Genetic Algorithms for Ambiguous Labelling Problems

Richard Oliver Myers, M.Sc.

Submitted for the degree of Doctor of Philosophy

Department of Computer Science

THE UNIVERSITY of York

March, 1999

Abstract

This thesis develops an optimisation framework within which the principle of least commitment can be applied to the solution of ambiguous consistent labelling problems. The natural optimisation method for such an approach is a hybrid genetic algorithm. The starting point is the refinement of an existing Bayesian approach to inexact labelling problems. Two criteria are presented which measure consistency in terms of the edit distance between labelled configurations and a dictionary of legal labellings. The new criteria have significantly lower time and space requirements than the original in the worst case.

The genetic algorithm is applied to labelling problems, and its feasibility for ambiguous labelling is demonstrated. The behaviour of the algorithm when labelling ambiguous and impossible line drawings is studied. Adapting the algorithm for ambiguous graph matching problems necessitates a reformulation of the posterior measurement probability to take measurement ambiguity into account. The hybrid genetic algorithm with a gradient ascent step is found to offer the best combination of optimisation performance and solution yield. The hybrid algorithm is a significantly better optimiser than gradient ascent or the standard genetic algorithm alone. The hybrid offers higher solution yields than other evolutionary algorithms.

The setting of the several parameters which control the genetic algorithm is studied using sets of factorial experiments. Empirical models for both labelling problems are derived. These models are used to find optimal conditions for the algorithm, and are found to be stable under extrapolation to problems up to three times as large as those in the original test set. The smaller than expected population size required for successful optimisation suggests that the gradient ascent step is responsible for the optimisation performance of the algorithm, and that the genetic algorithm operators work to furnish good initial guesses for gradient ascent.

Three measures of the algorithm's population diversity are presented. They are the Shannon entropy, the total inter-cluster Hamming distance and the size of the gene pool. These measures indicate that population diversity decreases rapidly over the first few iterations, and inexorably thereafter. It is shown that solution yield can be improved without the introduction of any new parameters to the algorithm. Since the gradient ascent step appears to be mainly responsible for the optimisation performance, selecting the next generation at random without replacement increases diversity without compromising search.

Contents

List of Figures

List of Tables

Abbreviations

ANOVA	analysis of variance
ARG	attributed relational graph
CHC	genetic algorithm variant (Eshelman 1991)
COST	labelling criterion
CROSS	crossover type
GA	genetic algorithm
GEN	genetic algorithm iteration limit
GENITOR	genetic algorithm variant (Whitley 1989)
HUX	half-uniform crossover
LINES	number of lines in a drawing
MAP	maximum *a posteriori* probability
NODES	number of nodes in a graph
POP	genetic algorithm population size

Symbols

β	a scale parameter, or inverse temperature
Γ	a labelling
δ	an elementary edit transformation
η	linear predictor
Θ_j	a dictionary of legal labellings for neighbourhood j
$\mathbf{\Lambda}$	a set of labels
ϕ	the null label
$\mathbf{\Psi}$	a bag (population) of labellings
$\mathbf{C}$	a set of "constraint neighbourhoods"
D	a distance measure
Fc	fraction of mappings correct
f	a match between two graphs
f_i, f_Γ	raw fitness of an individual, or a labelling
G	an attributed relational graph
g	size of gene pool
H_T	total inter-cluster Hamming distance
P_e	label error probability
P_m	mutation probability
P_x	crossover probability
$\Re$	the set of real numbers
S	Shannon entropy
S_i	a dictionary item
$\mathbf{V}$	a set of objects
$\left(\begin{array}{c} n \\ r \end{array} \right)$	binomial coefficient

Acknowledgments

Firstly I would like to thank my supervisor, Prof. Edwin Hancock, for his support, patience and guidance over the past three and a half years. I would also like to thank Prof. Jim Austin for his assessments of my work.

Thanks also go to EPSRC for the funding without which I would not have been able to undertake this work. I also wish to express my gratitude to the British Council, JISTEC and the NEC corporation for giving me the opportunity to practice what I preach at NEC's "Incubation Center" in Kawasaki, Japan for two months in the Summer of 1997.

I would also like to thank Dr. Andrew Cross for getting me started with genetic algorithms and labelling problems, and Dr. Richard Wilson for helping me become acquainted with graph matching. Benoit Huet's knowledge of Unix proved invaluable while I was getting to grips with the departmental computer system. Simon Moss provided a number of useful mathematical hints and tips.

My parents have given me tremendous support over the years, and I thank my father, Dr. David Myers, in particular for his general advice on experimental design and analysis techniques.

Last, but by no means least, I would like to thank my wife, Daisy, for the best things in life.

Declaration

I declare that all the work in this thesis is solely my own, except where attributed and cited to another author. Some of the material in the following chapters has been published previously: a list of the publications is given in appendix B.

Chapter 1

Introduction

David Marr proposed a simple experiment to distinguish between perception and cognition (Marr 1982). Consider the richness of the visual world around you in terms of light, form, colour and texture. Close your eyes and the richness disappears, open them again and it returns. Marr suggested that this phenomenon occurs because the internal representation of the world is an abstraction which contains much less information than is actually available from the scene. The task of mid-level vision is to bridge the gap between the perceptually rich world of the scene and the cognitively rich world of high-level internal representations. The behaviour of the real world is stereotyped to quite a large extent. Solid matter tends to be collected into distinct objects, which occupy space and tend to have differing surface properties. There may be a limited number of ways in which these objects can be juxtaposed. For example, it is usual for houses to be on the ground, and for the sky to be above the grass, and for the grass to be green, and so-on. The importance of such syntactic constraints is that they allow parts of a scene to be *labelled* without the need for detailed knowledge of their physical properties. A labelling of a scene is effectively an interpretation, and is thus one way of linking low-level percepts with high-level concepts.

Historically, a major problem with syntactic labelling schemes has been the large number of possible interpretations of a given scene. This can be addressed by augmenting the labelling process with evidence from the scene, or with *a priori* knowledge of the likely interpretation. When appropriate, early disambiguation can greatly simplify the labelling problem. However, inappropriate disambiguation at too early a stage risks making an expensive wrong decision. Marr's principle of least commitment states that such decisions should be postponed as long as possible (Marr 1982). This thesis argues that the labelling

algorithm should perform as little disambiguation as possible, serving up a variety of plausible interpretations to higher level processes.

This thesis studies the use of genetic algorithms to extract multiple solutions to ambiguous labelling problems. The genetic algorithm is a global optimisation method inspired by Darwinian evolution (Holland 1975). The idea is that better solutions to a problem are likely to evolve and survive when a population of potential solutions is subjected to natural selection. This thesis takes the view that using a genetic algorithm as a framework for labelling would allow a vision system to follow the principle of least commitment. High-level processes could select the best solutions from the genetic algorithm's population. To realise this goal it will be necessary not only to establish the capability of the algorithm as a labelling technique, but also to demonstrate that it can produce sufficiently many good quality labellings. Two important labelling problems are considered. Line labelling applies geometric constraints to interpret line drawings as three-dimensional objects. Graph matching applies structural constraints to compare triangulations of point sets.

Chapter 2 surveys the literature on consistent labelling, ambiguity and genetic algorithms. A method due to Hancock and Kittler (Hancock and Kittler 1990a) for solving the consistent labelling problem by optimisation is given in chapter 3, which further develops the method and considers efficient implementations. Chapter 4 uses this method in a genetic algorithm framework to solve ambiguous line labelling and graph matching problems. The graph matching framework is augmented with a consideration of the ambiguous measurements on extracted features, and by the introduction of metric based crossovers. Chapter 5 presents a large scale factorial study of genetic algorithm control parameters. Empirical models which can be used to set parameters for a given problem size are obtained. Chapter 6 considers ways in which the number of good solutions recovered by the algorithm can be increased. Finally, chapter 7 presents some conclusions and suggests some future directions for research. A short paper on the application of the genetic algorithm to a more difficult interpretation problem appears as appendix A.

Chapter 2

Literature Review

This chapter surveys the literature on consistent labelling, ambiguity, and genetic algorithms. Consistent labelling is a constraint satisfaction problem which has been studied extensively in the computer vision literature (Mackworth 1977; Haralick and Shapiro 1979; Haralick and Shapiro 1980; Haralick et al. 1978). Effective methods of solving this problem have been developed over the last 25 years. The prototypical consistent labelling problem is line labelling, which has been of interest to machine vision researchers for about 30 years (Huffman 1971; Clowes 1971; Waltz 1975). A more significant consistent labelling problem is graph matching, which occurs frequently in computer vision applications (Barrow and Popplestone 1971; Ballard and Brown 1982; Koenderink and van Doorn 1979; Dickinson et al. 1992; Wilson 1995; Cross 1998). It is usual to solve consistent labelling by optimising some global measure of consistency (Hummel and Zucker 1983; Faugeras and Berthod 1981; Hancock and Kittler 1990a). Gradient ascent is the method of choice where appropriate, but relies on an initial guess being close to the global optimum. More robust, global optimisation techniques must be used when no such initial guess can be made.

