on the left. Absolute fitness is calculated as fitness $= \Pi_i \bar{m}_i$. The different quality measures which belong to the individual with the highest fitness are shown on the right. The first quality measure (number of matches) was normalized to the range of [0,1] to integrate the measure into the same diagram.

5. J. M. Daida, J. D. Hommes, T. F. Bersano-Begey, S. J. Ross, and J. F. Vesecky. Algorithm discovery using the genetic programming paradigm: Extracting low-contrast curvilinear features from sar images of arctic ice. In P. J. Angeline and K. E. Kinnear, Jr., editors, *Advances in Genetic Programming Volume II*, pp. 417–442, Cambridge, Massachusetts, 1996. The MIT Press.
6. M. Ebner. On the evolution of edge detectors for robot vision using genetic programming. In H.-M. Groß, editor, *Workshop SOAVE '97 - Selbstorganisation von Adaptivem Verhalten, VDI Reihe 8 Nr. 663*, pp. 127–134, 1997. VDI Verlag.
7. M. Ebner. On the evolution of interest operators using genetic programming. In R. Poli, W. B. Langdon, M. Schoenauer, T. Fogarty, and W. Banzhaf, editors, *Late Breaking Papers at EuroGP'98: the First European Workshop on Genetic Programming*, pp. 6–10, Paris, France, 1998. The University of Birmingham, UK.
8. C. M. Fonseca and P. J. Fleming. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, 3(1):1–16, 1995.
9. C. Harris and B. Buxton. Evolving edge detectors with genetic programming. In J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, editors, *Genetic Programming 1996, Proceedings of the First Annual Conference*, pp. 309–314, Cambridge, Massachusetts, 1996. The MIT Press.
10. R. Jain, R. Kasturi, and B. G. Schunck. *Machine Vision*. McGraw-Hill, NY, 1995.
11. M. P. Johnson, P. Maes, and T. Darrell. Evolving visual routines. In R. A. Brooks and P. Maes, editors, *Artificial Life IV, Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, pp. 198–209, Cambridge, Massachusetts, 1994. The MIT Press.
12. T. Kalinke and W. von Seelen. Entropie als Maß des lokalen Informationsgehalts in Bildern zur Realisierung einer Aufmerksamkeitssteuerung. In B. Jähne, P. Geißler, H. Haußecker, and F. Hering, editors, *Mustererkennung 1996, 18. DAGM-Symposium, Heidelberg*, pp. 627–634, Berlin, 1996. Springer-Verlag.
13. A. J. Katz and P. R. Thrift. Generating image filters for target recognition by genetic learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(9):906–910, September 1994.
14. J. R. Koza. *Genetic Programming, On the Programming of Computers by Means of Natural Selection*. The MIT Press, Cambridge, Massachusetts, 1992.
15. J. R. Koza. *Genetic Programming II, Automatic Discovery of Reusable Programs*. The MIT Press, Cambridge, Massachusetts, 1994.
16. J. Lampinen and E. Oja. Distortion tolerant pattern recognition based on self-organizing feature extraction. *IEEE Transactions on Neural Networks*, 6(3):539–547, May 1995.
17. M. S. Lew, T. S. Huang, and K. Wong. Learning and feature selection in stereo matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(9):869–881, September 1994.
18. R. Lohmann. Selforganization by evolution strategy in visual systems. In H.-M. Voigt, H. Mühlenbein, and H.-P. Schwefel, editors, *Evolution and Optimization '89*, pp. 61–68. Akademie-Verlag, 1990.
19. H. A. Mallot. *Sehen und die Verarbeitung visueller Information, Eine Einführung*. Vieweg, Braunschweig, 1998.
20. H. P. Moravec. Towards automatic visual obstacle avoidance. In *Proc. of the 5th International Joint Conference on Artificial Intelligence*, Vision–1: p. 584, 1977.
21. Riccardo Poli. Genetic programming for image analysis. In J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, editors, *Genetic Programming 1996, Proceedings of the First Annual Conference*, pp. 363–368, Cambridge, Massachusetts, 1996. The MIT Press.

22. R. Poli and S. Cagnoni. Genetic programming with user-driven selection: Experiments on the evolution of algorithms for image enhancement. In J. R. Koza, K. Deb, M. Dorigo, D. B. Fogel, D. B. Fogel, M. Garzon, H. Iba, and R. L. Riolo, editors, *Genetic Programming 1997, Proceedings of the Second Annual Conference*, pp. 269–277, 1996. Morgan Kaufmann Publishers.
23. A. R. Pope and D. G. Lowe. Vista: A software environment for computer vision research. In *Proceedings of the 1994 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 768–772. IEEE, 1994.
24. D. Reisfeld, H. Wolfson, and Y. Yeshurun. Detection of interest points using symmetry. In *Proceedings of the International Conference on Computer Vision, Osaka, Japan*, pp. 62–65. IEEE, December 1990.
25. M. M. Rizki, L. A. Tamburino, and M. A. Zmuda. Evolving multi-resolution feature-detectors. In D. B. Fogel and W. Atmar, editors, *Proceedings of the Second American Conference on Evolutionary Programming*, pp. 108–118. Evolutionary Programming Society, 1993.
26. G. Roth and M. D. Levine. Geometric primitive extraction using a genetic algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(9):901–905, September 1994.
27. M. A. Shah and R. Jain. Detecting time-varying corners. *Computer Vision, Graphics, and Image Processing*, 28:345–355, 1984.
28. J. Shi and C. Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 593–600, 1994.
29. S. Smith. A new class of corner finder. In *Proceedings of the 3rd British Machine Vision Conference 1992*, pp. 139–148, 1992.
30. W. A. Tackett. Genetic programming for feature discovery and image discrimination. In S. Forrest, editor, *Proceedings of the Fifth International Conerence on Genetic Algorithms*, pp. 303–309. Morgan Kaufmann, 1993.
31. M. J. Tovée. *An introduction to the visual system.* Cambridge University Press, Cambridge, 1996.
32. S. Ullman. Visual routines. In M. A. Fischler and O. Firschein, editors, *Readings in Computer Vision: Issues, Problems, Principles, and Paradigms*, pp. 298–328, Los Altos, California, 1987. Morgan Kaufmann Publishers.
33. J. F. Winkeler and B. S. Manjunath. Genetic programming for object detection. In J. R. Koza, K. Deb, M. Dorigo, D. B. Fogel, D. B. Fogel, M. Garzon, H. Iba, and R. L. Riolo, editors, *Genetic Programming 1997, Proceedings of the Second Annual Conference*, pp. 330–335, San Francisco, California, 1997. Morgan Kaufmann Publishers.
34. M. Xie. Automatic feature matching in uncalibrated stereo vision through the use of color. *Robotics and Autonomous Systems*, 21:355–364, 1997.
35. H. Zabrodsky, S. Peleg, and D. Avnir. Symmertry as a continuous feature. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(12):1154–1165, 1995.
36. Q. Zheng and R. Chellappa. Automatic feature point extraction and tracking in image sequences for arbitrary camera motion. *International Journal of Computer Vision*, 15:31–76, 1995.
37. D. Zongker and B. Punch. *lil-gp 1.01 User's Manual (support and enhancements Bill Rand).* Michigan State University, March 1996.

Generation and Selection of Sensory Channels

Edwin D. de Jong[1] and Luc Steels[1,2]

[1] Vrije Universiteit Brussel
Artificial Intelligence Laboratory
Pleinlaan 2, 1050 Brussels, Belgium
{edwin, steels}@arti.vub.ac.be
http://arti.vub.ac.be
[2] Sony Computer Science Laboratory Paris
6, Rue Amyot, 75005 Paris, France
http://www.csl.sony.fr

Abstract. Sensory channels determine the way an agent views the world. We investigate the question of how sensory channels may be autonomously constructed using generation and selection. The context is the discrimination of geometric shapes. In a first experiment, elements of a solution were attributed fitness based on the part of the problem they solved. In two subsequent experiments, cooperation between elements was respectively required and encouraged by means of a fitness function which only rewards complete solutions. Differences between the approaches are discussed, and generation and selection is concluded to provide a successful mechanism for the autonomous construction of sensory channels.

Introduction

The discrimination game, introduced in [7], was developed for the investigation of category formation. The direct objective of a discrimination game is to distinguish a randomly selected object, called the *topic*, from the rest of the objects in a scene. Information is distilled from the objects by *sensory channels*, known as *feature extractors* in pattern recognition. A category is a range of values that a certain sensory channel may yield. Two objects can be distinguished if there is at least one sensory channel for which the objects' readings are in different categories. In the course of playing discrimination games, agents adapt their collection of categories. The eventual goal for an agent then is to acquire a set of categories that allow it to be successful in its discrimination task. In related research on language evolution, these categories are used by agents in lexicon formation experiments, see [8] for an overview.

In previous experiments, e.g. [3], [10], [2], sensory channels have always been determined by the experimenter. Here, we want to investigate whether it is possible to let the agent itself construct useful sensory channels for its discrimination task. The mechanism for this construction is generation and selection based on a set of rudimentary functions that can be used in combination with each

R. Poli et al. (Eds.): EvoIASP'99 and EuroEcTel'99, LNCS 1596, pp. 90–100, 1999.

other. The approach can be seen as a form of genetic programming [5] without crossover. This implies that the agent assembles a program which calculates information about the objects. In the experiments reported here, the objects are geometric shapes. An example of a sensory channel that could be generated is `(BREAK-FIGURE ANGLES AVG)`. The functions are applied in order, each one taking the result of the previous function as input. This function breaks a figure into curves, as will be explained later, computes the angles between subsequent curves, and returns the average of these angles.

We are not aware of any related work on generation and selection of sensory channels within the context of discrimination games. The discrimination game could be viewed as a special case of unsupervised learning. In [1], a supervised learning problem is approached using a hybrid method for feature selection. The system uses a genetic algorithm to search for subsets of features to be used in a pattern recognition task. The fitness of features is partly evaluated by computing the performance of a classifier built by inductive learning from these features. In [9], visual routines are described that are oriented towards a specific goal, rather than aimed at building a general model as known from David Marr's work [6]. Johnson et.al. [4] evolve this kind of visual routines using genetic programming with point and edge based functions as primitives.

The structure of the paper is as follows. Section 1 briefly describes the basics of the discrimination game. Section 2 is concerned with the generation of sensory channels; selection is discussed in section 3. The results of the experiments are presented in sections 4, which is followed by conclusions.

1 The Discrimination Game

The categories associated with a sensory channel are organized as a tree. During normal operation, this tree grows and shrinks dynamically as the agent refines categories and removes ineffective refinements. Figure 1 shows an example of such a tree. The root node represents the entire range of values that the sensory channel may produce. At each internal node, this interval is split into its two halves.

The course of the discrimination game is as follows. For each object in a scene, the agent receives the values extracted by its current set of sensory channels. The agent randomly selects one object to be the topic. The value of a sensory channel for a certain object lies in several categories of that channel's category tree. This is because categories in any branch of the tree are subranges of strictly monotonously decreasing size of the complete range, which is represented by the root of the tree. For each category it can be determined which objects fall within it and which do not. If there exists a single category in which the topic and no other objects lie, or in which all of the other objects lie but not the topic, then this category allows the agent to perform the desired discrimination. If this is not the case, all possible sets of categories up to a specified size are considered. This may produce a set of categories where each category discriminates the topic from some, but not all, of the other objects, and each object can be discriminated

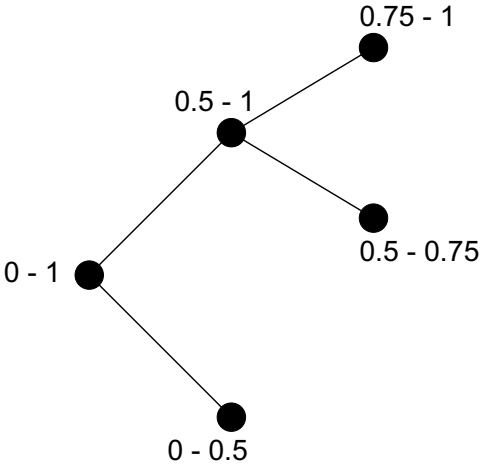


Fig. 1. Example of a tree representing the categories of a sensory channel

from the topic by at least one category. When no such combination is found, the game ends in a failure. The discrimination success is the ratio between successful and unsuccessful games.

2 Generation of Sensory Channels

We want to investigate whether it is possible to let the agent itself construct sensory channels using generation and selection. Sensory channels are generated by an essentially random process of combining basic functions. To speed up the process, impossible combinations of functions are excluded. Selection takes place on the basis of discrimination success.

As was stated above, agents normally adapt the category trees by adding and removing categories. Since we are interested in the feasibility of autonomous sensory channel construction, rather than in the specific categories that are used, sensory channels should be treated equally, so that the evaluation of the channels is not biased by other factors than the quality of the channel. Therefore, each channel's category tree is explored up to a certain depth, which is equal for all channels, and trees are not adapted by the agent.

Scenes contain a random number of these randomly generated geometrical shapes: circles, rectangles, squares, triangles, trapezia. Figure 2 shows an example of such a scene. Table 1 lists the functions used for generating sensory channels. Channels are sequences of functions that are applied sequentially to the internal representation of a shape. They are created by assuming 'figure' as the first input type and randomly selecting functions with appropriate input types until the final result is of type 'number'. Since there is a limited number of possibilities to choose from at every step, this leads to functions of a finite length in practice, and it is not necessary to limit the number of functions in a channel,

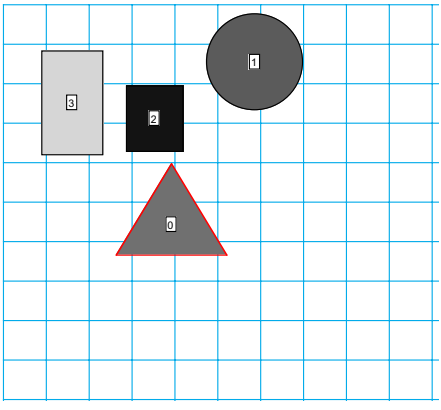


Fig. 2. Example of a scene

even though this may become infinite in theory. Therefore, it is not necessary to develop a genetic code for the channels, and no crossover is performed.

The only function applicable to a complete shape is `break-figure`, which yields a list of *curves*. The idea of this function is that a contour is split into several segments based on the degree of curvature. Segments may be curved, but, when seen as a circle segment, the allowed variation in the radius of this circle is limited. Whenever the curvature does change substantially, this signifies the end of the segment and the beginning of a new one. In general, this decision procedure would need to be defined more precisely, but since all figures here are geometric and functions are based directly on the internal descriptions of the shapes, their segmentation is known beforehand; a circle is a single segment, since by definition its degree of curvature (the reciprocal of the radius) is constant when measured along its contour, a triangle has three discontinuous (straight) curves, and squares, rectangles and trapezia four. After segmenting, the angles between subsequent segments can be computed, as well as the lengths and curvatures of the segments themselves. All these functions yield a list of numbers, to which arithmetical functions and set functions may be applied.

The generation aspect of our approach is determined by the choice of functions and the ways in which they may be combined. Ideally, all information contained in a figure should be extracted by some function. In this case, the choice of functions was rather straightforward; the angles between curves, their lengths and their degree of curvature are all important properties that should be available as a possible basis for a channel. Although more sophisticated functions are conceivable, these functions described seemed sufficient as a basis for the generation process. Furthermore, standard set operations and arithmetical functions are provided. Concerning the combination of the functions, we opted for the most basic form, i.e. combining functions sequentially.

Table 1. Functions used for generating sensory channels and their applicability

Input type	Function	Output type
figure	`break-figure`	list of curves
list of curves	`curvatures`	list of numbers
list of curves	`lengths`	list of numbers
list of curves	`angles`	list of numbers
list of numbers	`nth subsequence of length m`	list of numbers
list of numbers	`sum`	number
list of numbers	`difference`	number
list of numbers	`average`	number
list of numbers	`first`	number
list of numbers	`nth element`	number

3 Selection of Sensory Channels

Having fixed the generation part, the remaining degree of variation is the selection component. This is the more interesting part. Standard genetic algorithms assume an n-dimensional search space where solutions are points in this space, and solutions are compared with respect to fitness. In these terms, a solution in our case is a set of sensory channels. A good set of sensory channels will allow an agent to achieve high discrimination success, which can be seen as the fitness of that solution. Thus, one approach would be to randomly generate sets of sensory channels, evaluate them by playing discrimination games, and select the successful ones.

Although it would certainly be possible to take this approach, there is a possibility to exploit the structure of the problem by making better use of the feedback provided by the discrimination game. Since the function of a sensory channel is to discriminate the topic from other objects, the degree to which it succeeds in this can be determined *per channel*. Thus, it appears more efficient to evaluate *each* sensory channel apart, rather than a set of channels. The successful sensory channels are then allowed to remain, while ineffective channels are removed and replaced by new, randomly generated sensory channels. When this approach is adopted, the question that remains is how to calculate the success of a sensory channel. The results of three approaches that have been investigated are reported in the next section.

4 Experiments

4.1 Rewarding Partial Solutions

In the first experiment, success is attributed at the lowest possible level, i.e. the category level. Every category can be used to discriminate the topic from zero or more of the other objects. It would seem that the higher the ratio between

distinguished and non distinguished objects is, the more desirable the sensory channel is. Although the outcome of a discrimination game is binary (1 if all objects can be distinguished from the topic, zero if not), the fraction of distinguished objects provides more detailed information, and therefore this quantity is stored for each category. At the end of a series of discrimination games, the sensory channel for which categories have the lowest average fraction is discarded and replaced by a new sensory channel. The number of sensory channels thus remains constant, and was limited to 5 in these experiments.

Figure 3 shows the discrimination success over time. Each of the 500 datapoints represents the fraction of successful discrimination games in a series of 100 games. Scenes are randomly generated and contain a random number, from 2 up to 11, of geometric figures. After every series of games, the set of sensory channels is adapted using generation and selection as described above. In order to obtain reliable results, the experiment has been repeated ten times; the graph shows the average value over these runs.

Although the absolute success is acceptable, there is a consistent drop of the success after an initial rise. Apparently, the generation and selection process is not functioning effectively. An analysis of the selection process revealed three phases. First, ineffective elements in the (random) initial set of sensory channels are replaced by more successful channels. This causes the initial increase in the success. Then there is a phase where four of the five channels are better than average. The remaining channel may be any channel, since after each series of games the least effective channel is replaced by a randomly generated new channel. This phase is characterized by a high succes rate. Eventually however, in all of the ten runs, the system converges to a situation where four of the five channels are identical. Apparently, there is a single sensory channel which is more successful than any other channel. This channel calculates the length of the third curve of a figure, and has been encountered in several equivalent forms, a.o. `(BREAK-FIGURE LENGTHS (NTH 2))` and `(BREAK-FIGURE LENGTHS (NTH-M-TUPLE 1 4) (NTH-M-TUPLE 2 2) (NTH 1))`. This last channel breaks a figure into curves, calculates the lengths of these curves, takes the first subsequence of length 4, then the second subsequence of length two of that sequence, and finally the second element of this list of numbers, which produces the same answer as the first channel.

Clearly, merely selecting the most successful elements of solutions does not yield optimal results. The semi-stable situation in the middle phase of the experiments produced better solutions because of the diversity of the sensory channels. This diversity allowed the agent to combine channels which, although each only discriminates a subset of the objects, together do the whole job. This led us to think that the complementary aspect of elements in a solution may sometimes be more important than the quality of the elements themselves. This hypothesis was investigated in two subsequent experiments.

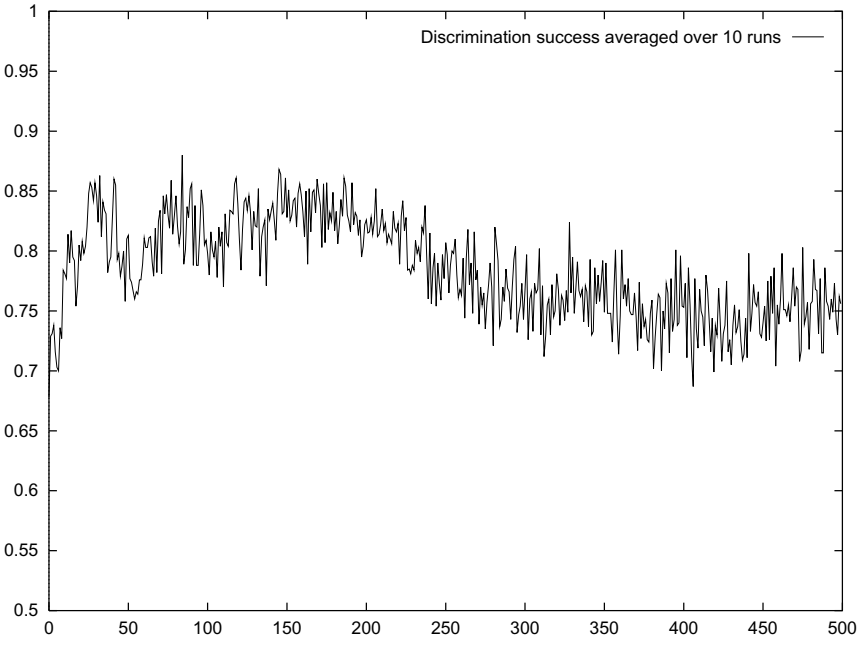


Fig. 3. Discrimination success when sensory channels are evaluated based on their individual capacity to discriminate

4.2 Cooperation between Sensory Channels

Two experiments have been performed where cooperation between sensory channels is investigated. In the first of these experiments, cooperation is *required* of solutions by only attributing fitness to solutions which consist of two of more channels and which resulted in a successful discrimination game, i.e. all objects could be discriminated from the topic. In the second experiment, we also only reward solutions which resulted in a successful discrimination game, but no constraint is placed on the number of sensory channels. This still favours groups of channels that complement each other, resulting in complete discrimination success, over groups of similar channels where each channel individually achieves high discrimination success but whose combination does not yield complete success. Thus, cooperation is encouraged in this scheme, but not required.

Enforcing Cooperation In this experiment, cooperation between sensory channels is enforced by only rewarding solutions consisting of more than one sensory channel. In these solutions, cooperation must take place, since the fact that a solution contains more than one channel implies that no single discriminative sensory channel has been found. Fitness was only attributed in cases where the discrimination game is successful. Thus, information about partial discrimination (e.g. when 6 out of 8 objects can be discriminated from the topic) is not

used. At first sight, it may seem that this method of selection makes less effective use of the feedback on discrimination. However, the results show that even if this may be the case, it is overshadowed by a larger advantageous effect of cooperation between channels.

Figure 4 shows the success, again averaged over ten runs, of 500 series of 100 discrimination games. This time, discrimination success increases steadily over time, without the fallbacks that characterized the first method. Furthermore, the level that is eventually reached is substantially higher; the average discrimination success surpasses 0.90, whereas in the first experiment it continued to vary around 0.75.

Whereas the first experiment consistently converged to four identical sensory channels based on the length of the curves, this experiment showed that encouraging cooperation leads to diversity. In each of the ten runs, the resulting set of sensory channels contained four different channels [1]. In all but one case, these four channels included the two channels based on curve length that were observed in the semi-stable solutions of the successful middle phase in the first experiment. Moreover, solutions now consistently included channels based on the angles between subsequent curves.

Encouraging Cooperation In the final experiment, fitness is attributed in cases where the discrimination game is successful. In contrast with the previous experiment, no constraint is placed on the number of channels in a solution. This has the effect that for the solutions containing more than one channel, the ones containing complementary channels have a higher chance of being rewarded than solutions of equally strong channels that do not complement each other. Thus, cooperation between channels is encouraged, but in contrast with the second experiment, it is not required, since single channels allowing perfect discrimination are also rewarded.

Figure 5 shows the success of this scheme averaged over ten runs of 500 series of 100 discrimination games. As in the previous experiment, success increases steadily over time, but here the level that is eventually reached is still higher, around 0.95. In contrast with the enforced cooperation experiment, these functions were not always based on different basis functions. The majority of the functions was based on length. However, the solutions were still diverse; in every case, the final set of functions did not contain any duplicate function.

The success of this experiment can be explained by the nature of the solutions; these contain selective channels, and are diverse at the same time. Apparently, the strong focus on complementariness in the enforced cooperation experiment rules out some solutions with functions that can be discriminative on their own. Furthermore, the encouragement of cooperation in the last experi-

[1] The sensory channel that was added most recently is not taken into account, since it was randomly generated at the previous time step. As the newly added channel is usually less effective then the remaining channels, it is normally this channel that is removed again at the next selection step. Thus the four remaining channels come to form a stable part of the solution.

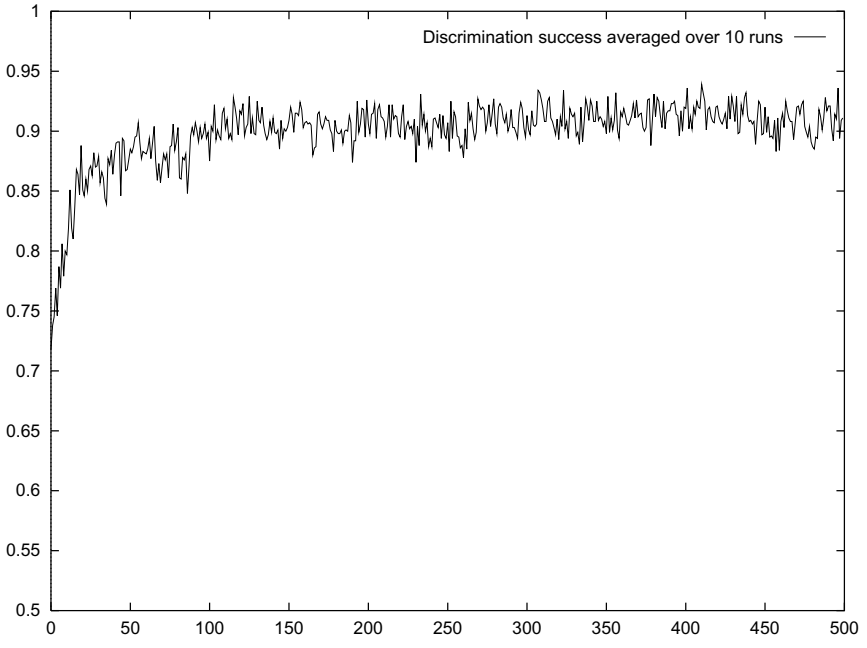


Fig. 4. Discrimination success when cooperation between the sensory channels in a solution is *required*

ment secures a diversity of the solutions that explain its advantage over the first experiment.

Conclusions

In previous work on discrimination games, sensory channels, which determine the way an agent sees the world, have always been programmed by the experimenter. The goal of this research was to investigate whether it is possible to let the agent construct these sensory channels autonomously by means of generation and selection based on a set of basic functions. This question can be answered positively.

As part of the investigation, three experiments on generation and selection of sensory channels have been performed. From an evolutionary computation point of view, the element of variation in these experiments is a set of sensory channels. Since the discrimination games provide information about the fitness of each separate channel, the first experiment made use of this information as much as possible. This resulted in solutions containing multiple instances of equivalent sensory channels, and hence a lack of useful complementary channels. In two other experiments, cooperation between sensory channels was investigated. First, cooperation of sensory channels was *required* of the solutions. This improved performance substantially. However, it had the drawback of ruling out some

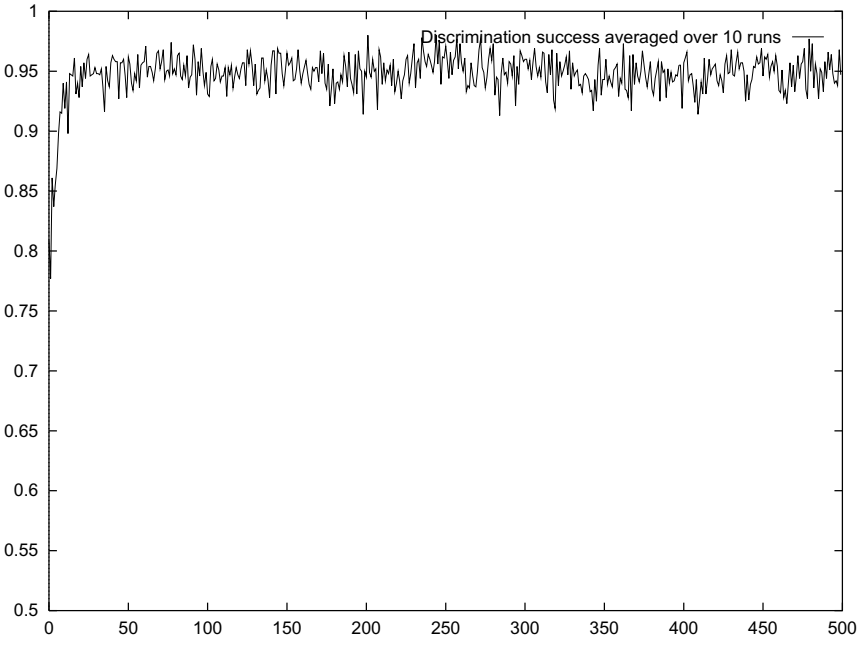


Fig. 5. Discrimination success when cooperation between the sensory channels in a solution is *encouraged*

solutions which contain highly selective channels and are diverse at the same time. This problem was not present in the third experiment, where cooperation was not required but encouraged, by attributing success only when the complete discrimination task was solved. The judicious encouragement of cooperation turned out to result in a desirable combination of diversity and selectiveness, which explains the favourable outcome of the final experiment.

It may be concluded that autonomous construction of sensory channels is possible using generation and selection, and that encouraging, but not requiring cooperative solutions may result in more diversity and better performance than attributing fitness at the lowest possible structural level.

5 Acknowledgements

For this research, grateful use was made of the BABEL package, developed by Angus McIntyre at Sony CSL, Paris. Thanks furthermore to Bart de Boer for useful discussion and to Paul Vogt and Tony Belpaeme for comments on an earlier version.

References

1. J. Bala, K. De Jong, J. Huang, H. Vafaie, and H. Wechsler. Using learning to facilitate the evolution of features for recognizing visual concepts. *Evolutionary Computation*, 4(3), 1996.
2. T. Belpaeme, L. Steels, and J. Van Looveren. The construction and acquisition of visual categories. In A. Birk and J. Demiris, editors, *Proceedings of the 6th European Workshop on Learning Robots EWLR-6*, Lecture Notes on Artificial Intelligence, Berlin, 1998. Springer-Verlag.
3. E.D. de Jong and P. Vogt. How Should a Robot Discriminate Between Objects? A comparison between two methods. In *Proceedings of the Fifth International Conference of The Society for Adaptive Behavior SAB'98*, volume 5, Cambridge, MA, 1998. The MIT Press.
4. M. P. Johnson, P. Maes, and T. Darrell. Evolving visual routines. In R. A. Brooks and P. Maes, editors, *ARTIFICIAL LIFE IV, Proceedings of the fourth International Workshop on the Synthesis and Simulation of Living Systems*, pages 198–209, Cambridge, MA, 6-8 July 1994. The MIT Press.
5. J. R. Koza. *Genetic Programming*. MIT Press, Cambridge, MA, 1992.
6. D. Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. W. H. Freeman, New York, 1982.
7. L. Steels. Perceptually grounded meaning creation. In M. Tokoro, editor, *Proceedings of the International Conference on Multi-Agent Systems ICMAS'96. Kyoto, Japan.*, 1996.
8. L. Steels. The synthetic modeling of language origins. *Evolution of Communication*, 1(1):1–34, 1997.
9. S. Ullman. *Readings in Computer Vision*, chapter Visual routines, pages 298–328. Morgan Kaufmann Publishers, 1987.
10. P. Vogt. Perceptual grounding in robots. In A. Birk and J. Demiris, editors, *Learning Robots, Proceedings of the EWLR-6, Lecture Notes on Artificial Intelligence 1545*. Springer, 1998.

Selecting Filter Banks to Enhance Evoked Potentials Recordings Using Evolutionary Algorithms

SJ Turner, PD Picton, JA Campbell
University College Northampton, Northampton, NN2 6JD, UK
scott.turner@nene.ac.uk

Abstract. Evoked potentials are electrical signals produced by the body in response to a stimulus. In general these signals are noisy with a low signal to noise ratio. In this paper a method is proposed that uses sets of filters, whose cut-off frequencies are selected by an evolutionary algorithm. An evolutionary algorithm was investigated to limit the assumptions that were made about the signals. The set of filters separately filter the evoked potentials, and are combined as a weighted sum of the filter outputs. The evolutionary algorithm also selects the weights. Inputs to the filters are sets of averaged signal, 4 or 10 signals per average. Even though there is likely to be variations between the signals, this process can improve the extraction of potentials.

1. Introduction

Evoked potentials (or evoked responses) are electrical signals recorded from a human body in response to a stimulus to the nervous system. Somatosensory evoked potentials in particular are recorded at sites such as the scalp or spine, ordinarily due to direct electrical stimulation of the nerves in the arms or legs. The features looked for are negative or positive peaks at certain known values, e.g. at 20msec or 300msec. The main problem with evoked potentials is the presence of noise from, for example, other sources within the body, recording equipment, or the local environment [1]. Noise can dominate the recorded signal, leading to a very low signal to noise ratio. There are several difficulties with this noise, one of which is that the spectral components of the noise overlap the same region as those of the evoked potential. This means that just applying a bandpass filter will not extract the evoked potential, and the noise components are often larger than those of the evoked potential. Ensemble averaging is the most commonly used method of reducing the noise in evoked potential recordings. The main disadvantage of this method is that to produce a reasonably noise-free signal, a large number of signals need to be averaged. Collection of a large number of signals means that signals need to be collected over relatively long periods of time. Taking a long time to collect the data may be undesirable for the subject under going the tests, or even impractical, and variations between signals can lead to distortion of features in the averaged signal. After ensemble averaging, a

R. Poli et al. (Eds.): EvoIASP'99 and EuroEcTel'99, LNCS 1596, pp. 109–119, 1999.

single bandpass filter is often applied. The aim of this work is to investigate using a set of bandpass filters to reduce the number of signals that are needed to extract the evoked potential. The searching abilities of evolutionary algorithms were used to select appropriate filter parameters and weights.

2. Method

2.1 Equipment And Data

All the signals were recorded on FM tape, using a STORE 4 FM tape recorder (RACAL Recorders), from spinal recorded evoked potentials in response to stimulation of the median nerve at the wrist. The data was collected using a Gateway 2000 Pentium P90 computer via an interface card and data acquisition software (PC30F, Eagle Technology). All the filters and evolutionary algorithms were developed and implemented in MATLAB (MathWorks, USA). Before being recorded the signals were passed through a bandpass filter (0.016-750Hz).

Recorded data consisting of 222 responses were collected from the tape. A total of 38 responses were excluded from the experiments as they were found to contain artifacts such as 'clipping.' Using the remaining 184 recorded responses, two sets of data with 92 responses in each were formed into a test and a training set. These sets are referred to here, as the recorded data. An average of the 184 recorded responses was used to form a reference signal which was the target signal that the filters aim to extract. In addition it was possible to make simulated data by adding noise to this reference signal. Pre-stimulus recordings, i.e. electrical activity recorded just before stimulation occurred, was the source of the added noise. This was chosen as it represents electrical activity recorded at the same site as where the evoked responses were to be recorded and should therefore contain similar kinds of electrical activity as the background noise on the evoked response recordings. This simulated data (target signal + noise) set was split into a training set (55 responses) and a test set (56 responses).

2.2 Filter Banks

The arrangement of the filter bank is shown in Figure 1. The signal was passed through each filter separately. The output of this system was the response produced by a weighted sum of the individual filter outputs.

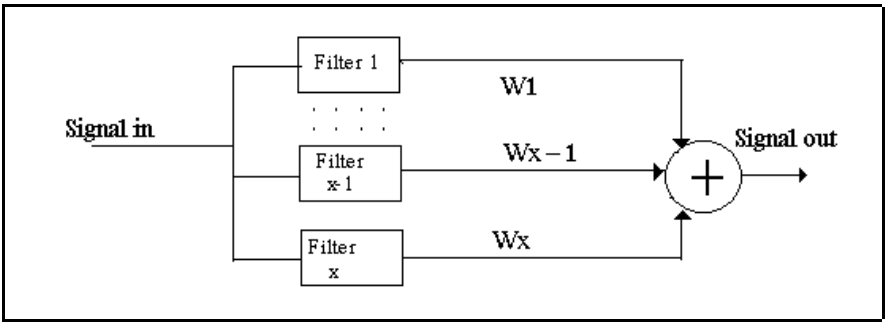


Fig. 1. Modeling the response as a set of x parallel filters

The results shown in this paper are those obtained using 3 filters in the filter bank. The filters were 4th order Butterworth bandpass filters, implemented using the MATLAB command FILTFILT. This command produced a zero-phase shift filter, which means that the filter itself did not produce a phase shift in the signal. Butterworth filters were selected because of their relatively smooth pass-band.

All the filters were set up randomly so that initially the low frequency cut-off was within the range 0-200Hz, and the high frequency cut-off was selected to be up to 300Hz higher than the low frequency cut-off. Subaverages (averaging small sets of signals) of the input sets were created to reduce the noise level. Again, the results shown in this paper are those obtained when 10 responses were used in each sub-average. The stimulation rate was set at two stimulations per second, so sub-averages of 10 responses equate to 5 seconds worth of evoked responses. In the training process every example in the sub-averaged training set was used to measure the fitness of the 'individual' set of filters and weightings in the population of possible solutions. The mean of all the example fitness values for that individual solution was used. Both simulated data and recorded data were used to develop and test the filter banks.

2.3 Evolutionary Algorithm

The filter parameters were encoded as a sequence of floating point numbers on the chromosomes. Michalewicz [3] suggests that floating point values are "intuitively closer to the problem space."

Fitness Function. The fitness function used here was the Mean Squared Error (MSE) between the results of the evolutionary algorithm and the known target response.

$$mse = \frac{1}{N}\sum_{k=1}^{N} e(k)^2 \qquad (1)$$

The error is the difference between the target signal and the test sequence at a point in time, $e(k)$.

Selection and Mutation After the fitness of each of the filter banks has been calculated the top quarter of the original population go through to the next generation unchanged i.e. those with the highest fitness (i.e. lowest MSE values). The remaining three quarters of the population in the next generation were produced by a selection process using crossover. The selection process used in this work is the roulette wheel approach where the higher the fitness of an individual sequence, the greater the probability that the sequence's genes will be used in the next generation. A set of pairs of random numbers, ranging from 1 to the sum of all portions of roulette wheel, was used to select the sequences that were the 'parents' of the next generation. A third random number was produced that determines where along the sequence the swapping occurs, so the two original sequences produce two new sequences. A second matrix was formed that was the same size and shape as the population matrix, and contained values in the range 0 to 1. If the value in the matrix was less than or equal to the mutation rate, then a change was made to the value in the population matrix at the corresponding position. The value in the population matrix was altered by up to +/- 12.5% of the current value. The population size was chosen as 80 and the mutation rate was set at 0.05. The evolutionary algorithms were all stopped after 200 generations.

3. Results

Figure 2 shows the averaged signal used both as the underlying signal of the simulated data sets and as the target signal.