The fact that there are many optimal solutions to typical line labelling problems has been recognised since the mid 1970s (Waltz 1975). However, the difficulty of using global contextual information in scenes to resolve them has led researchers away from this topic. The focus has been on disambiguating early and arriving at a single solution to the problem (Hummel and Zucker 1983; Faugeras and Berthod 1981). It is unclear how such methods would cope if the interpretation were genuinely ambiguous. There has been some interest recently in applying the principle of least commitment in these situations (Callari and

Ferrie 1996; Ezquerra et al. 1998), but no general framework for this has been proposed.

Genetic algorithms (Holland 1975; Goldberg 1989) have been found to be good global optimisers. However, the algorithm is very complex, and despite a massive literature, there has been relatively little theoretical or empirical work of significant practical value which is generally applicable. The algorithm is potentially useful for problems with many solutions because it maintains a population, as opposed to considering a single solution (Goldberg and Richardson 1987). Genetic algorithms have been used for such "multimodal optimisation" with some success. There have been a few applications of the algorithm to labelling problems (Fleurent and Ferland 1996; Cross 1998), but none have exploited the potential for multimodal optimisation of ambiguous labelling problems.

The next section considers consistent labelling, and gives line labelling and graph matching as examples. Section 2.2 considers ambiguity in vision. The final section reviews some of the vast literature on genetic algorithms.

2.1 Consistent Labelling

Constraint satisfaction problems are among the most widely studied in computer science. The problem is to make an assignment from some domain to a set of variables, subject to constraints between subsets of the variables. If the domain of assignment is discrete, the problem is equivalently one of "labelling" each of the variables in the set. If there are only two labels, the consistent labelling problem is equivalent to the general satisfiability problem (Nudel 1983; Garey and Johnson 1979). It is possible to transform any consistent labelling problem so that the constraints only apply to pairs of variables (Nudel 1983). Consistent labelling problems in which the constraints are binary were studied intensively in the computer vision literature of the mid to late 1970s (Mackworth 1977; Haralick and Shapiro 1979; Haralick and Shapiro 1980). Examples of binary consistent labelling problems include graph colouring, subgraph isomorphism, the Boolean satisfiability problem and scene labelling. The problem is known to be NP-complete in its general form (Cook 1971; Haralick et al. 1978), but a number of special cases exist. Indeed, finding tractable subproblems of satisfiability is an active area of research in complexity theory, for example see (Jeavons 1998).

In the context of computer vision, both Mackworth, and Haralick and Shapiro indepen-

dently formulated consistent labelling in terms of network consistency (Mackworth 1977; Haralick and Shapiro 1979; Haralick and Shapiro 1980). The variables to be labelled formed the nodes in a graph, and the edges of the graph represented binary constraints. Suppose that such a graph has n nodes, e edges and a labels. Mackworth and Freuder pointed out in (Mackworth 1977) and (Mackworth and Freuder 1985) that depth first search with backtracking, which was the standard method for such problems, has a worst case complexity of $O(ea^n)$. However, they showed that it is possible to effectively reduce the number of labels, a, by a preprocessing step in which inconsistent assignments are screened out. For binary consistent labelling problems, it is possible to achieve "arc consistency", i.e. to remove from the search space all labels which would violate the binary constraints, in $O(a^3n)$ time, as long as the problem can be represented by a planar graph. Mackworth and Freuder showed that Waltz's seminal contribution to line labelling, made several years earlier in (Waltz 1975), is just such a case.

Given time, search will enumerate the entire solution space for exact problems, but is of little use when no globally consistent solution exists. This is the case with inexact matching, and analysis of "impossible" scenes, situations which can arise when input is corrupted or incomplete. Furthermore, neither search nor arc consistency use evidence derived from the input, relying merely on pre-defined constraint relations. To overcome these problems, Rosenfeld, Hummel and Zucker formulated probabilistic relaxation in (Rosenfeld et al. 1976). However the convergence properties of their algorithm are not easily characterised (Kittler and Illingworth 1985). In (Hummel and Zucker 1983), Hummel and Zucker adopted a more information-theoretic approach in their reformulation of relaxation as an optimisation problem. Faugeras and Berthod developed a framework for labelling in which a measure of local ambiguity was minimised as consistency was maximised (Faugeras and Berthod 1981).

Much of the work involving consistent labelling has adopted Hummel and Zucker's optimisation paradigm (Faugeras and Berthod 1981; Lloyd 1983; Mohammed et al. 1983; Wilson and Hancock 1997). The problem is to find a set of label assignments which optimises some global consistency measure (Hummel and Zucker 1983). This is typically done by iteratively applying a local operator to the solution until no further improvement can be made. The most straightforward optimisation technique is gradient ascent, in which the update operator is required to monotonically increase the quality of the solution: this was the approach used by Hummel and Zucker in their original work (Hummel and Zucker

1983), and by many others (Hancock and Kittler 1990a; Faugeras and Berthod 1981; Lloyd 1983; Mohammed et al. 1983; Wilson and Hancock 1997). Gradient ascent is appropriate when an initial guess can be made, which is close to the final solution in the sense that there are no intervening local optima. This is not usually the case, so it is often preferable to use global optimisation techniques such as simulated annealing (Kirkpatrick et al. 1983; Geman and Geman 1984), mean field annealing (Geiger and Girosi 1991; Yuille and Kosowsky 1994) or genetic search (Holland 1975).

Hancock and Kittler have built on the work of Faugeras and Berthod (Faugeras and Berthod 1981) and Hummel and Zucker (Hummel and Zucker 1983) by developing a Bayesian framework for measuring consistency (Hancock and Kittler 1990a). This framework can be applied to many image analysis tasks, including pixel labelling, edge detection, line labelling and graph matching. The input is assumed to have been the result of the action of noise on an initially perfectly labelled scene, and the problem is to recover the original labelling. The framework uses an explicit dictionary representation of constraints, as adopted by Waltz, in conjunction with a Bayesian model of the label corruption process. Hancock and Kittler took the dictionary to be a quite general constraint representation: it is an exhaustive compilation of the consistent labellings of conveniently sized subunits of a scene. The scene corruption model is based on the premise that consistent dictionary items are subject to the action of a memoryless label corruption process, which yields the current label configuration. Modelling this corruption mechanism results in an expression for the probability of a particular labelling, which gauges consistency by an exponential function of the Hamming distances between configurations and dictionary items. This probability can be combined naturally with measurements made on the scene to allow both statistical and structural considerations to influence the labelling process. Scene interpretation is achieved by finding the label configuration which optimises the combined probability criterion. This was originally done in (Hancock and Kittler 1990a) by gradient ascent, but global optimisation techniques have also been successfully applied (Wilson 1995; Cross 1998). This approach has been applied to scene labelling (Hancock and Kittler 1990a), edge detection (Hancock and Kittler 1990b), graph matching (Wilson and Hancock 1997), and line labelling (Hancock 1994).

2.1.1 Line Labelling

Line drawing interpretation has been an active area of investigation in machine vision for over twenty-five years. Historically, one of its rôles has been to stimulate work in the area of consistent labelling via the development of relaxation techniques. In fact, it was the work of Huffman and Clowes on the consistent labelling of line drawings of polyhedral scenes that led Waltz to his seminal discrete relaxation algorithm (Huffman 1971; Clowes 1971; Waltz 1975). Waltz's contribution was to show how a dictionary of consistent junction labellings could be used in an efficient search for consistent interpretations of polyhedral objects.

Polyhedral scenes are composed of a variety of junction types. The junctions can be classified according to their topologies. Each line in a scene must be labelled according to whether it is the concave intersection of two planes, the convex intersection of two planes, or an occluding boundary. Waltz associated with each junction type a dictionary listing the consistent label configurations that can be assigned to junctions of that type (Waltz 1975). These dictionaries are derived from the geometric constraints on the projection of 3D scenes onto 2D planes (Huffman 1971; Clowes 1971; Sugihara 1978). Following the work of Huffman, Clowes and Waltz, Sugihara developed a grammar for skeletal polyhedra (Sugihara 1978). Malik has extended the theory to include curved surfaces (Malik 1987), and Williams has used labelled line drawings to reconstruct smooth objects (Williams 1992). Kirousis has developed several efficient algorithms for determining "labellability" and labelling (Kirousis 1990). Parodi and Piccioli have developed a method for reconstructing 3D scenes from labelled line drawings given known vanishing points (Parodi and Piccioli 1996). Hancock has applied the dictionary based Bayesian framework to labelling polyhedral scenes in (Hancock 1994).