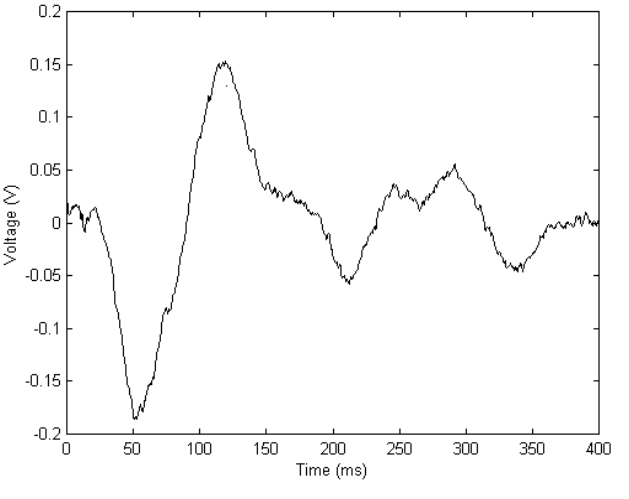


Fig. 2. Target signal

This spinal recording was chosen as it has small but important early components and much larger later components which were more time invariant. It therefore combines many of the features that need to be taken into account in the extraction of the evoked potentials from the background noise.

Five unfiltered sub-averages of simulated test data and the effects of filtering are shown in Figure 3. The signals have been shifted along the voltage axis to aid the visual presentation. Three filters were developed using the simulated training set. Comparing the results of the set of filters and the target signal, resemblance between the target signal and these filters can be seen. The most noticeable feature of these filtered signals is that they have negative peaks at around 50ms and 200ms. Two positive peaks in the region 100-200ms were also observed. These peaks are present in the target signal (Figure 2). Later components around 250, 300, 350ms do not appear in the majority of the signals. At the beginning of the signal, features are not present or have been 'flattened.'

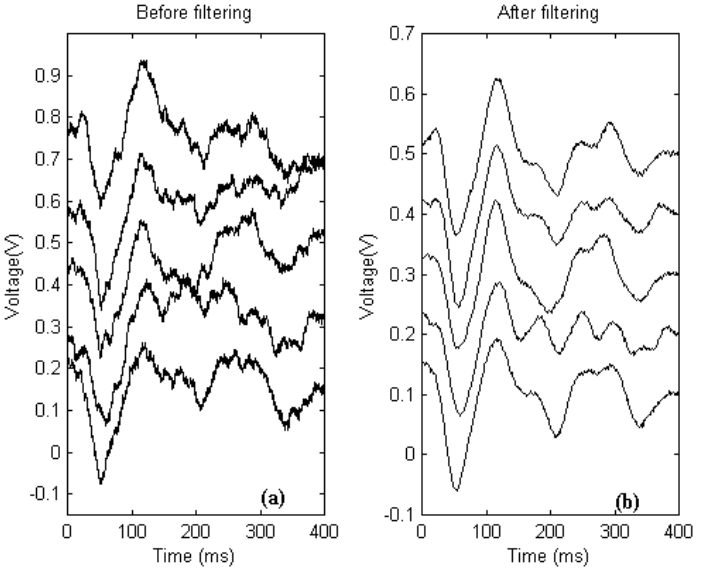


Fig. 3. (a) Averages of simulated activity (10 simulated evoked response per average). (b) The signals in (a) after being passed through a set of filters whose parameters were selected by an evolutionary algorithm, based on training with a different set of simulated responses.

Recorded evoked potentials were passed through the filters used previously. Figure 4 shows both the unfiltered and filtered responses. As in Figure 3, some of the features can be seen, but the similarities with the target signal are not as clear as when the simulated test data were filtered. Figure 5 shows the unfiltered averaged test recorded data again, but this time the signals are passed through a set of filters developed using the recorded data training set. In comparison with Figure 4, these are essentially the same shape but smoother. Table 1 contains the filter parameters and weightings for both sets of filters. Table 2 contains the minimum, maximum, mean and standard deviations of the MSE values.

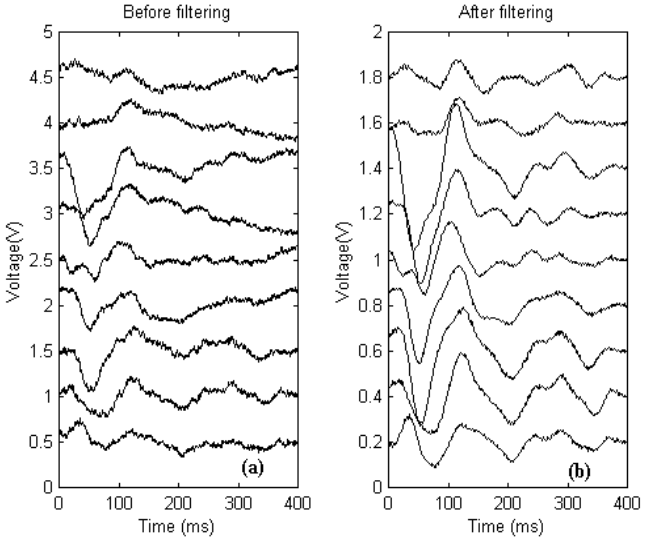


Fig. 4. (a) Averages of recorded activity (10 recorded evoked response per average). (b) The signals in (a) after being passed through a set of filters whose parameters were selected by an evolutionary algorithm (same filters as used in figure 3.)

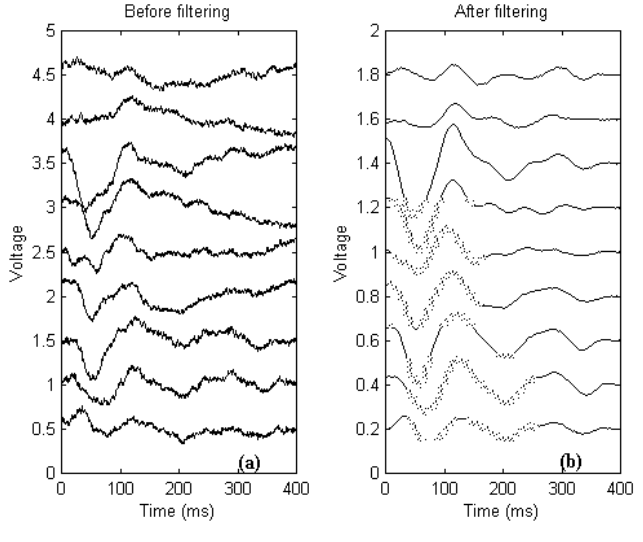


Fig. 5. (a) Averages of recorded activity (10 recorded evoked response per average). (b) The signals in (a) after being passed through a set of filters whose parameters were selected by an evolutionary algorithm, based on training with a different set of recorded responses.

	Cut-off frequencies (Hz)		
	fx	fx+1	weight
3 filter bank (simulated)	4.022	26.9667	0.9307
	66.751	576.816	0.2848
	133.398	637.093	0.0563
3 filter bank (recorded)	3.71	18.4259	0.6253
	52.788	55.649	0.31
	42.545	260.143	0.086

Table 1. Filter parameters selected using the evolutionary approach

Training Data	Test Data	min 10^{-3}	max. 10^{-3}	mean 10^{-3}	STD 10^{-3}
simulated	simulated	0.39	0.81	0.61	0.18
	recorded	1.6	5.9	2.7	1.4
recorded	simulated	1.0	1.4	1.2	0.2
	recorded	0.6	3.8	1.7	1.1

Table 2. Mean Squared Error values for the two filters.

4. Discussion

Evolutionary algorithms enable less specific assumptions to be made about the frequency properties of the signal beforehand. Using an evolutionary algorithm the algorithm can select cut-off frequencies for the filter, and weights, based on how well the filtered signal matches the shape of the target signal. A filter bank approach was selected so that spectral areas that are not important to extracting the evoked potential are less likely to be included in the filtered result.

A noticeable feature of all the processed responses, whether from simulated or recorded data, was that features at the beginning of the response were not as large as they are in the target signal. The reason for this was that these components were small compared with the rest of the response, and have higher frequency components than

those in the rest of the response. The dominant features were therefore these larger components, and this was the feature the algorithms have found. Changes in the larger low frequency components produced larger changes in MSE than the higher frequency components of the smaller early components. In Table 1, a list of the cut-off frequencies of the filter banks developed are given. Common to both set of filters is a high weighting on low frequencies, which would help to explain why components at the beginning of the response were smoothed out or reduced. This fits with groups such as Rossini et al. (1981) [4], who used a bandpass filter with relatively high frequency parameters (i.e. 150-3000 Hz) to extract short latency components (early components of the responses). The idea of a bandpass filter to extract these components does therefore seem relevant. Increasing the number of filters was investigated, but the results were no better, and so did not justify the extra processing needed. A reduction in the number of responses per average was tried, but the combination of the 'noisier' inputs signals and filters produced were not as effective as those where 10 responses were used.

5. Conclusions

Filtering simulated responses produced better results than filtering recorded responses (Table 2). This was believed to be due to the simulated response being time invariant. They were produced by taking the target responses, repeating it several times, and adding recorded noise. This means that the underlying response was not changing between the responses. In the recorded data, the underlying response can vary between the responses. The results of the filter developed using recorded data suggest that it was better than the filter developed using simulated data, for filtering recorded data (Table 2), at least for the later components of the signals. An assumption has to be made about the response that the frequency components were the same throughout the response, i.e. that it is stationary. A visual inspection of the responses suggests this is not true, as does the work by various groups using high-pass filtering to extract short latency components (e.g. Rossini et al. (1981) [4], McCabee et al. (1983)[2]). A possible way around this problem is to allow the filters to contribute to the overall final response at different times. These are time-varying filters, and work is on-going to investigate these. The effects of using other sets of intraspinal recordings and scalp recordings are also needed to investigate the effects of variations in recordings between subjects. A particular problem area is that the later components of the signals are likely to vary more than the earlier components. It is possible that other data sets may have signals that vary more than these, but this would also be a problem for the conventional ensemble average method.

References

1. Harrison SAB, Lovely DF (1995) "Identification of noise sources in surface recording of spinal somatosensory evoked potentials" *Medical & Biological Engineering & Computing* pp. 299-305.

2. McCabee PJ, Pinkhasov EI, Cracco RQ (1983) "Short latency somatosensory evoked potentials to median nerve stimulation effect of low frequency filter" *Electroencephalography and Clinical Neurophysiology* Vol. 55 pp 34-44.

3. Michalewicz Z (1996) *Genetic Algorithms + Data Structures = Evolution Programs* 3rd Edition, Springer.

4. Rossini PM, Cracco RQ, Cracco JB, House WJ (1981) "Short latency somatosensory evoked potentials nerve stimulation: scalp topography and the effect of different frequency filters" *Electroencephalography and Clinical Neurophysiology*

Evolution of Vehicle Detectors for Infrared Line Scan Imagery

Simon C. Roberts and Daniel Howard

Software Evolution Centre
Systems & Software Engineering Centre
Defence Evaluation and Research Agency (DERA)
Malvern, Worcestershire WR14 3PS, UK
dhoward@dera.gov.uk

Abstract. The paper addresses an important and difficult problem of object recognition in poorly constrained environments and with objects having large variability. This research uses genetic programming (GP) to develop automatic object detectors. The task is to detect vehicles in infrared line scan (IRLS) images gathered by low flying aircraft. This is a difficult task due to the diversity of vehicles and the environments in which they can occur, and because images vary with numerous factors including fly-over, temporal and weather characteristics. A novel multi-stage approach is presented which addresses automatic feature detection, automatic object segregation, rotation invariance and generalisation across diverse objects whilst discriminating from a myriad of potential non-objects. The approach does not require imagery to be preprocessed.

1 Problem Requirements

Airborne reconnaissance by low flying aircraft usually results in vast amounts of imagery from the aircraft's on-board sensors. Human analysts use their intuition and expertise to understand the imagery, but it is not possible for an analyst to inspect all of the imagery in reasonable time. Therefore, a timely automatic detector which can bring areas of interest to an analyst's attention is very attractive. The objective being not to classify objects but rather to bring the most promising areas in the imagery to the attention of the analyst.

The ultimate aim of this work is to develop automatic vehicle detectors for use by analysts of infrared line scan (IRLS) imagery. The problem is very challenging because the images vary with season, time of day, strength of sunlight, terrain, altitude and bank of the aircraft, and important esoteric reasons, e.g. the sensors on different aircraft will never be identical. Furthermore, the images are captured on video tape and suffer from quality degradation if not digitised within the first few plays. The automatic detector must find all the vehicles in the image without too many false alarms, it must be fast, and also it should preserve generality for detection of unusual vehicles and certain suspicious vehicle-like objects.

R. Poli et al. (Eds.): EvoIASP'99 and EuroEcTel'99, LNCS 1596, pp. 110–125, 1999.

2 Staged Genetic Programming

Genetic Programming for image analysis and object detection was pioneered by Tackett [1]. Investigators [2,3] have consistently expressed concern over the amount of computing time required to obtain practical results in this area. By pre-computing statistics of pixel data, Poli [4] both reduced the size of the search space and the computing time requirement. Other researchers have attempted to prevent the 'bloating' behaviour of GP, but GP's generality and search power can be compromised with such an intention.

In [5,6] we have proposed a two-stage Genetic Programming method and tested it in a hard near real world setting in a very computationally expensive domain. The two-stage method provided:

- small evolution times with no recourse to 'parsimony' or new crossovers;
- very efficient, i.e. fast, object detectors.

A novel multi-stage version is presented in this paper where GP is used to evolve a number of detectors which are then fused together to produce an overall object detector. The first stage is a coarse detection which is required to identify all of the objects in the imagery at the expense of producing many false alarms. Later stages provide a finer level of detection to reduce the false alarm rate. Each stage of detection processes irrotational statistics drawn directly from the imagery - no pre-processing is required. Thus, this work demonstrates additional uses for the staged evolution method:

- it can automatically discover features for vehicles such that these need not be prescribed *a-priori*;
- it can begin to address the so-called stability-plasticity dilemma [7].

3 Experimental Conditions

The traditional approach of using training, test and validation data sets has not yet been adopted because this requires a more uniform set of imagery and a more accurate account of each video tape that the imagery is extracted from. The problem being that there is no record of how many times these tapes have been used, and each time a tape set is looked at by an analyst the quality of the tape in use depreciates to an extent.

In addition, the volume of imagery required to construct comprehensive training, validation and test sets was not available at the time of these experiments. The rationale became to use GP on progressively larger training sets in order to learn about the likely elements of a solution, e.g. the limitations of the solution strategies adopted, and the power of the terminals and the functions used, i.e. the appropriateness of the search bias [8].

Two experiments were carried out. The first involved only four IRLS images taken from the same aircraft on the same flight and at roughly the same height and flight bank. The theme of these images was diverse. The second experiment involved ten IRLS images but at different heights and flight banks.

Four Samsung Alpha 633MHz machines and four 450MHz Pentium machines ran many combinations of random seeds for the various stages of Genetic Programming - c++ code. And other machines were used to study the evolved detectors, running an application with a *Mathematica* back-end [9]. This software traced the paths taken by detectors, i.e. down the $min()$ and $max()$ nodes in GP trees, to extract the dominant expressions responsible for the detection. Those findings will be presented elsewhere.

4 GP Specifications

The steady-state GP parameters used in the experiments were as in Table 1. (α and β are fitness function variables as described is section 4.3)

Table 1. Genetic Programming settings

Population	10000
Max tree size	1000
Max generations	10
Function nodes	+ , - , * , % , $min()$, $max()$
Terminal nodes	integer constants (-128 to 127)
	floating-point constants (-0.5 to 0.5)
	statistics as described in section 4.1
Kill tournament size	2
Breed tournament size	4
α	1, 2, 3
β	1 to 7

4.1 Statistics

Statistics of pixel information served as the underlying primitives for the GP trees as with [4,5,6,15]. However, these were defined to be rotationally invariant statistics based on concentric circles of single pixel thickness as in Figure 1. The circles were drawn with a diameter in pixels, e.g 11, with statistics defined on the pixel values in the perimeter. Four types of irrotational statistics were defined on each circle:

- a perimeter average, P_d,
- a perimeter standard deviation, S_d,
- the number of edges found on the perimeter, N_d,
- an edge distribution measure, E_d.

where the subscript d stands for 'diameter'. For example, for a diameter of 35 pixels the statistics are: P_{35}, S_{35}, N_{35} and E_{35} respectively.

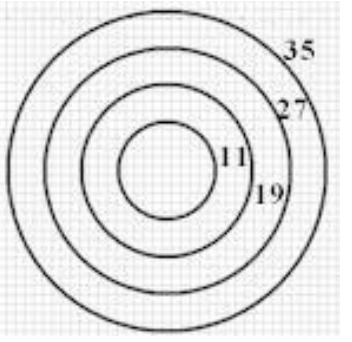


Fig. 1. Concentric statistics of pixel data

The number of edges was computed by taking the difference between consecutive pixels on the circle, and comparing against a threshold. An edge was identified when the difference between consecutive pixels on a given ring exceeded half the standard deviation of the entire image. The edge distribution measure E_d was defined as:

$$E_d = \sum_{i=1}^{N_d} (\frac{s_i}{L})^2 \tag{1}$$

where s_i is the arc length between the ith and the (i-1)th edges, L is the total arc length or perimeter, and N_d is the number of edges detected on the perimeter. If no edges were detected E_d was set to 1.

This choice of statistics must provide the evolutionary process with: (a) an irrotational character; (b) robustness to high variability in pixel data; and (c) both angular and radial information.

4.2 Initialisation

Each individual was represented as a tree consisting of functions and terminal nodes. The population was initialised using the ramped-half-and-half technique [11] with a maximum initial tree size of 4 nodes.

4.3 Fitness Function

A fitness function was used to evolve all detectors. A 'hit' or positive evaluation by the detector on a vehicle in the 'truth' is a 'true positive' or TP. A 'false alarm' or positive evaluation by the detector of a non-object is a 'false positive' or FP. The fitness function f aimed to maximise TP and to minimise FP:

$$f = \frac{\alpha\, TP}{\alpha\, N_v + \beta\, FP} - 1 \tag{2}$$

where N_v is the total number of vehicles in the 'truth' and α and β control the position along the Pareto curve. Higher α to β ratios thus bias detectors to hit

objects at the expense of yielding false alarms. Although these variables were not rigorously optimised, they were varied across different evolution runs.

4.4 Genetic Operators

These were standard cross-over and mutation operators (95% cross-over and 5% mutation) using tournament selection. No mating radius restrictions were applied, i.e. no 'speciation' was used - an individual could potentially breed with any other individual. Whenever the maximum allowed tree size was reached the shorter side of the swap was selected in crossover.

4.5 Choice of Parameters

Earlier work concluded that detection accuracy increases with population size [10]. Following this, the population was fixed at 10,000 which was found to give reasonable computation times.

The number of generations was restricted to 10 due to the computational expense of evolving for more generations. The average individual size tends to increase with each generation as has been observed by many, e.g. [12,13,14], thus the computational demands increase with generation. Furthermore, small solutions are likely to generalise better and it was thought that 10 generations was low enough to avoid over-training.

The maximum detector size was limited to 1000 nodes but no other scheme encouraged small detectors. However, the best detector in a population was found by ordering the individuals firstly in decreasing fitness and secondly in increasing size. Therefore the best detector was the smallest of the fittest detectors.

Each GP specification was run with at least 10 different randomiser seeds in order to ascertain general performance.

5 Training Strategy

In order to give the reader an appreciation of the difficulty of the task, Figure 2 shows a selection of vehicles from some of the IRLS images considered. It serves to illustrate the variability of vehicle subimages, e.g. some vehicles are colder than others, some are thermal shadows of vehicles, some appear in perspective while others are top down views of vehicles. Some vehicles are so bright that no features are recognisable. Note also that they come in many sizes, and can be seen at different heights. The vehicles have been aligned in this figure but they can appear at any angle. Any template matching algorithm would find it extremely difficult to generalise across these vehicles.

5.1 Definition of a Motorised Vehicle

This work tried to avoid introducing an *a-priori* answer to the questions: "what is a car?" or "what is a lorry?" For example, the training points were not focused

on distinct features of vehicles (e.g. bumpers, windscreens, hot engines, etc.) but a more generic approach was taken in order to encourage automatic feature detection. The reasons for not training on distinct features are numerous. From an isolated subimage it is difficult to clearly specify what it is about that subimage that indicates the presence of a vehicle. In other words, it is very difficult to define features which distinguish all vehicles from all non-vehicles. The training scheme would also require the precise locations of characteristic features to be specified.

Fig. 2. Selection of vehicles from the IRLS images.

Furthermore, the generalisation of features *a-priori* was undesirable because this generalisation is inextricably linked to the solution of the problem. Instead vehicles were simply represented by pixels subjectively placed towards the centre of each vehicle. A 'vehicle box' was centred on each of these points to contain most of each vehicle. GP was rewarded whenever it produced a detection within these boxes. An 'indifference zone' surrounded each vehicle box which may or may not have overlapped a vehicle depending on vehicle orientation and size. GP was neither rewarded nor punished for detections in this zone. The concept of a vehicle box allowed GP to discover characteristic vehicle points. This was facilitated by the multi-stage evolutionary process. Upon completion of the first stage, all detections within vehicle boxes were defined as vehicle points for the second evolution stage. GP then sacrificed many of these in order to trade off vehicle detections against fewer false alarms. Vehicle pixels that survived this process were to be found in parts of the motorised vehicle that GP considered to best represent the vehicle. Thus GP was able to discover both the characteristic

features and their location by generalising across these vehicle subimages. This procedure thus overcame the need manually specify distinct features of vehicles.

5.2 Standard Training Procedure

The following simple training procedure was a default for the experiments:

1. transform the image using 64.0 for mean and a standard deviation of 16.0.
2. define a 'truth' for each image to specify a single vehicle point for each vehicle.
3. evolve many first detectors using different initial random seeds.
4. select the best first detector from these, i.e. the one that produces the most hits and the least false alarms over all images.
5. apply the first detector to locate discovered vehicle points (i.e. hits within vehicle boxes) and false alarms with which to train the second detector.
6. evolve many second detectors using different initial random seeds.
7. select the best second detector, i.e. the one that gives the highest reduction in false alarms whilst sacrificing the least number of hits.
8. fuse the selected first and second detector to process all images.

6 Experiments Involving Four Images

For this experiment GP trained on four images, shown in Figures 3 and 4, that contain examples of vehicles in diverse environments. These raw IRLS images were in no way manipulated other than for a standard aspect ratio correction which has to do with the nature of the linescan sensor. Figure 3 is image number 1 in Table 2, and those of Figure 4 correspond to images 2, 3 and 4 in Table 2. From the table note that these were at similar altitude and on the same flight.

6.1 Implementation of the Training Strategy

Detectors process information about a subimage centred on a given point, therefore training points need to be selected for each vehicle. Ideally, these points would allow detectors to discover features which generalise across vehicles and discriminate from non-vehicles. However, as mentioned in section 5.1 these features are not known *a-priori*.

The selection of training points was restricted by the potential proximity of vehicles to each other, e.g. note how close the vehicles are in image 2. If training points had been selected on vehicle perimeters, there would have been a danger that detectors may not have generalised across proximate and isolated vehicles. To avoid this, a single training point was centrally positioned on each vehicle.

6.2 Results

Figure 3 shows the results of applying the detectors to image 1. The rate of detection is high and the false alarm rate is low. Furthermore, as is shown in

Fig. 3. Results of the two-stage evolution strategy: (left) application of first stage detector, (right) application of fused first and second stage detectors. White areas denote positive returns of the detectors. (corresponds to image 1 in Table 2)

Figure 4, this detector is able to identify disparate vehicles such as cars and lorries, vehicles at different projection, and of different brightness and tone.

For image 1 the detector managed to point to the thermal shadow of a car parked in a cool area towards the bottom of the image. In that area, the detector avoided detecting two people standing near these vehicles. Vehicle-like objects such as what is believed to be a skip at the top-right corner of this image were correctly flagged as suspicious objects. All three lorries in image 2 were detected and the large wall in the obstacle course that adjoins the car park was correctly missed. The detector missed very few vehicles in these first two images.

When the detectors were applied to image 3 they obtained a perfect detection of a vehicle in an urban situation, including a building with a bright roof, air vent, sharp edges and corners. Bright areas by the edge of the road were correctly classified as not being vehicles. Note that the detector settled for pixel areas on the front and back of the vehicle.

Image 4 was the most difficult image for this detector because:

- the canopy above the vegetation appears as a great number of textured shapes and edges that can readily be confused with the shapes of vehicles.
- the vehicle in this image is unlike the others in that it contains a different type of vehicle - a tank - and is of dull tone.

The detectors on image 4 produced a number of false alarms. This, however is consistent with a scenario likely to contain hidden objects. The image analyst may spend more effort on such an image. And results for Image 4 in Figure 4

are misleading because pixels flagged by the detector needed to be shown at much larger than pixel size. For example, the reader may conclude that image 4 contains eight false alarms and three hits. However, in actuality there is an equal number of false alarms as hits. It is interesting to note that a simple clustering method would eliminate those false alarms.

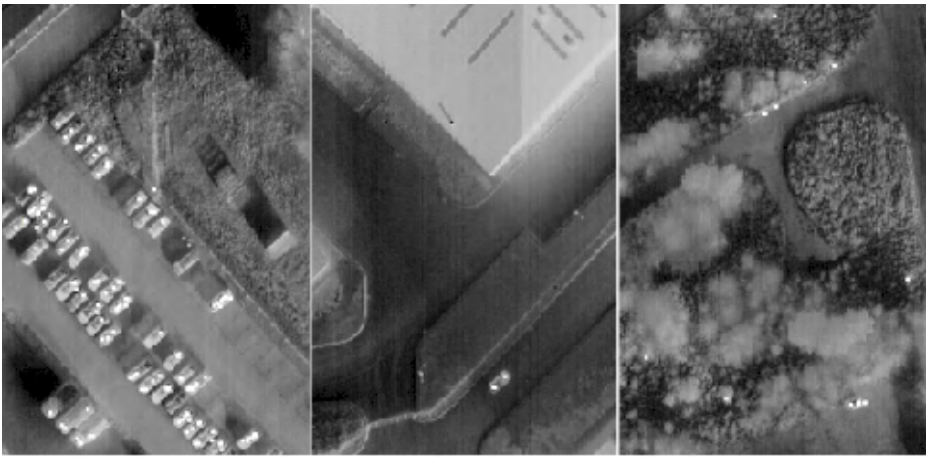


Fig. 4. Results of the fused two-stage detectors. White areas denote positive returns of the detectors. (from left to right these images correspond to images 2-4 of Table 2)

7 Experiments Involving Ten Images

The vehicle generalisation task becomes more difficult when many images are considered. This is not only due to the larger variability of vehicle types (e.g. cars, lorries, etc.) but is also due to variable aircraft altitude and bank, terrain environment, and atmospheric conditions.

Using the approach of sections 5.2 and 6.1 to process many more images with a greater variety of characteristics proved disappointing. The best results from a brief experiment involving 30 evolution runs using different random seeds, hit approximately two-thirds of the vehicles whilst giving about the same number of false alarms. Generalising across such a variety of vehicles whilst discriminating from a greater variety of non-vehicles is clearly too difficult a task.

7.1 Multiple Stages Method

The generalisation task is simplified if the vehicles are segregated such that only a subset of the vehicles needs to be discriminated from non-vehicles by a single second detector. Although vehicles can be manually segregated into

training subsets by their visual characteristics (e.g. by separating bright and dark vehicles, cars and lorries, etc.), automatic segregation avoids the need to manually define category boundaries. Automatic segregation is simply achieved by training a second detector to hit as many vehicles as possible whilst giving no false alarms. In theory, this detector should hit the most common vehicles which are most easily distinguishable from non-vehicles. These vehicles are then omitted from the training set and another second detector is evolved to hit the residual vehicles, whilst again giving no false alarms. Continuing this procedure should give multiple second detectors which specialise in hitting different types of vehicle. These second detectors can then be combined with OR and fused to the first detector using AND.

This approach is beneficial with regard to the so-called stability-plasticity dilemma [7]. To solve this dilemma a pattern-recognition system must be adaptive to changes in the input whilst previously encoded patterns must be stable against these changes. For example, a system which can detect vehicles from an industrial environment should be able to be adapted to detect vehicles from a rural environment without losing the ability to detect the industrial vehicles. In other words, it is desirable for a detection system to perform incremental learning where old pattern-encodings are retained even when new patterns are learned. Evolving a new second detector to process a new type of vehicle gives the system the ability to perform incremental learning, providing that the new vehicles can be generalised by the first detector.

A potential disadvantage with this technique is a reduction in detection speed on test images due to the need to apply multiple second detectors. However, as the generalisation task for a given second detector is simplified, smaller second detectors should be produced. The best results on ten images using the former approach were produced by detectors with a size of about 400 nodes, whereas the multiple-second-detector approach often produced much smaller detectors. Furthermore, fusing the second detectors using OR avoids the need to apply all second detectors to every subimage.

Table 2. Details of imagery

image	type	altitude (ft)	bank	total vehicles	flight
1	industrial	296	1R	29	A
2	car park	300	1R	43	A
3	industrial	300	1R	1	A
4	forest	288	0	1	A
5	rural	612	71L	9	A
6	rural	608	71L	3	A
7	residential	456	0	1	A
8	rural	612	72L	3	A
9	residential	460	9L	1	B
10	rural	604	71L	1	A

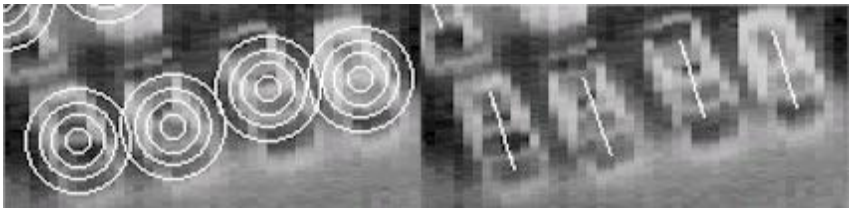


Fig. 5. Large outer ring and coverage of vehicles (left); multiple training points along a line for the first GP stage (right).

7.2 Implementation of the Training Strategy

The selection of a single training point per vehicle proved insufficient. In order to provide more training data whilst avoiding difficulties arising from the proximity of vehicles, multiple training points were used for each vehicle such that the points fell on a straight line with the same orientation as the vehicle, as shown in Figure 5. It was ensured that all points on the line were sufficiently distant from the vehicle edges.

It was deemed that shape information was important for vehicle detection. Whilst there was no statistic which explicitly specified vehicle shape, it was thought that shape information could be extracted if the diameter of the outer ring were greater than the typical vehicle width. However, if the outer ring were excessive it could overlap neighbouring vehicles. Preliminary experiments proved the importance of large outer rings by comparing the results obtained from using an outer ring with a diameter equal to a typical vehicle width, and using an outer ring which clearly exceeded this width.

This experiment highlighted the importance of shape information and verified that shape was primarily captured by the statistics of the outer ring. The ring diameters clearly provide vehicle size information. The training images corresponded to different altitudes hence ring diameters were linearly scaled according to altitude. For example, the diameters 11, 23, 33 and 45 corresponded to 300 feet, thus scaling to 5, 11, 17, and 23 for 600 feet. The coverage of vehicles by the rings is also shown in Figure 6.

7.3 Results

The overall results for all runs are summarised Table 3. 'Hits' is TP or the number of or detected vehicles and 'FAs' is FP or number of false alarms from all images. FOM is the 'figure of merit' which is equivalent to equation (2) but with $\alpha = \beta = 1$. For the evolution of first detectors using different randomiser seeds and α values, β was fixed at 1. Coverage is the average number of hit points per vehicle. Each column gives the average and standard deviation (in brackets) across all seeds.

Table 3. Summary of evolution: detector 1 (top); detector 2i (centre left) ; detector 2ii (centre right); detector 2iii (bottom) for different random seeds, α or β.

α	Hits	FAs	FOM	Coverage
1	90 (1)	7298 (2480)	.0137 (.0049)	85 (14)
2	91 (1)	10622 (3052)	.0095 (.0039)	103 (22)
3	91 (1)	13260 (4480)	.0078 (.0032)	115 (26)

1

β	Hits	FAs	FOM
4	42 (4)	8 (3)	.4201 (.0379)
5	37 (3)	4 (3)	.3868 (.0315)
6	37 (6)	4 (2)	.3841 (.0625)
7	35 (4)	3 (1)	.3672 (.0410)

β	Hits	FAs	FOM
1	73 (4)	37 (12)	0.5647 (.0248)
2	67 (2)	22 (6)	0.5881 (.0093)
3	67 (3)	18 (6)	0.5994 (.0137)
4	63 (1)	13 (1)	0.5921 (.0117)

2i,2ii

β	Hits	FAs	FOM
1	82 (2)	32 (3)	.6523 (.0032)

2iii

For the evolution of $2i$ detectors using different randomiser seeds and β values, α was fixed at 1. The best first detectors were deemed to be those that hit all vehicles with fair coverage and had relatively low false alarm rates. A number of detectors satisfied this requirement, mostly corresponding to $\alpha = 2$. For the evolution of 2ii detectors using different randomiser seeds and β values, α was fixed at 1. For the evolution of 2iii detectors using different randomiser seeds, both α and β were fixed at 1. All four of these tables show a general insensitivity to randomiser seed and a clear trend for the number of hits and false alarms to increase with the α to β ratio, as expected. Figure 6 shows that the individual second detectors identified particular features of the vehicle, to some degree the staged evolution method behaves as an automatic feature detector. It appears that the 2i detector is processing internal vehicle edges and is sensitive to vehicle width; 2ii follows external edges; 2iii detects bright areas.

Figure 7 shows that two vehicles were missed in image 1 because they are located close to the image edges, thus, preventing a sub-image being centred on the vehicles. Image 9 picks up more false alarms that most images perhaps because it is from a different flight. It is interesting to note that the two false alarms to the left of this image are centred on an area with a width equal to that of a typical vehicle.

Figure 8 combines images 5 and 8 and can be considered to represent the average performace that can be expected of the detectors developed to date.

8 Conclusions

The techniques have potential but a better understood and consistent set of imagery is required to advance this work. Vehicle shape is insufficiently represented by the current statistics. Further work could improve the statistics in this regard by relating information from multiple rings centred on different points. However, the scheme must generalise across proximate and isolated vehicles.

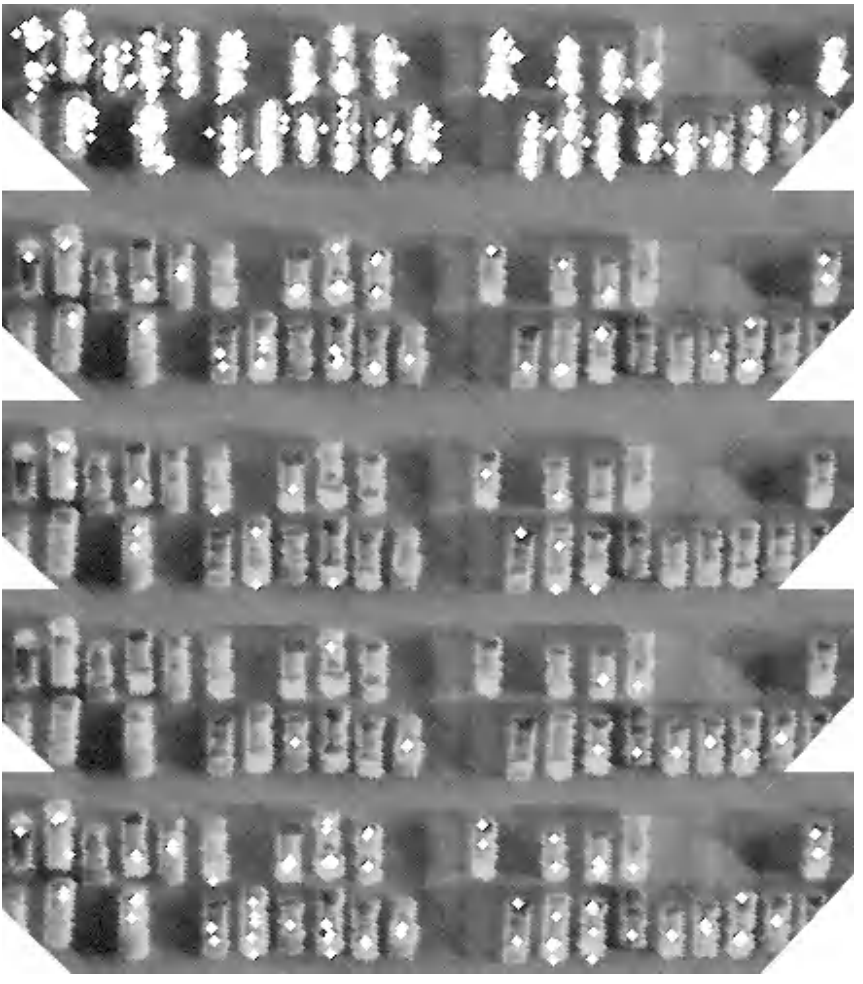


Fig. 6. Some vehicles from Image 2. From top to bottom: 1st, 2i, 2ii, 2iii, 1st AND (2i OR 2ii OR 2iii). Note how different second detectors target different features of cars.

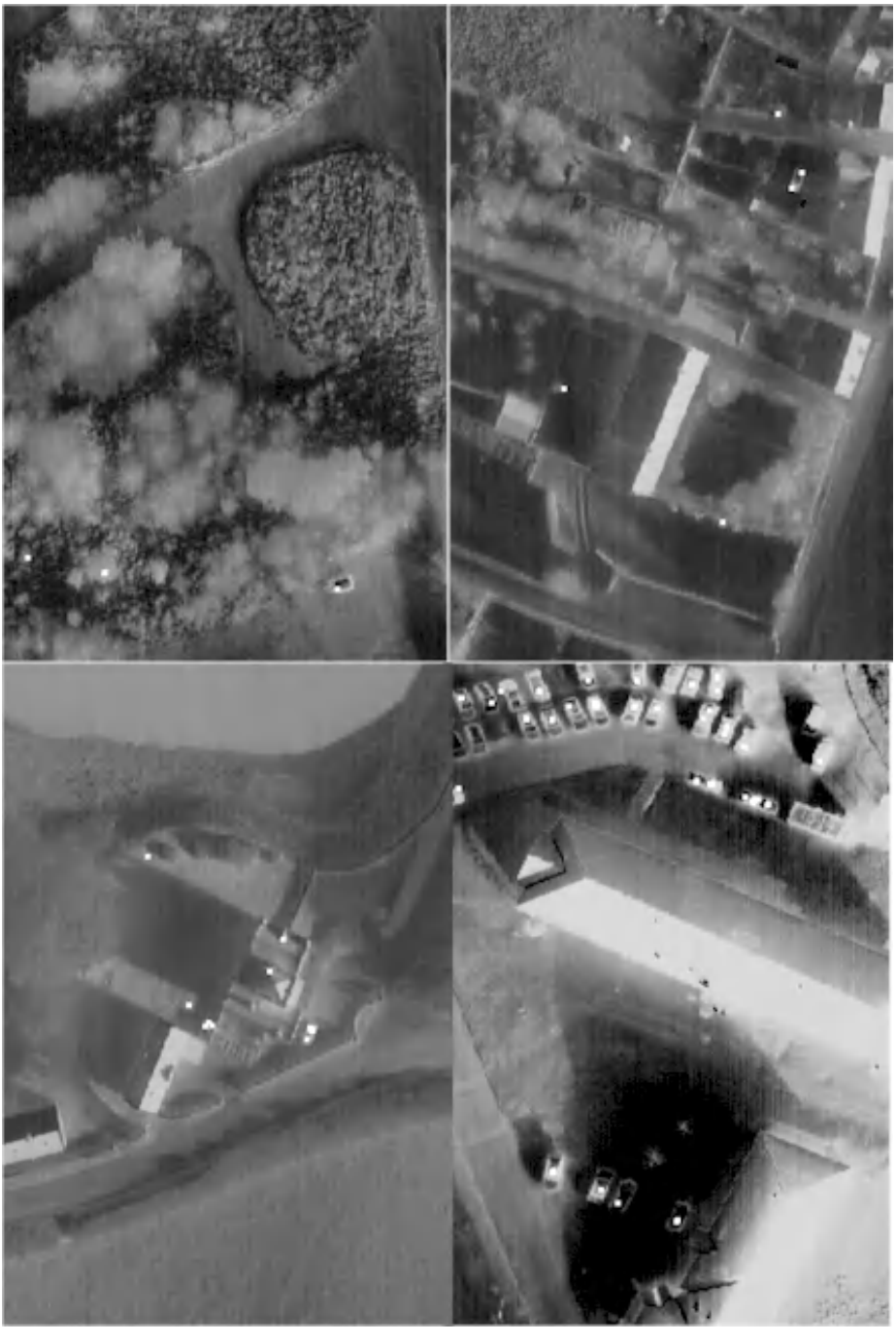


Fig. 7. Fusion of 1st AND (2i OR 2ii OR 2iii) detectors; image 4 (top left); image 7 (top right); image 9 (bottom left); image 1 (bottom right)

Fig. 8. Expected performance. Most cars and lorries are detected with a few misses and false alarms (from images 5 and 8). Note that the content of IRLS images can be unclear even to the trained eye. The linescan and line replication of IRLS images for aspect ratio correction are just visible at this level of magnification.

Acknowledgements

The authors wish thank Computing Devices Ltd of Hastings for their assistance.

References

1. Tackett W. A.: Genetic Programming for feature discovery and image discrimination. Proceedings of the Fifth International Conference on Genetic Algorithms. Morgan Kaufmann, (1993).
2. Winkeler J. F. and Manjunath B. S.: Genetic Programming for Object Detection. In Koza, Deb, Dorigo, Fogel, Garzon, Iba and Riolo (editors). Genetic Programming 1997: Proceedings of the Second Annual Conference.
3. Daida M., Bersano-Begey T.F., Ross S.J. and Vesecky J.F.: Computer-assisted design of image classification algorithms: dynamic and static fitness evaluations in a scaffolded Genetic Programming environment. In Koza, John R., Goldberg, David E., Fogel, David B., and Riolo, Rick L. (editors). Genetic Programming 1996: Proceedings of the First Annual Conference, pp. 279-284, Cambridge, MA: The MIT Press.
4. Poli R.: Genetic Programming for Image Analysis. In Koza, John R., Goldberg, David E., Fogel, David B., and Riolo, Rick L. (editors). Genetic Programming 1996: Proceedings of the First Annual Conference Cambridge, MA: The MIT Press.
5. Howard D.: Application of Genetic Programming to target detection and CFD problems. DERA Malvern technical report DERA/CIS/SEC/TR980322, (1998).
6. Howard D. and Roberts S.C.: Evolution of Ship Detectors for Satellite SAR Imagery Proceedings of Second European Workshop in Genetic Programming EvoGP, Lecture Notes in Computer Science, Göteborg, Sweden, (1999).
7. Carpenter G. A. and Grossberg S.: The adaptive resonance theory of adaptive pattern recognition by a self-organizing neural network. IEEE Computer, 77-88, (1988).
8. Tom M. Mitchell. Machine Learning. McGraw-Hill International, 1997. ISBN 0-07-042807-7.
9. Mathematica v.3.0 software: ©1996 Wolfram Research, Inc.
10. Roberts S.C., Howard D., Brankin R.: Genetic evolution of automatic ship detection in SAR imagery. DERA Malvern report DERA/CIS/SEC/TR980323, (1998).
11. Koza J.R.: Genetic Programming. Cambridge, MA: The MIT Press, 1992.
12. Langdon W.B. and Poli R.: Fitness Causes Bloat: Mutation. In Proceedings of the First European Workshop on Genetic Programming, pp 37-48, (1998).
13. Rosca, J. P.: Generalty versus size in Genetic Programming. In Koza, John R., Goldberg, David E., Fogel, David B., and Riolo, Rick L. (editors). Genetic Programming 1996: Proceedings of the First Annual Conference, pp 381-387. Cambridge, MA: The MIT Press.
14. Soule T. and Foster J.A.: Code size and depth flows in genetic programming. In Koza, Deb, Dorigo, Fogel, Garzon, Iba and Riolo (editors). Genetic Programming 1997: Proceedings of the Second Annual Conference, pp 313-320.
15. Target detection in SAR imagery by genetic programming, D. Howard, S. C. Roberts, R. Brankin, Genetic Programming 1998: Late Breaking Papers, pp 67-75, (1998).

Genetic Programming for Channel Equalisation

Anna Esparcia-Alcázar* and Ken Sharman

Department of Electronics & Electrical Engineering
University of Glasgow, Scotland, UK
aesparcia@ieee.org, kenshar@elec.gla.ac.uk

Abstract In this paper we investigate the application of a combined Genetic Programming - Simulated Annealing (GP-SA) solution to a classical signal processing problem which arises in communications systems. This is the so called channel equalisation problem where the task is to construct a system which adaptively compenstates for imperfections in the path from the transmitter to the receiver. Our primary interest is to examine the recosntruction of binary data sequences transmitted through distorting channels. We measure the performance of a generic GP-SA equaliser and compare it to that of standard methods commonly used in real systems. In particular, we consider special cases which are known to be difficult for the existing methods, such as non-linear and partial response channels. Our results show that the GP-SA method generally offers superior signal restoration but at the expense of computational effort.

1 Introduction

In previous work [3, 4], the authors have described a variant of Genetic Programming (GP) [10] that is well suited to applications in the signal processing domain. This approach uses GP to adapt the structure of a signal-flow graph in conjunction with gain parameters that are optimised by Simulated Annealing (SA). This technique is particularly powerful in that it enables simultaneous adaptation of both the structure and the parameters of a signal processing system. This is in contrast to classical adaptive signal processing systems where the system structure is chosen *a priori*. Although fixed system structures are adequate in many applications, there are equally many applications where an adaptive structure has the potential for better performance - especially when the operating environment is unknown and hence an optimal structure cannot be determined. Furthermore, a large class of signal processing systems employ control algorithms that only work with certain types of structure (e.g. linear filters). It is generally difficult to extend these control algorithms to structures known to offer better performance (e.g. non-linear filters). Hence, the opportunity for flexible structure adaptation is appealing.

* presently with the Industrial Control Centre, University of Strathclyde, Glasgow, Scotland

R. Poli et al. (Eds.): EvoIASP'99 and EuroEcTel'99, LNCS 1596, pp. 126–137, 1999.

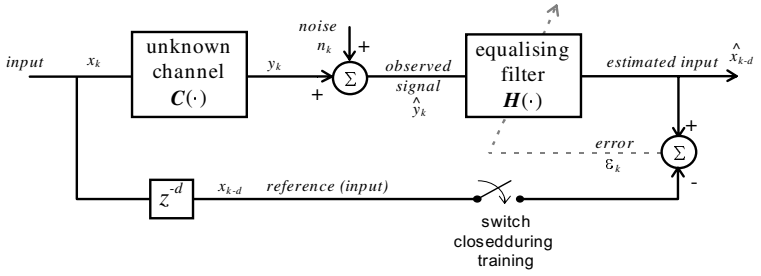


Figure1. The Channel Equalisation System.

The objective of this paper is to present experimental results comparing the GP-SA approach with some standard fixed-structure adaptive systems applied to the classical problem of channel equalisation. This extends previous work in this area and complements that of other authors who have proposed alternative solutions based on neural networks (which are still fixed structure, but have larger operating domains).

The paper is structured as follows. Sect. 2 briefly describes the channel equalisation problem and some of the existing solutions. In Sect. 3 we describe three special cases that have been addressed by other authors, pointing out the why they are difficult to solve by conventional techniques, thus justifying the GP-SA approach. In Sects. 4, 5 and 6 we describe the experiments and present results comparing GP-SA to the classical solutions for these three special cases. The results are collated and summarised in Sect. 7, and general conclusions are given in Sect. 8.

2 Background on Channel Equalisation

In a communications system, the original transmitted signal is modified by the characteristics of the propagation medium (the channel) as it travels to the receiver. Effects such as multipath, bandwidth limitations, non-linearities and so on all contribute to making the received data different to that which was transmitted. The aim of *channel equalisation* [16, p. 636][8, p. 217] is to compensate for the channel's imperfections and recover the original data.

Figure 1 shows a block diagram of a generic communications system and the usual form of the equaliser. The basic problem is complicated by the fact that the channel's characteristics are unknown (and possibly time-varying). A common approach uses a *training* period where a known signal is sent into the channel. The output of the equaliser can be compared to this reference to assess its performance. A feedback signal then adjusts the equaliser in an adaptive loop. Once the training process is finished, the transmission of the data begins.

At this stage some form of *test signal* can be transmitted, in order to measure the performance of the equaliser and the success of the training.

2.1 Linear equalisation

A simple and widely used method of equalisation is based on linear systems theory. This approach is attractive as it is fairly easy to develop an adaptive control algorithm for adjusting the parameters of the equalising filter. The control objective is to minimise a cost function such as a measure of the difference between the transmitted and equalised signals. The approach is tractable if the cost function is diferentiatable in the equaliser's parameters. A popular choice is the mean squared error (MSE).

Let the error signal $\{\epsilon_k\}$ be the difference between some desired response $\{d_k\}$ and the actual filter output, $\{y_n\}$.

$$\epsilon_n = y_n - d_n \tag{1}$$

The MSE is defined as

$$MSE \;=\; E(\epsilon_k^2) \tag{2}$$

where $E(\cdot)$ is the mathematical expectation.

From Fig. 1, note that that optimum equaliser has a transfer function $H(z)$ which is the inverse of $C(z)$, i.e.

$$H(z) = \frac{1}{C(z)} \tag{3}$$

This is referred to as *inverse filtering*.

If the channel impulse response is modelled as an Auto Regressive (AR) process, the effective transfer function of the channel is

$$C(z) = \frac{1}{b_0 + b_1 \cdot z^{-1} + b_2 \cdot z^{-2} \cdots + b_n \cdot z^{-n}} \tag{4}$$

In this case the appropriate equaliser is a finite impulse response (FIR) filter as follows

$$H(z) = b_0 + b_1 \cdot z^{-1} + b_2 \cdot z^{-2} \cdots + b_n \cdot z^{-n} \tag{5}$$

A widely used method for adapting the coefficients $\{b_i\}$ for this class of problems is the recursive least squares (RLS) algorithm.

When the channel is better modelled as a moving average (MA) or, more generally, as an auto regresive moving average (ARMA) system, an equaliser such as the one given in Eqn 5 may not be sufficient, even for high values of n. In such cases, an infinite impulse response (IIR) equaliser may be employed, such as the one given by Eqn 6

$$H(z) = \frac{b_0 + b_1 \cdot z^{-1} + b_2 \cdot z^{-2} \cdots b_n \cdot z^{-n}}{a_0 + a_1 \cdot z^{-1} + a_2 \cdot z^{-2} \cdots a_m \cdot z^{-m}} \tag{6}$$

Despite providing reduced computational complexity, IIR equalisers have been traditionally less employed because the two fundamental approaches to the adaptation of the coefficients a_i and b_j present major problems [17]. They can lead to biased estimates of the coefficients or converge to a local minimum of the error surface (which is not quadratic and may have multiple local minima), resulting in an incorrect estimate of the coefficients. A trade-off must be found between the two. Furthermore, an added problem is guaranteeing the stability of $H(z)$ during the adaptation process.

Recently other approaches have been employed to address these problems, such as Evolutionary Programming (EP) [1], Genetic Algorithms (GAs) [5, 11] and Simulated Annealing [14].

As can be deduced from Eqns 3 to 6, inverse filtering with FIR and IIR equalisers is dependent on the channel being linear (with transfer function $C(z)$) and its inverse $H(z)$ being realisable (for instance, $H(z)$ must be the transfer function of a stable system). Additional problems of linear equalisers have been reported in [9].

2.2 Nonlinear equalisation

In other situations the channel C will have nonlinear distortion. Such channels are found, for example, in data transmission over digital satellite links, especially when the signal amplifiers operate near their high gain limits [9]. If the distortion is severe, linear equalisers perform poorly. Nonlinear equalisers must then be employed but, in general, the mathematical treatment of such models is complex.

An alternative has been found in neural networks. Neural networks, such as multilayer perceptrons (MLPs) and Radial Basis Function (RBF) networks have been applied to channel equalisation. This is done at the expense of turning the equalisation problem into a classification one: the transmitted data are assumed to be symbols belonging to some finite alphabet and the network, acting as a classifier, must determine which symbol was transmitted.

One important drawback of NNs lies in the determination of the structure: there exists no established procedure for determining the number of layers and nodes [12].

The second main problem of MLPs and RBF networks is their being feedforward structures. When nonlinearity is the main impairment, feedforward NNs perform well. This is the case of the examples reported in [2, 7, 18]. It was shown how, for the high levels of noise involved in these examples, nonlinear classifiers were required.[1]

However, for higher values of the signal to noise ratio (SNR) (as should be expected in a telephone channel, for instance) the need for nonlinear compensation is balanced or overcome by the need for recurrence, or feedback.

To be able to cater for this, feedforward NNs require a large number of nodes, which increases their complexity. This hinders the study of their behaviour, as

[1] As pointed out above, these constitute cases of the detection problem, rather than equalisation.

well as their hardware implementation, which prevents their use in real time applications [13].

More recently, equalisation with recurrent neural networks (RNNs) has also been reported in the literature [9, 15]. Their less wide spread use is due to the complexity of the training algorithm, which may become unstable.

RNNs have the advantage of being more compact than their feedforward counterparts, but the issue of determining the structure remains.

2.3 A new approach

In view of all the problems involved in linear and nonlinear equalisation methods, it is desirable to find an equalisation technique that allows for adaptation of the structure, while catering at the same time for recurrence and nonlinearity.

Thus, taking into account the properties of Genetic Programming [10] and the tree representation described in previous work by the authors [3, 4], the scene is set for addressing the channel equalisation problem with GP.

3 Overview

The examples discussed here are cases for which it has been shown [2, 7, 6, 18] that nonlinear equalisation techniques can provide better results than linear ones. The unknown channel to equalise will be:

- a linear channel with high levels of noise.
- a linear partial response channel.
- a nonlinear channel.

The results yielded by the proposed method will be compared to those obtained by training a 19^{th} order FIR equaliser with the RLS algorithm, as done by [9].

The set up for the experiments is as follows. A training signal consisting of a 250 bit pseudo random binary signal (PRBS), applied at a rate of one bit per sample, is used to train both a FIR-RLS and a GP+SA equalisers. For the latter, the first 50 samples are rejected as transient in the calculation of the fitness during the evolution process.

After adaptation, a further 100100 samples of a signal of the same noise realisation are processed by both filters and the bit-error rate (BER)[2] calculated (rejecting the first 100 samples in both cases).

For the settings of the GP and SA algorithms the reader is referred to Table 4 and [3, 4].

[2] The BER is defined as the number of incorrectly classified bits divided by the total number of transmitted bits

Table1. Values of the bit-error rate obtained by equalisers for the channel $H(z) = 1 + 0.7z^{-1}$ over 30 runs.

	average BER		minimum BER	
SNR(dB)	FIR-RLS	GP+SA	FIR-RLS	GP+SA
2.5	0.126724	0.129697	0.11284	0.10519
5	0.071622	0.082892	0.06121	0.05491
7.5	0.033807	0.034803	0.02731	0.01299
10	0.009246	0.014868	0.00653	0.00120

4 Linear channel with high noise

Let us consider the linear minimum phase channel described by the transfer function

$$H(z) = 1 + 0.7z^{-1} \tag{7}$$

In the low noise situation it is possible to find an equaliser for this channel to any specified accuracy by employing a FIR filter of sufficient order. When the noise is high, however (i.e. the signal to noise ratio, SNR, is lower than 10 dB) the phenomenon of noise enhancement appears, which means that any increase in order results in a decrease in efficiency of the equaliser [7]. It is therefore interesting to try and find low order equalisers which employ some form of nonlinearity.

Results for different values of the SNR are given in Table 1 and shown graphically in Fig 2. Average and minimum values of the BER for 30 runs (per point) are presented, showing that the GP+SA method can obtain lower minimum values than the FIR-RLS, especially for values of the SNR of 7.5 and 10 dB.

For values of the SNR below these the noise is too high for the equalisation method to make a significant difference. For values above, both methods give consistently BERs of zero. In these cases it is interesting to note that several GP+SA equalisers yielded a fitness of 1 (i.e. MSE = 0), while this was never achieved by any of the FIR-RLS equalisers.

5 Partial response channel

The transfer function of partial response channels has zeros on the unit circle. Such channels are frequently encountered in magnetic recording [9] and since the inverse of the channel is undefined, there exists no linear filter that would sufficiently equalise them. Therefore nonlinear methods have to be used to reconstruct the originally transmitted signal.

Following [9] the channel employed in the experiments had a double zero on the unit circle, its transfer function being

$$H(z) = 1 - 2 \cdot z^{-1} + z^{-2} \tag{8}$$

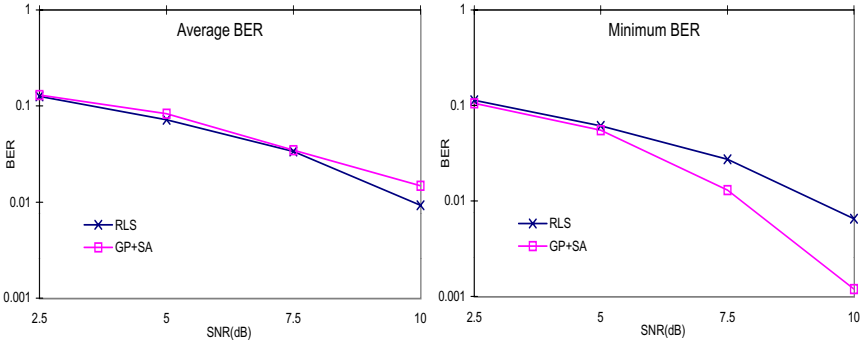


Figure2. Average and minimum values of the BER for linear channel (30 runs per point)

Table2. Values of the bit-error rate obtained by equalisers for the partial response channel $H(z) = 1 - 2 \cdot z^{-1} + z^{-2}$ over 30 runs.

SNR(dB)	average BER		minimum BER	
	RLS	GP+SA	RLS	GP+SA
10	0.07537	0.10266	0.06197	0.0014
12.5	0.06866	0.09047	0.05944	0
15	0.05895	0.08109	0.04295	0
17.5	0.05135	0.11199	0.04356	0
20	0.04471	0.07155	0.03426	0
22.5	0.03498	0.06836	0.02914	0
∞	0.00623	0.07644	0.00515	0

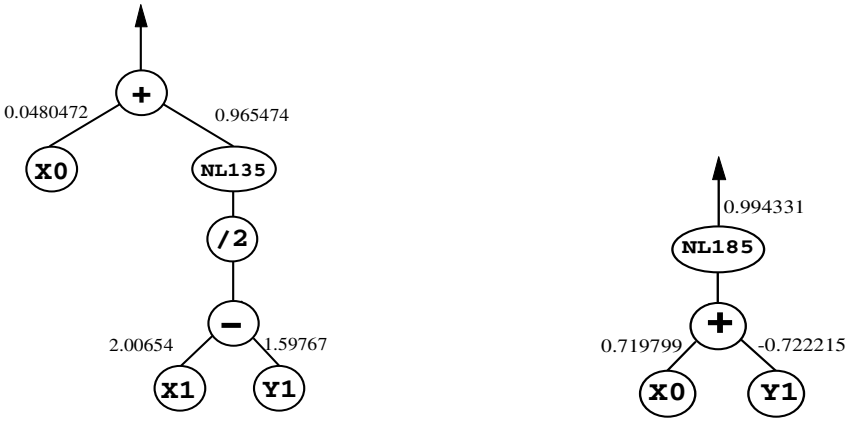
The performance of the proposed method was compared to that of the RLS algorithm, first for the noiseless case and then for different realisations of the signal to noise ratio. Thirty runs were performed in all cases and the values of the bit-error rate obtained are given in Table 2.

A *t*-test was used to compare the two methods in the absence of noise showing that the average BER of the solutions was lower for the RLS algorithm. However, none of the RLS solutions obtained a BER of zero, while, on the other hand, this was achieved by a number of the GP+SA solutions.

One of the solutions obtained by the proposed method is shown in Fig 3(b).

6 Nonlinear channel

Nonlinear channels are the obvious cases to tackle with nonlinear equalisation techniques. The channel used in the experiments follows the model shown in [9, 2] and is given by Eqns 9 and 10.



(a) Linear channel

(b) Partial response channel

Figure3. GP equalising filters for linear and partial response channels. The node `NL135` represents the function $\tanh(2.67x)$ and the node `NL185` represents the function $\tanh(3.64x)$.

$$\tilde{y}_n = 0.3482x_n + 0.8704x_{n-1} + 0.3482x_{n-2} \tag{9}$$

$$\hat{y}_n = \tilde{y}_n + 0.2 \cdot \tilde{y}_n^2 \tag{10}$$

The channel output, $\hat{y}$, is further corrupted by the addition of white Gaussian noise.

The experimental procedure is as explained for the linear channel, with the same GP and SA settings. The values of the bit-error-rate obtained are given in Table 3 and displayed graphically in Fig 5. From it we can conclude that GP equalisers outperform linear equalisers trained by the RLS algorithm, both on the average and the minimum values.

An example of a solution obtained by the proposed method is given in Fig 6. This tree was obtained with a training signal whose SNR was 15 dB. Other solutions obtained for various SNR realisations had a similar structure.

7 Comparison

A comparison with the results obtained by [9] and [15] is attempted here. Such a task is not easy, due to the difficulty in obtaining reliable measures from published results. Furthermore, the methods employed by these authors are based on a different philosophy than that of the GP+SA method presented here.

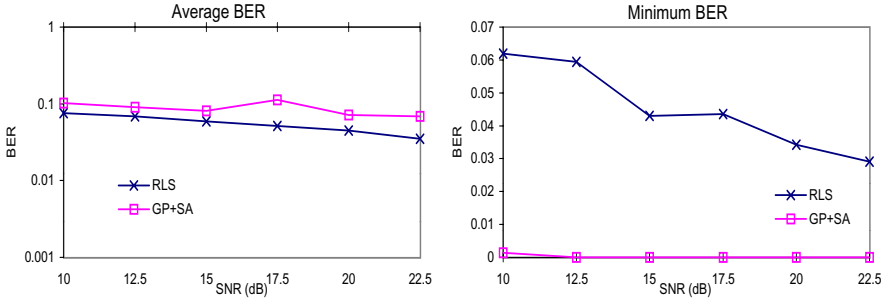


Figure4. Average and minimum values of the BER for partial response channel (30 runs per point). Note that the scale for the minimum BER plot is linear and not logaritmic, as is customary, so as to be able to show values of zero

Table3. Values of the bit-error rate obtained by equalisers for the nonlinear channel given by Eqns 9 and 10.

SNR(dB)	average BER		minimum BER	
	RLS	GP+SA	RLS	GP+SA
5	0.153082	0.131444	0.13183	0.12431
7.5	0.111524	0.093784	0.09914	0.08232
10	0.084465	0.064212	0.06722	0.04311
12.5	0.055214	0.044546	0.04454	0.01466
15	0.039135	0.026311	0.02155	0.00349
17.5	0.028946	0.016541	0.01095	0.00025

The minimum BER results achieved by the proposed method were consistently lower that the average values obtained by [9] and minimum values of [15] for the linear and partial response channels, and similar for the nonlinear one.

However, the average results only outperformed those of [9] for the high noise cases (SNR $\leq$ 7.5 dB) of the linear channel.

This is consistent with our expectation. Further adjustments in the method are needed to achieve optimal performance in these problems.

8 Conclusions

Previous work by the authors showed the possiblity of using node gain GP+SA for channel equalisation. This work has addressed the performace issues involved. We compared GP+SA equalisers to those obtained by classical techniques and also to results provided in the equalisation literature. Our results, which have been obtained for three classical examples, show that the performance of node

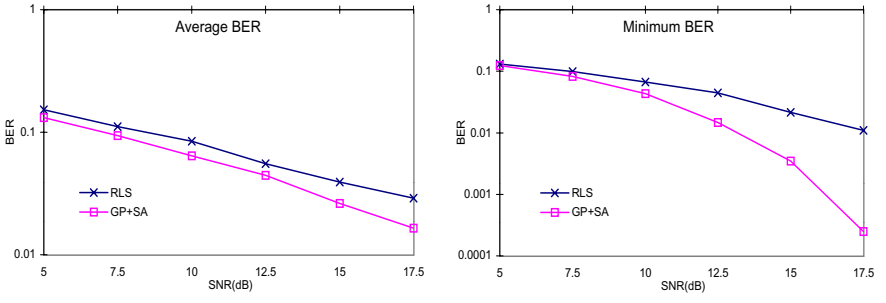


Figure5. Average and minimum values of the BER for the channel $H(z) = 0.3482 + 0.8704z^{-1} + 0.3482z^{-2}$ with nonlinear gain $d_2 = 0.2$. Results obtained by GP+SA and RLS equalisers over 30 runs (per point).

gain GP+SA is variable but we observed that in many cases the proposed method provided much better performance.

The main drawback of GP+SA when compared to the RLS algorithm is its computational expense. We have not been able to compare it with that of neural network-based methods due to the lack of data available. In any case computational expense is only an issue in certain applications, such as communications; other applications of equalisation do not have such time constraints and GP+SA could be successfully applied in those cases. Furthermore it must be pointed out that techniques that are widely employed nowadays, such as the RLS algorithm, were regarded as impractically time-consuming in their early days.

References

1. K Chelapilla, DB Fogel, and SS Rao. Gaining insight into evolutionary programming through landscape visualisation: an investigation into IIR filtering. In PJ Angeline, RG Reynolds, JR McDonnnell, and R Eberhart, editors, *Evolutionary Programming VI*, Springer–Verlag, Berlin,Germany, 1997.
2. S Chen, GJ Gibson, CFN Cowan, and PM Grant. Adaptive equalization of finite non-linear channels using multilayer perceptrons. *Signal Processing*, 20:107–119, 1990.
3. AI Esparcia-Alcázar. *Genetic Programming for Adaptive Digital Signal processing.* PhD thesis, University of Glasgow, Scotland, UK, 1998.
4. AI Esparcia-Alcázar and KC Sharman. Some applications of genetic programming in digital signal processing. In *Late Breaking Papers at the GP'96 conference*, pages 24–31, Stanford University, USA, July 1996.
5. DM Etter, MJ Hicks, and KH Cho. Recursive adaptive filter design using an adaptive genetic algorithm. In *Proceedings of the 1982 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP82*, pages 635–638, 1982.

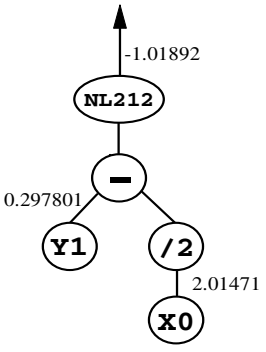


Figure6. An equalising filter for the channel $H(z) = 0.3482 + 0.8704z^{-1} + 0.3482z^{-2}$ with nonlinear gain $d_2 = 0.2$. The node `NL212` represents the function $\tanh(4.16x)$.

6. GJ Gibson, S Siu, and CFN Cowan. Multilayer perceptron structures applied to adaptive equalisers for data communications. In *Proceedings of the 1989 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP89*, pages 1183–1186, 1989.
7. GJ Gibson, S Siu, and CFN Cowan. The application of nonlinear structures to the reconstruction of binary signals. *IEEE Trans. on Signal Processing*, 39(8):1877–1884, August 1991.
8. S Haykin. *Adaptive Filter Theory*. Prentice Hall, 3rd edition, 1996.
9. G Kechriotis, E Zervas, and ES Manolakos. Using recurrent neural networks for adaptive communication channel equalization. *IEEE Trans. on Neural Networks*, 5(3):267–278, March 1994.
10. JR Koza. *Genetic Programming: On the programming of computers by means of natural selection*. The MIT Press, Cambridge, Massachusetts, 1992.
11. Q Ma and CFN Cowan. Genetic algorithms applied to the adaptation of IIR filters. *Signal Processing*, 48:155–163, 1996.
12. B Mulgrew. Applying radial basis functions. *IEEE Signal Processing Magazine*, 13(2):50–65, March 1996.
13. SK Nair and J Moon. Data storage channel equalization using neural networks. *IEEE Trans. on Neural Networks*, 8(5):1037–1048, September 1997.
14. R Nambiar and P Mars. Genetic and annealing approaches to adaptive digital filtering. In *Proceedings of the 26th Asilomar Conference on Signals, Systems and Computers*, pages 871–875, IEEE Computer Society Press, 1992.
15. R Parisi, ED Di Claudio, G Orlandi, and BD Rao. Fast adaptive digital equalization by recurrent neural networks. *IEEE Trans. on Signal Processing*, 45(11):2731–2739, November 1997.
16. JG Proakis. *Digital Communications*. McGraw–Hill, 3rd edition, 1995.
17. JJ Shynk. Adaptive IIR filtering. *IEEE Acoustics, Speech and Signal Processing Magazine*, pages 4–21, April 1989.
18. S Theodoridis, CFN Cowan, CP Callender, and CMS See. Schemes for equalisation of communication channels with nonlinear impairments. *IEE Proceedings on Communications*, 142(3):165–171, June 1995.

Table4. GP and SA set up for equalisation experiments

Genetic Program Settings	
Function set	`+ - * / +1 -1 *2 /2 1 Z` `PSH NL0...NL255`
Terminal set	`X0...X3 Y1 Y2 C0...C255 STK0...STK4`
β limits for `NL` nodes	$\beta_{hi} = 10 \qquad \beta_{lo} = 0.1$
Nodes with Gain	`X Y STK C NL`
Population size	500
Mutation probability	0.01
Size restrictions	at creation: maximum depth = 4 at crossover: maximum length = 25
Fitness function	$\frac{1}{1+MSE} \qquad 0 \leq f \leq 1$
Input signal (X)	output of the channel when fed with a Pseudo-Random Binary Signal (PRBS)
Reference signal	the same PRBS delayed by one sample
Number of training samples	250 550 (partial response channel, noiseless only) the first 50 are not considered for fitness calculation.
Number of runs	30
Termination criterion for each run	30 minutes of CPU time 60 min (partial response channel, noiseless only)
Number of test samples	100100 the first 100 are not considered for fitness calculation.
Simulated Annealing Settings	
Perturbation Distribution	Cauchy $C(0,1)$
Starting Temperature (T_0)	1.5
Cooling Schedule (temperature variation law)	Inverse linear in n: $\quad T = \frac{T_0}{n+1}$
Scale perturbation by	0.2
Annealing Iterations (n)	100
Trials per temperature	5

`PSH`: push node's input value to a stack, return same.
`NLn`: implements the sigmoid function, or $\tanh\left(\frac{\beta}{2}\right)$ with n an integer in the range [0, n_{max}] and $\beta = n\frac{\beta_{max}-\beta_{min}}{n_{max}} + \beta_{min}$.
`XN`: system input delayed by N samples; `YN`: system output delayed by N samples.
`STKn`: retrieve $\mathtt{n}^{th}$ position of the stack.
The 256 entries in the constant table are chosen at random uniformly within the interval [-1,1].

Improving Mutation Capabilities in a Real-Coded Genetic Algorithm

Cristian Munteanu and Vasile Lazarescu

Electronics and Telecommunications Department, "POLITEHNICA" University of Bucharest, 1-3 Iuliu Maniu Blvd., Sector 6, 7000 Bucharest, Romania
{munteanu, vl}@vala.elia.pub.ro

Abstract. This paper introduces a new method of performing mutation in a real-coded Genetic Algorithm (GA), a method driven by Principal Component Analysis (PCA). We present empirical results which show that our mutation operator attains higher levels of diversity in the search space, as compared to other mutation operators, meaning that by employing our mutation operator we maintain diverse populations that increase the chances of finding better solutions during evolution of the GA. The performances of the real-coded GA with PCA-mutation were checked on the problem of designing IIR filters by Deczky method, which is a well known direct design method of IIR filters. Results obtained show that our PCA-mutation GA has been more successful in keeping diverse populations during search, the consequence being the finding of better solutions to the filter design problem, compared to solutions found using GA with classical mutation operators.

1 Introduction

Genetic Algorithms (GAs) are strategies that mimic the evolution of populations of individuals, and as their natural counterparts, GAs behave according to the principle "Survival of the fittest will win". GAs have been applied in solving various problems including difficult optimization tasks where the function to be optimized is highly multimodal, nonlinear, with no gradient information available. Many technical design problems may fall into the "difficult optimization task" paradigm, such as adaptive IIR filter design, robust control, optimal path planning in robotics, pattern recognition, circuit layout and general optimal engineering design, as well as applications in scheduling, time-tabling and in economics. GAs may yield interesting insights into biological behavior of natural organisms, thus enlarging the spectrum of applicability.

GAs are complex evolving systems, and since their first theoretical motivations made by Holland in [4], there was an increasing strive to bring together various theoretical considerations concerning the way GAs work. In his seminal work [4], Holland explained how GAs process information by applying simple genetic operations, such as selection, crossover, mutation and inversion to a population of

R. Poli et al. (Eds.): EvoIASP'99 and EuroEcTel'99, LNCS 1596, pp. 138-149, 1999.

chromosomes. He named the information blocks processed by GAs, *schemata* and he showed through the *Schema Theorem*, that high performance schemata tend to spread exponentially into the population (performance or *fitness* of a schema, or chromosome, being defined with respect to the function to be optimized by the GA). Therefore, the optimal chromosome (solution of the optimization problem) consists of such performant schemata. Later, Goldberg in [1] advanced the *Building Block Hypothesis*, which states that a GA seeks near-optimal performance through juxtaposition of short length and low order, performant schemata, called *building blocks*. However, many classical theoretical results, have been questioned by recent advances in GA theory, and we will briefly mention the investigation of the Building Block Hypothesis' validity in [11]. Most of the theoretical work done so far, concerns the variants of GAs with binary coded chromosomes, or binary-coded GAs. However, in practice, real-coded GAs, that are GAs for which genes in the chromosomes are coded as real variables, have proven to perform quite well in many applications. Goldberg, in [2], talks about the "paradox of real codings" saying that "theoreticians have wondered why practitioners have paid so little heed to the [classical, binary-coded GA] theory, and practitioners have wondered why the theory seems so unable to come to terms with their findings [concerning real-coded GAs]".

In this paper we will focus on real-coded GAs which, as previously mentioned, perform very well in many applications. In the remainder of this paper we will introduce a new mutation operator based on Principal Component Analysis (PCA-mutation) and we will check that a PCA-mutation real-coded GA performs better than real-coded GAs with other mutation operators, on a digital filter design test problem. Specifically, the test problem is a IIR filter design problem by Deczky method. Results obtained support our initial claim that PCA-mutation searches in a more thorough manner than classical real-coded GA mutation operators: Uniform and Non-Uniform mutation operators.

2 Outline of Real-Coded GAs

Real-coded GAs employ a population **x** of N chromosomes, each chromosome having l genes (usually, N and l are fixed), with:

$$\mathbf{x} = \left[x_1, x_{2,}, \ldots, x_N\right]^T \tag{1}$$

where $x_i = [x_{i1}, x_{i2}, \ldots, x_{il}]$, $\forall i = \overline{1 \ldots N}$ denotes a chromosome in the population with real-coded genes $x_{ij} \in [lb_j, ub_j]$ with $lb_j, ub_j \in \Re$, $\forall i = \overline{1 \ldots N}, j = \overline{1 \ldots l}$.

The bounds on the genes' values (lower bound lb_j and upper bound ub_j) are taken equal to the bounds of the parameters of the problem to be optimized. The main genetic operations in a real-coded GA are the same as those in a binary-coded GA. Thus, we have a selection mechanism that picks the highly fitted chromosomes into the mating pool. The individuals in the mating pool, are then recombined by applying crossover and mutation. The selection mechanism is the same as in the binary-coded

GA case. We may employ any selection scheme valid for binary-coded GAs, such as: proportional selection, rank-based selection, tournament selection, elitist mechanisms, etc. The remaining genetic operations (i.e. crossover and mutation), differ from their binary-coded counterparts, because they have to operate on real valued genes, but they are similar in spirit with the classical binary-coded operators [5, 6].

The main crossover operators are:

1. *Simple crossover*: defined in the usual, binary-coded chromosome's way, that randomly picks two parents from the mating pool and exchanges genetic information between one random split point in the chromosomes.
2. *Arithmetical crossover*: defined as a linear combination of two chromosomes seen as vectors: Let x_1^p, x_2^p be two parents in the mating pool. The offspring after recombination are calculated as:

$$x_1^o = a \cdot x_1^p + (1-a) \cdot x_2^p,\ x_2^o = (1-a) \cdot x_1^p + a \cdot x_2^p \quad \textbf{(2)}$$

   with a a random value in [0, 1].
3. *Heuristic crossover*: defined as a linear extrapolation of the two parents based on fitness value. Let x_1^p, x_2^p be two parents in the mating pool, with fitness $f(x_1^p) \geq f(x_2^p)$ in a maximization problem. The offspring are calculated as:

$$x_1^o = x_1^p + a \cdot \left(x_1^p - x_2^p\right)\ x_2^o = x_1^p \quad \textbf{(3)}$$

   with a a random number in [0, 1]. The offspring have to be feasible with respect to their genes' values: $x_{ij}^o \in [lb_j, ub_j],\ \forall i \in \{1,2\},\ j = \overline{1 \ldots l}$. If not feasible, the offspring are regenerated following the same rule given in (2), with another randomly picked constant a.

The main mutation operators are:

1. *Uniform mutation*: which randomly selects one gene x_{ij} and sets its value equal to a uniform random number in $[lb_j, ub_j]$.
2. *NonUniform mutation*: which randomly selects one gene x_{ij} and sets its value according to the following rule:

$$x_{ij}^o = \begin{cases} x_{ij} + \left(ub_j - x_{ij}\right) \cdot \Gamma(t) \text{ if } a_1 < 0.5 \\ x_{ij} - \left(x_{ij} + lb_j\right) \cdot \Gamma(t) \text{ if } a_1 \geq 0.5 \end{cases} \quad \textbf{(4)}$$

   where $\Gamma(t) = (a_2(1 - t/t_{\max}))^b$ with a_1 and a_2 two random numbers in [0, 1], b a constant parameter, t the generation number, $t_{\max}$ the maximum run time.