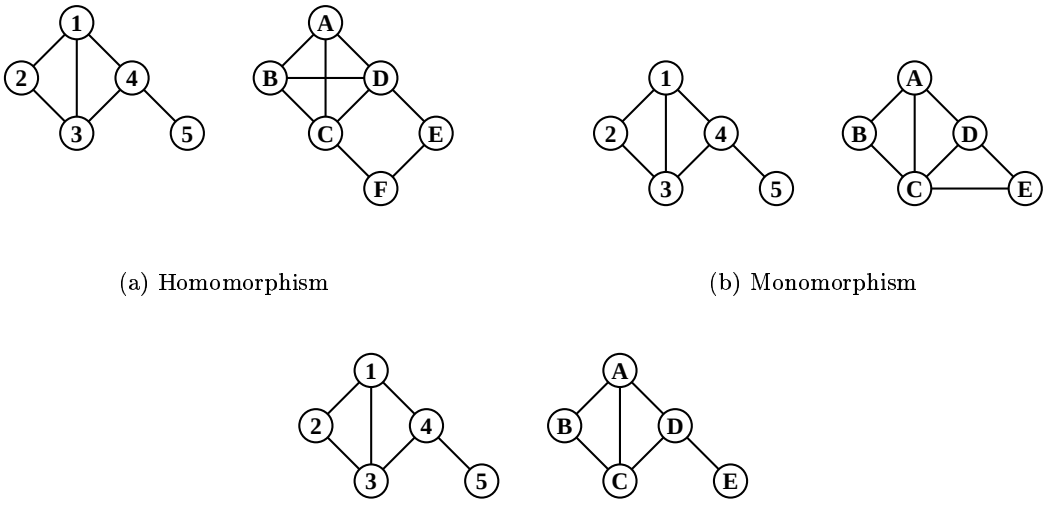
The interpretation of line drawings has applications in, among other areas, document analysis, processing architects' sketches, and automatic interpretation of engineering drawings. For example, Lipson and Shpitalni generated three dimensional object descriptions from freehand line drawings in (Lipson and Shpitalni 1996).

2.1.2 Graph Matching

Like line labelling, graph matching has a long history in computer vision. Graphs have been used as representations since the early 1970s, when Barrow and Popplestone used them in (Barrow and Popplestone 1971) to represent spatial relationships between scene components. Much of the early work on graphs in computer vision focused on the use of semantic networks, since they could simultaneously model the scene and be used to make inferences (Ballard and Brown 1982). The idea of making associations between objects explicit in the representation was also psychologically plausible (Bartlett 1932; Sowa 1984). Minsky suggested a hierarchical representation with image features at the bottom level and scene knowledge at the top, which he called "frames" (Minsky 1975). However, trying to cram too much knowledge into such representations had its disadvantages (Minsky 1975; Ballard and Brown 1982), and the trend since the early 1980s has been to use more "lightweight" representations such as the aspect graphs of Koenderink and van Doorn (Koenderink and van Doorn 1979), or part-primitive hierarchies. An interesting combination of these two representations was used by Dickinson and co-workers in (Dickinson et al. 1992). As the goals of computer vision research have become less lofty, the relational representations used for scene objects have become more syntactic than semantic. A modern graphical representation of a scene might be constructed purely from geometric properties of extracted features, for example the Delaunay triangulations used by Wilson in (Wilson 1995). A variety of other triangulations were considered by Tuceryan and Chorzempa in (Tuceryan and Chorzempa 1991).

Graph matching problems arise whenever relational descriptions need to be compared. Indeed, Barrow and Popplestone's original work involved matching the relational descriptions of scenes to semantic models (Barrow and Popplestone 1971). In (Shaprio and Haralick 1981), Shapiro and Haralick give the taxonomy of matching problems shown in figure 2.1.

The homomorphism problem arises when the graphs are similar but not identical. This is common in real world matching problems, and is also known as inexact matching. Relational monomorphism occurs when a copy one graph is embedded in the other: this is the same as subgraph isomorphism, which is known to be NP-complete. Isomorphism, or exact matching, is not known to be NP-complete, but no polynomial time algorithm has yet been found.



(a) Homomorphism

(b) Monomorphism

(c) Isomorphism

Figure 2.1: Graph Matching Problems. This figure is a copy of that given in (Shaprio and Haralick 1981).

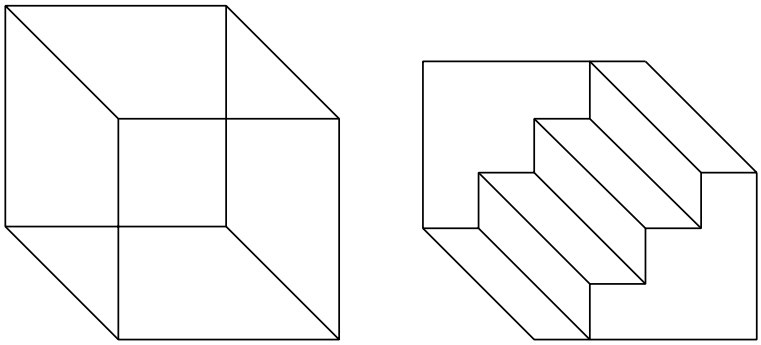
Ambler et al. proposed that inexact matching be accomplished by searching for maximal cliques in the association graph, which is a classical problem in graph theory (Ambler et al. 1975). The association graph contains nodes representing every possible mapping between each node of the two original graphs. Association graph nodes are linked if the mappings which they represent are consistent. The largest clique in the association graph thus represents the best match possible between the two original graphs. In (Ullman 1976), Ullman treated subgraph isomorphism as a constraint satisfaction problem, and applied a filtering operation similar to arc consistency.

Throughout the 1980s and 1990s, the emphasis has been on optimising measures of similarity between graphs. The proposed mappings effectively transform one of the graphs; the closer this "relational image" is to the other graph, the better the match. Various similarity measures have been proposed, based either on editing processes (Fischler and Elschlager 1973; Sanfeliu and Fu 1983) or on corruption models (Shaprio and Haralick 1985; Wilson 1995). In (Wilson 1995), Wilson adapted Hancock and Kittler's Bayesian framework (Hancock and Kittler 1990a) to derive a probabilistic measure of the quality of a match, which was optimised by gradient ascent. Cross has investigated several global optimisation approaches to this problem in (Cross 1998), including simulated annealing and genetic search.

The practical applications of graph matching are myriad, but include object recognition (Dickinson et al. 1992), feature-based stereo correspondence (Horaud and Skordas 1989), image registration (Wilson 1995), and medical imaging (Dumay et al. 1992).

2.2 Ambiguity

Ambiguities exist at all levels of visual perception. Of particular interest in this thesis are ambiguities which arise in the syntactic processing of scene elements. These ambiguities manifest themselves in two ways. First, there may be several alternative and equally good interpretations of scenes such as those shown in figure 2.2. Second, there may be no totally consistent interpretation, but several which are quite "close", as can be the case in inexact matching.



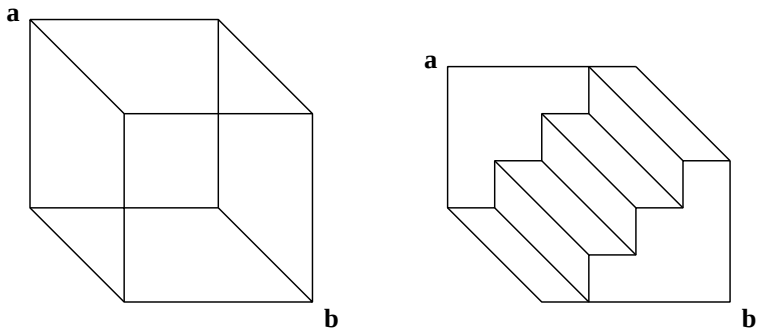
(a) Necker Cube (b) Shröder Staircase

Figure 2.2: Two Ambiguous Drawings.

Connectionist models have been used to model perceptual alternation of ambiguous visual stimuli, in which the interpretation of a drawing such as the Necker cube or Shröder staircase, shown in figure 2.2, periodically switches between several alternatives (Feldman and Ballard 1982; Masulli et al. 1990; Riani and Simonotto 1994; Bialek and Deweese 1995). Kawamoto, Masulli and coworkers, and Riani and Simonotto have all suggested that noise drives the perceptual alternation process (Kawamoto 1993; Masulli et al. 1990; Riani and Simonotto 1994). Presumably the noise allows the network to escape from local energy minima in a manner analogous to annealing. However, Bialek and Deweese have shown that the alternation rate is independent of noise, but rather depends on *a priori* hypotheses. The alternation itself is a random event and therefore accounts for the

requirement that the network be noisy (Bialek and Deweese 1995).