Real-coded GAs have been seldom studied by GAs theoreticians, mainly because small alphabets, that are used to code genes in a chromosome, maximize the number of schemata available for genetic processing [1]. Therefore, a small alphabet (e.g. binary) would be the best choice, while an infinite alphabet as in the real-coding scheme, would be the worst choice. Real-coded GAs have been successfully applied in many practical tasks for a number of reasons, such as: their suitable coding scheme

(one parameter represents one gene), the avoidance of Hamming cliffs and other artifacts of mutation acting on bit strings treated as unsigned binary integers, their fewer generations to population conformity [2]. Goldberg in his study [2] develops a theory of *virtual alphabets* for real-coded GAs. Virtual alphabets may be roughly defined as slices in the search space for which the fitness function has above average values. These finite number virtual alphabets are the information processing units of the real-coded GAs in the same way schemata are the basic building blocks of a binary-coded GA search mechanism. Goldberg also identifies some problems, related to premature convergence of the real-coded GAs and *blocking*. The latter phenomenon is similar to deception in a binary-coded GA, in that virtual alphabets already discovered and combined during evolution, prevent from further improvement of the best solution found, the GA being lead away from the actual global optimum. To avoid blocking and premature convergence, new genetic operators have to be designed, operators that maintain genetic diversity in the population at a suitable high level, knowing that premature convergence or convergence in a general sense, means high conformity of the individuals in the population. From this theoretical point of view, seeking more effective mutation operators, that avoid loss of diversity, makes sense. We adopt this approach in the next section, by introducing a new mutation operator, that is the PCA-mutation.

3 Principal Component Analysis and PCA-Mutation

Principal Component Analysis (PCA) has been used recently in Evolutionary Algorithms, specifically in Evolutionary Strategies (ESs), by Hansen and Ostermeier to adapt the mutation distribution [3]. Their Covariance Matrix Adaptation algorithm (CMA) was designed to achieve a generalized step size control scheme for the mutation operator in ESs. Our approach to PCA utilization with Evolutionary Algorithms is entirely different from theirs, both in terms of implementation details (our PCA-based mutation operator acts on a GA rather than an ES) and most important in terms of the goals we seek. While Hansen and Ostermeier's CMA strategy seeks to better adapt to the search space, increasing the *exploiting* power of mutation in ESs [3], our PCA-based mutation operator seeks to increase the *exploratory* effect of mutation in real-coded GAs.

Principal Component Analysis (PCA) is a well known statistical technique that has been widely used in data analysis and compression. The goal of the method is the compression of a high-dimensional input data into a lower dimensional space, without loss of relevant information. The input data set $\mathbf{x} = [x_1, x_2, \ldots, x_N]^T$ with $x_i = [x_{i1}, x_{i2}, \ldots, x_{il}]$, $\forall i = \overline{1 \ldots N}$ may be viewed as a cloud of N points in the l-dimensional Euclidean space, centered around the mean $E(\mathbf{x})$. To capture the main features of the data set, PCA is looking for directions along which the dispersion or variance of the point cloud is maximal. These "principal" directions form a subspace of lower than l dimension, and the projection of the data $\mathbf{x}$ onto the respective subspace will yield a transformation similar to compression, that minimizes the loss of

information according to the Minimum Mean Square Error criterion. Before proceeding, let us note that the definition of the data cloud previously given, is the same as the definition of the genetic population in (1). This means that we think of the genetic population in a real-coded GA, as a cloud of *l*- dimensional points, the number of points in the cloud being *N*. We apply PCA on this genetic population (**x**) in the usual manner, as follows: First we compute the covariance matrix **S** of the data cloud **x**:

$$\mathbf{S} = E\left\{(\mathbf{x} - \mu_{\mathbf{x}})(\mathbf{x} - \mu_{\mathbf{x}})^T\right\} \tag{5}$$

with $\mu_{\mathbf{x}} = E(\mathbf{x})$. As matrix **S** is a symmetrical matrix its eigenvalues have real and positive values [8]. We may calculate the eigenvalues $\lambda^{(i)}$ and the respective *l*-dimensional eigenvectors $v^{(i)}$ and we have:

$$\mathbf{S}v^{(i)} = \lambda^{(i)}v^{(i)} \text{ for } i = \overline{1 \ldots l}. \tag{6}$$

Because the eigenvalues $\lambda^{(i)}$ are real and positive we may order them such that $\lambda_1 > \lambda_2 > \ldots > \lambda_i > \ldots > \lambda_l > 0$ will be the set of ordered eigenvalues, and v_i with $i = \overline{1 \ldots l}$ the respective eigenvectors. The PCA method states that if we project the data **x** onto a subspace consisting of a few directions given by those eigenvectors v_i at the top of the rank (v_i for $i = \overline{1 \ldots p}$ with $p < l$) we get an optimal linear transformation (better than any reduction transformation to *p* arbitrary directions), also called the Karhunen-Loève Transform.

The PCA-mutation proceeds as follows: after computing the covariance matrix **S,** the ordered set of eigenvalues λ_i and the corresponding eigenvectors v_i, we calculate the projection of the population **x** onto the orthogonal basis formed by *all l* eigenvectors v_i. First, we subtract the mean $\mu_{\mathbf{x}}$ from the population **x** in order to "center" the data cloud into the origin of the coordinates system. The projected population **y** onto the new coordinates system **V**, is:

$$\mathbf{y} = (\mathbf{x} - \mu_{\mathbf{x}}) \cdot \mathbf{V} \tag{7}$$

where $\mathbf{V} = [v_1\ v_2 \ldots v_l]$ is the orthogonal eigenvector matrix, each v_i being an *l*-dimensional column vector.

Mutation is performed on the projected population $\mathbf{y} = (y_{ij})$ by computing the squared length of the projections along each direction v_i, that is $||\mathrm{Pr}_{vi}(\mathbf{x})||^2$. We have:

$$\left\|\mathrm{Pr}_{v_i}(x_j)\right\|^2 = y_{ij}^2,\ i = \overline{1 \ldots l},\ j = \overline{1 \ldots N}. \tag{8}$$

It can be shown [8] that the mean squared length of the projection along one direction v_i is equal to the respective eigenvalue λ_i:

$$E\left(\|\mathrm{Pr}_{v_i}\|^2\right) = \lambda_i,\ \forall i = \overline{1 \ldots l}. \tag{9}$$

The mean in (9) is taken over all *N* points in the population. Equation (9) and the fact that $\lambda_1 > \lambda_2 > \ldots > \lambda_i > \ldots > \lambda_l > 0$ imply that the mean squared projection is

biggest along the first direction v_1, called the first Principal Component, the second biggest component being along direction v_2, and so on. The mean length of each projection quantifies the level of diversity along the respective direction. PCA is used in order to extract the most important information present in the data set, along the Principal Components Directions. However, our mutation operator seeks a totally different goal: the genetic populations after applying mutation, should exhibit *close in value* components and not some important components while others are negligible. The homogeneity of the components after applying mutation may be viewed as equivalent to a high diverse genetic population. A population less diverse will exhibit a few important principal components while the rest are close to zero. PCA-mutation should prevent this situation.

From equation (9) and the ordering of the eigenvalues λ_i, we have:

$$E\left(\left\|\mathrm{Pr}_{v_{i-1}}\right\|^2\right) > E\left(\left\|\mathrm{Pr}_{v_i}\right\|^2\right) \text{ for } i = \overline{1\ldots l}. \tag{10}$$

Consider the following *non-negative* quantities:

$$c_j^i, \text{ so that } c_j^{i-1} \le c_j^i, \forall i = \overline{1\ldots l},\ j = \overline{1\ldots N}. \tag{11}$$

As before, i is the index of the respective ordered eigenvalue λ_i. The quantities c_j^i are taken as random numbers between 0 and $c_{\max}$, and then sorted to fulfill (11). $c_{\max}$ is a constant parameter of the mutation operator. Taking the mean value in (11) we have:

$$E\left(c_j^{i-1}\right) \le E\left(c_j^i\right)\ \forall i = \overline{1\ldots l}. \tag{12}$$

The mutation operator adds the quantities c_j^i to each projected squared coordinate, as follows:

$$\forall i \quad \begin{aligned} \left\|\mathrm{Pr}^{\text{mutated}}{}_{v_{i-1}}(x_j)\right\|^2 &= \left\|\mathrm{Pr}_{v_{i-1}}(x_j)\right\|^2 + c_j^{i-1} \\ \left\|\mathrm{Pr}^{\text{mutated}}{}_{v_i}(x_j)\right\|^2 &= \left\|\mathrm{Pr}_{v_i}(x_j)\right\|^2 + c_j^i \end{aligned} \quad j = \overline{1\ldots N}. \tag{13}$$

Taking the mean value in (13) over all points in the population $j = \overline{1\ldots N}$, subtracting the equations, and letting $\Delta_i = E(||\mathrm{Pr}_{vi-1}(x_j))||^2) - E(||\mathrm{Pr}_{vi}(x_j))||^2)$ and $\Delta_i^{\text{mutated}} = E(||\mathrm{Pr}_{vi-1}^{\text{mutated}}(x_j))||^2) - E(||\mathrm{Pr}_{vi}^{\text{mutated}}(x_j))||^2)$, we have:

$$\Delta_i^{\text{mutated}} = \Delta_i + \left(E\left(c_j^{i-1}\right) - E\left(c_j^i\right)\right). \tag{14}$$

From (10) we have $\Delta_i > 0$ and from (12) we have $(E(c_j^{i-1}) - E(c_j^i)) \le 0$. Combining this latter result with (14) we obtain:

$$\Delta_i^{\text{mutated}} \le \Delta_i,\ \forall i = \overline{1\ldots l}. \tag{15}$$

The main result of applying mutation is given in equation (15): after applying mutation, the difference between two adjacent mean squared projections tends to become smaller compared to the same quantity before mutation. As the mean squared

projection quantifies the importance of the component along the respective direction (v_i), we conclude from (15) that our mutation operator satisfies our design goal, that is: mutation tends to homogenize the components to avoid having few important principal components while the rest are negligible. As discussed before, homogeneity of components means higher diversity in the genetic population, and this is the objective our mutation operator has to achieve. After adding the quantities c_j^i to the projected squared coordinates in (13), we perform an operation that is inverse to the projection on the orthogonal **V** basis. Before doing this, we must ensure that each negative projected coordinate corresponds to a negative projected mutated coordinate. Therefore, we first compute the sign of each element in matrix **y**, that is we compute the matrix signum(**y**). The actual mutated coordinates in $\mathbf{y}^{\text{mutated}}$ are the square roots of the mutated square projections $||\text{Pr}_{vi}^{\text{mutated}}(x_j)||^2$ in (13), multiplied by the corresponding sign signum(y_{ij}). Finally, we have:

$$\mathbf{x}^{\text{mutated}} = \mathbf{y}^{\text{mutated}} \cdot \mathbf{V}^T + \mu_{\mathbf{x}} \text{ with } \mathbf{V} = \left[v_1 \;\; v_2 \;\; \ldots \;\; v_l\right]. \tag{16}$$

The resulting mutated population is $\mathbf{x}^{\text{mutated}}$. PCA-mutation as any mutation operator is applied with some probability P_{m}. By generating a matrix $A = (a_{ij})$ of uniform random variables in [0, 1], if $a_{ij} > P_{\text{m}}$ then the respective quantity c_j^i is set to zero in (13), otherwise it is left unchanged. To compute the homogeneity level of the components, apart from the differences Δ_i, we may employ the components' ratios r_p defined as:

$$r_p = \frac{\sum_{k=1}^{p} \lambda_k}{\sum_{k=1}^{l} \lambda_k}, \quad p < l \text{ with } \lambda_i, i = \overline{1 \ldots l} \text{ the ordered eigenvalues of } \mathbf{S}. \tag{17}$$

A high level of components homogeneity implies a low level of r_p, while the existence of some dominant principal components, while the rest of the components are close to zero, means that $r_p \cong 1$. Therefore, PCA-mutation should achieve low values for the ratios r_p.

4 Test Case: IIR Filter Design by Deczky Method

Deczky method is a direct method of designing IIR filters, the main advantages over the indirect methods are the simplicity (it does not require analogue prototypes, as the indirect methods do) and the possibility to specify a desired group delay characteristic together with a desired magnitude characteristic [9]. The filter to be designed has the following transfer function:

$$H(z) = K_o \prod_{k=1}^{N_s} \frac{(1 - z_k z^{-1})(1 - z_k^* z^{-1})}{(1 - p_k z^{-1})(1 - p_k^* z^{-1})} \tag{18}$$

with N_s the number of singularities: zeros z_k and poles p_k and K_o a constant. The parameters to be found are: K_o, z_k and p_k. The error to be minimized is:

$$E(\omega_i) = \alpha_1 \sum_{q=1}^{L} W_1(\omega_i) \left| A(\omega_i) - A_d(\omega_i) \right|^m + \alpha_2 \sum_{q=1}^{L} W_2(\omega_i) \left| \tau(\omega_i) - \tau_d(\omega_i) - \tau_o \right|^m \quad \textbf{(19)}$$

with ω_i being L digital frequencies in the interval [0, π], A the magnitude characteristic of the transfer function in (18), τ the group delay characteristic of $H(z)$ in (18), τ_o an acceptable lag (also a parameter to be found) and W_1, W_2 some error weighting functions. In the original Deczky method, constants α_1, α_2 were taken so that $\alpha_1 + \alpha_2 = 1$.

Deczky method implies solving a system of nonlinear equations to find the parameters that minimize the error function in (19). Usually, gradient or second order methods are employed for this task, like the Fletcher-Powell algorithm [9], but these methods require many restarts, as they usually fall into local optima, depending on the initial solution taken. Therefore, a global search approach that does not require gradient information, is more appropriate for the Deczky method. This should be a reasonable task for a GA. As the number of parameters in the Deczky method is relatively big, a real-coded GA with the most suitable one parameter-one gene coding, is the best choice. We, therefore, employed our real-coded GA with PCA-mutation (PCAmGA). Our strategy was compared to a real-coded GA with Uniform mutation (UmGA), to a real-coded GA with NonUniform mutation (NUmGA) and to a modified Newton method. The crossover operator was arithmetic crossover. All employed classical mutation and crossover operators are defined in section 2. The selection mechanism for all strategies involved in the comparison is a combination of binary tournament selection, that picks the best individual from a randomly taken pair of parents, with an elitist scheme that automatically copies the k best individuals into the next generation (we used k = 5). The elitist scheme is necessary because we used a relatively high mutation rate (P_m = 0.05) that can destroy the useful genetic information found during GA's evolution, unless we apply additional preserving mechanisms. The fitness function is taken with respect to the error in (19), to be minimized by the GA. Because our implementations of GAs search for the maximum rather than minimum value, we took the fitness as: *fitness* = 1/*error*. The coding of the parameters into real genes is the following: the amplitudes of the singularities z_k, p_k are taken so that to obtain a minimum phase filter: $|z_k|, |p_k| \in [0, 1]$ (the rightmost value for the pole's amplitude is taken less than unity to insure stability), the phases of the respective singularities are covering the whole unit circle [0, 2π); K_o and τ_o are taken in [-10, 10]. Each parameter is represented by a single real gene in the chromosome. The constants in Deczky method are: N_s = 5, L = 30, m = 2 (we minimize a quadratic error function), $\alpha_1 = 8$, $\alpha_2 = 0.02$, $W_1(\omega) = W_2(\omega) = 1$; the desired amplitude characteristic $A_d(\omega_i)$ and desired group delay characteristic $\tau_d(\omega_i)$ are plotted in Figure 1. All strategies compared, have the following common parameters: population size N = 100, chromosome length l = 22, mutation rate P_m = 0.05, crossover rate P_c = 1, maximum number of generations to run is t_{max} = 300. For the PCAmGA we have

the additional parameter c_{max} = 0.05 (see discussion on equation (11)) and for NUmGA we have the constant b in (4) with b = 3 (also t_{max} = 300 is a parameter of NUmGA). We performed 20 independent runs for each strategy. Results are given in Table 1, in terms of Best, Average and Standard Deviation of the solution found (the solution is the chromosome with the maximum fitness, or minimum error value equivalently, found in a run), statistics being calculated over 20 independent runs.

Table 1. The Best, Average and Standard Deviation of the solution found, statistics being calculated over 20 independent runs for PCAmGA, UmGA, NUmGA

Fitness:	Best	Average	Std.
PCAmGA	0.5083	0.36623	0.1082
UmGA	0.4301	0.30031	0.0558
NUmGA	0.2208	0.1254	0.067
Newton	0.4518	0.3098	0.0859

From Table 1, it follows that PCAmGA outperforms UmGA, NUmGA, and the modified Newton method, on average, and considering the best solutions found. Assuming normal distributions of the solutions found, we applied a Student's t-Test for significance of means difference, and found that the average solution obtained by PCAmGA is significantly different from the average solution found by UmGA, at a significance level of 0.05. A better result was obtained when comparing PCAmGA to NUmGA: the average solution found by PCAmGA is significantly better than the one obtained by NUmGA, at a significance level of 0.05. Strategies employing an evolutive search (i.e. PCAmGA, UmGA) performed better, on average, than the modified Newton search method. In Figure 1, we plot the magnitude and group delay characteristics of the best solution found by PCAmGA.

To show that PCAmGA achieves higher levels of conformity for the components in PCA, which as discussed in the previous section, implies higher levels of population diversity, we computed the PCA ratios r_p for $p = \overline{1 \ldots l}, l = 22$. We identified the minimum levels of r_p denoted by Λ_p. Roughly, these are mean levels obtained after 150 generations for PCAmGA and UmGA, and before 100 generations for NUmGA. In the case of NUmGA, after nearly 100 generations the population diversity starts to decrease significantly, due to the diminishing effect of NonUniform mutation. Therefore, levels of PCA ratios for NUmGA were computed before generation 100. Around Λ_p the value of r_p oscillates with a small variance. These levels may be viewed as "minimum" levels for r_p during the evolution of a GA. Comparing these levels (Λ_p), a lower level means higher population diversity (see comments on equation (17)). These levels, averaged over 20 independent runs, are given for each strategy, in Table 2. The ratio r_2 is plotted in Figure 2 for PCAmGA and UmGA and we may note that Λ_2 for PCAmGA is around 0.25, which is less than 0.4, the level for UmGA.

From Table 2 it is apparent that PCAmGA reaches the lowest levels Λ_p for each p, proving experimentally that our mutation operator maintains highly diverse populations, increasing the chances of finding better solutions.

The PCAmGA was implemented in the Matlab™ programming environment, and it took approximately 20 minutes on a 150 MHz Pentium computer, to obtain good solutions to the filter design problem.

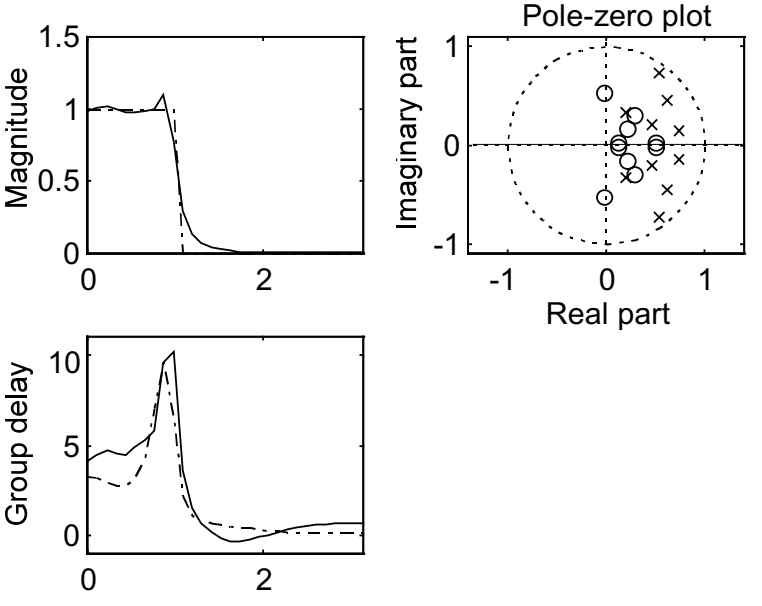


Fig. 1. Best solution found by PCAmGA (*continuous line* = solution characteristic, *dot line* = desired characteristic)

Table 2. Levels Λ_p of the PCA ratios r_p for PCAmGA, UmGA and NUmGA

Λ	**PCA mGA**	**Um GA**	**NUm GA**	Λ	**PCA mGA**	**Um GA**	**NUm GA**
Λ_1	0.1334	0.2266	0.4181	Λ_{12}	0.8133	0.9031	0.9889
Λ_2	0.2413	0.3667	0.6140	Λ_{13}	0.8437	0.9238	0.9919
Λ_3	0.3334	0.4718	0.7075	Λ_{14}	0.8710	0.9416	0.9940
Λ_4	0.4132	0.5567	0.7776	Λ_{15}	0.8957	0.9564	0.9951
Λ_5	0.4843	0.6273	0.8326	Λ_{16}	0.9179	0.9690	0.9965
Λ_6	0.5475	0.6868	0.8760	Λ_{17}	0.9375	0.9792	0.9976
Λ_7	0.6038	0.7377	0.9109	Λ_{18}	0.9548	0.9872	0.9984
Λ_8	0.6546	0.7814	0.9380	Λ_{19}	0.9698	0.9930	0.9991
Λ_9	0.7003	0.8188	0.9581	Λ_{20}	0.9825	0.9969	0.9995
Λ_{10}	0.7420	0.8512	0.9730	Λ_{21}	0.9928	0.9991	0.9999
Λ_{11}	0.7795	0.8791	0.9830	Λ_{22}	1	1	1

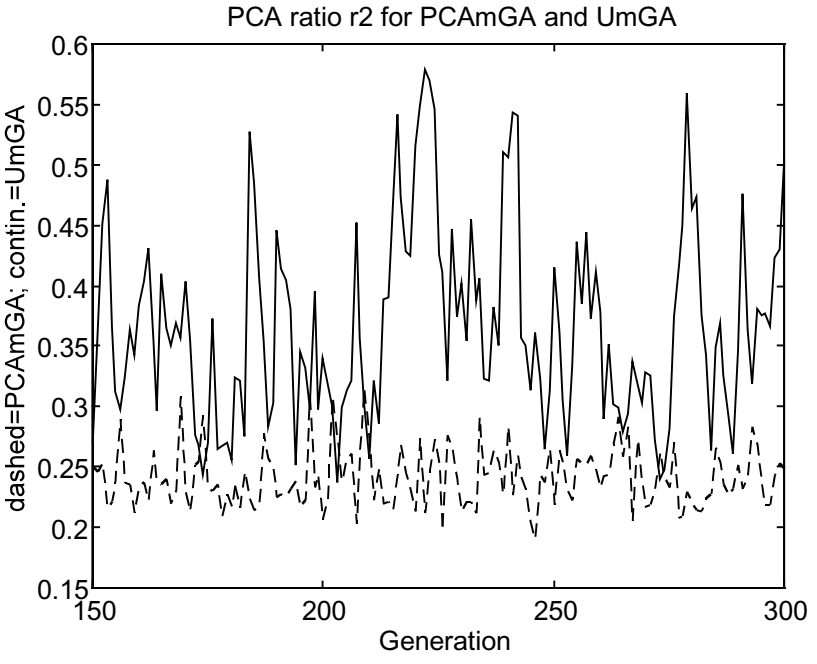


Fig. 2. PCA ratio r_2 for PCAmGA (*dash line*) and UmGA (*continuos line*)

5 Conclusions

In this paper we introduce a new mutation operator for a real-coded GA, based on PCA. The mutation operator searches more effectively, outperforming the classical mutation operators by keeping high levels of genetic diversity in the population, resulting in better solutions found. The tests were performed on a Deczky IIR design method that requires a global search strategy, and we obtained good solutions employing our new mutation operator. The IIR filter found by our strategy (PCAmGA) was better than the solution found by the classical Newton search method.

One may argue that, as our operator involves computations over multiple parents in the population to produce the offspring, it should be regarded as a multiparent recombination operator rather than a mutation operator which traditionally requires one parent to produce one offspring. However, it is also traditionally accepted that the global effect of mutation is that of randomly *changing* the genes' values, while the effect of recombination is that of *exchanging* genetic information between two or more parents. Tacking into account the latter argument, we have chosen to name our operator "mutation", because its end effect is that of randomly changing the genes' values in the gene pool, rather than exchanging the information between parents.

For future work we will focus on improving our method by designing repairing algorithms, as PCA-mutation may yield unfeasible chromosomes. We will also consider a Non Uniform PCA-mutation by decreasing the parameter $c_{\max}$ over time.

We will consider a comparison of PCAmGA to other powerful strategies acting on real-coded chromosomes, such as The Breeder Genetic Algorithm [7], and to mutation-orientated evolutionary methods such as the Evolutionary Strategies [10]. In the light of the No Free Lunch (NFL) Theorems of Wolpert and Macready [12], that basically state that there is no general better optimization strategy, the respective comparison will be done on an extended test set, to clearly identify the problems most suitable for the application of PCAmGA.

References

1. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. NY: Addison-Wesley (1989)
2. Goldberg, D.E.: Real-coded GAs, Virtual Alphabets, and Blocking. Complex Systems. 5 (1991) 153-171
3. Hansen, N., Ostermeier, A.: Convergence Properties of Evolutionary Strategies with the Derandomized Covariance Matrix Adaptation: The (μ/μI, λ)-CMA-ES. In: Proceedings of EUFIT'97, Vol. 1, Verlag Mainz, Aachen (1997) 650-654
4. Holland, J.H.: Adaption in Natural and Artificial Systems. Ann Arbor (1975)
5. Houck, C.R., Joines, J.A., Kay, M.G.: A GA for Function Optimization: A Matlab Implementation. Technical Report. North Carolina State University (1995)
6. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. 3rd edn. Springer-Verlag, Berlin Heidelberg New York (1996)
7. Mühlenbein, H., Schlierkamp-Vosen, D.: Predictive Models for the Breeder Genetic Algorithm. Evolutionary Computation. 1(1) (1993) 25-49
8. Neagoe, V., Stanasila, O.: The Theory of Pattern Recognition. Romanian Academy Publishing House (1992)
9. Oppenheim, A., Schafer, R., W.: Discrete-Time Signal Processing. Prentice-Hall International (1989)
10. Schwefel, H.-P.: Numerical Optimization of Computer Models. Wiley, Chichester (1981)
11. Thornton, C.: Why GAs are Hard to Use. Complexity International. 4 (1997)
12. Wolpert, D.H., Macready, W.G.: No Free Lunch Theorems for Optimization. IEEE Trans. on Evolutionary Computation. 1(1) (1997) 67-82

Model-Based Object Recognition from a Complex Binary Imagery Using Genetic Algorithm

Samarjit Chakraborty[1], Sudipta De[2], and Kalyanmoy Deb[3⋆]

[1] Institut TIK, Eidgenössische Technische Hochschule, Zürich, Switzerland.
[2] KanGAL, Indian Institute of Technology Kanpur, India.
[3] Department of Computer Science, University of Dortmund, Germany.

Abstract. This paper describes a technique for model-based object recognition in a noisy and cluttered environment, by extending the work presented in an earlier study by the authors. In order to accurately model small irregularly shaped objects, the model and the image are represented by their binary edge maps, rather then approximating them with straight line segments. The problem is then formulated as that of finding the best describing match between a hypothesized object and the image. A special form of template matching is used to deal with the noisy environment, where the templates are generated on-line by a Genetic Algorithm. For experiments, two complex test images have been considered and the results when compared with standard techniques indicate the scope for further research in this direction.

1 Introduction

Finding the best transformation that maps an object model into the image of a scene is a central issue in object recognition. There are several approaches to this problem which explicitly rely on results from computational geometry. Among them are geometric hashing [17], alignment [14] and voting [2]. The Hough transform [22], which is recognized as a powerful tool for curve as well as object detection falls into the third category. A different line of approach involves the development of cost functions for measuring the difference between two sets of points or line segments under various transformations. Such cost functions based on the Hausdorff distance have been extensively investigated in both computational geometry [1,3,12] and computer vision [13,24] literatures. Although these methods give good results in the presence of small amounts of noise and occlusion, they do not scale well when applied to complex cluttered scenes, and in the presence of a lot of noise. For example, in a study on the noise sensitivity of the generalized Hough transform by Grimson and Huttenlocher [9], it was concluded that even for moderate amounts of noise and occlusion, these methods can hypothesize many false solutions, and their effectiveness is dramatically reduced. Similar conclusions were made by Sarachik [25] for the geometric hashing

⋆ Author to whom all correspondence should be directed.
e-mail: deb@ls11.informatik.uni-dortmund.de

R. Poli et al. (Eds.): EvoIASP'99 and EuroEcTel'99, LNCS 1596, pp. 150–161, 1999.

paradigm applied to 2-D object recognition. So these techniques are reliable only for relatively simple tasks in the absence of excessive noise and clutter, where the image data corresponding to correct solutions is a large fraction of the total data. In an effort to address this problem, in [6] we proposed a scheme for detecting analytic curves using a Genetic Algorithm (GA) [7] in combination with the Randomized Hough transform [27]. The present paper extends that work to consider the detection of any binary object model in a binary edge map of a scene image. There exists a large volume of literature on detecting curves and objects in noisy as well as cluttered images. But most of them assume a predefined error model, either uniform bounded for feature displacement or a 2D Gaussian. Additionally, several approaches also assume the presence of the model in the image and the worst case search time in the presence of noise is exponential in the problem size [10]. The proposed method in this paper is flexible, and does not assume any error model. It is particularly effective in the case of complex images where the number of pixels belonging to the object being searched for is a very small fraction of the total number of edge pixels. In image processing literature there is a mathematical distinction between *clutter* and *noise*. The former might refer to all features or points that come from something different than the model, where as *noise* usually refers to the phenomenon in which the identified locations of the image points are slightly displaced from where they should be. Coupled with these, there might be several spurious data points in the image arising out of various sources, for example, edge points arising out of brightness discontinuities and imperfect edge detection. For the purpose of this paper it is not required to distinguish between these different errors and we will refer to all such points jointly as *false attractors*.

The concept of using GA for curve extraction has been explored in the past [11,23]. But the problems of noise or clutter were not considered. Object recognition in a complex image using GA has also been attempted [26]. The method that we present is more flexible in terms of the allowable similarity between the model and the object in the scene image. This has important advantages for successful recognition of real life images, since it results in a flexibility in evaluating an hypothesis about the occurrence of the object in the scene. We illustrate this through examples presented in Section 4.

There has been an enormous amount of research in automatic object recognition. But despite this fact the problem remains largely unsolved. A comprehensive overview of this subject from a variety of perspectives can be found in [21]. We believe that the use of GA can help in dealing with the uncertainties that arise in any practical object recognition system. Further, since such a task involves a very large search space, a suitably designed GA approach can reduce the search time by several orders of magnitude with respect to an exhaustive search.

In this paper, the object recognition task is performed by representing the model and the image in the form of their binary edge pixels. This representation has a number of benefits. Edge pixels are robust to changes in sensing conditions and edge-based techniques can be used with many imaging modalities. Several

previous approaches have considered modeling objects as a set of straight line segments, and matching these to the straight line segments extracted from an image [5,19]. Our use of the complete edge map to model objects, rather than approximating them as straight line segments, allows irregularly shaped objects to be modeled accurately. We specifically address images with a very large fraction of points constituting the false attractors by using a special form of template matching and compare our results with standard methods. Our templates are generated on-line, guided by the GA.

In the next section we briefly identify cases where standard methods fail due to the presence of a large number of false attractors. Towards this we use an example of straight line detection, following which we describe our method. In Section 4 we describe test results with two images and compare the performance with standard methods. Section 5 concludes the paper.

2 Motivation

The various approaches towards searching for the occurrence of an object in a scene can be roughly classified depending on whether the search is performed in the *correspondence space*, *transformation space*, or both. Correspondence space is the space of *matches*, which are sets of pairings between model and image features or points. Transformation space is the space of possible object poses.

The interpretation tree approach [8] exemplifies those methods that search entirely in the correspondence space. Its name refers to a search tree of choices concerning the interpretation of each image feature. Proceeding from the root of the tree, the match search examines an additional image feature at each level of the tree. Branches at each level represent different choices among model features that can be matched to that image feature, plus the choice of matching nothing at all to it. A complete interpretation of the image, assigning some subset of image features to corresponding model features, is associated with each of the tree's leaves. This method is computationally very costly and is generally exponential to the number of image and model pixels. Hence in the presence of excessive number of false attractors, such a method is rendered infeasible.

The generalized Hough transform is an example of a method that searches the transformation space. An accumulator array indexed by parameters of object pose, is first initialized as empty. Then, for each possible match between one image feature and one model feature, poses consistent with that match are determined and votes are cast in the bins of the accumulator array corresponding to those poses. The second stage is an exhaustive search for parameters in the accumulator array which are local maxima. Each such local maximum represents a candidate match between the model and the image. In this approach, points on the same object occurring in the image result in points in the parameter space which are close together, whereas the false attractors result in randomly distributed points in the parameter space. Thus a large cluster of points in the parameter space represent a match between the model and an object in the image. The validity of this assumption, however, depends on there being a low

likelihood that clusters due to false attractors will be comparable or larger in size than clusters due to points on genuine objects. We believe that in many real life images, this assumption does not hold. Fig. 1(a) shows two straight lines L_1 and L_2, where each line is composed of a small number of disconnected points. In Fig. 1(b), random noise is superimposed on the line L_1 (Fig. 4 in Section 4 shows one example where such a situation really arises in practice). Let us call the lines in Fig. 1(a) as *true lines* and the line in Fig. 1(b) that corresponds to line L_1 of Fig. 1(a), as a *pseudo line*. Line L_2 in this figure still remains a true line. If our model is a simple straight line, then ideally the recognition algorithm should detect both L_1 and L_2 from Fig. 1(a) but only L_2 from Fig. 1(b). Note that there are a large number of pseudo lines in the noise region in Fig. 1(b). Since the number of points on each of these pseudo lines is comparable or more than than the number of points on the line L_2, it gets masked in the parameter space by these pseudo lines.

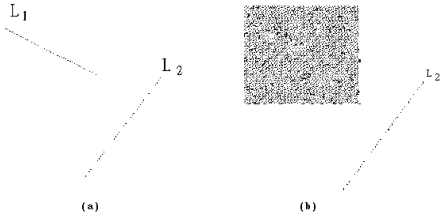


Fig. 1. A binary edge image (a) Two straight lines (b) Noise superimposed on one of the lines

3 A Genetic Algorithm for Object Recognition

To overcome the effects of noise in curve detection, the Window RHT and Random Window RHT due to Kälviäinen *et al.* [16], randomly place a window on an edge point and try to locate a curve within the window. Similarly template matching [4,28] has been widely used in computer vision for object recognition. An object in an image is defined to be recognized if it correlates highly with a template image of the hypothesized object. The template image is usually a transformed version of the model of the hypothesized object. Our technique is conceptually similar to this. We place a weighted template on an edge point and measure the weighted difference between pixels on a real object and the spurious points surrounding it. The templates are constructed online, guided by the GA.

A crucial problem with ordinary template matching is the size of the search space [20,18]. An attempt to overcome this is through the randomized versions like Window RHT and Random Window RHT. We feel that a search guided by a GA is more superior than a simple random search and can reduce the search time by orders of magnitude.

3.1 Generating Templates from Model Images

Given a binary edge map and a model, or possibly a library of models, our objective is to identify the occurrence of these models in the image. If a model is represented by the set of its edge pixels $\mathcal{M}$, then a template $\mathcal{T}$ is generated from $\mathcal{M}$ by choosing three parameters that describe a transformation of $\mathcal{M}$ into $\mathcal{T}$, along with some additional parameters which determine the quality of allowable matches. The parameters used for transformation are translation, rotation and scaling, and possibly also mirror image about any arbitrary line. We say that the model $\mathcal{M}$ occurs in the given image at the location indicated by the template $\mathcal{T}$ if $\sum_{x \in \mathcal{T}} Z_x \geq N_{min}$, where Z_x is the gray level of the pixel x (0 or 1 in a binary edge map) in the binary edge map of the image. The template $\mathcal{T}$ is the set of points $\{x : d(x, \mathcal{I}(x')) \leq \delta$ and $x' \in \mathcal{M}\}$, where $\mathcal{I}$ is some composition of translation, rotation and scaling, and $d(x, \mathcal{I}(x'))$ is the Euclidean distance between the points x and $\mathcal{I}(x')$. δ is a parameter which describes the width of a strip or band around the transformed model, which allows for certain tolerance. N_{min} is the minimum number of pixels of the edge detected image that must occur within the template so that the presence of the hypothesized object corresponding to the model $\mathcal{M}$ can be ascertained. A relatively large value of δ allows objects to be detected which are fuzzy or have a weak similarity with the model $\mathcal{M}$.

For images with relatively less or no spurious points such as Fig. 1(a), this formulation is sufficient and is in fact similar to the Window RHT used for curve detection, except for the fact that we do not use any transformation mapping from the image to the parameter space as is common in Hough transform. Rather, we simply count the number of points lying within the template $\mathcal{T}$. But in the case of images with a large proportion of false attractors such as Fig. 1(b), whenever the template is placed on a region consisting of such points, a false alarm in the form of a pseudo object will be raised. To extend this method to include such images, we formulate a weighted template rather than the simple one described above. The response of the template $\mathcal{T}$ under this formulation is given by $R = \sum_{x \in \mathcal{T}} W_x Z_x$, where W_x is the weight or coefficient of the pixel x. We shall say that the model $\mathcal{M}$ occurs in the image at the location indicated by template $\mathcal{T}$ if the response R of the template is greater than a constant R_{min}, fixed, depending on the dimensions of the model, template width δ, and the coefficients W_x. The coefficients of pixels that lie away from the transformed model i.e. $\mathcal{I}(\mathcal{M})$, are assigned negative values. So when a lot of spurious points are present in the neighborhood of $\mathcal{I}(\mathcal{M})$, as in the case of the pseudo lines in Fig. 1(b), the positive response due to the points on and near $\mathcal{I}(\mathcal{M})$ is offset by the negative response due to the spurious points surrounding it. As a result, false alarms are avoided.

3.2 Parameter Search Space and the Use of GA

Even if a particular object is known to be present in the image a priori, the space of transformations from the model to the image is extremely large. Hence an exhaustive search of this space would take too long to find a good match between

templates and images. A random search in the presence of excessive noise and clutter is also not beneficial. So instead of randomly choosing the transformation parameters to generate a template, we use a genetic algorithm to search the parameter space for all instances of objects for which the template response is greater than R_{min}. For this, each of the parameters - x and y-coordinates of the translation vector, the rotation angle, and the scaling factor, are coded as fixed length binary strings. The resulting string, obtained by concatenating all these strings, gives the chromosomal representation of a solution to the problem. Note that the domains of each of the parameters may be different and the length of the string coding a given parameter depends on the required parameter resolution. The fitness of a solution is taken to be the response of the weighted template, as described in the previous section.