In (Kawabata 1978), Kawabata observed that the visual fixation point, i.e. the point to which one's gaze is directed, determines the perception of depth and alternation rates in ambiguous figures such as the Necker cube and the Shröder staircase, shown in figure 2.3. Kawabata suggested that the local interpretation at the fixation point tends to propagate to generate a stable global interpretation. This observation chimes with the selective attention hypothesis (Kawabata and Mori 1992; Horlitz and O'Leary 1993), in which *a priori* expectations combined with focussed attention lead to relatively stable unambiguous interpretations of ambiguous figures.



(a) Necker Cube (b) Shröder Staircase

Figure 2.3: Locally Unambiguous Drawings. Fixating on points a and b yields relatively stable percepts.

Calliari and Ferrie have recently described a model-based vision system which can cope with ambiguity (Callari and Ferrie 1996). The system makes a set of initial guesses which are refined by subsequent data gathering. This approach has produced promising results, and would seem to complement an active vision strategy.

Many consistent labelling problems which arise in computer vision have more than one possible solution. This was recognised in Waltz's original paper (Waltz 1975), but no strategy for handling ambiguity was developed. Ambiguity has historically been seen as a "bad thing", to be resolved as quickly as possible, rather than as a necessary part of scene interpretation. Waltz used search to extract an arbitrary solution from the set of possible solutions to his labelling problem (Waltz 1975). Faugeras and Berthod developed a direct measure of ambiguity which was minimised in their relaxation scheme (Faugeras and Berthod 1981). Hummel and Zucker used a simple definition of "unambiguous labelling"

as a *sine qua non* for their convergence proof (Hummel and Zucker 1983). Almost all of the work on consistent labelling has focussed on the robust extraction of a unique solution. The classical approach in vision has been to resolve local ambiguities at a relatively early stage to force a single interpretation and then backtrack if necessary (Faugeras and Berthod 1981). A more recent example can be found in (Qin and Luh 1994). This may be appropriate if there is compelling local evidence for a particular interpretation such as is ideally the case in probabilistic relaxation. However, backtracking is generally inefficient (Mackworth 1977; Haralick and Elliott 1979), and is only possible in search. It is not at all clear how backtracking should be implemented for deterministic labelling schemes such as gradient ascent.

Although the use of global contextual information to resolve ambiguity is a major unsolved problem in machine vision, the early commitment to a particular interpretation which occurs in most techniques does not help. In cases where there is no clear single interpretation, a vision system should have an efficient method of entertaining various options. The initial stage of interpretation should yield several possible solutions from which the system can choose without having to backtrack. This is Marr's principle of least commitment (Marr 1982). One of the few examples of a labelling application in which disambiguation is delayed is furnished by Ezquerra et al.'s system for labelling coronary angiograms in (Ezquerra et al. 1998).

2.3 Genetic Algorithms

Genetic algorithms belong to a family of stochastic global optimisation methods based on the concept of Darwinian evolution in populations. They have been proposed independently by several authors (Fraser 1957; Bremermann 1958; Reed et al. 1967; Holland 1975), but it is Holland's formulation in (Holland 1975) which is regarded as the standard. The population is composed of individuals which interact with the environment and each other over a number of algorithm iterations or "generations". Interaction with the environment takes two forms: first, mutation in which parts of the individual are changed at random, and second, selection in which individuals are selected for future reproduction based on their quality. Individuals may also interact with each other via crossover, which is analogous to genetic recombination, and in which information is exchanged between (two) "parent" individuals to form (two) "offspring" individuals. The idea is that over

time, individuals better suited to the environment will emerge, and come to dominate the population. The quality measure used is usually called the "fitness": high values of fitness correspond to better individuals. Most implementations terminate when either a specified number of iterations or fitness evaluations has taken place, or a maximally fit individual has emerged. The algorithm is controlled by the population size, terminating criterion, and mutation and crossover probabilities, in addition to crossover type (q.v.) and selection mechanism. Most of the early work on genetic algorithms was based on numerical optimisation problems (DeJong 1975; Whitley et al. 1995).

In principle, solving a particular problem with a genetic algorithm is straightforward. Evolutionary algorithms usually operate at two levels. The lower "genotypic" level is the level of encodings or "chromosomes"; the smallest semantically meaningful part of a chromosome is a "gene". At this level, crossover and mutation operate on the individuals regardless of their interpretation, and may thus be regarded as syntactic operators. The higher "phenotypic" level is the level of observed characteristics. The selection operator operates on the phenotype since it is this which determines the fitness of the individual; selection is thus a semantic operator. The only stage in the algorithm at which specific knowledge of the problem is required is the translation from genotype to phenotype, which is abstracted via the fitness function.[1] To solve a problem with a genetic algorithm, it is necessary to define a suitable encoding and fitness function for the problem, choose or define a crossover operator, decide on a selection strategy, choose an algorithm variant, and finally choose suitable values for population size, crossover rate, mutation rate, and a terminating criterion.

It is conventional to use a binary encoding for the problem (Holland 1975; Goldberg 1989; Mitchell 1996), so that the chromosomes are in fact bit strings. However, in (Michalewicz 1996), Michalewicz criticises this approach, and suggests that more natural, "real value" encodings be used.

Crossover is the recombination step: parts of two chromosomes are exchanged. There are two ways in which this occurs. First, genes may be swapped on an individual basis or

[1] In biology, genotype and phenotype are very different. The phenotype arises from complex interactions between the individual's genetic makeup and the environment, and the relationship between genotypes and phenotypes is typically many to many. In a genetic algorithm, however, this relationship is usually one to one, so the distinction between genotype and phenotype may appear arbitrary. It is common for the same entity to be referred to as an "individual" at the phenotypic level, and as a "chromosome" at the genotypic level.

in contiguous runs. The most popular example of the first method is Syswerda's uniform crossover in which bits are exchanged on a random basis (Syswerda 1989). The other kind of crossover is multipoint, where homologous points on the chromosomes are chosen and the substrings between them exchanged. The classical example is two point crossover (DeJong and Spears 1990). Uniform crossover tends to be more disruptive and therefore more exploratory than multipoint. Multipoint crossover is appropriate when there are large "areas of optimality" on the chromosomes: in this case, the offspring ought to resemble the parents.

Mutation is a background operator, which introduces new information to the population. It can be seen as a noise source: the mutation rate should be significantly lower than the crossover rate in order not to disrupt the progress of the algorithm.

2.3.1 Modelling Genetic Algorithms

The genetic algorithm is clearly a very complex system. There are many different parameters to set and operators to choose. There has been considerable research interest in establishing theoretical models to describe and predict algorithm behaviour. The first model was based on Holland's "schema theorem" (Holland 1975; Goldberg 1989). A schema is a partially specified chromosome: several different schemata are instantiated in any given chromosome, and a given schema can be instantiated in many different ways. Holland suggested that the main function of crossover is schema processing, and demonstrated that 2^n schemata per individual are processed at every generation for binary encodings of length n (Holland 1975). He derived the schema theorem, which predicts that the numbers of small schemata with few fixed loci which have greater than average performance should increase exponentially. This effectively predicts the convergence of the algorithm. There are a number of weaknesses in the schema theorem, the main one being the strength of its underlying assumption, that there is a schema which always has above average performance by some constant factor. The schema theorem was criticised by Mühlenbein in (Mühlenbein 1994).

Several authors have attempted rigorous mathematical analysis of simple genetic algorithms, for example Vose in (Vose 1995) and Qi and Palmieri in (Qi and Palmieri 1994a; Qi and Palmieri 1994b). These models usually require simplifying assumptions such as infinite population sizes, or involve very large calculations (Mitchell 1996). Prügel-Bennett

and Shapiro adopted a physical analogy in (Prügel-Bennett and Shapiro 1994), in which the evolving population is seen as a thermodynamic system, the states of which could be described by statistical mechanics equations. This allowed Prügel-Bennett and Shapiro to predict the behaviour of the algorithm, but the approach is highly problem-specific. There has also been some interest in applying mathematical models used in the study of biological genetics to evolutionary optimisation, for example, Mühlenbein in (Mühlenbein 1994; Mühlenbein and Schlierkamp-Voosen 1995) and Bedau in (Bedau 1995). Recently, Cross has given an accurate theoretical account of the search properties of the genetic algorithm for inexact matching (Cross 1998).

While general theoretical models are beneficial in understanding the processes at play in genetic algorithms, their use in practical settings is limited. Any deviation from the assumptions made in the model could invalidate it, and the model may not be particularly informative as to what the best algorithm configuration might be for a given problem. By taking advantage of specific features of the problem to be solved, both Cross and Prügel-Bennett and Shapiro were able to make accurate predictions about the behaviour of the algorithm.