Since in practical situations an exact match between the model and a hypothesized object is not expected, we construct the template $\mathcal{T}$ such that points near the transformed model $\mathcal{I}(\mathcal{M})$ are associated with positive coefficients and points lying further away have negative coefficients. This, along with a suitably chosen value of the minimum response R_{min}, offers considerable flexibility regarding the quality of the resemblance between the model and the detected objects.

Creation of initial population. In most GA applications, the initial population consists of entirely random structures to avoid convergence to a local optima. However, in this problem, the question is not of finding the global optima, but of finding all solutions with fitness greater than R_{min}. To identify prospective regions of the search space, the *hypothesize and test* paradigm commonly used in visual object recognition might be effectively used. In [6] we used a Randomized Hough transform for this purpose. For object recognition, a variation of this method similar to the generalized Hough transform might be used to generate an initial set of hypotheses. Towards this, pairs of points are randomly chosen and possible transformations which map these two points onto points in the image are computed, as in the alignment method [14]. However instead of explicitly testing such transformations, the *count*, in the accumulators representing the parameter space, corresponding to such transformations are incremented by one. After repeating this process for a predefined number of times, points in the parameter space with counts exceeding a predefined threshold represent candidate hypotheses. The GA searches the entire parameter space with a bias towards these hypotheses. Corresponding to each candidate hypothesis, a suitable number of solutions are introduced into the initial population. Further, a fixed number of random samples from the solution space are also introduced. The total number of solutions is kept fixed over all the generations.

It should be noted that the above mentioned method of generating candidate hypotheses is rendered ineffective in the presence of excessive clutter and extreme scaling, where this scheme is no better than randomly generating the initial population. However, for images with even moderate amounts of noise and clutter, it can lead to a considerable speedup.

Selection. The selection used here falls into the category of dynamic, generational, preservative, elitist selection [27]. Let there be M distinct solutions in a given generation, denoted by $S_1, S_2, \ldots, S_M$. The probability of selecting a solution S_i into the mating pool is given by :

$$P(S_i) = \frac{\mathcal{F}(S_i)}{\sum_{j=1}^{M} \mathcal{F}(S_i)}$$

Where $\mathcal{F}(S_i)$ is the fitness of the solution S_i. A fixed number of solutions are copied into the mating pool according to this rule and a small number of remaining solutions are randomly generated. In each generation, a fixed number of best solutions of the previous generation are copied in place of the present worst solutions, if they happen to be less fit compared to the former. This is a slight modification of the Elitist model where only the best solution is preserved.

Crossover and Mutation. Because of the number of parameters involved, it is intuitive that the single point crossover operation may not be useful. So crossover is applied to each substring corresponding to each of the parameters - x and y-coordinates of the translation vector, the rotation angle, and the scaling factor, the operation being the usual swapping of all bits from a randomly chosen crossover site of the two parents, chosen randomly from the mating pool [29]. Hence this crossover is similar to the standard single-point crossover operator, but operated on substrings of each parameter. Therefore, there are four single-point crossovers taking place between two parent strings.

We have used a classical mutation operator in which each bit position of the solution strings is complemented with a small mutation probability.

The overall algorithm. The initial population consisting of a fixed number of solutions is created as already described. In each generation, the entire population is subjected to selection, crossover and mutation. At the end of each generation, edge pixels corresponding to solutions having fitness greater than R_{min} are removed from the edge map. After fixed number of generations, the accumulators corresponding to the parameter space used for generating the candidate hypotheses are reset and the voting process is repeated to generate a fresh set of hypotheses. Candidate solutions corresponding to these are then introduced into the population and whole process is once again repeated. This iteration is continued until no new curve segments are extracted for a given number of generations, which in our experiments was set to 200.

4 Test Results and Comparisons

We have experimented with two different images. For the ease of comparison with standard methods, in our first experiment the model is a simple straight line. Although this is the simplest possible case, as evident from the previous sections, our algorithm is blind to this fact. For comparing the performance of our method with Hough transform which is the most popular method for straight line detection, we used a public domain software package XHoughtool

[15], where a number of non-probabilistic and probabilistic Hough transform algorithms have been implemented.

As indicated in the previous section, there are various parameters that our algorithm uses. The parameters related to the template are its width, the coefficients or weights associated with each pixel, and the threshold response R_{min}. The allowable quality or degree of correspondence between the model and the objects extracted from the image is determined by the template coefficients and its width. A wide template with more than one row of positive coefficients will detect objects whose pixels are spread out along its width compared to the model in question. Thus, a suitably designed template, along with a proper threshold value R_{min}, will be able to distinguish between an object having a relatively weak similarity with the model, and a false attractor. In our first experiment where the model is a straight line, we have used a template width of 3, to detect only perfect straight lines. The coefficients of all pixels lying on the straight line were set to 2 and the others to -1 as shown in Fig. 2. Too low a value, R_{min}, of the threshold might detect a pseudo line where as a too high value might miss a faint, disconnected, but visually detectable line. The results shown in this section were obtained with R_{min} set to the length of the transformed line, i.e. $\mathcal{I}(\mathcal{M})$.

-1	-1	-1	.....	-1	-1	-1
2	2	2	.....	2	2	2
-1	-1	-1	.....	-1	-1	-1

Fig. 2. A mask of width 3

For the GA parameters, we used a mutation probability of 0.1 and any population size around 100 was found to work well. In each generation, 25% of the solutions were randomly created and the rest copied from the mating pool in accordance with the fitness proportionate selection. Further, the best 10% solutions of the previous generation were copied in place of the worst solutions of the current generation.

Fig. 3(b) shows a 512 by 512 binary image obtained after edge detection of the corresponding gray scale image shown in Fig. 3(a). Our model in this case consists of a simple straight line. Note the three disconnected, but visible real lines in the image, two at the center and one the the extreme left end. The straight lines detected by our algorithm are shown in Fig. 3(c). Altogether seven different Hough transform algorithms are implemented in the XHoughtool package. In spite of a serious attempt being made to select the test parameters for each method as optimally as possible, none of the algorithms gave useful results because a large number of pseudo lines were detected. A typical result is shown in Fig. 3(d). Since the number of edge points lying on the real lines are much less compared to those lying on many of the pseudo lines, no suitable accumulator threshold value exists which can detect only the real lines. Generally these algo-

rithms work well even in the case of noisy images, where the lines are connected and the number of edge points lying on these lines are at least comparable to the number of noise points. It is to be noted in this example that there are edge detectors which if used for Fig. 3(a) along with proper thresholding, would eliminate most of the false attractors now appearing in Fig. 3(b). In such a situation, a simple HT algorithm would suffice. However, we have used the edge detection algorithm incorporated in *xv*, the interactive image viewer available on any X-window system. This, in some way artificial route, was adopted only to illustrate a situation where the proposed method might be useful. Secondly, the 'lines' detected in Fig. 3(c) are actually edges, with no width. But for the purpose of this algorithm we do not distinguish between an edge and a line.

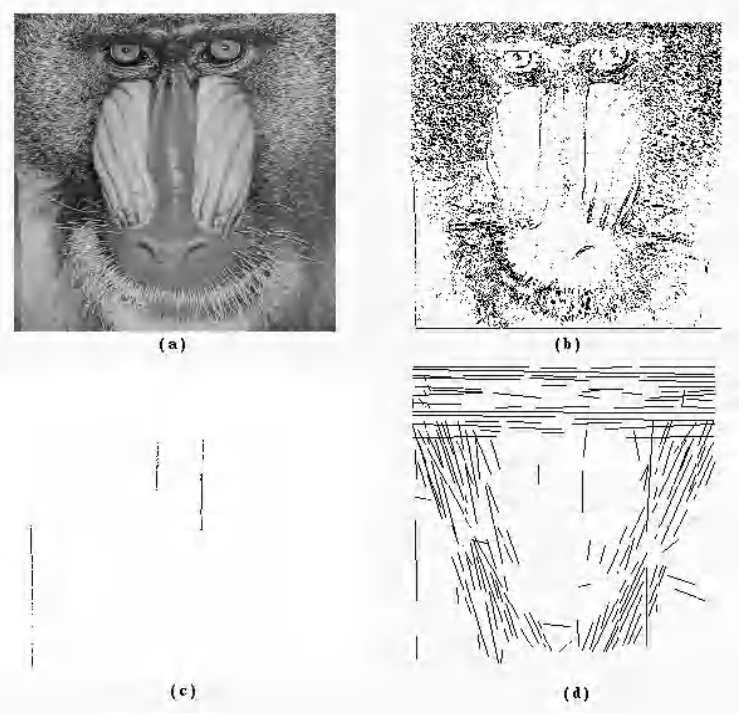


Fig. 3. Test results with the model being a simple straight line (a) A 512 by 512 gray scale image (b) The corresponding binary edge map obtained after edge detection (c) Straight lines detected by the proposed method (d) A typical result obtained using a Hough transform algorithm

Our second example is a synthetic image shown in Fig. 4 consisting of instances of the letters **A** and **Z** under various orientations and scaling, along with random noise. The model is a letter **A**. Note that there are seven instances of the letter **A** in the scene image. In most of the test runs the algorithm could

detect all the seven **A**s and avoid any false alarms. However, it was crucial to approximately choose the value of the threshold R_{min}. In this case also we used a template width of three as in the previous example.

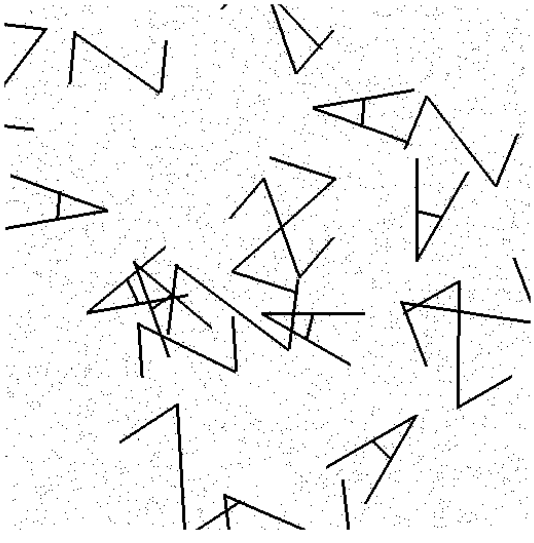


Fig. 4. A binary scene image where the model **A** is to be detected

We should emphasize here that the procedure for generating the initial solutions described in the last section, is much more effective in the case of our second example where the proportion of false attractors is much less compared to the first.

5 Summary

The Hough transform and its variants are the most popular methods for detecting analytic curves from binary edge data. However, they do not scale well when applied to complex environments in the presence of excessive noise and clutter. In [6] we presented a GA in combination with the Randomized Hough transform but using a different scoring function, to deal with such environments. This paper extended that technique to incorporate model based object recognition. Towards this we used a special form of template matching which offers a considerable flexibility regarding the quality of the allowable matches. Although there has been attempts to use simple random search for several computer vision problems, a search guided by a GA is probably superior in this case.

For future work, further experimentation could be performed using a variety of different image and model pairs to illustrate the general applicability of this

method. One possible application domain might be automatic target recognition where, because of its military applications, the goal is to avoid the object being detected. It should also be possible to utilize other methods than the one described here for generating the set of hypotheses used for initializing the population. Further, it would be interesting to extend the set of transformations considered here with shearing for example, to test weak similarities between the model and the image.

Acknowledgements

The authors are grateful to the anonymous referees for their constructive criticism and suggestions.

References

1. P.K. Agrawal, M. Sharir, and S.Toledo. Applications of parametric searching in geometric optimization. In *Proc. of 3rd. ACM SIAM Symp. on Discrete Algorithms*, pages 72–82, 1992.
2. T. Akutsu, H. Tamaki, and T. Tokuyama. Distribution of distances and triangles in a point set and algorithms for computing the largest common point sets. In *Proc. 13th. Annual ACM Symp. on Computational Geometry*, pages 314–323, Centre Universitaire Méditerranéen, Nice, France, 1997.
3. H. Alt, B. Behrends, and J. Blömer. Measuring the resemblance of polygonal shapes. In *Proc. of 7th. Annual ACM Symposium on Computational Geometry*, pages 186–193, 1991.
4. Dana Ballard and Christopher M. Brown. *Computer Vision*. Prenctice Hall, 1982.
5. Ross J. Beveridge. *Local Search Algorithms for Geometric Object Recognition: Optimal Correspondence and Pose*. PhD thesis, University of Massachusetts, Amherst, May, 1993.
6. S. Chakraborty and K. Deb. Analytic curve detection from a noisy binary egde map using genetic algorithm. In *Proc. 5th. International Conference on Parallel Problem Solving from Nature (PPSN V)*, pages 129–138, 1998. Lecture Notes in Computer Science 1498.
7. D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, M.A., 1989.
8. W. E. L. Grimson. *Object Recognition by Computer : The Role of Geometric Constraints*. MIT Press, 1990.
9. W. E. L. Grimson and D. P. Huttenlocher. On the sensitivity of the Hough transform for object recognition. *IEEE Trans. Pattern Anal. Machine Intell.*, PAMI-12:255–274, 1990.
10. W. Eric L. Grimson. The effect of indexing on the complexity of object recognition. Technical Report A.I. Memo No. 1226, Artificial Intelligence Laboratory, MIT, 1990.
11. A. Hill and C. J. Taylor. Model-based image interpretation using genetic algorithms. *Image and Vision Computing*, 10:295–300, 1992.
12. D.P. Huttenlocher, K. Kedem, and M. Sharir. The upper envelope of voronoi surfaces and its applications. *Discrete and Computational Geometry*, 9:267–291, 1993.

13. D.P. Huttenlocher, G.A. Klanderman, and W.J. Rucklidge. Comparing images using the Hausdorff distance. *IEEE Trans. Pat. Anal. and Mach. Intel.*, 15:850–863, 1993.
14. D.P. Huttenlocher and S. Ullman. Recognizing solid objects by alignment with an image. *Inter. Journal of Computer Vision*, 5(2):195–212, 1990.
15. H. Kälviäinen, P. Hirvonen, L. Xu, and E. Oja. Houghtool–a software package for Hough transform calculation. In *Proceedings of the 9th Scandinavian Conference on Image Analysis*, pages 841–844, June 1995. (http://www.lut.fi/dep/tite/XHoughtool/xhoughtool.html).
16. H. Kälviäinen, L. Xu, and E. Oja. Recent versions of the Hough transform and the Randomized Hough transform : Overview and comparisons. Technical Report 37, Department of Information Technology, Lappeenranta University of Technology, Finland, 1993.
17. Y. Lamdan and H.J. Wolfson. Geometric Hashing: A general and efficient model-based recognition scheme. In *International Conference on Computer Vision*, pages 238–249, 1988.
18. Z. Q. Liu and Terry M. Caelli. Multiobjective pattern recognition and detection in noisy backgrounds using a hierarchical approach. *Computer Vision, Graphics, and Image Processing*, 44:296–306, 1988.
19. Chris Loader. Local search algorithms for 2d geometric object recognition. Master's thesis, Department of Computer Science, The University of Western Australia, 1995.
20. Avrahm Mergalit and Azriel Rosenfeld. Using probabilistic domain knowledge to reduce the expected computational cost of template matching. *Computer Vision, Graphics, and Image Processing*, 51:219–234, 1990.
21. Arthur R. Pope. Model-based object recognition : A survey of recent research. Technical Report TR-94-04, Department of Computer Science, University of British Columbia, January, 1994.
22. J. Princen, J. Illingworth, and J. Kittler. A formal definition of the Hough transform : properties and relationships. *J. Math. Imaging Vision*, 1:153–168, 1992.
23. G. Roth and M. D. Levine. Geometric primitive extraction using a genetic algorithm. *IEEE Trans. Pattern Anal. Machine Intell*, PAMI-16(9):901–905, 1994.
24. W.J. Rucklidge. Locating objects using the Hausdorff distance. In *Proc. of 5th. International Conference on Computer Vision*, pages 457–464, 1995.
25. Karen B. Sarachik. Limitations of geometric Hashing in the presence of gaussian noise. Technical Report A.I. Memo No. 1395, Artificial Intelligence Laboratory, MIT, 1992.
26. D. L. Swets, B. Punch, and W. John. Genetic algorithms for object recognition in a complex scene. In *Proceedings of the International Conference on Image Processing*, pages 595–598, October, 1995. Washington, D.C.
27. L. Xu and E. Oja. Randomized Hough transform (RHT) : Basic mechanisms, algorithms, and computational complexities. *CVGIP : Image Understanding*, 57(2):131–154, 1993.
28. Leonid P. Yaroslavsky. *Digital Picture Processing.* Springer Verlag Berlin, 1985.
29. K. S. Y. Yuen, L. T. S. Lam, and D. N. K. Leung. Connective Hough transform. *Image and Vision Computing*, 11(5), 1993.

TestPatternGenerationunder Low Power Constraints

Fulvio Corno, Maurizio Rebaudengo, Matteo Sonza Reorda, Massimo Violante

Politecnico di Torino
Dipartimento di Automatica e Informatica
Corso Duca degli Abruzzi 24 I-10129, Torino, Italy
{corno, reba, sonza, violante}@polito.it

Abstract. A technique is proposed to reduce the peak power consumption of sequential circuits during test pattern application. High speed computation intensive VLSI systems, as telecommunication systems, make power management during test a critical problem. A Genetic Algorithm computes a set of redundant test sequences, then a genetic optimization algorithm selects the optimal subset of sequences able to reduce the consumed power, without reducing the fault coverage. Experimental results gathered on benchmark circuits show that our approach decreases the peak power consumption by 20% on the average with respect to the original test sequence generated ignoring the power dissipation problem, without affecting the fault coverage.

1 Introduction

Testing a VLSI circuit is often performed through the application to its Input Pins of a sequence of values, such that the values that one can observe on the Output Pins of the fault-free circuit are different from the ones appearing on the same pins of any faulty circuit. One of the main problems is how to generate a suitable sequence of values to be applied to the Input Pins (Automatic Test Pattern Generation, or *ATPG*).

The economical importance of ATPG tools for digital circuits is continuously growing, and the demand for efficient algorithms and tools able to handle the current circuits is thus very strong. Correspondingly, there have been significant research efforts in this field, which produced tens of proposals in terms of ATPG algorithms and techniques.

In the last few years, several methods have been proposed for sequential circuits, which exploit Genetic Algorithms (GAs). In fact, Evolutionary Techniques allow to tame randomness and successfully exploit it for finding optimal solutions. Results showed that the approach is very flexible and provides good results for large circuits, where other methods fail. An overview of such techniques is presented in [1].

Due to the great increase in circuit size and complexity, the Test Pattern Generation problem is traditionally considered critical from the point of view of the required computational power, and a significant amount of research activities has been devoted to it in the past years [2]. The ATPGs are thus evaluated according to three parameters: the attained Fault Coverage, the required CPU time, and the number of generated Test Vectors.

R. Poli et al. (Eds.): EvoIASP'99 and EuroEcTel'99, LNCS 1596, pp. 162–170, 1999.

Now, something is changing due to technological novelties and market's stimuli. The recent development of complex, high-performance, low-power devices implemented in deep submicron technologies creates a new class of more sophisticated electronic products. Telecommunication systems are an emblematic example of this new class of systems: they operate at high frequency, mix analog and digital components and require high computation capabilities (e.g. wireless communications systems). This new class of systems makes power management a critical parameter.

Starting from the observation that the power consumption during test is significantly higher than that during normal circuit operation [3], researchers have devoted their efforts to the definition of new test algorithms, able to consider the power consumption along with the previously mentioned parameters.

Several approaches have been proposed, which can be classified as:

- *ATPG-integrated optimization* : the test pattern is optimized for low-power during the test generation phase [4][5].
- *post-ATPG optimization*: the test pattern is first generated by a classical ATPG, then it is optimized for power [6][7][8][9].

The method proposed in this paper stems from the observation that, given a fault to be tested, several sequences may exist able to detect it. The test sequences are equivalent form the point of view of Fault Coverage, but they may show a significantly different behavior as far as the power consumption is concerned. In particular, as motivated in Section 2, *peak power* consumption is considered.

Our method proposes to exploit a traditional GA-based ATPG to compute a *redundant* test pattern, i.e., a test pattern where a single fault is covered by several *different* sequences. Then, an optimization algorithm is applied to select an optimal subset from the pool of previously computed test sequences, in order to minimize the peak power consumption without affecting the Fault Coverage.

To enlarge the search space of the optimization algorithm, an analysis of the fault list is preliminarily performed to identify the faults detected only by sequences responsible for the peak power consumption (defined as *critical faults*). Then the redundant test pattern is generated targeting the critical faults, only. This approach allows also a reduction of the CPU time overhead introduced by the redundant ATPG.

The proposed method is organized in four independent steps: critical fault identification, redundant test pattern generation, peak power estimation and optimal sequences selection. To demonstrate the feasibility of the approach and to evaluate its performance we made some preliminary experiments: switch level simulation is exploited to evaluate peak power consumption, and genetic algorithms were implemented both to generate redundant test pattern and to select the optimal subset of sequences for minimal power consumption. The experimental results gathered on a subset of the ISCAS benchmark circuits show a reduction ranging from 1% to 52% with respect to the original test patterns generated ignoring the power consumption problem, without affecting fault coverage.

The remainder of our paper is organized as follows. Section 2 presents the problems raised by power consumption during test application. Section 3 describes the approach adopted to minimize the power dissipation during test application. Experimental results on benchmark circuits are presented and discussed in Section 4. Section 5 draws some conclusions.

2 Motivation

In this paper, we assume that the circuit under test has been implemented by using a static CMOS technology. The source of power dissipation in a CMOS circuit can be classified as static and dynamic. Static dissipation is due to leakage currents, and can be neglected since it has a small magnitude. Dynamic dissipation is due to the current required to charge and discharge the load capacitance within the circuit, and is the dominant term of power dissipation for CMOS circuits.

Given a CMOS gate *g*, its dynamic power consumption can be expressed as:

$$P_g = 0.5V_{dd}^2 C_L E_g(sw) f_{switch} \quad (1)$$

where V_{dd} is the supply voltage, C_L is the physical capacitance at the output of the gate *g*, $E_g(sw)$ is the number of time the output of gate *g* toggles, and f_{switch} is the clock frequency. For high frequency, computation extensive applications, such as telecommunication systems, the term $E_g(sw)f_{switch}$ has a significant magnitude, leading to high power consumption.

High power consumption produces high temperatures that tend to exacerbate several silicon failure mechanisms, in particular electro-migration [10]. To improve circuit performance and portability, the current trend is to adopt low-power design techniques and to reduce the package size by exactly matching the power dissipation during the normal mode operation [4]. It is therefore necessary to apply test vectors causing a power consumption not higher than that during normal operation or to remove any excessive heat generated during test using cooling equipment. Due to the constantly increasing density of circuits, the use of cooling equipment is increasingly difficult to adopt.

The problem becomes even more evident when addressing the test of bare-dies since the power dissipation capability of a bare-die is lower than that of a packaged chip [11]. The test applied before packaging stresses the chip much more than in any later stage. Hence, some of the bare-dies may fail during this test session even if they don't have manufacturing defects, if power dissipation constraints are not taken into account when preparing the test sequences. Another factor that must be taken into account is the impedance of the probes used to carry the input and power supply signals to the bare-die, which is normally higher than that of the pins of a circuit package [12]. High power consumption corresponds to high currents through the power supply and ground probes. Due to the high probe impedance, the bare-die is subject to high power and ground noise, which are given by:

$$V_{noise} = L \cdot \frac{dI}{dt} \quad (2)$$

where *I* is the current through the power supply and ground probes, and *L* is the probe impedance. When *L* is higher than usual, the term *dI/dt* which is closely correlated to the circuit switching activity) must be reduced in order to maintain V_{noise} under a threshold. Otherwise, the circuit under test could erroneously change its logic state, failing the test, and good dies could be classified as faulty due to excessive noise. From another point of view, this means that the test sequences for a bare-die have to satisfy more stringent power constraints than those for a packaged chip.

The above mentioned phenomena reduce the die-yield (which is the ratio of good dies available for packaging to the total number of dies etched) and hence rise the IC cost. Reduction in die-yield may not be significant for small, low-density circuits. However, for large and high density circuits, as in the case of Multi-Chip-Modules (MCMs), the problem becomes more significant.

In the past, the problem of power dissipation during test was a minor issue since the test was performed at a speed lower than the normal operation speed. Conversely, today circuits are tested at higher clock rates, if possible at the circuit normal clock rate (at-speed testing). Power dissipation during test is therefore expected to rise [4].

During circuit design, the peak power consumption is a critical issue since it determines the thermal and electrical limits of component, the system packaging requirements, and heat sinks dimensions [13]. We can thus conclude that when dealing with high-density systems as the modern ASICs or MCMs, to perform a non-destructive test we have to satisfy all the power constraints defined in the design phase. Therefore, the peak power consumption during test must be kept under a well defined threshold.

3 Proposed approach

Our method exploits a GA-based test pattern generator to compute a *redundant* test pattern, i.e., a test pattern where a single fault is covered by several *different* sequences. Then, an optimization algorithm is applied to select an optimal subset from the pool of test sequences previously computed, in order to minimize the peak power consumption without affecting the Fault Coverage. The existence of multiple sequences able to detect a given fault and the significantly different power consumption of the sequences motivate our approach.

To minimize the CPU time required by the ATPG for computing a redundant test pattern, an analysis of the fault list is preliminarily performed. The *critical* faults for power consumption are identified: they are those faults detected only by sequences leading to the peak power consumption. Then the redundant test pattern is generated targeting the critical faults, only.

The whole process has been organized in four independent steps:

1. *critical fault identification*
2. *redundant test pattern generation*
3. *peak power estimation*
4. *optimal test sequence selection*.

In Fig. 1 the general environment is reported. In the following, the different steps are analyzed.

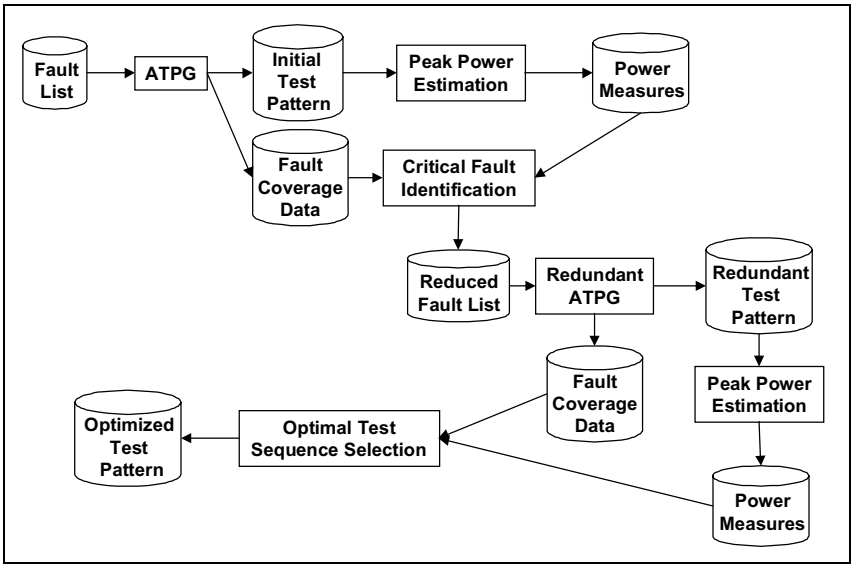


Fig. 1. The environment for low power testing.

3.1 Critical fault identification

A fault is said *critical* if all the sequences able to detect it have a high peak power consumption. The peak power optimization process has thus to target critical faults, only, with the intent of replacing the sequences that have a high power consumption with others less consuming covering all the critical faults.

To identify the critical faults, a first ATPG experiment is performed. In this step no redundancy is introduced in the test pattern. The peak power consumption of the obtained test pattern is measured, and the faults covered only by sequences corresponding to the peak power consumption are selected: the obtained reduced fault list is then used as the target for the following redundant test pattern generation.

3.2 Redundant test pattern generation

A test pattern composed of several independent sequences is said to be *redundant* if one or more sequences can be removed without affecting the fault coverage attained by the test pattern.

The redundant ATPG module inserted in the flow on Figure 1 has the peculiarity of purposely introducing a redundancy in the test pattern. This redundancy allows us to ignore the power consumption problem during test pattern generation with the fault coverage as the main goal of the algorithm.

The adopted ATPG does not need to be adapted to the low power objective, the only required modification being the introduction of an artificial redundancy in the test pattern. A fault is considered as detected only if it covered by at least *M*

sequences, *M* being the *redundancy factor*. To effectively explore the search space of test sequences, a GA-based ATPG [14] is used.

The advantage of this approach is to simply adopt a standard fault coverage oriented ATPG with minimal modifications. The only cost introduced is an increase in CPU time required to generate the redundant test pattern. On the other hand, being the critical faults a small portion of the fault list, we expect to have a reduced CPU time overhead.

3.3 Peak power estimation

The IRSIM [16] switch-level simulator is adopted to measure the peak power consumption of a test sequence. The power consumption of each pattern belonging to a test sequence is first computed. Then, the peak power consumption of the sequence is computed by identifying the maximum among the values of each vector of the sequence.

3.4 Optimal test sequence selection

The goal of this step is to generate the optimal test pattern satisfying the following constraints:

- same fault coverage of the redundant test pattern
- minimal peak power consumption.

This goal is reached starting from the redundant test pattern generated during the previous step. The minimization algorithm has to select the optimal subset of sequences that satisfies the constraints. The problem is a classical *set covering* problem. For the purpose of this paper we devised a Genetic Algorithm to solve the optimal test sequence set selection. The following sub-section details the adopted fitness function.

3.5 Fitness

The fitness function definition is a critical choice since it allows to tune the algorithm to find an optimal solution. As it has been said before, the goal of the algorithm is to minimize the power consumption satisfying the constraint to cover the complete set of faults covered by the original fault list. The following fitness function has been adopted:

- if the Fault Coverage attained by the *i-th* individual (FC_i) is lower than the Fault Coverage (FC) attained by the redundant test pattern, then the fitness function is:

$$f(i) = FC_i - FC < 0 \quad (3)$$

- if the Fault Coverage attained by the *i-th* individual is equal to the Fault Coverage attained by the redundant test pattern, then the fitness function is:

$$f(i) = PP - PP_i > 0 \quad (4)$$

where *PP* is the Peak Power consumption of the redundant test pattern and PP_i is the peak power consumption of the *i-th* individual.

As a consequence, individuals reducing the Fault Coverage have a high probability to be removed from the population, while the power consumption of individuals that do not affect the Fault Coverage is minimized.

4 Experimental Results

To implement the critical fault identification step we exploited the GA-based ATPG GATTO [14] for test pattern computation and IRSIM for peak power estimation. To implement the remaining steps of the algorithm described above, two tools have been written:

- *RED-GATTO* (*REDundant GATTO*) produces the redundant test pattern and for each sequence generates the list of covered faults (without fault dropping). RED-GATTO has been implemented starting from the GA-based ATPG GATTO [14]. Some changes have been introduced to implement the modified fault dropping mechanism: the redundancy factor is set to 5, i.e., a fault is dropped only if it has been detected by at least 5 sequences.
- *POPTIM* (*Power OPTIMizer*) implements the Genetic Algorithm for optimal test sequence set selection: its input is the set of sequences generated by RED-GATTO (in particular the list of faults covered by each sequence) and the power consumption measure for each sequence computed by IRSIM.

A subset of the standard set of benchmarks for sequential ATPG problems ISCAS'89 [15] has been adopted. All the experiments have been run on a Sun SparcStation 5/110 with 64 Megabytes of RAM. Table 1 shows the obtained results. The attained fault coverage (FC), the number of vector in the test pattern (Vect.) and the peak power consumption (PP) are reported for the original ATPG (GATTO) and for the power optimized ATPG.

The results demonstrate that the approach allows to effectively reduce the peak power consumption: 20% on the average, but circuits exist for which the reduction reached 50%.

From the CPU time point of view, it is important to specify that the largest part of the reported CPU time is spent by the ATPG step. The redundancy requires a lot of elaboration time and the CPU time is approximately proportional to the redundancy factor used by the ATPG.

From the test length point of view, we can observe that the proposed method has a reduced impact on the number of required test vector.

When compared to an alternative post-ATPG optimization method [9], the proposed approach performs 10% better.

5 Conclusions

Starting from the observation that for high performance VLSI systems, such as telecommunication systems, the power consumption is becoming a problem, an approach to generate test patterns for low power has been proposed. It is based on

four separated steps namely critical faults identification, test pattern generation, peak power estimation and optimal test sequence set selection. This approach is independent on the adopted ATPG and it is based on the concept of redundant test pattern generation. The ATPG to be exploited is not required to be particularly suited to low power minimization, only the fault dropping mechanism has to be modified. In our approach we used a modified version of the GA-based ATPG GATTO [14].

Experimental results show the effectiveness of the proposed method in reducing the peak power consumption of test pattern.

6 References

1. F. Corno, M. Rebaudengo, M. Sonza Reorda, "Experiences in the Use of Evolutionary Techniques for Testing Digital Circuits", *Invited Paper*, Applications and Science of Neural Networks, Fuzzy Systems, and Evolutionary Computation, B. Bosacchi, D. Fogel, J. Bezdek, Editors, *Proceedings of SPIE*, Vol. 3455, pp. 128-139, 1998
2. M. Abramovici, M. A. Breuer, A. D. Friedman: *Digital systems testing and testable design*, Computer Science Press, New York, NY (USA), 1990
3. Y. Zorian, "A Distributed BIST Control Scheme for Complex VLSI Devices", *IEEE 11th VLSI Test Symposium*, 1993, pp. 4-9
4. S. Wang, S.K. Gupta, "ATPG for Heat Dissipation Minimization during Scan Testing", *IEEE Design Automation Conference*, 1997, pp. 614-619
5. J. Silva, J. Monteiro, K.A. Sakallah, "Test Pattern Generation for Circuit Using Power Managment Techniques", *IEEE European Test Workshop*, 1997
6. J. Costa, P. Flores, H. Neto, J. Monteiro, J. P. Marques Silva, "Power Reduction in BIST by Exploiting Don't Cares in Test Patterns", *IEEE Internation Workshop on Logic Synthesis*, 1998
7. P. Girard, C. Landrault, S. Pravossoudovitch, D. Severac, "Reducing Power Consumption during Test Application by Test Vector Ordering", *IEEE International Symposium on Circuits And Systems*, 1998
8. F. Corno, P. Prinetto, M. Rebaudengo, M. Sonza Reorda, "A Test Pattern Generation methodology for low power consumption", *IEEE 16th IEEE VLSI Test Symposium*, 1998, pp. 453-457
9. F. Corno, M. Rebaudengo, M. Sonza Reorda, M. Violante, "Transformation-based Peak Power Reduction for Test Sequences", to be presented at *IEEE Alessandro Volta Memorial Workshop on Low Power Design*, 1999
10. P.C. Li, T.K. Young, "Electromigrations: The Time Bomb in Deep-Submicron ICs", *IEEE Spectrum*, Vol. 33, No. 9, 1996, pp. 75-78
11. S. Chakravarty, V. Dabholkar, "Minimizing Power Dissipation in Scan Circuits During Test Application", *IEEE Workshop on Low Power Design*, 1994, pp. 51-56
12. S. Wang, S. Gupta, "ATPG for Heat Dissipation Minimization During Test Application", IEEE Trans. on Computers, Vo. 47, No. 2, February 1998, pp. 256-262
13. M. Pedram, "Power Minimization in IC Design: Principles and Applications", *ACM Transaction on Design Automation of Electronic Systems*, Vol. 1, No. 1, 1996, pp. 3-56
14. F. Corno, P. Prinetto, M. Rebaudengo, M. Sonza Reorda, "GATTO: a Genetic Algorithm for Automatic Test Pattern Generation for Large Synchronous Sequential Circuits", *IEEE Transactions on Computer Aided Design*, Vol. 15, No. 8, August 1996, pp. 943-951
15. F. Brglez, D. Bryant, K. Kozminski, "Combinational profiles of sequential benchmark circuits," Proc. *Int. Symp. on Circuits And Systems*, 1989, pp. 1929-1934
16. A. Salz, M. Horowitz, "Irsim: An incremental mos swith-level simulator", *Design Automation Conference*, 1989, pp. 173-178

Table 1. Experimental results

	Original ATPG			Low Power ATPG			
Circuit	FC [%]	Vect. #	PP [mW]	Vect. #	PP [mW]	ΔPP [%]	CPU [s]
s208	69.3	269	1.51	157	1.38	8.6	77
s298	88.6	756	1.65	194	1.27	23.0	216
s344	98.1	122	1.93	90	1.40	27.5	126
s349	97.9	148	1.94	110	1.60	17.5	120
s382	85.2	822	2.00	646	1.62	19.0	707
s386	76.6	523	3.42	359	2.42	29.2	143
s400	87.3	936	1.92	884	1.89	1.6	940
s420	47.5	243	1.62	84	1.49	8.0	100
s526	77.8	1890	1.76	658	1.47	16.5	1383
s526n	80.8	2101	1.89	1482	1.55	18.0	1521
s641	83.2	337	2.65	279	2.02	23.8	485
s713	81.7	504	3.45	325	2.58	25.2	601
s820	39.8	219	5.05	142	2.49	50.7	134
s832	38.9	179	5.34	195	2.58	51.7	358
s838	37.2	316	2.10	219	1.67	20.5	210
s938	37.2	316	2.09	223	1.67	20.1	209
s953	98.7	1236	3.48	1058	2.64	24.1	1308
s967	89.0	547	3.32	529	2.90	12.7	973
s1269	99.6	539	18.48	371	10.45	43.4	1438
s1423	54.1	399	9.68	516	8.61	11.1	2251
s1488	76.0	354	13.98	513	10.98	21.5	1863
s1494	95.4	1492	15.13	1794	15.02	0.7	4318
s1512	54.7	393	3.43	303	3.07	10.5	810
s5378	67.8	613	19.01	730	17.35	8.7	9147
s13207	20.4	1307	22.34	584	18.11	18.9	14998
s15850	5.5	127	19.18	42	15.06	21.5	1445
s35932	76.3	407	355.37	330	326.20	8.2	59939
				Average		20.1	

A Genetic Algorithm for Designing Networks with Desirable Topological Properties

Andrew Webb[1], Brian Turton[1], and John Brown[2]

[1] Cardiff School of Engineering, Division of Electronic Engineering, Cardiff University, Queen's Buildings, PO Box 689, Newport Road, Cardiff, CF2 3TF, UK
[2] Magellan Business Networks, Northern Telecom House, Maidenhead, SL6 8XB, UK
email: turton@cf.ac.uk, webba@cf.ac.uk, john.brown.jmbrown@nortel.co.uk

Abstract. The network design problem discussed in this paper deals with optimising network parameters that characterise the following topologies: ring, chordal ring, torus and hypercube. These topologies have known, advantageous characteristics that may be useful in a final solution. By devising a system that can measure the extent to which an arbitrary mesh approaches these topologies multi-objective genetic algorithms that include topology as a dimension can be developed. Multi-objective genetic algorithms allow the designer to choose the 'ideal' design from a pareto-optimal surface. This paper describes a method by which such a measure can be obtained for a topology from a set of network parameters namely: minimum hop count, node eccentricity, node degree and the number of links. In order to prove that these measures are effective in the context of a genetic algorithm, test results are given for applying these measures as part of a fitness function for evolving the specified topologies from an 'arbitrary' mesh network. The results obtained show that the measures used are suitable for measuring the extent to which an arbitrary mesh matches a known topology, within a fitness function. As a consequence the designer can be guaranteed a range of acceptable but different choices.