The alternative to theoretical modelling is empirical modelling. Possibly because of the computational expense, very few substantial experimental studies of genetic algorithms have been undertaken. The earliest was (DeJong 1975), in which DeJong considered a suite of five numerical optimisation problems. Problems from DeJong's test suite are still used to test and compare genetic algorithms, an approach strongly criticised by Whitley and others in (Whitley et al. 1995). In (Schaffer et al. 1989), Schaffer et al. presented what appears to be the only significant large scale experimental study of genetic algorithms to date. However, their experiments were mostly based on numerical test problems and have been shown by Mühlenbein in (Mühlenbein 1994) to be very sensitive to extrapolation. Indeed, regression equations are just as vulnerable as theoretical models to misinterpretation and overuse. Extrapolation of a regression model within the same problem domain should be undertaken with caution, but extrapolation outside the problem domain is totally meaningless.

2.3.2 Practical Issues

There are several practical problems with genetic algorithms. "Premature convergence" is the tendency of the population to converge to a set of very similar individuals between which the fitness function finds it hard to distinguish (DeJong 1975). "Epistasis" in biology is the interaction between genes. The contribution of a gene to the phenotype (fitness) depends on other genes. A special case of epistasis is "deception" (Goldberg 1987), where optimal points in the fitness landscape are sharp and isolated so that adjacent points lead the algorithm away from the optimum.

Many variations on the canonical algorithm have been tried, mainly in attempts to alleviate one or other of these problems. To avoid premature convergence, Shraudolf and Belew suggested a dynamic windowing approach (Schraudolf and Belew 1992). Fogel reported that this technique, which is essentially a coarse-to-fine strategy, works for unimodal objective functions but not for multimodal ones (the most common case) (Fogel 1994). Other methods for avoiding premature convergence, such as rank selection and "sigma scaling", are reviewed in (Srinivas and Patnaik 1994b): they are mainly concerned with constraining the standard deviation of the fitness over the population.

As well as variations in one or more of the operators, there are several distinct algorithmic variants including Syswerda's Steady State GA, in which only one offspring at a time is produced and replaces the worst member of the population (Syswerda 1989); Whitley's GENITOR, which combines rank based selection with the steady state algorithm (Whitley 1989); Ackley's Iterated Genetic Search (Ackley 1987); and Eshelman's CHC algorithm, in which recombination can only occur between chromosomes when the Hamming distance between them is above a certain threshold (Eshelman 1991).

An important class of genetic algorithm is the hybrid genetic algorithm, sometimes called "memetic algorithm", in which evolutionary optimisation is coupled with a local search step (Davis 1991; Cross et al. 1997; Whitley et al. 1995), or even simulated annealing (Yip and Pao 1995). It is believed that the benefit of this approach is that the population is pushed into its best possible state at each iteration, so that the algorithm is more likely to find the global optimum by the time the population has converged.

Genetic algorithms have been applied over a wide range of disciplines. The most famous applications are possibly "artificial life", in which evolution and competition between

organisms is simulated (Jefferson et al. 1991), and Koza's "genetic programming", where the individuals are actual computer programs, which are evaluated on the quality of their output (Koza 1992). In vision, genetic algorithms have been used for image segmentation (Andrey and Tarroux 1994), object recognition (Tsang 1997), stereo matching (Saito and Mori 1995), edge extraction (Bhandarkar et al. 1994), and graph matching (Cross 1998).

2.3.3 Multimodal Optimisation

The idea that genetic algorithms can be used to simultaneously find more than one solution to a problem was first mooted by Goldberg and Richardson in (Goldberg and Richardson 1987). They attempted to prevent the formation of large clusters of identical individuals in the population by de-rating the fitness function.

In (Cedeño et al. 1995), Cedeño and coworkers presented a "multiniche crowding" algorithm, which successfully finds several optimal solutions after about twenty generations. However, since the optima do not share the same fitness, it is not clear whether the different "niches" are stable over long periods.

An alternative method inspired by Glover's tabu search (Glover 1989) was used by Beasley and coworkers in (Beasley et al. 1993). The algorithm is restarted with the fitness function modified to suppress solutions found in previous program runs. This method has the disadvantages that, first, it is inherently sequential (an optimum must be found before it can be suppressed), and, second, that it is difficult to determine an appropriate radius of suppression. Recently, Jelasity and Dombi claim to have solved this problem, but their solution involves the introduction of several extra parameters (Jelasity and Dombi 1998).

A more radical approach is the distributed genetic algorithm, in which the population is explicitly subdivided, with restricted communication between the sub-populations (Gorges-Schleuter 1991; Whitley and Starkweather 1990; Davidor 1991). It is not entirely clear that this approach is very different from parallel multiple restarts, and it has been argued that the sub-populations are not stable in the long run (Davidor 1991).

A common feature of these approaches has been the necessity for extra parameters. Niching and crowding strategies typically require two or three extra parameters to be controlled. These parameters are needed, for example, to determine when to de-rate the fitness of an individual, by how much, and the distance scale of the de-rating function. In distributed

algorithms, it is necessary to decide how to arrange the sub-populations, their sizes, and under what conditions migration between them may occur. In (Smith et al. 1993), Smith and co-workers demonstrated a situation in which niching could occur in a standard genetic algorithm, without the need for any extra parameters.

2.4 Summary

Consistent labelling problems have been an active area of research in computer vision for the last 30 years. If the problem is small enough and the constraints sufficiently strong, heuristic search can be used to enumerate arbitrary solutions. However, where the constraints are weak, it is preferable to solve the problem in an optimisation framework, and use evidence from the scene to guide the labelling process. This can be achieved in a very natural way by applying Hancock and Kittler's Bayesian framework to the problem (Hancock and Kittler 1990a). Considerable success has been reported for this approach in the unambiguous case (Wilson 1995; Cross 1998).

Unfortunately, many consistent labelling problems are ambiguous in the sense that there may not be sufficient information in the scene to identify a unique solution. Current approaches to consistent labelling tend to disambiguate at a very early stage, although there has been interest in least commitment techniques from some quarters (Callari and Ferrie 1996; Ezquerra et al. 1998).

The genetic algorithm is a global optimiser which develops a population of solutions, rather than a single solution. It is a very complex algorithm with many parameters and operators. It has proved difficult to provide an adequate theoretical account of the algorithm which is useful in practice. Recent experience suggests that theories which consider specific aspects of the problem are more useful (Prügel-Bennett and Shapiro 1994; Cross 1998). Similarly, the few attempts at developing empirical models of the algorithm have tended to be rather too general (DeJong 1975; Grefenstette 1986; Schaffer et al. 1989).

Because it maintains a population of solutions, the genetic algorithm has been proposed as a method of simultaneously obtaining several different solutions to ambiguous problems. However, adaptation of the algorithm for this specific goal involves the introduction of more parameters to an already complex method (Goldberg and Richardson 1987; Beasley et al. 1993; Davidor 1991).

It would seem that the genetic algorithm is a natural way of solving ambiguous consistent labelling problems, since it has the required global optimisation properties, and also the potential to be used as a "least commitment optimiser". Genetic algorithms have been applied to labelling problems before. Recently, Fleurent and Ferland used genetic algorithms for graph colouring in (Fleurent and Ferland 1996).[2] In the field of computer vision, Cross applied the genetic algorithm to inexact matching in (Cross 1998), but the emphasis there was on finding one optimal solution to an unambiguous labelling problem.

The main novel contribution in this thesis will be to apply genetic algorithms to ambiguous consistent labelling problems, with a view to extracting as many solutions as possible without having to restart the algorithm. Cross's theoretical model for graph matching will be complemented by specific empirical models for both line labelling and graph matching. Methods of increasing solution yield which do not require many additional algorithm parameters will be considered.

[2]Davis applied the genetic algorithm to graph colouring earlier in (Davis 1991), but the treatment there was of graph colouring as a combinatoric optimisation problem, rather than explicitly as a labelling problem.

Chapter 3

The Consistent Labelling Problem

This chapter considers the dictionary-based approach to the consistent labelling problem. It will build on the work of Wilson in (Wilson 1995) to develop a general framework for solving inexact consistent labelling problems by optimisation. The main novel contribution in this chapter is the use of the edit distance in inexact labelling criteria. A new linear formulation of the consistency criterion is given. Attention is also paid to reducing the worst and average case complexities of consistency criterion evaluation.

The next section introduces the consistent labelling problem. Section 3.2 formulates line labelling and graph matching, the two labelling problems considered in the thesis. The Bayesian formulation of consistent labelling given by Hancock and Kittler in (Hancock and Kittler 1990a) is reviewed in section 3.3. Section 3.4 gives the linear labelling criterion and considers some of its direct applications in optimisation frameworks. Section 3.5 considers the problem of inexact consistent labelling, and criticises Wilson's original approach in (Wilson 1995). Section 3.6 gives an account of the string edit distance, and applies it to the labelling problem. Section 3.7 considers algorithm complexity. Finally, section 3.8 describes experiments which evaluate the sensitivity and efficiency of the new labelling criteria.

3.1 Introduction

The consistent labelling problem can be formulated as follows. Given a set of objects, $\mathbf{V}$, and a set of labels, $\mathbf{\Lambda}$, a set of neighbourhoods, $\mathbf{C}$, can be defined, each element C_j

of which is defined as $C_j =< i \mid i \in \mathbf{V} \wedge \mathbf{Conn}(j,i) >$, where **Conn** is the connectivity relation which is reflexive but not necessarily symmetric or transitive. A labelling, Γ_j, of the j^{th} neighbourhood is a list of labels applied to its constituent objects, $\Gamma_j : C_j \times \mathbf{\Lambda}$. Each neighbourhood has a dictionary, Θ_j, of legal labellings; the dictionaries capture the structure of the problem.[1] Waltz filtering (Waltz 1975) in this context would remove globally inconsistent labellings from the dictionaries, Θ_j. The consistency of a labelling of all the objects, Γ, can be determined by considering separately the consistency of the labellings of the neighbourhoods, Γ_j. This leads naturally to parallel-iterative algorithms for assessing the consistency of a labelling in terms of comparisons between the local configurations, Γ_j, and the dictionary items, $S_i \in \Theta_j$. There are at least four different types of consistent labelling problem.

The first is the most trivial type in which the neighbourhoods are just the objects themselves - i.e. $C_j =< j >$ and the dictionary is global. A good example is Ackley's ONEMAX problem (Ackley 1987), which has appeared frequently in the genetic algorithm literature (Syswerda 1989; Louis and Rawlins 1993; Mühlenbein and Schlierkamp-Voosen 1995; Srinivas and Patnaik 1996; Miller and Goldberg 1997; Mühlenbein 1994). The idea is to fill a bit-string with 1s. In this case, the objects are the bit-positions, the label set is $\{0,1\}$, the neighbourhoods are $\{< 1 >, < 2 >, ...\}$, and all the dictionaries consist of the single item $< 1 >$. In other words, the labellings of the bit-positions are independent, and the only permitted label for each bit-position is 1.

The second type of consistent labelling problem is where only certain classes of neighbourhood are permitted. The classic example of this is Huffman-Clowes labelling of trihedral polyhedra (Huffman 1971; Clowes 1971), in which the objects to be labelled are the lines in a drawing, the labels reflect edge geometry, and the neighbourhoods are the junctions which are of four types. The dictionaries for each junction type and hence for each neighbourhood are global: they depend on the geometry of the junction not its constituent lines. Line labelling is described in section 3.2.1.

The third type is exact graph matching, in which the objects to be labelled are the nodes in one graph, and the labels are the nodes in another graph. The neighbourhoods and

[1] If all the neighbourhoods are the same size, this is equivalent to Haralick and Shapiro's original formulation of the consistent labelling problem as a network constraint satisfaction problem in (Haralick and Shapiro 1979; Haralick and Shapiro 1980). The neighbourhoods are the "unit-constraint relation" and the dictionaries are the "unit-label constraint relation".

dictionaries are typically defined over the edge sets of the graphs. Graph matching is harder than line labelling because first the label set is larger, second the dictionaries change from problem instance to problem instance, and third the neighbourhoods and dictionaries are larger and more complex: each line in a drawing only belongs to two junctions, but each node in a graph typically belongs to many neighbourhoods.

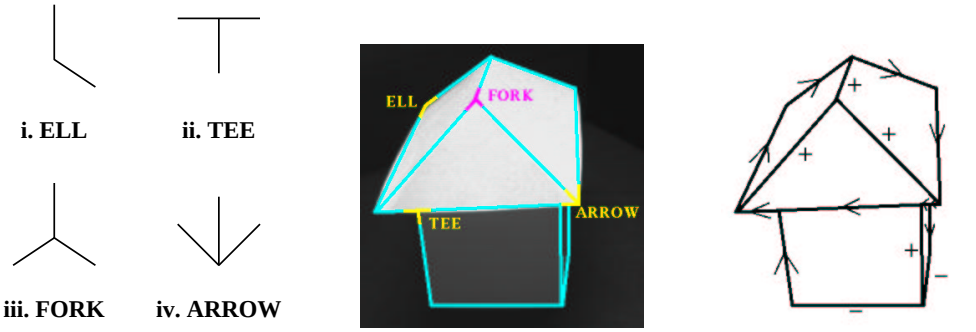
The fourth problem is inexact graph matching, which is a superset of exact graph matching. The crucial difference between this and "classical" consistent labelling problems such as the previous three is that the dictionary items, S_i, need not be the same size as the label configurations, Γ_j. This problem is very much harder than exact matching or line labelling because there may not actually be a consistent labelling. The goal here is to establish a maximally consistent labelling, a goal more conveniently sought via optimisation than by search. Graph matching is discussed in more detail in section 3.2.2.

3.2 Two Labelling Problems

This thesis considers two labelling problems: Huffman-Clowes line labelling, which is by now a classical problem in machine vision and artificial intelligence; and inexact attributed relational graph matching, which has greater contemporary relevance.

3.2.1 Line Labelling

Line labelling was independently formulated by Huffman and Clowes in the mid 1970s (Huffman 1971; Clowes 1971). Drawings representing scenes composed of trihedral polyhedra only contain four types of junction. The junctions can be classified according to whether they have ELL, TEE, FORK or ARROW shaped topologies. Each line in the drawing must be labelled according to whether it represents the concave intersection of two surfaces, $-$, the convex intersection of two surfaces, $+$, or an occluding boundary, $\rightarrow$ (the occluded surfaces are always to the right as one follows the arrow). An example is given in figure 3.1. In (Waltz 1975), Waltz associated a dictionary of consistent label configurations with each junction type. These dictionaries are derived from the geometric constraints on the projection of three-dimensional scenes onto two-dimensional planes. Thus, the problem of finding a plausible three-dimensional interpretation of a line drawing can often be solved by finding a consistent labelling of the lines in the drawing.



(a) Junction Types (b) Example (c) Obvious Labelling

Figure 3.1: The Four Junction Types Defined by Huffman in (Huffman 1971).

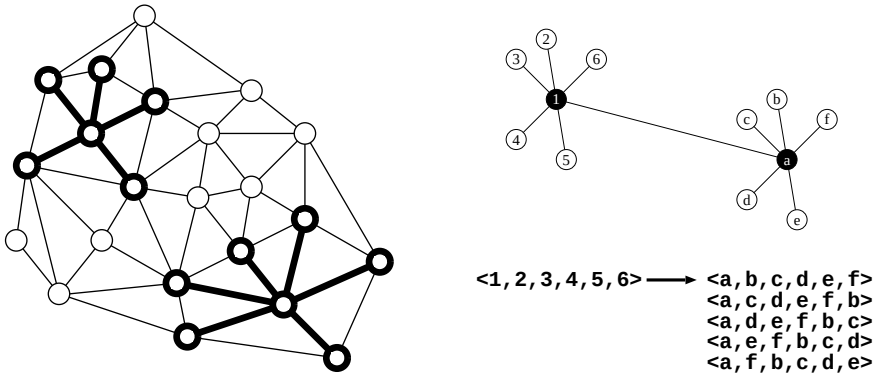
3.2.2 Graph Matching

Like line labelling, graph matching is a classical consistent labelling problem. The variant of concern here is inexact attributed relational matching. Attributed relational graphs were first described by Fu in (Fu 1983).

An attributed relational graph (ARG) is a triple, $G = (\mathbf{V}, \mathbf{E}, \mathbf{A})$, where $\mathbf{V}$ is the set of vertices or nodes, $\mathbf{E} \subseteq \mathbf{V} \times \mathbf{V}$ is the set of edges, and $\mathbf{A} \subset \mathbf{V} \times \Re^k$ is the set of measurement vectors relating to the original scene. Graph matching is the problem of establishing a correspondence between a data graph, $G_D = (\mathbf{V}_D, \mathbf{E}_D, \mathbf{A}_D)$, and a model graph, $G_M = (\mathbf{V}_M, \mathbf{E}_M, \mathbf{A}_M)$. This correspondence, $f : V_D \mapsto V_M \cup \{\phi\}$, is a labelling of the nodes in $\mathbf{V}_D$ with nodes from $\mathbf{V}_M$ or a special null label, ϕ, for unmatchable nodes. This can be seen as a consistent labelling problem since the structure of the model graph provides constraints on the labels applicable to data graph nodes. Indeed, graph matching was one of Haralick and Shapiro's original example problems in (Haralick and Shapiro 1979; Haralick and Shapiro 1980). When the data graph is identical to the model graph, the problem is said to be exact; otherwise it is inexact.