1. Introduction

The network design problem discussed in this paper deals with the optimisation of parameters associated with a particular topology. These parameters we seek to optimise include network diameter (maximum eccentricity), minimum hop count between node pairs, node degree and the number of links. There are a number of beneficial reasons for doing this. Certain specific topologies, particularly toroidal networks have a number of desirable topological properties that are useful when designing reliable and efficient telecommunication networks. These desirable properties include regular, symmetrical connectivity patterns, a guaranteed nodal degree, straightforward routing, guaranteed network diameter and measures of network vulnerability such as toughness and integrity. For example, the Manhattan

R. Poli et al. (Eds.): EvoIASP'99 and EuroEcTel'99, LNCS 1596, pp. 171-181, 1999.

Street Network, a form of torus, exhibits many of these properties and has already been proposed as a possible architecture for local and metropolitan area networks. The relatively large number of potential paths between node pairs makes such networks more reliable and enables heavily congested portions of the network to be avoided. By increasing the number of alternative paths in an appropriate manner, the mean and maximum distance between nodes decreases, messages use a smaller fraction of the available bandwidth and the overall throughput increases [1-3]. Thus more efficient and economic use is made of the available resources. A direct method of encouraging 'good' characteristics would be to measure network *toughnes*s and *integrity* [4]. The toughness is a measure of how tightly the sub-graphs of a graph G are held together, while integrity is a measure of the overall network vulnerability rather than local weaknesses. However, there is no known Polynomial algorithm for finding these values, consequently these measures are impractical for use within a genetic algorithm where many thousands of topologies need to be evaluated. The alternative is to encourage the network towards predetermined advantageous topologies.

2. Problem Description

This section will first provide an overview of the various terms and notations used in the subsequent sections of this paper. A description is also given of the various parameters used in the GA optimsation.

2.1 Notation

- AE = Mean maximum node eccentricity
- SE = Standard deviation of the maximum node eccentricity
- DE = Desired mean maximum node eccentricity
- N = Number of graph nodes
- D_{ij} = Minimum hop count between nodes (i) and (j)
- AD = Mean nodal degree
- DD = Desired mean nodal degree
- SD = Standard deviation of the nodal degree
- AH = Mean minimum hop count between all node pairs
- DH = Desired minimum hop count between all node pairs
- SH = Standard deviation minimum hop count
- W_1 - W_7 = Weighting factors used in fitness function

2.2 Definitions

The degree of a node is defined as the number of links incident on to the node. The hop count is the minimum number of hops between the nodes i and j. The mean hop count, *AH*, is defined as follows:

$$AH = \sum_{i=1}^{N} \sum_{j=i+1}^{N} D_{ij} \Big/ N(N-1)/2 \tag{1}$$

Where N is the number of graph nodes and D_{ij} is the distance in hops between the node i and node j. The eccentricity, *e(v)*, of a vertex *v* is the minimum distance between *v* and the vertex furthest from *v*.

The network diameter is the maximum eccentricity, and can be defined as follows:

$$Diameter = \max\{e(v)\} \tag{2}$$

For the graph G of figure 1, the nodes x, y and z have an eccentricity of 5, 3 and 4 respectively. The overall diameter (maximum eccentricity) is 5.

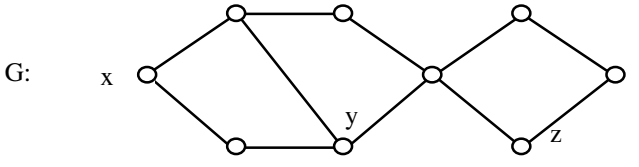


Fig. 1. Example graph

The toughness of a graph *G*, t(G), is defined as:

$$t(G) = \min\left\{ \frac{|S|}{k(G-S} \right\} \tag{3}$$

where S is the vertex cut-set of *G* and *k(G-S)* is the number of remaining graph components after the vertex cut-set has been removed.

The integrity of a graph *G*, *i(G)* is defined as:

$$i(G) = \min\{|S| + N(G-S)\} \tag{4}$$

where *N(G-S)* is the maximum order of a component of *G-S*, i.e. the number of nodes in the largest remaining component.

2.3 Problem Definition

The network design problem that we are interested in is that of generating graphs that resemble the given topology, i.e. graphs whose actual parameters values match the known ideal parameter values as closely as possible. One way of encouraging this in non-regular topologies is to use a fitness function that is a measure of how closely a graph resembles the chosen topology. Thus the fitness value is proportional to how closely the actual parameter values match the ideal parameter values. The metrics used in the fitness function were the mean and standard deviation of each of the previously mentioned parameters; an additional metric was used for the number of network links. The ideal values for each topology are shown below in table 1:

Table 1. Optimal topology parameter values

Nw Size	Topology	Mean Diam	Mean Node Deg	Mean Hop Count	No. Links
16 Nodes	Ring	8	2	4.266	16
	Ch Ring	5	3	2.666	24
	Torus	4	4	2.133	32
	Hcube	4	4	2.133	32
32 Nodes	Ring	16	2	8.258	32
	Ch Ring	5	3	3.161	48
	Torus	6	4	3.096	64
	Hcube	5	5	2.580	80
64 Nodes	Ring	32	2	16.29	64
	Ch Ring	8	3	4.698	96
	Torus	8	4	4.063	128
	Hcube	6	6	3.047	192

The fitness value used by the GA is simply a measure of how closely an individual graph resembles the topology of interested. The fitness function is as follows:

$$Fitness = W_1\left(1-\left|\frac{AH}{DH}-1\right|\right)+W_2(2-SH)+W_3\left(1-\left|\frac{AD}{DD}-1\right|\right)+W_4(2-SD)+$$
$$W_5\left(1-\left|\frac{AE}{DE}-1\right|\right)+W_6(2-SE)+W_7\left(1-\left|\frac{L}{DL}-1\right|\right) \quad (5)$$

The weighting factors W1, W3, W5 and W7 are set at 100, while W2, W4 and W6 are set at 20.

3. GA Implementation

3.1 Encoding Chromosome Solutions

The chromosomes comprise of a sequence of binary numbers to represent connections between node pairs [5-6]. The number of genes, L, is equal to the number of potential links that connect all possible node pairs and is given by:

$$L = N(N-1)/2 \qquad \textbf{(6)}$$

where N is the total number of graph nodes. The chromosome structure, C_i, is represented below,

$$C_i = \{C_0, C_1, C_2, ..., C_L\}$$

,where C_i is a binary number that represents the presence or absence of a link between a node pair. New offspring are allocated using a roulette wheel method, applied to linearised fitness values. After the new generation has been allocated, standard two-point crossover is applied to chromosome pairs at a probability p_c to produce new child chromosomes. In order to maintain genetically diverse populations, a number of mutation operators are applied with a very low probability p_m and are described pictorially in figures 2 and 3. The single link swap shown in figure 2 deletes an arbitrarily chosen link and adds another link at another arbitrarily chosen location that has no link. The double link swap shown in figure 3 deletes two arbitrarily chosen links and replaces them at two arbitrarily chosen non-link sites.

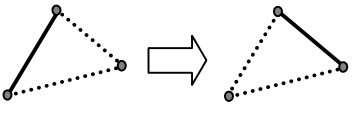


Fig. 2. Single link swap

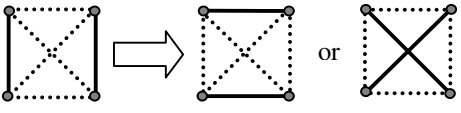


Fig. 3. Double link swap

3.2 Repair Heuristics

The GA can sometimes produce 'bad' results that need to be repaired to make them feasible. Unfeasible networks include disconnected networks, networks that violate the connectivity requirements and networks that cannot survive single link or node failures. Such networks are repaired in an arbitrary fashion, such that useful areas in the search space are not excluded. The repair mechanism uses a graph searching algorithm to detect if a graph is disconnected. Unfeasible graphs are 'repaired' by adding a suitable number of arbitrarily chosen links until the network is feasible. For networks to satisfy the connectivity requirements, the node degree, k, must fall within the specified limits, e.g. (1<k<6). Graphs that violate the connectivity condition are repaired in the manner described in figure 4.

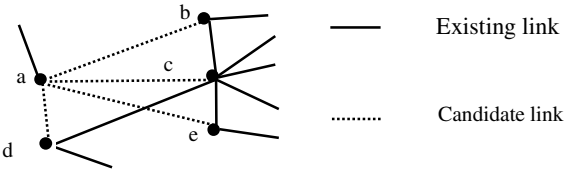


Fig. 4. Satisfying connectivity requirements

The graph shown in figure 4 shows how node (a) violates the minimum node connectivity constraint of degree 2. Adding a candidate link (shown as a dotted line) arbitrarily can rectify this degree constraint violation. Figure 4 also shows how node (c) exceeds the maximum connectivity constraint for this example. To rectify this, one of the existing links connected to node (c) is chosen arbitrarily and removed.

Occasionally, some networks become disconnected in such a way as to produce two or more distinct networks, even though all network nodes may satisfy the connectivity requirements. This can sometimes happen when generating initial random populations or as a result of applying genetic operators.

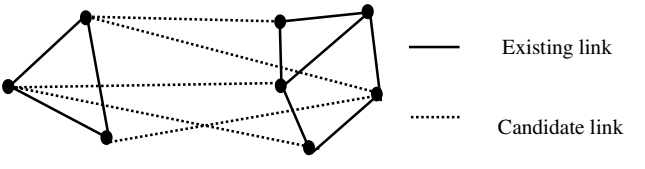


Fig. 5. Ensuring graph is fully connected

The dotted lines shown in figure 5 illustrate some of the many potential candidate links that may be added to the graph. To reconnect the graph, candidate links are added arbitrarily until the graph is connected. Another feasibility constraint could be to ensure that the loss of any single link or node should not bisect the network. The link and node that are indicated by the arrows in figure 6 illustrate how their removal could disconnect the graph. The dotted lines show some of the potential links that could be added to make these networks immune to a single link or node failure.

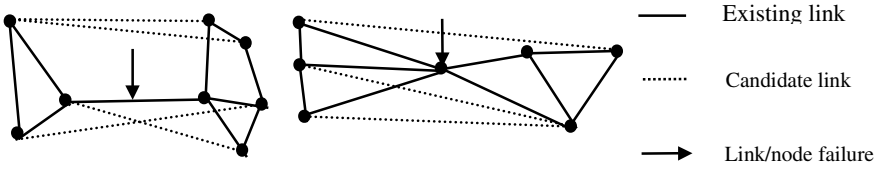


Fig. 6. Surviving single link/node failures

3.3 Fitness Evaluation

The chromosome fitness value is a measure of how much a graph resembles the topology of interest, i.e. its "toroidalness", "ringness", "hypercubeness", etc. The higher the fitness value, the more closely the network resembles the chosen topology. To compute the graph fitness values, we apply Dijkstra's shortest path algorithm to each graph to determine the minimum hop count between all node pairs. These hop counts are used to determine the following parameters that are used in part of the fitness evaluation: mean and standard deviation of the maximum node eccentricity and mean and standard deviation of the minimum hop count between all node pairs. The mean/standard deviation node degree and the total number of links are obtained from the graph. A summary of the algorithm is shown in the following pseudocode:

```
Evolutionary Algorithm
{
    Generate initial population of graphs;
    WHILE (Termination criteria NOT reached)
    {
            Apply graph repair mechanisms:
                    (i)  Satisfy connectivity constraints;
                    (ii) Ensure graph is biconnected;
            Evaluate chromosomes:
                    (i)  Apply Dijkstra's Algorithm;
                    (ii) Calculate fitness;
            Select new population: Recombination and Mutation;
    }
}
```

4. Computational Results

The graphs in figure 7 show the normalised mean and best fitness values obtained over 500 iterations of the GA for 16, 32 and 64 node topologies. In table 2 a summary is given showing the best overall parameter values obtained by the GA. The first column in table 2 represents the size of each network, while the second column represents the network topology. The following three columns give the mean and standard deviation of the parameters and also the percentage differences by which these values differ from the optimal. The final column shows the fitness values and their percentage difference from the optimal values.

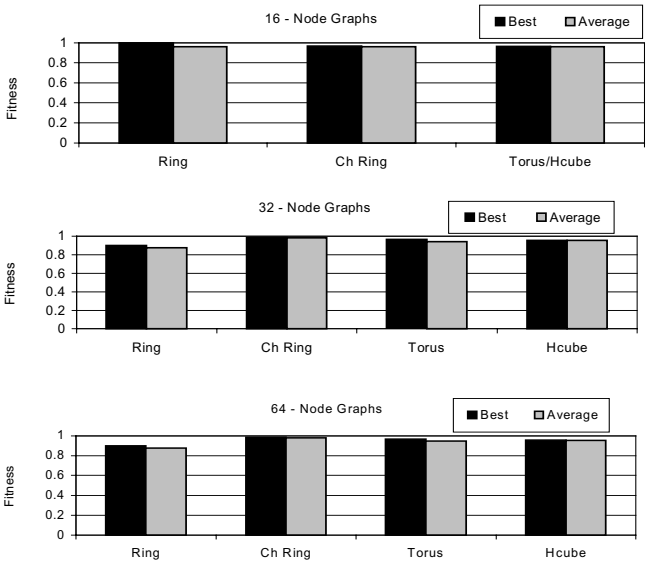


Fig. 7. Mean and best fitness values obtained by the GA.

Table 2. Best parameter values obtained by the GA

N	To	Eccentricity				Mean Node Degree				Hop Count				Fitness	
		Mean	% diff	Std Dev	% diff	Mean	% diff	Std Dev	% diff	Mean	% diff	Std Dev	% diff		% diff
16	R	4.26	0.00	0.00	0.00	2.00	0.00	0	0	8.00	0.00	0.00	0.00	520	0
	C	2.65	0.30	0.24	12.3	3.00	0.00	0.51	25.8	4.68	6.24	0.47	23.9	488	6.02
	T	2.11	0.75	0.17	8.55	4.00	0.00	0.51	25.8	3.56	10.9	0.51	25.6	484	6.86
	H	2.11	0.75	0.17	8.55	4.00	0.00	0.51	25.8	3.56	10.9	0.51	25.6	484	6.86
32	R	6.67	19.1	0.43	21.8	2.25	12.5	0.44	22	12.4	22.4	0.49	24.9	438	15.7
	C	3.15	0.06	0.21	10.5	3.00	0.00	0.50	25.4	5.00	0.00	0.00	0.00	505	2.77
	T	3.02	2.39	0.13	6.7	4.00	0.00	0.67	33.6	5.00	16.6	0.00	0.00	484	6.76
	H	2.40	6.78	0.10	5.1	5.00	0.00	0.56	28.4	4.00	20.0	0.00	0.00	479	7.72
64	R	9.64	40.8	0.43	21.7	2.21	10.9	0.41	20.8	18.1	43.2	0.88	44.1	390	24.9
	C	4.06	13.5	0.18	9.3	3.34	11.4	0.54	27.0	6.85	14.2	0.35	17.5	459	11.6
	T	2.87	29.1	0.08	4.4	4.46	11.7	0.59	29.5	4.00	50.0	0.00	0.00	415	20.0
	H	2.49	17.9	0.05	2.85	6.21	3.65	0.41	20.8	4.00	33.3	0.00	0.00	455	12.3

N = Number of nodes To = Topology
R = Ring C = Chordal Ring
T = Torus H = Hypercube

4.1 Example Results

The graphs shown in figure 8 show two examples of the best topology obtained by the GA after 1000 iterations. Table 3 compares the ideal parameter values with actual parameter values.

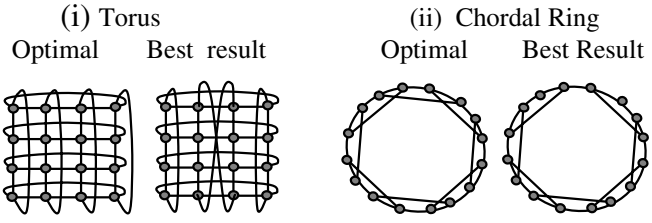


Fig. 8. Comparison of best topology found with ideal result

Table 3. Ideal and actual parameter values

	(i) Torus			(ii) Chordal Ring		
Parameter Value	Ideal	Actual	% diff	Ideal	Actual	% diff
Fitness:	520	487	6.20	520	498	4.23
Mean Hop Count:	2.13	2.00	6.25	2.66	2.76	6.25
Std Dev Hop Ct:	0.00	0.04	0.02	0.00	0.15	7.50
Mean Node Deg:	4.00	4.00	0.00	3.00	2.87	4.33
Std Dev Node Deg:	0.00	0.00	0.00	0.00	0.34	17.0
Mean Max Ecc:	4.00	3.00	25.0	5.00	5.00	0.00
Std Devn Max Ecc:	0.00	0.00	0.00	0.00	0.00	0.00
Number of Links:	32	32	0.00	24	23	4.16

4.2 Evaluating Toughness and Integrity

The toughness and integrity values were determined for a number of randomly selected chromosomes with different fitness values. Evaluation of toughness and integrity were achieved by determining the number of disconnected graph components, k(G-S), and the maximum order of the remaining components, N(G-S), for all possible vertex cut-sets. The target toughness and integrity values were 1 and 9 respectively. The graphs shown in figures 9 and 10 plot the fitness value against the toughness and integrity values for the 16 node torus and chordal ring target topologies.

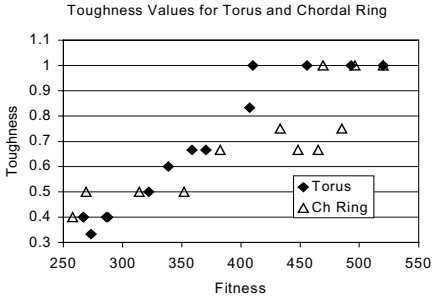


Fig. 9. Toughness vs Fitness

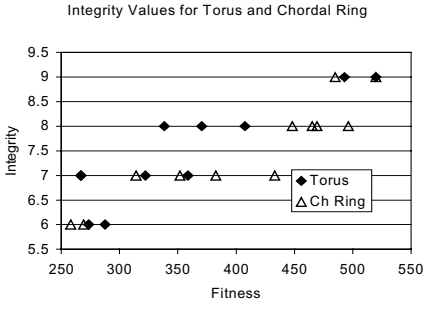


Fig. 10. Integrity vs Fitness

5. Discussion of Results and Conclusions

The quality of results obtained by the GA is established by comparing its results with the known optimal solutions. In all examples the results of the GA appears to converge after a few hundred generations. The graphs given in figures 9 and 10 show that as the GA approaches the ideal topology, in doing so it also reaches the target toughness and integrity values.

The results for the 64-node graphs have the slowest rate of convergence, due to the very large search space that this size of problem presents us with. There was one instance where the GA managed to generate an optimal result within 500 generations. Beyond this point the Genetic Algorithm seems to search randomly for better solutions and as a consequence further experiments indicate that at least 10,000 generations are required before an optimal result is reached. Ring topologies are unusual in that a connectivity of two is both necessary and sufficient to ensure a connected graph is a ring. Other topologies are more complex in that they require additional criteria. In particular this is true of the remaining topologies studied: chordal ring, torus and hypercube. Other improvements could be made to make the GA perform more efficiently, particularly when the GA has ceased to produce

improved results. These include more sophisticated mutation/link swapping techniques and also hill climbing techniques.

In order to design practical networks more suitable for real-world design problems, it is possible to use this idea as part of a multiobjective function so that the network designer could choose networks that have been simultaneously optimised over a variety of different parameters. This work shows how arbitrary mesh networks can be guided towards known topologies with good characteristics without limiting the network to precisely match a particular topology. In addition approaching a known topology has been shown to encourage toughness and integrity values that match those of the associated target topology.

References

1. Maxemchuck, N. F., "Routing in the Manhattan Street Network", IEEE Transactions on Communications, Vol. COM-35, No. 5, May 1987.
2. Maxemchuck, N. F., "Regular Mesh Topologies in Local and Metropolitan Area Networks", AT&T Technical Journal, Vol. 64, No. 7, September 1985.
3. Robertazzi, T. G., "Toroidal Networks", IEEE Communications Magazine, June 1983, Vol. 26, No. 6.
4. Chartrand, G., Lesniak, L., Graphs and Digraphs, Chapman & Hall, 1996, 3rd Edition.
5. Sinclair, M.C., "Minimum Cost Topology Optimisation of the COST 239 European Optical Network", Proceedings of the Second International Conference on Artificial Neural Networks and Genetic Algorithms, Alés, France, pp26-29, April 1995.
6. Sinclair, M.C., "NOMaD: Applying a Genetic Algorithm/Heuristic Hybrid Approach to Optical Network Topology Design", Proceedings of the IEE Colloquium on Multiwavelength Optical Networks: Devices, Systems and Network Implementations, London, June 1998.
7. Goldberg, D. E., Genetic Algorithms in Search, Optimization and Machine Learning. Addison Wesley, Reading, MA (1989).
8. Cahn, R. S., Wide Area Network Design: Concepts and Tools for Optimization, Morgan Kaufman Publishers, 1998.
9. Davis, L., Orvosh, D., Cox, A. and Qiu, Y., "A Genetic Algorithm for Survivable Network Design", Proceedings of the Fifth International Conference on Genetic Algorithms, pp405-415, 1993.
10. Davis, L., Coombs, S., "Genetic Algorithms and Communication Link Speed Design: Theoretical Considerations", Proceedings of the Second International Conference on Genetic Algorithms, pp. 252-256, 1987.
11. Kershenbaum, A, "Telecommunications Network Design Algorithms", McGraw-Hill. 1993.
12. Ko, K.T., Tang, K.S., Chan, C.Y., Man, K.F., "Packet switched Communication Network Designs using GA", Genetic Algorithms in Engineering Systems: Innovations and Applications, Conference Pub. No. 446, 2-4 September 1997.
13. Standish, T.A., "Data Structures, Algorithms and Software Principles in C", Addison-Wesley, 1995.
14. Tanenbaum, A.S., "Computer Networks", 3rd edition, Prentice/Hall International, Inc., 1996.

Approximate Equivalence Verification for Protocol Interface Implementation via Genetic Algorithms

Fulvio Corno, Matteo Sonza Reorda, Giovanni Squillero

Politecnico di Torino
Dipartimento di Automatica e Informatica
Corso Duca degli Abruzzi 24 I-10129, Torino, Italy
{corno, sonza, squillero}@polito.it

Abstract. This paper describes a new approximate approach for checking the correctness of the implementation of a protocol interface, comparing its low-level implementation with its high-level prototype. The possibility to validate protocol interfaces is extremely useful in many industrial design flows and the proposed methodology does not impose particular requirements and it is able to fit in existing design flows: the proposed approach is based on coupling a commercial simulator with a genetic algorithm that tries to disprove the equivalence of an implementation with its high-level prototype. The use of a commercial simulator guarantees a complete compatibility with current standards and the method is able to fit painlessly in an existing industrial flow. Moreover, the use of a genetic algorithm allows the analysis of large and realistic designs. Experimental results show that the proposed method is effectively able to deal with realistic designs, discovering potential problems, and, although approximate in nature, it is able to provide a high degree of confidence in the results.

1 Introduction

The development of protocol interfaces is a challenging task due to the increasing complexity of both digital circuits and protocol specifications. Starting from the protocol requirements (usually provided as a set of natural-language statements), common design flows include several steps (sketched in Figure 1) before producing the final, low-level hardware implementation of a protocol interface.

In the first steps, it is usual to prepare an abstract prototypical description of the interface in some high-level hardware description language; this description should then be validated to check if it is coherent with protocol requirements. This description neglects many details and optimizations, but it fully reflects the functionality of the interface. The validation process is commonly carried out by including some checks in the design and simulating the interface with functional test patterns. Alternatively, validation can be performed using model checkers, or other programs, although, in this case, it is necessary to suitably express protocol characteristic and designers tend to be unwilling to learn new formalism, such as temporal logics.

R. Poli et al. (Eds.): EvoIASP'99 and EuroEcTel'99, LNCS 1596, pp. 182–192, 1999.

Unfortunately, these descriptions are often not directly synthesizable, in the sense that it is not possible to derive directly and in an automated way the low-level description of the interface implicitly from the high-level one. Designers have to develop it with some manual transformation and checking the equivalence of the two descriptions is a critical issue that requires an accurate analysis in order to test whether all the characteristics of the high-level specification are correctly implemented.

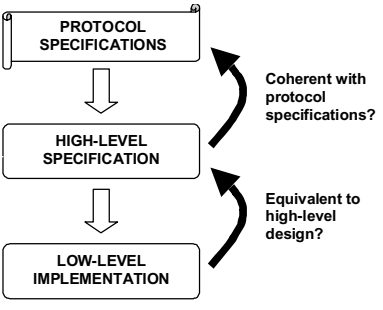


Figure1: *CommonProtocolInterfaceDesignFlow*

Consequently, two different types of checks are involved in the development of a protocol interface: validating the correctness of the high-level interface with respect to protocol requirements, and verifying the equivalence of the high-level specification of the interface with the low-level one. This paper addresses the second problem, proposing a methodology for checking the equivalence that is applicable to a wide set of descriptions and able to fit almost painlessly in an existing design flow.

An important characteristic of the high-level prototypical specifications is that, while *non-synthesizable*, they are usually *executable* (designers typically test them by simulation). The proposed method uses the possibility of executing these descriptions to heuristically check their equivalence with the low-level implementations. Thus, the method does not require to write any custom model in ad-hoc languages.

At the present time, exact equivalence verification of a large high-level system with respect to its low-level implementation is beyond the state of the art. Therefore, we propose a partial solution to this problem, which sacrifices *exactness* in favor of applicability. Indeed, all verification systems shall be considered not exact when run on a computer system. In practice, there is always the case that the tool runs out of memory or available CPU time before delivering a result. In our method, we simply give up exactness *explicitly*: we develop an approximate equivalence verification algorithm that is not always able to produce an answer, but is able to deal with sequential equivalence between high-level specifications and their low-level implementations.

The proposed approach belongs to a brand new framework that can be called *approximate equivalence verification*. Given the two circuits, approximate equivalence verification techniques employ simulation driven by heuristic algorithms to seek a *counterexample* of the equivalence to be proven, i.e., an input pattern able to produce an output behavior in one system different from the one produced by the other. Thus, they can only provide *negative* responses (using these techniques, no equivalence can ever be proved), and there is no mathematical certainty that the

counterexample, if it exists, will eventually be found by the algorithms. Nevertheless, experimental results show that in many practical cases the confidence obtainable with such methods on the whole set of real circuits can by higher than the one obtainable with more traditional, formal approaches.

In recent times, approximate equivalence verification techniques are achieving increasing interest in the research community. Two techniques able to deal with real-sized circuits described at low-level have recently been proposed [4] [5]. Both approaches are based on the same low-level simulator, but they implement completely different evolutionary algorithms to seek a counterexample. In those works, approximation techniques were shown to successfully complement the results obtained by exact techniques, thus improving the overall quality of the verification process. The problem of equivalence verification of one system described at the RT-level and one described at the gate-level was engaged in [7], while the problem of equivalence verification of two systems described at the RT-level was tackled in [6].

The paper describes VEGA.PI (*Verification of Equivalence through Genetic Algorithm of a Protocol Interface*), a new approach for checking the equivalence of a generic high-level prototype and its low-level implementation. VEGA.PI exploits the analysis of internal activity of the low-level description, and it can take advantage from designers' knowledge for identifying known correspondences in the two descriptions.

Experimental results show that the proposed approach is effectively able to verify large and realistic circuits, discovering problems in the design process. Furthermore, another set of experiments, performed on smaller circuits and compared with an equivalence verification algorithm based on exact techniques, showed that VEGA.PI is able to provide results with a high degree of confidence despite its approximate nature.

The paper is organized as follows: Section 2 describes the algorithm, with particular emphasis to the heuristic; Section 3 contains the experimental evaluation of a prototype; Section 4 concludes the paper.

2 The VEGA.PI Algorithm

For the purpose of this paper, we chose to describe protocol interfaces as VHDL processes. VHDL is widely adopted in industrial design flows, and it is ideal to describe protocol interface specifications, since it is well known by designers and it allows to easily include checks in the form of *assert* statements directly into the description.

Given two descriptions, a *distinguishing sequence* is an input pattern able to produce an output behavior in one system different from the one produced by the other. In this paper, we assume that both circuits start from the reset state. If such a sequence exists, an equivalence verification tool should be able to provide it as a *counterexample*; otherwise, it should provide the *proof* of its non-existence. The approximated approach presented in this paper will never be able to provide non-existence (i.e., circuit equivalence) proofs, but is meant to be effective in finding the counterexample, when existing. Experimental results will show that, even with this limitation, the degree of confidence that the approximated result can provide is quite high.

An important fact to emphasize is that both descriptions share the concept of temporization and vectors composing sequences are applied simultaneously. Thus, we are assuming that both systems use the same clock.

Given two descriptions that are to be proven equivalent, the information needed by VEGA.PI is the list of corresponding primary inputs and primary outputs (that must coincide in the two descriptions). The designer has also the opportunity to specify a set of *checkpoints*, i.e., mappings between VHDL signals or variables and netlist lines that should correspond in the two implementations. Usually, checkpoints are limited to a subset of the memory elements, and it is straightforward to identify them, for instance knowing the convention adopted by the synthesis tool for assigning gate names from RT-level signal names.

Checkpoints are only used as *hints* to the algorithm, and their functional equivalence is not assumed, thereby allowing the user to take benefit even from partial or hypothetical correspondences. The higher number of checkpoints the designer is able to provide, the more effective the heuristic algorithm is in evaluating sequences. In the worst case, if no checkpoints are provided, a pure black box verification is performed.

During verification, the RT-level description is considered as an almost black box, where primary inputs are controllable and primary outputs and checkpoints are observable. On the contrary, the internal behavior of the gate-level description is completely accessible by the tool. The genetic algorithm aims at deriving a sequence that causes a difference on the primary outputs, and it guides the search by analyzing the differences that may appear on the checkpoints and by monitoring the internal activity of the gate-level circuit.

VEGA.PI adopts a heuristic search algorithm for constructing the distinguishing sequence (as other approximate equivalence verification algorithms); therefore, the more it runs without finding a counterexample, the likelier it does not exist. In this way designers are able to easily trade off CPU time with confidence on the result.

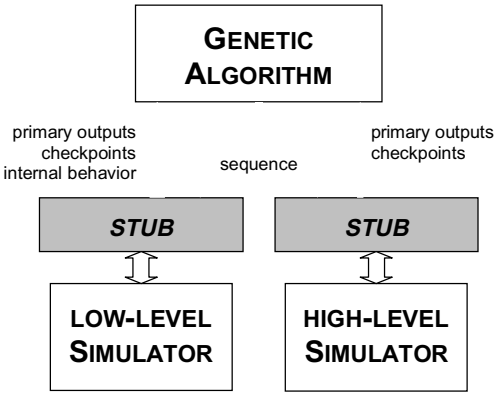


Figure 2: *VEGA.PI Architecture*

The architecture of VEGA.PI is shown in Figure 2. VEGA.PI constructs counterexamples through a *genetic algorithm* (detailed in the next Section) which exploits information coming from the simulation of sequences. The genetic algorithm uses two different stubs to communicate with different simulators, one for gate-level

descriptions and one for RT-level descriptions. These two stubs have different characteristics because the required interaction is different: the gate-level simulator must be highly customizable, since the internal state of the circuit is heavily observed during the simulation. On the contrary, the main requirement for the RT-level simulator is to be fully compatible with the VHDL standard. For this reason, we propose the adoption of commercial tools that, while sacrificing some ability to interact with the simulation, guarantee more applicability.

3 The Genetic Algorithm

Evolutionary Algorithms aim at applying techniques derived from biological systems, in particular Natural Selection, to search and optimization problems. VEGA.PI, in particular, uses a specific type of evolutionary algorithm, called *Genetic Algorithm* (GA). GAs were first introduced by Holland in [10] and they are today well known to be suited for finding nearly optimal solutions of very large problems [8]. Moreover, the use of a GA for generating input sequences has already been widely exploited in the literature (e.g., for Automatic Test Pattern Generation [[2] [12] [11]).

The goal of the GA in VEGA.PI is to discover a distinguishing sequence, i.e., an input pattern able to produce an output behavior that is different in the two systems being compared. The GA evolves a *population* of *sequences*, i.e., binary vectors to be applied to the circuit primary inputs in consecutive clock cycles starting from the initial state.

Each sequence is characterized by its *fitness*, i.e., its closeness to the goal. The goal is to excite some internal differences and to propagate them to a primary output of the circuit. In the algorithm, the goal can be rephrased as follows: excite all possible behaviors in the circuit while trying to retain any difference found in some checkpoints. The following fitness function $F(s)$ for a sequence s is thus used in VEGA.PI:

$$F(s) = \sum_{i=1}^{\text{len}(s)} \left(\alpha \cdot C_i + \beta \cdot C_i^* + \gamma \cdot A_i - \delta \cdot U_i \right)$$

where the sum extends over all the *len(s)* vectors of the sequence *s*. After simulating the *i*-th vector in sequence *s*:

- C_i is the number of checkpoints where the two circuits assume a different value for the first time
- C^*_i is the number of checkpoints already counted in C_{i-1} or in C^*_{i-1} whose difference is still present after the current vector
- A_i is the gate-level circuit activity, i.e., the number of gates and flip-flops which have assumed a binary value never assumed before
- U_i is equal to 1 if the vector is useless, i.e., if all C_i, C^*_i, and A_i are equal to 0. Otherwise it is equal to 0.

The coefficients $\alpha > \beta > \gamma > \delta$ set the relative importance of the sub-goals, in decreasing order: forcing new differences on checkpoints; retaining differences on checkpoints; letting the circuit explore new configurations; avoiding useless vectors.

Since all C_i, C^*_i, and A_i count an event only the first time it occurs, the coefficient δ has the effect to penalize long sequences: vectors that are not able to force new

differences on checkpoints neither to explore new configurations receives a negative value.

During evolution, sequences mate and mutate to generate new sequences in the population and best sequences are selected for survival on the basis of their fitness function. Using this mechanism, sequences in the population tend to become fitter and fitter as generations pass.

The mating of sequences is performed through *crossover* operators, that select two parents and generate a new sequence by taking random parts of each parent; four crossover operators are defined in VEGA.PI and they are chosen with equal probability:

- **Horizontal 1-cut crossover:** the new sequence is composed of some vectors coming from either parent, according to the position of one cut point randomly generated in the first individual (x_1), and another one randomly generated in the second (x_2).
- **Horizontal 2-cut crossover:** the new sequence is composed of some vectors coming from either parent, according to the position of two cut points randomly generated in the first individual (x_1 and x_2). The length of the new sequence is the longest between the two parent ones.
- **Horizontal uniform crossover:** each vector in the new sequence is taken randomly from the first or from the second parent. The length of the new sequence is the longest between the two parent ones.
- **Vertical uniform crossover:** each vector in the new sequence inherits some bit columns from the first parent and some from the second. The length of the new sequence is the longest between the two parent ones: inputs taken from the shortest parent are completed with random values where needed.

Sequences can also undergo *mutation*, where some bits are randomly modified, inserted, or deleted. Three mutation operators are implemented in VEGA.PI and are selected with equal probability:

- **Change mutation:** a vector in the sequence is replaced with a new randomly generated one.
- **Add mutation:** a random vector is added in a random position, shifting forward the subsequent vectors.
- **Delete mutation**: a randomly selected vector is removed from the sequence, shifting backward the subsequent vectors.

Individuals are selected for applying genetic operators using the roulette wheel technique on their linearized fitness: sequences with higher fitness are likelier to be selected for crossover or mutation, so that good sequences are given more chances to generate better ones. Evolution continues until a distinguishing sequence is found, until a maximum predefined number of generations has been stepped through, or until the system has reached stability (i.e., no fitness improvements are recorded for a given number of generations).

3.1 Implementation

A prototypical implementation of VEGA.PI has been developed using the ANSI-C language. VEGA.PI includes the genetic algorithm and the two separate stubs and amounts to about 2,000 lines of code.

For simulating the gate-level description, we adopted an in-house developed, 2-valued, event-driven, gate-level simulator. The stub allows examining the internal state of each gate during simulation.

The adopted parameters for the fitness function are shown in Table 1.

Table 1: *Parameter Values*

Parameter	Value	Meaning
α	# of gates	Weight for exciting a difference on a checkpoint
β	$\frac{\alpha}{10}$	Weight for keeping a difference on a checkpoint
γ	1	Weight for exploring new configurations
δ	1	Penalty for useless vectors

The GA simulates a population of 30 individuals, creating 15 new individuals at each generation (thus preserving the best half of the population). Mutation or crossover operators are chosen with equal probability (50%). The maximum number of possible generations was set to 300, and the system is assumed to have reached stability after 30 generations without any fitness improvements.

To avoid any limitations in the syntax of the descriptions, for simulating RT-level descriptions we choose a commercial VHDL simulator: *V-System 5.1* developed by Model Technology. For the sake of efficiency, VEGA.PI runs the simulator as a concurrent process and communicates with it via unix pipes or sockets. The developed stub can be accommodated to interact with different simulators.

4 Experimental Evaluation

To evaluate VEGA.PI, two different sets of experiments were performed. The first set of experiments aims at evaluating the confidence of the results provided by VEGA.PI. The second shows the efficacy of VEGA.PI for validating the output of a synthesis tool in presence of designer errors.

All experiments were performed on Sun SPARC-Stations 5 with 64 Mbytes of memory, running SunOS 4.1.4.

4.1 Confidence Evaluation

VEGA.PI was used to disprove equivalence of high-level specifications with their gate-level implementations in presence of errors. The adopted error model aims at mimicking small and subtle mistakes: first an high-level description is synthesized; then a random fanout-free region (FFR) is selected from the gate-level description, and a randomly chosen single bit in its truth table is flipped; finally, the updated FFR is synthesized back in the description. Since no checks are performed to determine if the input configuration for the flipped bit is sequentially reachable or to determine if the flipped bit is sequentially observable, descriptions are often sequentially equivalent even after the modification.

To evaluate the confidence of the result, VEGA.PI is compared with AQUILA [9], a state of the art equivalence verification tool. This tool is not able to handle high-level descriptions, thus, it was used to verify the equivalence between the original gate-level implementation and the modified one. The complete experimental flow is sketched in Figure 3.