In (Wilson 1995), Wilson showed that considering the nodes alone was insufficient to solve the inexact version of the problem. Consideration of both the node attributes and neighbourhoods is necessary. Wilson defined the neighbourhoods as "supercliques". A superclique is the neighbourhood formed by a node together with all the nodes connected to it by an edge taken in cyclic order - i.e. $C_j =< j, u | (j, u) \in \mathbf{E}_D >$. Panel (a) of figure 3.2 gives examples of supercliques. The dictionaries are formed by considering the possible

mappings between supercliques in $\mathbf{V}_D$ and supercliques in $\mathbf{V}_M$. For planar graphs, it is necessary to consider all cyclic permutations of the model graph supercliques as shown in panel (b) of figure 3.2.



(a) Supercliques (b) Mappings

Figure 3.2: Supercliques as defined by Wilson in (Wilson 1995).

To summarise, attributed relational matching can be seen as a consistent labelling problem if the objects are the nodes of the data graph, $\mathbf{V}_D$, the labels are the drawn from the set formed by the union of the model graph nodes and a null label (i.e. $\mathbf{\Lambda} = \mathbf{V}_M \cup \{\phi\}$), and the neighbourhoods are the supercliques in the data graph. The dictionaries are cyclic permutations of the supercliques in the model graph, $S_i =< v, i|(i, v) \in \mathbf{E}_M >$, and $\Theta_j = \{S_i \mid |S_i| = |C_j|\}$.

3.3 Bayesian Formulation

This section follows and generalises the formulations of Hancock and Kittler, Wilson, and Hancock in (Hancock and Kittler 1990a; Wilson 1995; Hancock 1994). Imagine that a consistent labelling of the j^{th} neighbourhood has become corrupted through some imaginary error process to give the "observed" configuration, Γ_j. The initial consistent configuration could have been any of the dictionary items, $S_i \in \Theta_j$. The Bayes rule can now be applied, leading to an expression for the probability that configuration Γ_j arose by chance from the action of corruption on one of the dictionary items S_i:

$$P(\Gamma_j) = \sum_{S_i \in \Theta_j} P(\Gamma_j|S_i)P(S_i) \tag{3.1}$$

Since there is no evidence that any particular dictionary item was the "source" of the configuration, $P(S_i)$ must be taken to be uniformly distributed with value $\frac{1}{|\Theta_j|}$, giving

$$P(\Gamma_j) = \frac{1}{|\Theta_j|} \sum_{S_i \in \Theta_j} P(\Gamma_j|S_i) \tag{3.2}$$

Consider now the probability of the configuration given a dictionary item, $P(\Gamma_j|S_i)$. Following Hancock and Kittler (Hancock and Kittler 1990a), by assuming that the corruption process is memoryless and independent of the neighbourhood structure, and has a uniform probability, P_e, $P(\Gamma_j|S_i)$ can be factorised over the neighbourhood of j to give

$$P(\Gamma_j|S_i) = \prod_{0<k\leq|\Gamma_j|} P(u|v) \tag{3.3}$$

where u and v are the k^{th} items in Γ_j and S_i respectively. According to Hancock and Kittler's corruption model (Hancock and Kittler 1990a), if the configuration label u is not the same as the dictionary label v, it is as a result of a corruption event which happens with probability P_e. Therefore, the unit labelling probability, $P(u|v)$, is determined solely by whether or not the labelling was corrupted at that position, hence

$$P(u|v) = \begin{cases} P_e & \text{if } u \neq v \\ 1 - P_e & \text{otherwise} \end{cases} \tag{3.4}$$

This allows $P(\Gamma_j|S_i)$ to be written in terms of the number of corruption events, or distance, from Γ_j to S_i, $D(\Gamma_j, S_i)$.

$$P(\Gamma_j|S_i) = P_e^{D(\Gamma_j,S_i)}(1 - P_e)^{|\Gamma_j|-D(\Gamma_j,S_i)} \tag{3.5}$$

In the exact case, the distance, $D(\Gamma_j, S_i)$, is the Hamming distance; the inexact case is discussed later. A little rearrangement of equation 3.5 gives

$$P(\Gamma_j|S_i) = K_{C_j} \exp\left[-k_e D(\Gamma_j, S_i)\right] \tag{3.6}$$

where $K_{C_j} = (1 - P_e)^{|\Gamma_j|}$ and $k_e = \ln\left(\frac{1-P_e}{P_e}\right)$. This leads to

$$P(\Gamma_j) = \frac{K_{C_j}}{|\Theta_j|} \sum_{S_i \in \Theta_j} \exp\left[-k_e D(\Gamma_j, S_i)\right] \tag{3.7}$$

This last expression has a number of interesting properties. First, every dictionary item contributes to the probability of the configuration. This gives a finely graded criterion in which well matched dictionary items carry greater weight than poorly matched items. Second, the differential weighting is controlled by the label error probability, P_e. As $P_e \to 0$, $k_e \to \infty$, so only exact matches will have non-zero probabilities. Conversely, when $P_e = 0.5$, $k_e = 0$ and any match will have probability K_{C_j}. In (Hancock 1994), Hancock observed that when P_e is small, the sum in equation 3.7 is dominated by the minimum distance configuration, so the equation can be rewritten as follows:

$$P(\Gamma_j) \approx \frac{K_{C_j}}{|\Theta_j|} \exp\left[-k_e \min_{S_i \in \Theta_j} D(\Gamma_j, S_i)\right] \tag{3.8}$$

In (Hancock 1994), Hancock drew an analogy between the minimum distance, $\min_{S_i \in \Theta_j} D(\Gamma_j, S_i)$, and the Gibbs potential, in which k_e plays the rôle of inverse temperature. This means that reducing P_e at every iteration of a labelling algorithm would harden the constraints and impart a deterministic annealing-like behaviour to the algorithm. The analogy is not so simple when the approximation breaks down and the minimum distance no longer dominates the sum of exponentials. This case has been studied rigorously by Finch, Wilson and Hancock in (Finch et al. 1998).

To obtain the global probability of the entire labelling, Γ, Hancock and Kittler, Wilson, and Hancock take the arithmetic mean of equation 3.7 over all neighbourhoods to give

$$P(\Gamma) = \frac{1}{|\mathbf{V}|} \sum_{j \in \mathbf{V}} \frac{K_{C_j}}{|\Theta_j|} \sum_{S_i \in \Theta_j} \exp\left[-k_e D(\Gamma_j, S_i)\right] \tag{3.9}$$

This criterion can be optimised by gradient ascent, in which the label, v, for each object, u, is iteratively chosen to maximise consistency, according to the following maximum *a posteriori* probability (MAP) update rule:

$$f(u) = \arg\max_{v \in \mathbf{\Lambda}} P(\Gamma) \tag{3.10}$$

3.4 Linear Formulation

Suppose the global criterion in equation 3.9 were defined as a geometric rather than an arithmetic mean,

$$P_G(\Gamma) = \left(\prod_{j \in \mathbf{V}} P(\Gamma_j) \right)^{\frac{1}{|\mathbf{V}|}} \tag{3.11}$$

This can be expanded according to equation 3.7 to give

$$P_G(\Gamma) = \left(\prod_{j \in \mathbf{V}} \frac{K_{C_j}}{|\Theta_j|} \sum_{S_i \in \Theta_j} \exp\left[-k_e D(\Gamma_j, S_i)\right] \right)^{\frac{1}{|\mathbf{V}|}} \tag{3.12}$$

Applying Hancock's approximation for small P_e from equation 3.8 allows the inner sum of exponentials to be rewritten in terms of the minimum distance, $\min_{S_i \in \Theta_j} D(\Gamma_j, S_i)$. The product terms can then be separated to give:

$$\begin{aligned} P_G(\Gamma) &\approx \left(\prod_{j \in \mathbf{V}} \frac{K_{C_j}}{|\Theta_j|} \exp\left[-k_e \min_{S_i \in \Theta_j} D(\Gamma_j, S_i)\right] \right)^{\frac{1}{|\mathbf{V}|}} \\ &\approx k_v \exp\left[-\frac{k_e}{|\mathbf{V}|} \sum_{j \in \mathbf{V}} \min_{S_i \in \Theta_j} D(\Gamma_j, S_i) \right] \end{aligned}$$

where $k_v = \left(\prod_{j \in \mathbf{V}} \frac{K_{C_j}}{|\Theta_j|} \right)^{\frac{1}{|\mathbf{V}|}}$. Recalling Hancock's thermodynamic analogy in equation 3.8, it is tempting to regard the sum of minimum distances as a Gibbs potential, with $\frac{k_e}{|\mathbf{V}|}$ playing the rôle of inverse temperature. As before, this relies on the approximation from equation 3.8 that either k_e is large or the distances, $D(\Gamma_j, S_i)$, are large. Given this approximation, it is now possible to express the consistency of a labelling in terms of a cost function,

$$E(\Gamma) = \sum_{j \in \mathbf{V}} \min_{S_i \in \Theta_j} D(\Gamma_j, S_i) \tag{3.13}$$