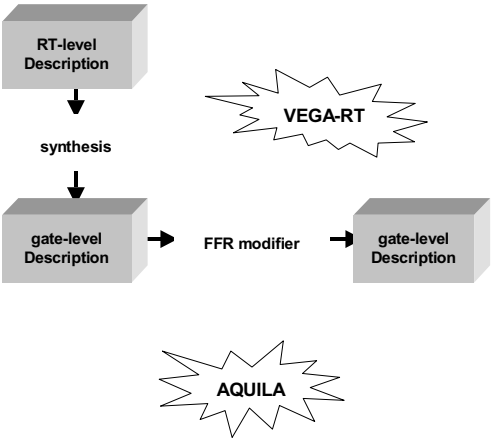


Figure 3: *Experiment Flow*

Some benchmark descriptions have been selected among publicly available VHDL synthesizable descriptions [1]. Their characteristics are summarized in Table 2. "#VHDL lines" and "#GATES" report the size of the specification and of the implementation, respectively. Last three columns better detail benchmarks, in term of number of memory elements ("FF"), primary inputs ("PI") and outputs ("PO").

Table 2: *FFR Benchmark Characteristics*

circuit	#VHDL lines	#GATES	#FF	#PI	#PO
b06	128	66	9	4	6
b08	89	168	21	11	4
b13	296	309	53	12	10

Table 3 reports experiment results. In the first column ("Circuit") the name of the circuit is showed. The first column group describes the characteristics of the FFR selected for injecting the fault: the output gate name ("stem"), the number of gates belonging to the FFR ("size") and the number of its inputs ("#in"). The next column group contains the different answers given by the two different tools to the question "is the modified circuit equivalent to the original one?". VEGA.PI is an approximate algorithm, thus the answer can be "NO" (a counterexample was found) or "probably" (since no distinguishing sequence has been found, the circuits should be assumed equivalent). AQUILA is an exact tool and it is able to provide both answers: "YES" (proven equivalent) and "NO" (proven different). Unfortunately, for the largest circuit it fails due to memory explosion without providing any answer. Finally, column "CPU" reports the seconds of CPU time consumed by VEGA.PI.

Table 3: *Experimental Results*

Circuit	FFR			EQUIVALENT?		CPU [s]
	stem	size	#in	VEGA-RT	AQUILA	
b06	U206	9	6	NO	NO	10.0
b06	U206	9	6	NO	NO	3.0
b06	U208	6	4	NO	NO	6.0
b06	U208	6	4	probably	YES	1,014.0
b08	U331	17	11	probably	YES	625.1
b08	U331	17	11	NO	NO	122.2
b08	U332	17	10	NO	NO	95.6
b08	U332	17	10	probably	YES	1,102.0
b08	U334	26	16	probably	YES	1,595.2
b08	U334	26	16	probably	YES	1,366.6
b13	GT_255_U5	11	7	NO	*unknown*	96.7
b13	GT_255_U5	11	7	NO	*unknown*	546.8
b13	U695	6	4	NO	*unknown*	430.8
b13	U695	6	4	NO	*unknown*	637.6
b13	U697	7	4	NO	*unknown*	9.0
b13	U697	7	4	NO	*unknown*	489.0
b13	U710	8	4	NO	*unknown*	420.0
b13	U710	8	4	NO	*unknown*	5.0
b13	U755	5	3	probably	*unknown*	1,526.0
b13	U755	5	3	probably	*unknown*	1,210.0
b13	U826	7	4	probably	*unknown*	830.9
b13	U826	7	4	probably	*unknown*	769.7

Experimental results show that results provided by VEGA.PI have a high degree of confidence. When AQUILA demonstrates the non-equivalency, VEGA.PI is always able to find a valid distinguishing sequence: in no case the exact tool contradicts the hypothesis made by VEGA.PI.

It should be noted that the main goal of VEGA.PI is to deal with RT-level designs without imposing severe syntax limitations. Thus, the proposed approach sacrifices some efficiency in the interaction with the simulators to guarantee more applicability. It can be noted that, when verifying small descriptions, the overhead caused by the two stubs is significantly big. However, on larger descriptions almost all CPU time is used to run simulations, and simulation time, usually, does not increase exponentially with circuit complexity.

4.2 Verification of Synthesis Results

To evaluate VEGA.PI on realistic problems, we tested it on two high-level descriptions of simplified microprocessors realized by students, where the critical part is the correctness of the bus interface. Students were not experienced designers, and the resulting VHDL specification contains ambiguous statements that may cause a misinterpretation. As a result, the gate-level implementation does not correctly implement protocol specifications.

Table 4: *Microprocessors Benchmark Characteristics*

circuit	#VHDL lines	#GATES	#FF	#PI	#PO
p-viper	518	3,461	247	34	54
p-80386	648	6,931	447	37	70

Table 4 summarizes the characteristics of the benchmarks in terms of number of VHDL lines and the number of processes. For the gate-level descriptions Table 4 reports the number of gates, flip-flops, primary inputs and primary outputs. For both circuits, VEGA.PI disproves the equivalency of the RT-level with the synthesized gate-level implementation, thus exposing a problem in the synthesis step.

In each case, by analysis of the provided counterexample, the students were able to pinpoint the location of the problem in the VHDL code. For the p-viper circuit, an assignment of a 32-bit signal (declared as a VHDL `integer`) to a 20-bit variable (declared as `integer range 2**20-1 downto 0`) caused an inconsistency in the upper bits of the variable, since in the netlist they were not present. In the p-80386 benchmark, the pointer registers in the FIFO instruction queue were compared as signed values in the VHDL source (since they were declared as `integer range`) and as unsigned values in the netlist (since the synthesizer inferred that they always contained positive values, i.e., addresses).

The errors were corrected by inserting a truncation operator in p-viper to drop the upper bits in the VHDL, too, and by declaring the address pointers as unsigned in p-80386.

5 Conclusions

This paper presented a new approximate approach for verifying the equivalence of an RT-level design with its gate-level implementation. This task is particularly important when designers deal with circuits (or sub-circuits) implementing protocols; in this case, equivalence between the implementation and its high-level specification has to be carefully verified. This verification would significantly benefit several design steps, if the methodology do not impose particular requirements. The proposed approach is based on coupling a heuristic algorithm with a commercial VHDL simulator and an in-house developed gate-level simulator, thus the proposed method guarantees a complete compatibility with VHDL standards and it is able to fit painlessly in an existing industrial flow.

Experimental results show that the proposed approach is effectively able to verify large and realistic RT-level designs, discovering problems in the synthesis process. Furthermore, another set of experiments, performed on smaller descriptions and compared with an exact equivalence verification algorithm, showed that VEGA.PI is able to provide results with a high degree of confidence despite its approximate nature.

6 References

1. Circuits downloadable at `http://www.cad.polito.it/tools/`
2. F. Corno, P. Prinetto, M. Rebaudengo, M. Sonza Reorda , "GATTO: a Genetic Algorithm for Automatic Test Pattern Generation for Large Synchronous Sequential Circuits" , *IEEE Transactions on Computer-Aided Design*, Vol. 15, No. 8, August 1996, pp. 991-1000
3. F. Corno, P. Prinetto, M. Sonza Reorda, "Testability analysis and ATPG on behavioral RT-level VHDL," *IEEE International Test Conference*, 1997, pp. 753-759
4. F. Corno, M. Sonza Reorda, G. Squillero: "VEGA: A Verification Tool Based on Genetic Algorithms," *IEEE International Conference on Circuit Design*, Texas, 1998, pp. 321-326
5. F. Corno, M. Sonza Reorda, G. Squillero, "Approximate Equivalence Verification of Sequential Circuits via Genetic Algorithms," poster in *DATE'99 Design, Automation and Test in Europe*, 1999
6. F. Corno, M. Sonza Reorda, G. Squillero, "Approximate Equivalence Verification Techniques for RT-Level Descriptions," submitted to: *GLSVLSI'99, Great Lake Symposium on VLSI*, 1999
7. F. Corno, M. Sonza Reorda, G. Squillero, "Approximate Verification of RT- versus Gate-Level Sequential Circuits", submitted to: *DAC'99, ACM/IEEE Design Automation Conference*, 1999
8. E. Goldberg, "Genetic Algorithms in Search, Optimization, and Machine Learning," Addison-Wesley, 1989
9. S.-Y. Huang, K.-T. Cheng and K.-C. Chen, "AQUILA: An Equivalence Verifier for Large Sequential Circuits," *ASP-DAC*, 1997
10. J. H. Holland, *Adaption in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MC (USA), 1975.
11. M.S. Hsiao, E.M. Rudnick, J.H. Patel, "Automatic Test Generation Using Genetically-Engineered Distinguishing Sequences," *IEEE Transactions on Computer-Aided Design*, Vol. 16, N. 9, September 1997. pp. 1034-1044
12. D. G. Saab, Y. G. Saab, J. A. Abraham, "Automatic Test Vector Cultivation for Sequential VLSI Circuits Using Genetic Algorithms," *IEEE Transactions on Computer-Aided Design*, Vol. 15, N. 10, October 1996, pp. 1278-1285

Evolving Routing Algorithms with the JBGP-system

Eduard Lukschandl[1], Henrik Borgvall[2], Lars Nohle[2], Mats Nordahl[2], Peter Nordin[2]

[1] Ericsson Hewlett-Packard Telecom AB, P.O. Box 333, S-431 24 Mölndal, Sweden
[2] Institute of Physical Resource Theory, Chalmers University of Technology, S-412 96 Göteborg, Sweden

Abstract. This paper describes work in progress where we apply Genetic Programming to the problem of finding routing algorithms in telecommunication networks, using a network simulator and the Java Bytecode Genetic Programming System being developed at the EHPT lab.

1 Introduction

Circuit switched communication networks consist of switches connected by links. Normally, the topology of the network only contains connections from a switch to a small number of its neighbours. This means that a phone call originating in one switch and destined for a non-neighbour has to be routed via other switches. The routing problem consists in chosing algorithms for routing calls in a network with limited node and link capacities. Network operators lose substantial amounts of money because calls do not reach their destination. One of the reasons for this is that switches may get overloaded, blocking the attempted call, so that the caller gets a busy signal. A number of new techniques from artificial intelligence, such as artificial neural networks, genetic algorithms, and ant-like agents have been tried in attempts to improve routing algorithms.

Today, routing in telecommunications networks is often done by using a routing table to decide to which neighbouring switch a certain call should be routed. A routing table is a mapping from the set of all possible destination switches (determined, e.g., by the area code) to the set of neighbouring switches of the node in question. The routing tables are usually static, i.e., they are defined as part of the network configuration process, and are only rarely changed after that. A routing table may contain alternative choices, so that if the first neighbour chosen is overloaded the call is routed to the second choice listed in the routing table, and if the second neighbour is also overloaded, an alarm to a human operator may be generated. The operator has two alternatives: blocking the call, or manually (and temporarily) updating the routing table. Considerable losses may occur because the operator usually chooses the blocking alternative.

The aim of this project is to use GP to learn routing functions for telecommunication networks, utilizing the Java Bytecode GP framework developed by the group. Both static and dynamic routing will be considered in the project.

R. Poli et al. (Eds.): EvoIASP'99 and EuroEcTel'99, LNCS 1596, pp. 193–202, 1999.

2 Genetic Programming

An evolutionary algorithm maintains a population of structures such as computer programs, or binary strings. These are bred much like dogs or cattle, selecting for some desirable feature with a large number of generations passing each second. In this way one can breed computer programs that solve problems that no human programmer can solve, using completely new and innovative programming techniques.

One approach to evolutionary algorithms is Genetic Programming (GP) [4], which lets the computer program itself by evolving programs, e.g., in machine code. An overview is given in [2]. This method can be used, e.g., when theories are lacking, and a system cannot be designed from first principles, or when there is no time for human programming and complete programs have to be written in a few seconds in order to adapt to new situations.

Since its introduction in 1992 GP has been used to solve a number of hard problems, e.g., in speech and image understanding, robotic control and pattern finding. Most of the programs generated are difficult to analyze, and it is evident that the computer creates programs for itself in a very different way than human beings. However, some analyzable parts show impressive, ingenious and creative use of computer resources in solving a hard problem.

GP has not yet been extensively applied to routing problems in telecommunications networks (one exception is given by ref. [3, 8]).

3 The system

The system is built around a network simulator that allows us to model a network and the features we are interested in, such as telephone call patterns, and routing algorithms, and to study properties of the system such as the load inflicted on a switch when establishing a route, and the number of calls lost due to switch overloading.

The simulator simulates the behaviour of people making phone calls via the network for a certain period of time. Initially, very simple stochastic models have been used to generate calls, but the aim of the project is to move to more realistic models based on real data.

In general, the simulator can take into account various limitations built into the system, such as

- The nodes contain one or several limited resources.
- Data processing in the nodes may take a certain amount of time.
- Links between nodes may have limited transportation capacity.
- Data transmission along the links may take a finite amount of time.

In the initial runs described below, we restrict ourselves to the case of unlimited link transmission capacity, zero transmission time, and a single resource at each node that is used during the duration of a call and models a resource like buffers or ports (another example of a resource might be a processing unit

that is only used while connecting and disconnecting a call). This means that connecting and disconnecting calls takes zero time and zero resources.

In each node of the network, there is a single information processing and routing entity. This software entity or program gets input from an environment and processes it. It may produce an output, affect its environment, and/or change its state if equipped with internal memory. It could more formally be described as an finite automaton or Turing machine equipped with actuators.

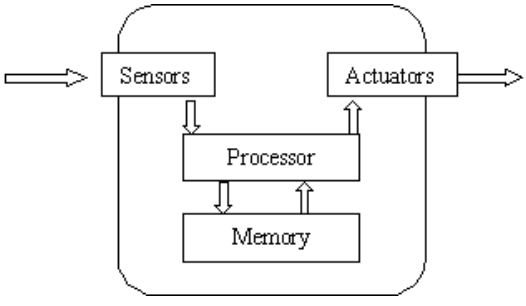


Fig. 1.: Schematic view of a node.

The environment contains of a number of such routing entities, each one representing a node in a communication network of predefined topology. The nodes are linked to each other by connecting the actuators of the routing program with the sensors of the other routing programs. All links are defined as being bi-directional.

3.1 The Call Simulation

We simulate the behaviour of people making telephone calls via the network for a certain period of time. The calls consist of normal circuit switched connections. The simulator contains an event queue and a clock or counter simulating the progress of time. There are two basic kinds of events: call-connect events and call-disconnect events. A large number of random calls are generated before the simulation starts. These calls are then sorted and fed into the simulated network. Calls are taken from the event queue and dispatched to the appropriate node.

The following tasks are performed when a call, or *call-connect event*, arrives at a network node:

– If the *load capacity* of node is exceeded then the call fails and we update the counter for failed and blocked calls. The information about the failed call is propagated backwards in the net decreasing the load of earlier nodes.

- If it is detected that this call already has passed the node then the call fails due to *circular routing*. The failure counter for circular calls is updated and infomation is propagated backwards just as for the blocked call.
- If none of the problems above are encountered the call is considered successful and if the node is the destination node of the call then the success is noted and a call-disconnect is scheduled after the prescribed time.
- If the node is not the destination node the routing program is used to foward the call to a neighbouring node.

A call-disconnect event causes the load of the nodes in question to be decremented.

3.2 The Evolutionary System

The genetic programming framework used to represent the routing programs is the Java Bytecode GP system developed at EHPT. Descriptions of this system are given in [5, 6]. For the purposes of our current research we have streamlined it to evolve functions, represented as Java methods in bytecode format, that take only numbers as parameters and return only numbers. Thus the instructions used in the programs are a subset of the JVM instruction set. Technically speaking, this GP-engine is a Java method evolver (JME).

When the system is run, the JME asks the environment to evaluate the individuals of the population, i.e. the candidate methods, by calling the environment's evaluate-method and passing it the individual (method) to be evaluated. The environment takes the individual and plugs it into the processor-slot of one or more nodes of the network. Then a simulation run starts and the calls are dispatched by the scheduler and routed according to the program of the individual. During the simulation various values are collected, such as number of failed calls or the money earned. From these values in different experiments fitness values can be computed which are returned to the JME. In the initial version

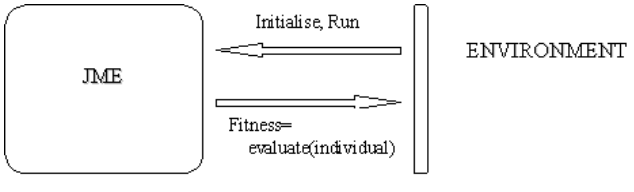


Fig. 2.: The relationship between JME and the environment.

of the system, one algorithm is evolved for each class of nodes with a certain out-degree. The algorithms initially have access to information about the load at neighbouring nodes, possibly information about distance to the goal for the

different neighbours, or other local information. The task of the system is to find routing algorithms for each node type that optimise the performance of the system, measured either by the number of lost calls for a certain call sequence or a variety of call sequences, or by the revenue with different price tags on different calls, e.g., differentiating between local, long distance, and international calls.

In another version of the system, separate algorithms will be evolved at each node of the network, which means having a population at each node, where each individual program is tested for a fairly short time. Similar strategies have been successful when using GP for robot control (e.g., [7]), and it will be interesting to attempt to extend these ideas to a multi-agent situation. This version will provide an interesting testing ground for different ideas on credit assignment.

4 Experiments

4.1 Restrictions in the Model

In these initial experiments we made the following choices of parameters and restrictions compared to the final objectives:

- The network used was based on the Synchronous Digital Hierarchy network of British Telecom used as a test case by Schoonderwoerd and others [1, 9]. In the intial experiments we have however reduced the network topology by taking only a subset consisting of the 13 northernmost nodes of the network into account, see Figure 1.
- The number of calls was 250 per simulation, evenly distributed over 250 time units.
- The mean call duration was 20 time units.
- The number of call patterns was held to one: all nodes have the same probability of being the source of the calls and the same probability of being the destination.
- The load capacity of the nodes was set to 8.

We also made a few other simplifications, e.g., we did not use any state variables or internal memory in the nodes. Furthermore all knowledge transmission was considered to be instantaneous, and we did not consider any topological differences between nodes. This paper describes initial experiments showing the basic feasibility of the experiments but the evaluations are not, at present, extensive enough to enable statistically significant conclusions.

4.2 Experiment Setup

The following experiments were performed:

In Experiment A each node ran a routing program. The output of this node-internal routing program specified the target neighbour node of an incoming call, so that each node could directly tell which of its neighbouring nodes it should route to. The input parameters to the routing program in each node were the

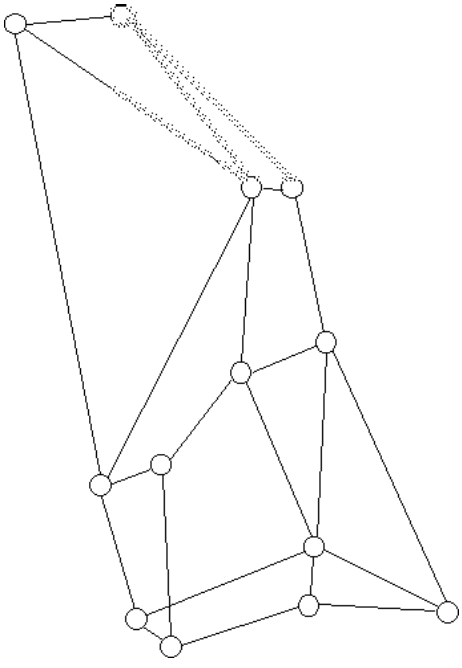


Fig. 3.: The northen part of the SDH network of British Telecom.

destination node, the loads of the neighbouring target nodes, and their distances to the destination node. Preliminary evaluations showed that this approach was computationally expensive and complex to evolve, and the remaining experiments were performed in a quite different manner:

A node which wants to know which neighbouring node to route to does the following. It asks each of the neighbouring nodes to run a "busyness program" and return a *busy-factor* to the asking node. The node then simply choses the neighbouring node with the lowest busy-factor. We performed three different experiments with this set-up:

- Experiment B: The input parameters to an evolved "busyness program" were the load of a node, and its distance to the destination node of the call to be routed. The number of lost calls was minimised.
- Experiment C: The input parameters were the load of a node, and its distance to the destination node, as above. But in these experiments, calls to a certain node were given a tenfold value compared to the other calls. In this case, the revenue was maximised.
- Experiment D: The input parameters were the load of a node, and its distance to the destination node, as above; and in addition, a value, either 1 or 10, of the call to be routed. The revenue was maximised.

In all experiments we used the following GP-parameters:

- number of individuals: 500
- cross-over rate: 85%
- mutation rate: 5%
- percentage of randomly chosen individuals copied: 5%
- percentage of best individuals copied: 5%

5 Results

The following results were observed:

Experiment A: This experiment was abandonded as described above.

Experiment B: In this case the objective was minimising the number of lost calls using the load and distance to the destination as parameters. The experiment was performed over 100 generations, which means that a total of 50000 individuals were evaluated, or in other words that 50000 simulations of 250 calls between 13 switches were run.

As can be seen in Figure 4 the problem is quite amenable to random search, and the best of the 500 randomly generated algorithms in the initial generation causes the loss of 7.6% of the 250 calls, i.e., 19 calls. Furthermore we conclude that the improvement over time in the evolutionary process is rather moderate leading to a call loss of 6.8% after 100 generations.

The function associated with the best simulation can be expressed as follows:

$$L(\frac{1}{2}d + 2 + \frac{1}{2}d(d + L/(d - L^2)) + d \tag{1}$$

where L is the load relative to the maximum load, and d is the distance to the destination. Most of the calls are lost because of blocking situations.

Experiment C: In this case the objective was maximising the revenue using the load and distance to the destination as parameters, see Figure 5. An interesting observation in this experiment is that the percentage of lost calls is higher than in Experiment B, 8% after 80 generations compared to 6.8%. But as can be seen in Figure 7 showing the revenue, the amount of money earned at generation 80 is 410 units for 230 calls, compared with 404 units for the 233 successfully routed calls in Experiment B. This suggests that the algorithm is able to discriminate between normal and expensive calls, without having explicit knowledge about the price, and only getting implicit information via the fitness value.

Experiment D: Here the objective was maximising revenue using the load, the distance to the destination, and the price of the call as parameters. The results are shown in Figure 6. As expected, providing the algorithm with the price parameter and maximising revenue leads to even better performance revenue-wise, while the percentage of lost calls is the same as in Experiment B.

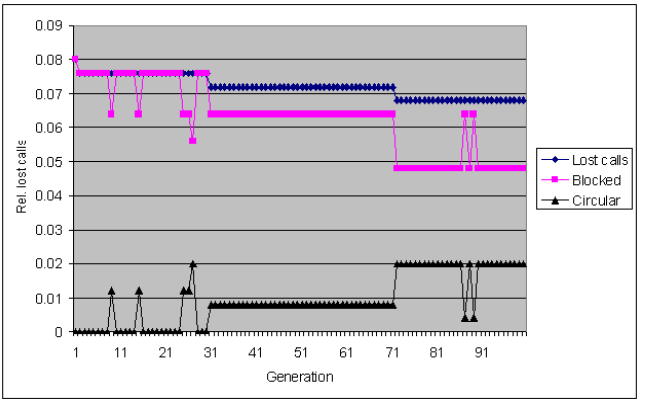


Fig. 4.: Experiment B: Minimising the number of lost calls.

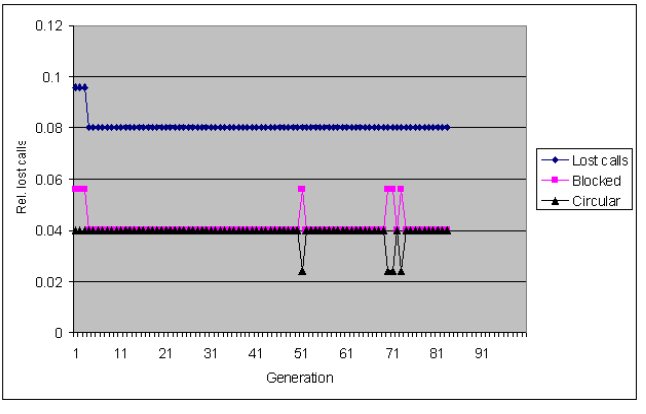


Fig. 5.: Experiment C: Maximising the revenue using two parameters.

6 Conclusion and Discussion

The preliminary results show that Genetic Programming is a feasible method for the induction of routing algorithms and possibly a viable alternative to other methods. The flexibility in adding arbitrary parameters, such as the price of calls, and the ease of introducing the evolved algorithm into a running system, encourage us to continue with this line of research.

In future work we will extend this work to study networks of more realistic size (such as the complete SDH network mentioned above). We also intend to study the performance of the algorithms under the influence of non-random call-

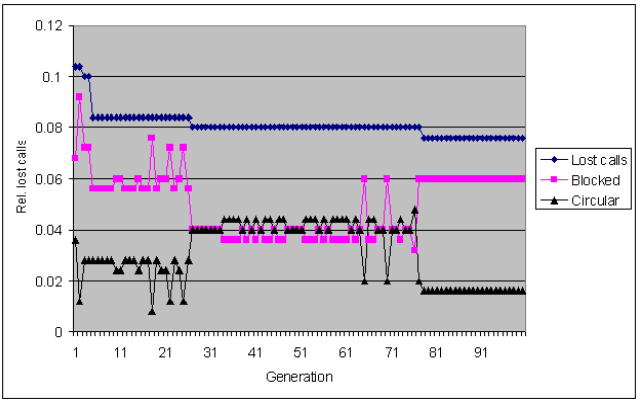


Fig. 6.: Experiment D: Maximising the revenue using three parameters.

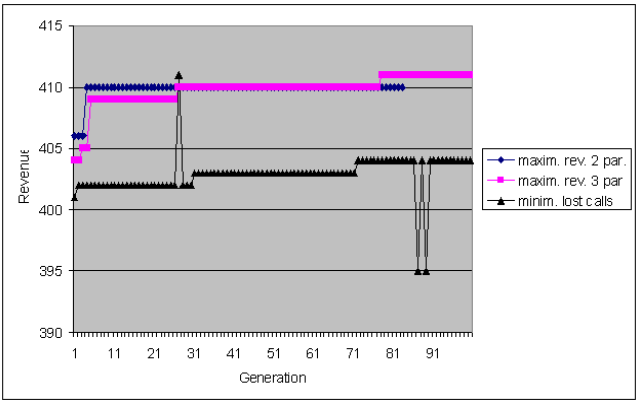


Fig. 7.: The revenue from experiments B-D.

patterns (e.g., more calls originating from one node or more calls destined for one node), in particular call-patterns taking more statistical features of real data into account.

We will also investigate the sensitivity of the algorithm to topology changes in the network, such as broken links or nodes, where the genetic approach may turn out to be useful in relearning and adapting the routing algorithm.

The experiments should also be extended to make the results truly statistically significant. The simulation itself could easily be parallelized (the system can already run transparently across a network in a distributed fashion). We

will also carry out experiments where the programs are expressed in binary machine code, which will both improve speed in the evolution and be essential for applications in real problem domains. The efficiency could also be further improved by the use of time parsimony, i.e., including execution speed as part of the fitness function. The results should of course also be compared more carefully against other approaches, from simple routing tables to more complicated adaptive methods.

References

1. Appleby, S., Steward, S. : Mobile Software Agents for Control in Telecommunication Networks, BT Technology Journal, **12(2)** (1994).
2. Banzhaf, W., Nordin, P., Keller R.E., Francone, F.D. : Genetic Programming - An Introduction. Morgan Kauffmann, San Fransisco, and d-punkt, Heidelberg, Germany (1998).
3. I. M. A. Kirkwood, S. H. Shami, and M. C. Sinclair: Discovering simple fault-tolerant routing rules using genetic programming. ICANNGA97, University of East Anglia, Norwich, UK, (1997).
4. Koza, J.R : Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press (1992).
5. Lukschandl, E., Holmlund, M., Modén, E.: Automatic Evolution of Java Bytecode: First Experience with the Java Virtual Machine: Late Breaking Papers at the First European Workshop on Genetic Programming (EuroGP'98) Paris, 14-15 April (1998).
6. Lukschandl, E., Holmlund, M., Modén, E., Nordahl. M., Nordin, P. :Induction of Java Bytecode with Genetic Programming. Late Breaking Papers at the Genetic Programming Conference. J.R. Koza (ed.), University of Wisconsin, July 22-25. Stanford, CA: Stanford University Bookstore (1998).
7. Nordin, P., Banzhaf, W. :An On-Line Method to Evolve Behaviour and to Control a Minature Robot in Real Time with Genetic Programming. Adaptive Behavior **5(2)** (1997) 107-140.
8. Shami, S. H. , Kirkwood, I. M. A. , Sinclair, M. C.: Evolving simple fault-tolerant routing rules using genetic programming. Electronics Letters, **33(17)** August (1997) 1440–1441.
9. Schoonderwoerd, R., Holland, O.E., Bruten, J.L., Rothkrantz, L.J.M.: Ant-like Agents for Load Balancing in Telecommunication Networks. Agents'97, Marina del Ray CA, USA (1997).

Optimising Self Adaptive Networks by Evolving Rule-Based Agents

Evaggelos Nonas[1] and Alexandra Poulovassilis[2]

[1] Department of Computer Science, King's College London, Strand, London WC2R 2LS, U.K.
vagelis@dcs.kcl.ac.uk
[2] Department of Computer Science, King's College London, Strand, London WC2R 2LS, U.K.
alex@dcs.kcl.ac.uk

Abstract. The need for networks that adapt autonomously to dynamic environments is apparent. In this paper we describe how self adaptive networks can be optimised by means of agents residing on the nodes of the network. The knowledge of these agents is a set of active rules. A genetic algorithm dynamically prioritises these rules in the face of dynamically evolving conditions. To our knowledge, this is the first time that GAs have been used for this purpose. We demonstrate the applicability of our method by presenting several experiments and results.

1 Introduction

As telecommunication networks have become bigger and more complex, the need for managing them effectively, optimising their capacity and reducing their operation costs has become apparent. This need is becoming more urgent as the telecommunications market is continuously changing and new services have to be constructed and provided quickly and cheaply. Moreover, since users are becoming mobile, networks have to adapt quickly to varying load conditions and traffic patterns.

There are many methods for optimising a network. Some of them solve this problem using an analytical approach, others using an evolutionary approach. Both categories give good results from a long-term point of view. They use statistical data to calculate average costs, which do not change over a long period of time: weeks or even months.

In contrast, what we propose is a method for optimising the network on the protocol level, by using costs that can be predefined or collected at run-time. For the optimisation procedure we combine the analytical with the evolutionary approach. Parts of the problem are solved using a deterministic algorithm, and other more

R. Poli et al. (Eds.): EvoIASP'99 and EuroEcTel'99, LNCS 1596, pp. 203–214, 1999.

complicated parts are solved using a genetic algorithm. Such a network will have the ability to adapt to dynamic environments with little or no human intervention, so it is termed a Self Adaptive Network.

We use software agents that reside on each node of the network and optimise it in real time. The knowledge of each agent is expressed in the form of active rules consisting of events and actions. The reactive part of the agent (the part that responds to external events) is dynamically optimised by a genetic algorithm. The rational part of the agent also uses active rules, but these are statically defined.

The outline of this paper is as follows: Section 2 describes the main features of self adaptive networks. Section 3 describes and compares two software architectures that work using active rules: beliefs, desires, intentions (BDI) agents and active databases. It then describes the active rules that our system uses. Section 4 describes the genetic algorithm that optimises the rule-based agents. Section 5 gives some experiments and results from our system. In section 6 we present conclusions and future research directions.

2 Self Adaptive Networks

A self adaptive network is a network that can automatically adapt to changes in its environment without human intervention being necessary. While load conditions change and nodes and links may fail, the network continues to operate near the optimum state, requiring little or no assistance from its operators. In other words, the network must be autonomous, intelligent and have distributed control. There should be no need for global knowledge in the network. On the contrary all information must be kept as local as possible.

Our network model is a simple yet powerful one. The network is composed of a set of nodes and a set of connections between them. Each node can exchange messages only with its immediate neighbours. There is no global knowledge of the topology of the network stored in any node. There is a set of services provided by the network and each node can provide some or all of the services. The task for every node is to provide the services requested from it with the minimum cost. The cost can be a function of the number of intermediate nodes and links the service is using, as well as of the load and spare capacity of those nodes and links respectively. Obviously, the larger the number of intermediate nodes and links a service is using, the larger the cost for the provision of that service.

Messages are exchanged between nodes to allow service establishment and service cancelling. Messages can be exchanged only between connected nodes. For the time being we use three kinds of messages, but we intend to add more in the future. The first kind of message requests a service from a node and has as parameters the requesting node, the service number as well as the hop-count (number of intermediate nodes the request has used). Messages with a hop-count greater than a specific number are canceled automatically, to avoid flooding the network with cyclic or very long requests. The second kind of message concerns the answer to a request for a

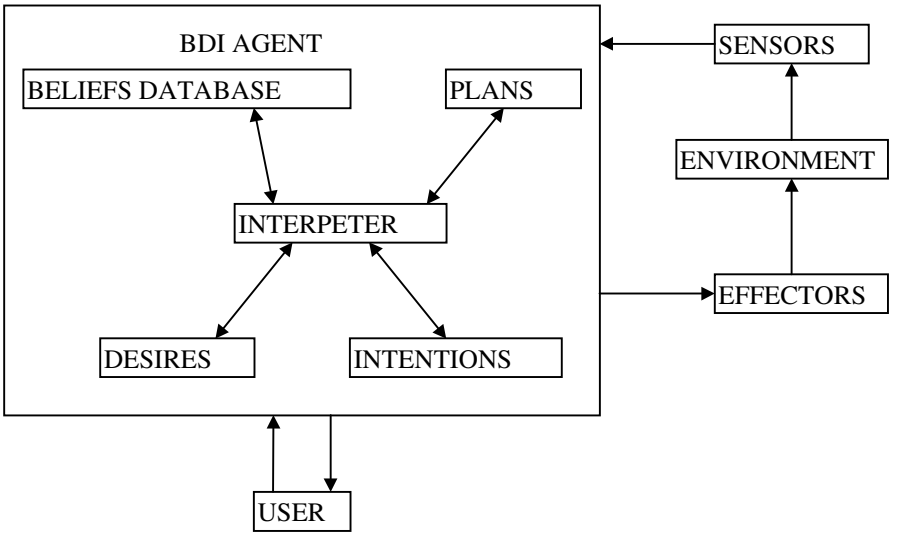


Fig. 1. Typical BDI System

service. If the service can be provided, the cost is returned, otherwise the message just rejects the request. The third kind of message cancels services already provided.

When a service is provided its cost is calculated as follows:

$$\sum_i N_i + \sum_k L_k \quad (\mathbf{1})$$

where i is the number of nodes and k is the number of links the service is using, and N_i and L_k is the load imposed by this service on each node and link, respectively. When the service can not be provided (because a node or link has reached its maximum capacity, or because the hop count has exceeded the maximum allowed limit), the same formula is used for the cost of the service, but i and k are now set to the total number of nodes and links in the network respectively. Clearly, more sophisticated cost functions can be used in dynamic environments.

3 Active Databases and BDI Agents

Beliefs-desires-intentions (BDI) agents have been extensively studied for some years [5], [9], [11], [2]. A BDI agent has the following components (see Figure 1):

- Beliefs Database: Contains facts about the sate of the world, as well as about the agent's internal state.
- Desires: Contains agent's goals expressed as conditions over some interval of time and are described by applying various temporal operators to state descriptions.

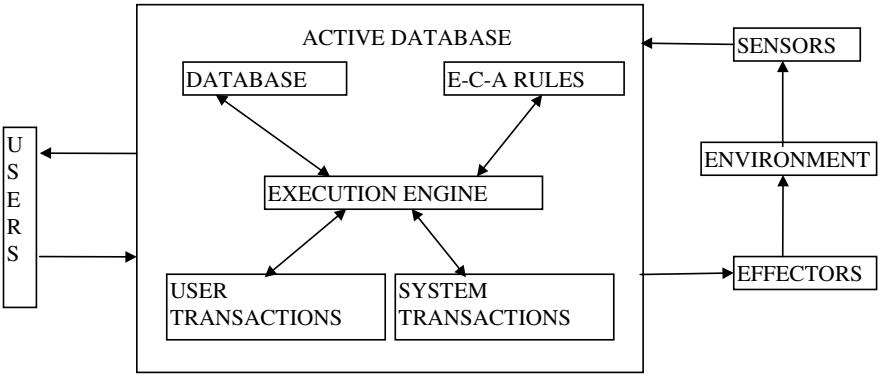


Fig. 2. Typical Active Database System

- Plans: Actions the agent has to take in order to fulfill its goals. They have an invocation condition which specifies upon which events the plan should be fired and a context condition which specifies under what condition the plan applies.
- Intentions: Plans that are valid for firing are placed in an intentions structure where they are executed. They can be hierarchically ordered.

Active databases are also based upon an Event-Action architecture [4] (see Figure 2). An active database system consists of the "traditional" components of a database system plus a component that is concerned with the firing of event-condition-action (ECA) rules. The meaning of an ECA rule is: "when an event occurs check the condition and if it is true execute the action". There is an event language for defining events and for specifying composite events from a set of primitive ones. The condition part of an ECA rule formulates in which state the database has to be, in order for the action to be executed. The action part of an ECA rule may start a new transaction which when executed may trigger new ECA rules. In this way we can have trees of triggering and triggered transactions.

By comparing Figures 1 and 2 we can very easily see the correspondence between the components of the BDI agent and active database architectures. The beliefs database of the BDI agent corresponds to the main database store of an active database. The desires of a BDI agent are expressed in an active database as transactions submitted by users. The plans of a BDI agent correspond to the ECA rules of the active database. Finally the intention structure of the BDI agent is expressed in the active database as transactions generated by the system, i.e. transactions generated from the activation of ECA rules.

Similarities and differences between BDI agents and active databases are discussed in more detail in [1], [12], where characteristics such as events, actions, consistency, query expressiveness, goal achievement and responsiveness are compared. The most important of their common characteristics is the way that actions are executed, in that

Table 1. A Simple Example Rule Set

Events	Actions
Provide service A	Local Remote
Provide service B	Local Remote
Provide service C	Local Remote
Send request for service A	Send to node 2 Send to node 3 Send to node 4
Send request for service B	Send to node 2 Send to node 3 Send to node 4
Send request for service C	Send to node 2 Send to node 3 Send to node 4

upon a certain event occurring, if a condition holds a rule is fired. There may be cases where more than one rule may be triggered by the same event occurrence. The system will then select the rule with the highest priority to fire, or will arbitrarily select a rule to fire if there are more than one with the same priority.

In this paper we assume that there is an agent running on each node of the network. The knowledge of each such agent is expressed using a set of rules. An event occurs at a node when it is asked to provide a service. There are two possible actions that can be triggered for this event: the service can be provided remotely, or the service can be provided locally. When a service is to be provided remotely, a new `send request' event is generated. The possible actions corresponding to this event are all the nodes that the requesting node is connected with. For instance, if node 1 is connected with nodes 2, 3 and 4 and the network provides services A, B and C, the events and actions for node 1 are shown in Table 1 (no order is shown, just all the events and all the possible actions for each event).

4 Using a GA to Optimise the Rule Based Agents

Our method provides an automatic way of selecting the "best" rule to fire upon an event occurring, using a genetic algorithm to determine which rule to fire if more than one rule is triggered. Genetic algorithms and genetic programming have been used before in the design of agent systems [8], [10]. The novelty of our work is that we are using GAs to dynamically optimise a set of rules in response to changes in the environment.

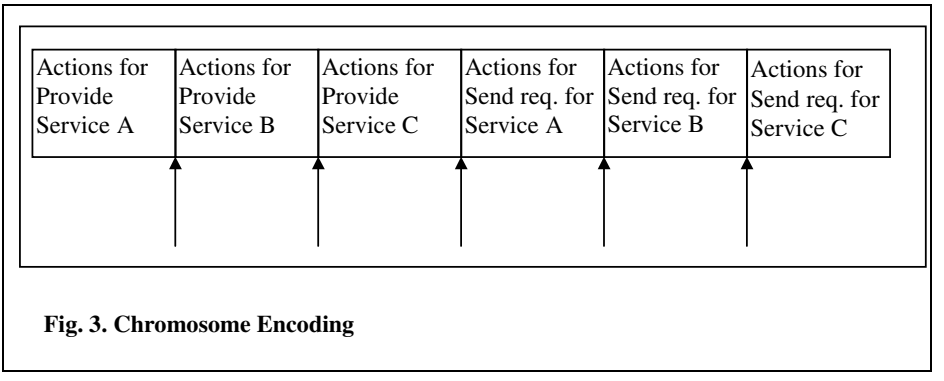


Fig. 3. Chromosome Encoding

For the moment we do not support conditions in our active rules, although we plan to cater for conditions in the future.

At each node, the system holds a list of possible actions that can be taken for each event that may occur. The first action is always selected, but a simple genetic algorithm running in parallel dynamically changes the order of the actions. Obviously this approach requires a measure of the performance of the agent, which must be available at run-time, to be given to the genetic algorithm.

The GA is used to try out several permutations of the rule set and finally find the best ordering. Permutations of the possible actions for each event are enumerated and placed in the chromosome one after the other. We assign each permutation an integer in the range 0..n!-1, where n is the number of actions. The binary representation of this number is placed in the chromosome to encode that permutation of actions for the event. The whole chromosome is composed of a sequence of K such numbers, in their binary representation, where K is the number of possible events. Thus one chromosome can encode all the rules with which each agent works[1].

Each agent has a chromosome pool which is initially randomly instantiated. These chromosomes are evolved by the genetic algorithm to better solutions. We use a constant population size, selection proportional to fitness, and full replacement of parents by their children. Multiple point crossover is used for breeding. Crossover points are set at the end of each event in the chromosome. The chromosome for the example of Table 1 is shown in Figure 3, where the arrows show the positions of the crossover points.

The fitness of each chromosome is calculated as follows: When a node provides a service to another node, it also sends to it the cost of this service. This cost is a function of the number of intermediate nodes and links the service is using as well as

[1] This particular encoding of the GA was chosen initially for ease of programming, but it was adopted since it performed well. Our current library that implements the Genetic Algorithm, does not support other than the binary encoding. An alternative method for describing permutations would be as ordered lists. We are in the process of adding this feature to our library. Once we've done this, we are planning to test the performance of PMX or other permutation crossovers ([3, pp. 72], [6], [13]).

their load and free capacity respectively. Obviously, when the service is provided locally, the cost is minimum. Each chromosome in the chromosome pool is used for service provision for some time and the costs of the services provided using it are averaged. The fitness, then, for this chromosome is inversely proportional to this average cost. So the larger the cost, the smaller the fitness of the chromosome and vice-versa.

The fitness of each rule set is given by:

$$F = 1000 \times \frac{(M - A)^2}{M^2} \quad (2)$$

where M is the maximum cost for service provision and A is the average cost for all the services provided using this rule set. Fitness is normalized between 0 and 1000. The squared term helps the GA to converge more quickly to a solution.

The current implementation of our architecture is in Borland C++ Builder and runs under Windows 95 or Windows NT. A network simulator as well as the actual agents running on each node of the network have been built. The genetic algorithms used by the agents have also been programmed. There is a graphical user interface that provides for the design of the network, the design of the rules the agents are using and the fine-tuning of the genetic algorithm that each agent runs.

Since the genetic algorithm controls the way the agents respond to events, we can say that the reactive behaviour of the agent is controlled by the genetic algorithm. But there can also be another part, the "rational" part, that controls the agent, for example if our architecture is part of an agent built partially using another method and controlled partially by the constructs this method provides. If for instance the agent is built conforming to the BDI model, it will have facts, goals, plans and intentions. Some of the plans will be selected for execution using the traditional approach, but some others using the GA approach. The rational part of the agent can also control several parameters of the GA, restart it when needed, or schedule it to be run when the load is low.

5 Experiments and Results

In this section we present some results for several network configurations. In all the graphs, the Y axis shows the mean fitness of the nodes' chromosome pools, averaged over all the nodes. The X axis shows the number of generations the genetic algorithm has been run. While nodes are being trained, service requests have a uniform distribution as far as type of service is concerned, across all nodes. Of course, real traffic data can ultimately be used for more effective training.

Our first experiment uses a network of 100 nodes and 200 links. The topology has been randomly created by our software. There are 40 services provided across the network. We examine three different cases with varying service distribution across nodes. In the first case all 40 services are provided by all the nodes. In the second case there is a random distribution of services across nodes. The number of different services provided by each node is drawn randomly from the range [1..40]. In the third

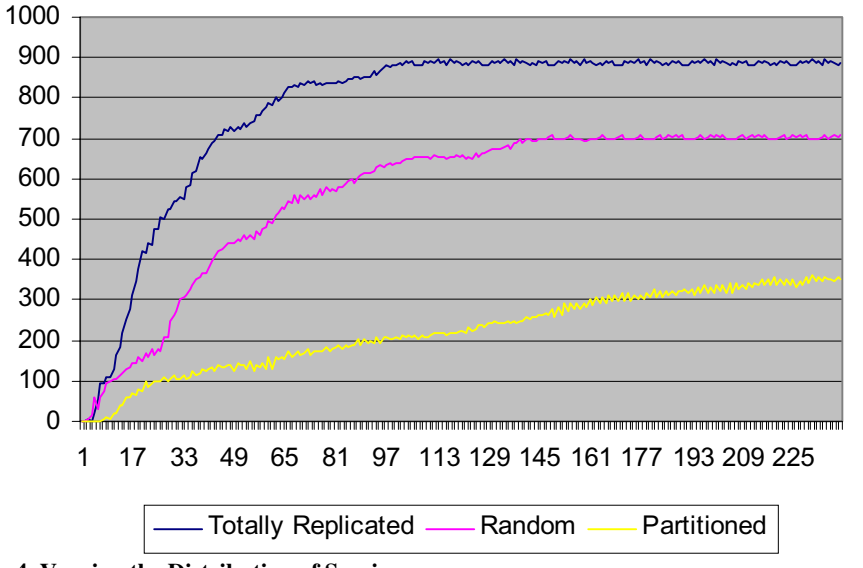


Fig. 4. Varying the Distribution of Services

case, services and nodes are split into 5 disjoint sets and eight services are provided by each node. For example, nodes 1 to 20 provide services 1 to 8, nodes 21 to 40 provide services 9 to 16, etc. Graphs for all three cases are shown in Figure 4 under the legends Totally Replicated, Random and Partitioned respectively.

As we would expect, the best performance is achieved when services are totally replicated across all nodes. The worst performance is achieved when services and nodes are partitioned into disjoint sets. This is because only a few of the total number of services can be provided locally, or with a small hop-count. Random distribution of services results in a performance between the two "extreme" cases.

Our second experiment demonstrates the fault tolerance of the network and its behaviour is illustrated in Figure 5. There is a network, Network A, consisting of 11 nodes and 10 services. 10 of the nodes are connected in a ring and provide only 3 services each, which vary from node to node. The 11^{th} node provides all 10 services and is connected with all the other nodes. So it is the most important node of the network.

The black curve in Figure 5 shows the performance of the network when node 11 is down from the beginning of the run until it finishes: we call this network Network B. Network B is optimised to an average fitness of approximately 380. The grey curve initially shows the behavior of Network A, which is optimised to a state higher than Network B. After 500 generations node 11 goes down and the performance of the network decreases initially but after approximately 500 more generations it reaches the expected performance for Network B. At that point node 11 comes up again and the performance of the network is restored to its original value. After 1400 generations from the beginning of the experiment node 11 goes down again but this

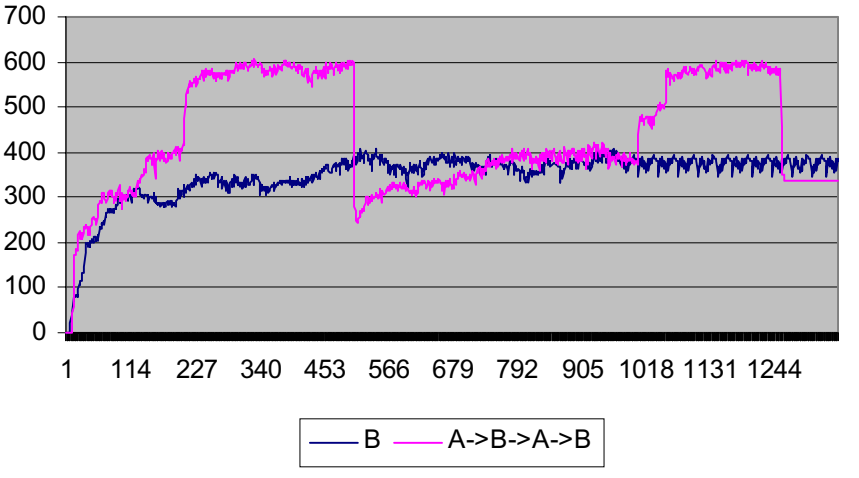


Fig. 5. Fault Tolerance Demonstration

time no GA is used for optimising the network. Instead the agents remove from their rule sets any dependencies they have on node 11. To do this they degrade the priority of actions involving node 11 to the last position in the rule set. The performance of the network after this point is shown by a flat line, since there is no evolution of the rule sets. The performance is about 350, which is close to the 380 mark that the GA can achieve after evolution and certainly near (but a bit lower) the optimum performance for this configuration. This last observation shows that our approach can be used for very quick network restoration and perhaps also for congestion control.

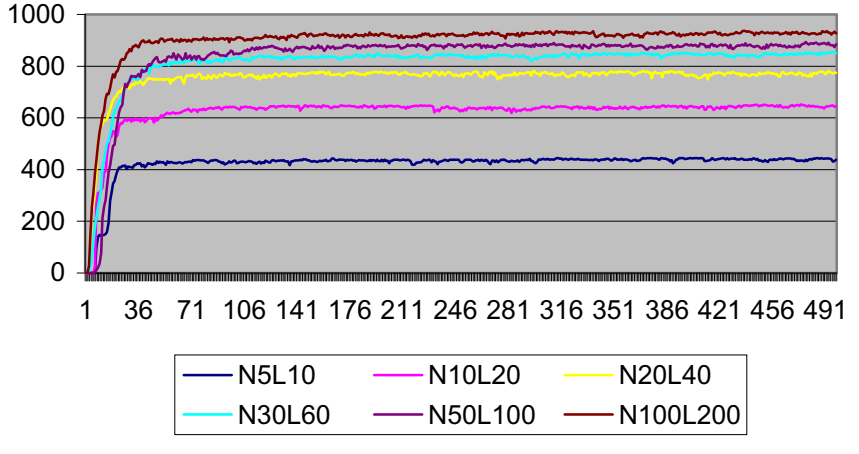


Fig. 6. Speed of Optimisation

It is for this reason that we keep a permutation in the chromosome, instead of a single "best" action for each event. As we have demonstrated in our experiments the second action for an event can be used for network restoration in case of failures. It could be argued though that the actions at the bottom of the event action table will be used rarely and thus that maintaining them is an unnecessary overheard. Thus, further investigation into the usefulness and fitness of the lower-order actions is necessary.

Finally we present a third experiment, which shows the speed the GA optimises a network according to its size. In figure 6 we present six different cases named NXLY, where X and Y are respectively the number of nodes and links the network has. All six networks have a links-to-nodes ratio of two to one. All the networks provide 5 services. We see that the time taken by the GA to optimise the network is not dependent on the size of the network. This is a very important fact that demonstrates the distributed solution and load balancing our method supports. One can also observe that bigger networks have better performance. This is to be expected since bigger networks have more alternatives for providing "cheap" (i.e. lower cost, so higher fitness) services.

6 Conclusions

In this paper we have described how self adaptive networks can be optimised by means of agents residing on the nodes. The knowledge of these agents is a set of active rules. A genetic algorithm dynamically prioritises these rules in the face of dynamically evolving environments. To our knowledge, this is the first time that GAs have been used for this purpose. We have showed that our approach is good for network failures and network restoration. We expect it to be well suited to more general conditions of varying load, and more experimentation is necessary into this. The advantages of our approach to optimising self-adaptive networks are apparent: distributed solution, load balancing and sharing, and self adaptation to varying load conditions and fault situations.

Our network model is connectionless and best effort. In other words it very much matches the TCP/IP routers used to handle traffic on the Internet. It will try to transmit a packet (provide a service in our model) using the best possible way. It will always take the first choice of the active rule set, but if this is unavailable, then it will take the second, and so forth. Another application domain for our approach is global query optimisation in distributed heterogeneous databases. Such systems consist of multiple autonomous databases, and there is little or no global information about local cost models and database contents. We envisage that agents residing on each node could use dynamically evolving active rules to determine the best way to process each type of query (i.e. service) requested at that node.

One could argue that in our system the genetic algorithm can find a local optimum and then stop. This is always a possibility with genetic algorithms, but in a network where service distribution across nodes is done in such way that neighbouring nodes have some services in common there are many good solutions and the genetic algorithm will find one of them. In extreme cases where there is only one good

solution the genetic algorithm may fail, but it can be restarted by the rational part of the agent with many chances of finding a better solution. Overall, the advantages of adaptation, autonomy and distributed operation are more important in self adaptive networks than the discovery of the best solution, especially in a dynamic and continually changing environment where keeping track of global information would be difficult if not impossible.

For further work we plan to construct the rational part of the agents. This too will be based on active rules. It will schedule, restart and fine tune the genetic algorithm. It will also feed it with a good initial population and will provide for knowledge exchange between neighbouring nodes. Scheduling and restarting can be done depending on changing load conditions, on changing network topologies and on the spare computational capacity of the nodes, since they also have to provide services to the network. Depending on those conditions, the rational part can either use the GA to re-optimise the network or based on the knowledge it already has can adjust the rule base for better performance. We believe that this combination of intelligence and heuristic search methods will lead to a much better performance than use of the latter alone.

Finally, we plan to apply our approach to the problem of query optimisation in distributed, heterogeneous databases, where there may be many possible ways for a query to be answered. In this context, the rational parts of the agents will facitate information sharing between the data sources while the reactive parts will optimise distributed access to the data.

Acknowledgments

E. Nonas is sponsored by B.T. Laboratories, Systems and Software Unit, Martlesham Heath, Ipswich.

References

1. J. Bailey, M. Georgeff, D. B. Kemp, and D. Kinny, “Active databases and agent systems --- A comparison”, Lecture Notes in Computer Science, 985, 342-356, (1995).
2. Michael Bratman, Intention, plans, and practical reason, Harvard University press, 1987.
3. Lawrence Davis, Handbook of Genetic Algorithms, Van Nostrand Reinhold, New York, 1991.
4. K. R. Dittrich, S. Gatziu, and A. Geppert, “The active database management system manifesto: A rulebase of ADBMS features”, Lecture Notes in Computer Science, 985, 3-17, (1995).
5. Klaus Fischer, Jorg P. Muller, and Markus Pischel, “A pragmatic BDI architecture”, in Proceedings on the IJCAI Workshop on Intelligent Agents II : Agent Theories, Architectures, and Languages, volume 1037 of LNAI, pp. 203-218, Berlin, (19-20 August 1996). Springer Verlag.

6. D. E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, Reading, Mass., 1989.
7. David Goldberg, Genetic Algorithms, Addison Wesley, Reading, 1989.
8. Thomas Haynes and Sandip Sen, “Evolving behavioral strategies in predators and prey”, in IJCAI-95 Workshop on Adaptation and Learning in Multiagent Systems, pp. 32-37, (1995).
9. David Kinny, Michael Georgeff, and Anand Rao, “A methodology and modelling technique for systems of BDI agents”, in Proceedings of the 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, volume 1038 of LNAI, pp. 56-71, Berlin, (22-25 January 1996). Springer Verlag.
10. Mauro Manela and J. A. Campbell, “Designing good pursuit problems as testbeds for distributed AI: A novel application of genetic algorithms, in Proceedings of the 5th European Workshop on Modelling Autonomous Agents in a Multi-Agend World (MAAMAW'93), volume 957 of LNAI, pp. 231-252, Berlin, GER, (August 1995). Springer.
11. Anand S. Rao and Michael P. Georgeff, “BDI agents: from theory to practice”, in Proceedings of the First International Conference on Multi—Agent Systems, pp. 312-319, San Francisco, CA, (1995). MIT Press.
12. Johan van den Akker and Arno Siebes, “Enriching active databases with agent technology”, in Proceedings ot the First International Workshop on Cooperative Information Agents, volume 1202 of LNAI, pp. 116--125, Berlin, (February26--28 1997). Springer.
13. G. Syswerda, “Schedule Optimisation using Genetic Algorithms”, In L. Davis, editor, “Handbook of Genetic Algorithms”, chapter 21, pp. 332-349, Van Nostrand Reinhold, 1991.

Genetic Construction of Optimal Circulant Network Designs *

E.A. Monakhova, O.G. Monakhov, and E.V. Mukhoed

Institute of Computational Mathematics and Mathematical Geophysics,
Siberian Division of Russian Academy of Sciences
Pr. Lavrentieva, 6, Novosibirsk, 630090, Russia
Phone: +7-3832-341066, Fax:+7-3832-324259
{emilia, monakhov}@rav.sscc.ru

Abstract. A solution of NP-hard optimization problem of constructing optimal circulant networks with the minimum diameter for given degree $\delta > 4$ and number of graph nodes $N > 1000$ is considered. The circulant networks and their different applications are the object of intensive investigations, and they are realized as interconnection networks in some parallel multicomputer systems. The application to solution of the problem of a genetic algorithm based on the simulation of natural evolution process and the comparison between it and a random and reduced search algorithms are considered. The catalogues of optimal (suboptimal) circulant networks are obtained.

Keywords: optimal networks, circulant graphs, genetic algorithms

1 Introduction

In this work we consider fundamental optimization problem of efficient interconnection networks design for parallel computer architectures: the construction of optimal networks having the minimum diameter (and, respectively, the optimum of transmission delays, reliability and connectivity, speed of communications and etc.) for a given number of nodes N and degree δ of a regular graph. The circulant graphs [1, 3, 4, 6, 10, 13, 17, 19], characterising by high scalability, survival and modularity, are realized as interconnection networks in multimodule supercomputer systems (MPP, Intel Paragon, Cray T3D, etc.). For degree of a graph $\delta = 4$, there exists an analytical solution for synthesis of optimal circulants [2,4,9]. But under $\delta > 4$ and any N the problem of optimal (suboptimal) circulants synthesis is known as NP-hard and an analytical approach to its solution meets certain difficulties. Heuristic algorithms are known for synthesis of suboptimal loop networks [1, 17] but diameters of graphs obtained are distinguished considerably from their exact lower bounds. The algorithms of search [5, 10] do not give solutions for large N and n, so it is necessary to develop new effective methods. Genetic algorithms were successfully used for solution of a number of problems in combinatorial optimization, artificial neural network learning,

* This work is partially supported by RFBR grant N97-01-00884

R. Poli et al. (Eds.): EvoIASP'99 and EuroEcTel'99, LNCS 1596, pp. 215–223, 1999.

graph theory, modelling evolution and Cayley graphs degree/diameter problem [7, 14, 15, 18]. The paper presents an application of genetic algorithm based on the simulation of natural evolution process for the synthesis of multidimensional circulants.

2 Optimal Circulants

Definition 1. *A circulant network is the graph* $G(N; s_1, s_2, \ldots, s_n)$ *with* N *nodes (*N *is an order of graph), labelled as* $0, 1, 2, \ldots, N-1$*, having* $i \pm s_1, i \pm s_2, \ldots, i \pm s_n$ *(mod* N*) nodes adjacent to each node* i*.*

The numbers $S = (s_i)$ $(0 < s_1 < s_2 < \ldots < s_n < (N+1)/2)$ are the generator set of the finite Abelian automorphism group associated to a graph. The degree of a node in an undirected graph G $\delta = 2n$, where n is dimension of this graph. The graph $G(N; 1, s_2, \ldots, s_n)$ when $s_1 = 1$ is known as a loop network [1, 3, 5, 6, 13, 16]. The optimization problem is to find a graph with the minimum diameter (and, possibly, with the minimum mean distance) among all possible circulants $(C(N, n))$ having N nodes and dimension n. The diameter of G is defined by $d = \max_{ij} d_{ij}$, where d_{ij} is the length of a shortest path from a node i to a node j. The average distance of G is $\bar{d} = \frac{1}{N(N-1)} \sum_{ij} d_{ij}$. Let $d(N) = \min_S \{d(G(N; S))\}$. Let for any graph $G \in C(N, n)$, $K_{n,m}$ denote the number of nodes that are reachable by at most m steps from the node 0, $K^*_{n,m}$ being the upper bound for $K_{n,m}$. Denote $L_{n,m} = K_{n,m} - K_{n,m-1}$, $L^*_{n,m}$ being the upper bound for $L_{n,m}$. The values of $K^*_{n,m}$, $L^*_{n,m}$ for any n, m were determined in [4, 8, 17].

In the literature, a different sense is applied to a term "optimal". For example, in [3, 6, 13, 16] a graph is optimal if $d(G) = d(N)$, and it is tight optimal if $d(G) = ulb(N)$ (exact lower bound for $d(N)$). We will use the following terminology.

Definition 2. *A graph* $G \in C(N, n)$ *is limit optimal, if* $L_{n,m} = L^*_{n,m}$ *for any* $0 \le m \le d^* - 1$ *and* $L_{n,d^*} = N - K^*_{n,d^*-1}$*, where the diameter* $d^* = ulb(N)$ *is given from the correlation* $K^*_{n,d^*-1} < N \le K^*_{n,d^*}$*.*

Limit optimal graphs achieve the exact lower bounds for the diameter and average distance, they have the maximum connectivity among all graphs from $C(N, n)$ [4,8] and the minimum number of steps for realization of communication algorithms [11] but exist not for all values of N and $n > 2$ [10].

Definition 3. *A graph* G *is optimal if* $d(G) = ulb(N)$*.*

Definition 4. *A graph* G *is suboptimal if* $d(G) = ulb(N) + 1$*.*

The diameters of optimal circulants are computed from the expression for $K^*_{n,m}$. In the case when $n = 2$ the exact lower bound for $d(N)$ should be $ulb(N) = \lceil (\sqrt{2N-1} - 1)/2 \rceil$ [2, 4, 9]. In [2, 4, 9] it was shown that for any $N > 4$ optimal graph $G(N; s_1, s_2)$ exists for the values of $s_1 = ulb(N)$, $s_2 = ulb(N) + 1$. The

example of limit optimal circulant with a given description is shown in Fig. 1. For $n = 2$, the necessary and sufficient conditions of existence of limit optimal double-loop networks [12], conditions of existence of optimal ones [3, 10, 13] were determined. For $n > 2$ the question about existence for any N at least of suboptimal graphs remains open. In [5] for loop networks with degree 6 and $N \leq 1237$ optimal and suboptimal graphs and their descriptions are obtained.

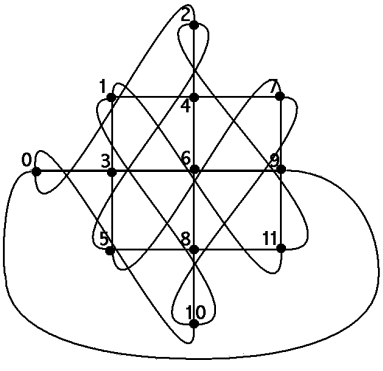


Fig. 1. Circulant graph G(12;2,3) shown as a lattice

3 Two Algorithms for Finding a Graph Diameter

The basic part of computations under synthesis of optimal networks is required for finding a graph diameter. Two different algorithms for finding a diameter were realized in the complex of genetic algorithms programs.

3.1 The First Algorithm for Finding a Diameter

Let N nodes of a tested graph G with numbers $0, 1, ..., N-1$ be cycling in a loop. An element $h[i]$ of array h will contain a distance from node 0 to node i. At the beginning of computations $h[0] = 0$, $h[i] = -1$ for all other i. Then, there follows a loop in which the paths are formed from node 0 to all nodes, the lengths of these paths and the number of nodes which may be reached from the node 0 at given step of loop are computed (Fig.2, left). And it goes on until there are the nodes in which a path can be built. In addition it is defined whether the graph G is connected or not. In the latter case a diameter of G gets the value equal to infinity. As soon as a step of the loop exceeds the value d^* under search of optimal circulants (or $d^* + 1$ for suboptimal) the loop is finished.

3.2 The Second Algorithm for Finding a Diameter

This algorithm is a generalization of the algorithm proposed in [16] for two-dimensional circulants and it is based on the constructive method of synthesis [9, 12, 16, 17], using the geometrical visualization and generating the constructions of optimal circulants (Fig.2, right). Consider the algorithm for circulants with

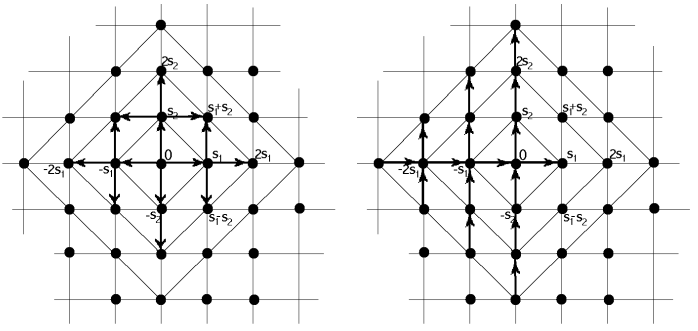


Fig. 2. Scanning nodes for first (left) and second (right) algorithms for finding of diameter

degree 6. A circulant $G(N;S)$ may be constructed as a octahedron-similar frame of lattice of unit cubes in $\mathbf{Z^3}$ in the following way. Label each lattice point (i,j,k) by number $m = (s_1 i + s_2 j + s_3 k) \pmod N$, m being the number of graph node. As a result every label $0 \le m \le N-1$ is repeated in the space infinitely many times, resulting in a tessellation of octahedron-similar constructions of $\mathbf{Z^3}$. All N node labels of optimal graphs with $d = d^*$ must lie inside a octahedron with a semidiagonal equal to d^* (for suboptimal graphs d is increased by a unit). So we scan the coordinates of lattice points in the following way: $i \in [-d, d]$ and, correspondingly, $j \in [-(d-|i|), d-|i|]$, and $k \in [-(d-|i|-|j|), d-|i|-|j|]$, and label obtained numbers of nodes. If the number of labelled (uncoinciding) nodes equals N then the optimal description is found.

3.3 The Results of Execution

As it was shown by the computer realization the improvement of the first algorithm for finding a diameter as compared to the second one is of a factor 2–2.7 in execution time. The results of genetic algorithms execution presented below in Tables were obtained under the realization of the first version of obtaining a diameter.

4 Genetic Algorithm for the Synthesis

The genetic algorithm is based on simulation of the survival of the fittest in the population of entities each of which presents a point in space of solutions of optimization graph problem. The entities are presented by strings of genes (generator sets). The function $\mathcal{F}$ named fitness function evaluates the degree of approximation of a graph diameter to its exact lower bound. The purpose of genetic algorithm is the search of global minimum of $\mathcal{F}$ when the initial structure of population is given through applying to it the genetic operators: selection, crossover, mutation.

4.1 The Structure of Data

While searching the optimum, we use two populations: the old and new one. The old population is produced on previous iteration (for the first iteration it is filled with randomly chosen generator sets) and is used for filling new population. Under the synthesis of optimal descriptions for N, changing in some range, the generator sets that were the best ones for value of N are used as the first population for value of $N + 1$.

The parameters of genetic algorithm are: N and n are the order and the dimension of a graph, M is the number of graphs in population, i_M is the number of iterations, p_m is the probability of mutation, p_c is the probability of applying crossover to pair of entities.

Each population is the set of generator sets for given N and n. Each generator set is represented by an vector of length $n + 1$. The genes contain the integers from 1 to $[N/2]$, the $[n + 1]$- term describes the diameter of a graph. The minimization of fitness function means the minimization of diameter. The iterations are finished if the best value of $\mathcal{F}$ is distinguished at most on one unit from the exact lower bound of a diameter or after a given number of steps.

4.2 The Mutation

The mutation is applied to randomly chosen generators in all set of generators of the current population (their number is determined by p_m) and produces new ones. The mutations of two types are applied: 1) a replacement of each chosen generator with random number from 1 to $[N/2]$ and 2) a replacement of a generator with random number from some neighborhood of replaced generator.

4.3 The Crossover

We apply the single point crossover to two generator sets and get two new ones. Some arbitrary pairs are chosen with probability p_c from population consisting of M graphs. In every pair the graph generators are partitioned into two parts in randomly chosen place and they exchange the parts between each other (see Fig. 3).

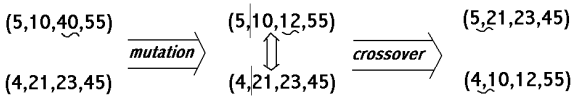


Fig. 3. Example of mutation and crossover

4.4 The Selection

The selection realizes the principle of the survival of the fittest. It is applied to the old population as a whole. The diameters of all graphs in population are computed. The selection consists in sorting the population on significance of a graph diameter and in copying several best generator sets to the new population which then is filled out by using crossover and mutation. The graphs with smaller diameters are remembered separately in order not lose them under mutation or crossover. The graphs with most diameters are displaced by the graphs with smaller diameters.

5 Experimental Results

In the Tables 1 and 2 some results of execution of genetic algorithm (GA) for the synthesis of descriptions of selected graphs are presented for degree 6 (Table 1) and for larger degrees (Table 2). The descriptions of optimal (suboptimal) graphs are represented: N is the number of nodes of a graph, s_i is the generator, d is the diameter of a graph, d^* is exact lower bound for the diameter. Under execution of genetic algorithm the following values of parameters were used: $p_m = 0.18$, $p_c = 0.5$, $M = 200$, i_M=3000.

Table 1. The fragment of catalogue of circulants with $\delta = 6$

N	$d(d^*)$	s_1	s_2	s_3	N	$d(d^*)$	s_1	s_2	s_3
1561	11(10)	43	645	650	1562	11(11)	130	301	350
2047	12(11)	19	575	974	2048	12(12)	237	464	541
2625	13(12)	1096	1244	1283	2626	13(13)	65	239	562
3303	14(13)	46	238	651	3304	14(14)	229	1047	1182
4089	15(14)	133	1309	1617	4090	15(15)	347	1130	1240
11521	22(20)	823	5135	5137	11522	22(21)	2129	2626	4003
13287	23(21)	1138	1586	6042	15225	24(22)	105	337	1247

Table 2. The selected circulants with different degrees

N	δ	$d(d^*)$	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
2236	8	8(7)	57	92	248	607				
1024	10	6(5)	49	64	367	462	476			
4096	12	7(6)	321	753	836	1380	1893	1990		
19825	12	9(7)	92	456	1735	3952	5366	6701		
16384	14	8(6)	490	1277	1645	2512	3832	4317	5448	
55000	16	8(7)	855	4380	7897	10132	20068	20175	22429	25671

We compared the genetic algorithm with the reduced search (RS) [12] and the random search (RA) algorithms. The measurements of arithmetic mean of execution time $t_{av}(RS)$, $t_{av}(GA)$ and $t_{av}(RA)$ in intervals $N_1 \leq N \leq N_2$ for

Table 3. Comparison between the RS, GA and RA of circulant synthesis

graph	δ	$N_1 - N_2$	$d(d^*)$	$t_{av}(RS)$	$t_{av}(GA)$	$t_{av}(RA)$
subopt	6	1562-1571	12(11)	0.6	0.5	0.5
		1800-1809	12(11)	1.3	2.7	0.7
		2048-2057	13(12)	1.2	0.8	0.2
		2616-2625	13(12)	476.3	198.3	147.7
subopt	8	682-691	7(6)	0.9	0.3	0.3
		980-989	7(6)	9.2	0.5	0.4
		1760-1769	8(7)	58.0	1.1	0.8
optimal	6	2048-2057	12(12)	568	23.3	21.8
		3304-3313	14(14)	2335	196.6	497.6
		4090-4099	15(15)	3867	1633	3705
optimal	8	1300	7(7)	–	8.0	15.0

obtaining the first encountered suboptimal (or optimal) description are presented in Table 3 (in seconds, Pentium-166 MMX).

These valuations were determined for different values of N including the beginning, the middle and the end of ranges for a given diameter. The choice of intervals is explained by the fact that a probability of existence of optimal (suboptimal) descriptions for circulants are decreased under increase of N for a given d. The minimum probability is observed at the bounds of transitions from a diameter to another one. The execution time of algorithms is considerably increased in these points on comparison with the beginning of range. The results show that the use of GA becomes preferable for obtaining optimal circulants under the increase of a diameter and a dimension. Under above-mentioned parameters genetic algorithm allowed to synthesize the whole continuous ranges of suboptimal (in most cases) or optimal circulant graphs with degrees $\delta = 6, 8, 10, 12$ and orders, respectively, $N \leq 3300, 3400, 3500, 3600$. The considered algorithm allows to synthesize rapidly suboptimal (or with the diameter differing two units from its exact lower bound) graphs for large degrees and orders. The explicit catalogues of optimal and suboptimal three-dimensional circulants and graphs with larger degrees are represented at Web-page: - $http://rav.sscc.ru/\sim emilia/$.

Fig. 4 shows the obtained number of suboptimal graphs vs number of mutations and crossovers for given $i_M = 1000$, $N = 2300$, $M = 100$ $n = 3$. The maximum number of the suboptimal graphs corresponds to application of the crossover operator to each graph of the population (cross_every=1).

6 Conclusion

The results of research of genetic algorithm application to the synthesis of optimal circulant network designs are presented. The genetic algorithm and random search are suitable for perspective areas search in spaces of solutions of considered optimization problem. Their efficiency depends on probability of existing the desired graphs for a given degree and order. Under increasing dimension and order of a graph the appropriateness of using genetic algorithm is increased.

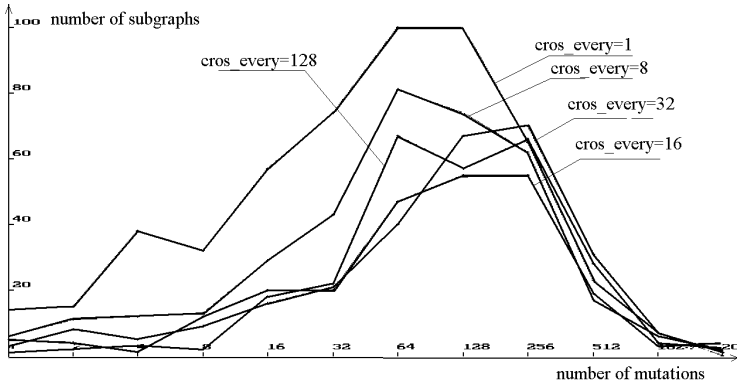


Fig. 4. Number of suboptimal graphs vs number of mutations and crossovers

The genetic algorithm allowed to get the descriptions of optimal (suboptimal) circulants for such values of N and δ which were inaccessible earlier for heuristic [1, 17] and exhaustive search [5, 10] algorithms. The following stage of advance in solution of the optimization problem for large degrees and orders is connected with researches developed in combination of genetic algorithms and more powerful heuristics and elaboration of parallel versions of the algorithms.

References

1. J.-C. Bermond, F. Comellas and D.F. Hsu, *Distributed loop computer networks: a survey*, J. Parallel Distributed Comput., **24**, 1995, 2–10.
2. J.-C. Bermond, G. Illiades and C. Peyrat, *An optimization problem in distributed loop computer networks*, Third International Conference on Combinatorial Math. New York, USA, June 1985, Ann. New York Acad. Sci., **555**, 1989, 45–55.
3. J.-C. Bermond and D. Tzvieli, *Minimal diameter double-loop networks: Dense optimal families*, Networks, **21**, 1991, 1–9.
4. F.T. Boesch and J.-F. Wang, *Reliable circulant networks with minimum transmission delay*, IEEE Trans. Circuits Syst., **CAS-32**, 1985, 1286–1291.
5. S. Chen and X.-D. Jia, *Undirected loop networks*, Networks, **23**, 1993, 257–260.
6. D.-Z. Du, D.F. Hsu, Q. Li and J. Xu, *A combinatorial problem related to distributed loop networks*. Networks, **20**, 1990, 173–180.
7. D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, 1989.
8. V.V. Korneyev, *Macrostructure of homogeneous computer systems*, Vychislitelnye sistemy, **60**, Novosibirsk, 1974, 17–34 (in Russian).
9. E.A. Monakhova, *On analytical representation of optimal two-dimensional D_n-structures of homogeneous computer systems*, Vychislitelnye sistemy, **90**, Novosibirsk, 1981, 81–91 (in Russian).
10. E.A. Monakhova, *Optimal circulant computer networks*, Proc. International Conference on Parallel Computing Technologies, PaCT-91, Novosibirsk, USSR, 1991, 450–458.

11. E.A. Monakhova, O.G. Monakhov, *Collective exchanges in circulant networks of parallel computer systems*, Optoelectronics, Instrumentation and Data Processing, New York, **6**, 1997, 91–106.
12. E. A. Monakhova, O. G. Monakhov, V. V. Korneyev. *Methods and algorithms for synthesis of optimal circulant structures of computer systems*, Proc. of the Sixth Inter. Workshop on Distributed Data Processing (DDP-98), Novosibirsk, Russia, 1998, 87–90 (in Russian).
13. K. Mukhopadhyaya and B.P. Sinha, *Fault-tolerant routing in distributed loop networks*, IEEE Trans. Comput., **44**, No. 12, 1995, 1452–1456.
14. M. Sampels, *Large Networks with Small Diameter*, Proc. of the Inter. Workshop on Graph Theoretic Concepts in Computer Science (WG'97), Berlin, 1997, 288—302.
15. H.-P. Schwefel, T. Baeck, *Artificial evolution: How and why?*, Genetic Algorithms and Evolution Strategy in Engineering and Computer Science - Recent advances and industrial applications. Wiley, Chichester, 1997, 1–19.
16. D. Tzvieli, *Minimal diameter double-loop networks. 1. Large infinite optimal families*, Networks, **21**, 1991, 387–415.
17. C.K. Wong and Don Coppersmith, *A combinatorial problem related to multimodule memory organizations*, J.Assoc. Comput. Mach., **21**, 1974, 392–402.
18. X. Yao, *Global Optimization by Evolutionary Algorithms*, Proc. of The Second Aizu Inter. Symposium on Parallel Algorithms/ Architecture Synthesis, Japan, IEEE Press, 1997, 282–291.
19. J. Zerovnik, T. Pisanski, *Computing the diameter in multiple-loop networks,* J. Algorithms, **14**,1993, 226-243.

Author Index