This function is appealing since it depends only on the closest dictionary item to the current configuration: $E(\Gamma)$ is the number of local inconsistencies in the labelling Γ and

therefore provides a more direct measure of consistency than the Bayesian formulation in equation 3.9. In contrast to the deterministic annealing implied by equation 3.8, this cost function can be optimised using stochastic simulated annealing (Kirkpatrick et al. 1983). Simulated annealing makes an analogy between minimising a cost function and finding low energy states of a system of particles with discrete energy levels. At thermal equilibrium, the probability of such a system being in a particular state would have a Boltzmann distribution. In simulated annealing, the labelling, Γ, is subject to random changes, which are accepted or rejected according to a decision rule based on the Metropolis algorithm (Metropolis et al. 1953). Provided certain conditions are satisfied, after many iterations the labelling will reach an "equilibrium" in which the probability of a particular configuration is governed by the Boltzmann distribution,

$$P_B(\Gamma) = \frac{1}{Z} e^{-\beta E(\Gamma)} \tag{3.14}$$

where $Z = \sum_{\Gamma \in \mathbf{\Omega}} e^{-\beta E(\Gamma)}$ is the partition function over all labellings in the configuration space, $\mathbf{\Omega} = \mathbf{\Lambda}^{|\mathbf{V}|}$, and β is the inverse temperature. Prügel-Bennett and Shapiro extend this idea to the genetic algorithm's selection scheme in (Prügel-Bennett and Shapiro 1994). Following their example, it is possible to select a particular labelling from a population with probability p_Γ, defined in terms of its "Boltzmann potential", $e^{-\beta E(\Gamma)}$:

$$p_\Gamma = \frac{1}{Q} e^{-\beta E(\Gamma)} \tag{3.15}$$

where Q is the sum of the potentials of all the labellings in the population. In this case, β behaves more like a scale parameter than a temperature. As long as β is set to $\frac{k_e}{|\mathbf{V}|}$ and the approximation in equation 3.8 is good, Boltzmann selection based on the cost function of equation 3.13 should be analogous to fitness proportionate selection based on the geometric mean in equation 3.12.

The approximation of equation 3.8 will be poor when both k_e and $D(\Gamma_j, S_i)$ are small. Fortunately, this is unlikely to be the case: at the start of the labelling process, k_e may be small but $D(\Gamma_j, S_i)$ will probably be large since the initial labelling will probably have many inconsistencies. By the end of the process, when $D(\Gamma_j, S_i)$ becomes small, k_e will be large. These observations are important because annealing schedules previously established for P_e by Wilson and Cross in (Wilson 1995) and (Cross 1998) can now be

used to control the inverse temperature for simulated annealing (equation 3.14) or for Boltzmann selection (equation 3.15) in a genetic algorithm.

3.5 Inexact Labelling Problems

Up to now, the formulation in terms of corrupted dictionary items has been relatively straightforward since it has been assumed tacitly that the dictionary items are the same size as the neighbourhoods, hence the simple definitions of $P(\Gamma_j|S_i)$ in equation 3.3, $P(u|v)$ by the distribution rule 3.4, and of $D(\Gamma_j, S_i)$ as the Hamming distance. This section considers the inexact case, in which this assumption is generally incorrect.

In (Wilson 1995), Wilson addressed this problem by padding the dictionary items with dummy labels so that it was the same size as the local configuration. For example, consider the configuration, $\Gamma =< u_1, u_2, u_3 >$, and the dictionary item, $S_i =< v_1, v_2 >$. In order to compare the configuration to the dictionary item, padding must be added to the dictionary item to give $S_i' =< v_1, v_2, \phi >$. If the dictionary item is larger than the configuration, the configuration must be padded. This approach to the problem entails several important drawbacks. First an additional parameter is required: the probability of extra or missing labels in a configuration. Second the number of such padded dictionary entries will be large. Third summing over very many dictionaries in equation 3.9 will distort the criterion.

To see the need for an extra parameter, consider the distribution rule in equation 3.4 in the case when dummy labels are present. Suppose that the consistent labelling, $< u_1, u_2, u_3 >$, has been corrupted by the addition of u_4 at the end. When comparing to the dictionary $< v_1, v_2, \phi, v_3 >$, one might conclude that the Hamming distance should be 2 whereas in fact only a single error has occurred. On the other hand, simply ignoring dummy labels would lead to considering $< u_1, u_2, u_3, u_4 >$ a perfect match for $< v_1, v_2, v_3, \phi >$ even though there is an error in the labelling. To avoid these difficulties, Wilson used the following distribution rule:

$$P(u|v) = \begin{cases} P_\phi & \text{if } u = \phi \text{ or } v = \phi \\ (1 - P_e)(1 - P_\phi) & \text{if } u = v \\ P_e(1 - P_\phi) & \text{otherwise} \end{cases} \tag{3.16}$$

The effect of this on the global criterion of equation 3.9 is to introduce as an additional

control variable the probability of a label being added to or deleted from a configuration, P_ϕ. Applying the thermodynamic analogy would necessitate a two-dimensional "temperature". Indeed, Wilson found that explicitly controlling P_ϕ did not give particularly good results (Wilson 1995): whereas the introduction of P_e allowed the labelling process to be controlled in a principled manner, the introduction of P_ϕ only caused problems. A major part of Wilson's work was the control of the process by which data graph nodes are assigned the null label. In (Wilson 1995) and (Wilson and Hancock 1997), it was found that the best method for null labelling was graph editing, closely followed by a constraint filtering post-processing step, both of which considerably outperformed explicit control of P_ϕ. However, Wilson did not address the consequences of dictionary padding for the space and time requirements of criterion evaluation.

3.5.1 Dictionary Size

Wilson's use of constraint filtering or graph editing in (Wilson 1995) and (Wilson and Hancock 1997) allowed P_ϕ to be held constant at some low value related to the average size difference between configurations and dictionary items. In other words, it can be treated as a prior probability rather than as a control variable. However, Wilson's approach still requires the addition of padding to dictionary items or label configurations. This section assumes without loss of generality that only dictionary items are being padded.

Consider the addition of k labels to a configuration by random corruption: the "original" size of the configuration was $|C_j|$, so the new size is $|C_j| + k$. These k labels can be added anywhere, so the dictionary must be augmented by adding $\left(\begin{array}{c} |C_j| + k \\ k \end{array} \right)$ new items.[2] Now consider labellings which might have had up to K labels added in the corruption process. The size of the dictionary is now

$$|\Theta_j| = \sum_{0 \leq k \leq K} \left(\begin{array}{c} |C_j| + k \\ k \end{array} \right)$$

[2]This actually depends on whether or not the neighbourhoods are cyclic, as they are for line labelling and graph matching. In cyclic neighbourhoods, $< a, b, c > \equiv < b, c, a >$, so there are only $|C_j| + k - 1$ places where a dummy label could be inserted. Without loss of generality, non-cyclic neighbourhoods are considered here, in which there are $|C_j| + k$ such places.

$$= \begin{pmatrix} |C_j| + K + 1 \\ K \end{pmatrix} \tag{3.17}$$

The worst case occurs when $k = |C_j|$, and both are large, in which case:

$$\begin{aligned}
|\Theta_j|_{\mathrm{MAX}} &= \begin{pmatrix} 2|C_j| + 1 \\ |C_j| \end{pmatrix} \\
\Rightarrow \ln |\Theta_j|_{\mathrm{MAX}} &= \ln (2|C_j| + 1)! - \ln |C_j|! - \ln (|C_j| + 1)! \\
&\approx \ln (2|C_j|)! - 2 \ln |C_j|! \\
&\approx 2|C_j| \ln 2|C_j| - 2|C_j| - 2(|C_j| \ln |C_j| - |C_j|) \\
&\approx 2|C_j| \ln 2 + 2|C_j| \ln |C_j| - 2|C_j| - 2|C_j| \ln |C_j| + 2|C_j| \\
&\approx 2|C_j| \ln 2 \\
\Rightarrow |\Theta_j|_{\mathrm{MAX}} &\approx 4^{|C_j|}
\end{aligned} \tag{3.18}$$

by Stirling's approximation for large $|C_j|$. Such a bad worst case might be expected to occur infrequently. However, the dictionary size can become large very quickly when even a moderate amount of padding is needed.

In (Wilson et al. 1998), Wilson, Cross and Hancock found that the Delaunay triangulation gave the best results for matching noisy point sets. For the two 70-node Delaunay graphs encountered in section 3.8, the mean superclique size of the model graph is 5.7 (s.d. = 1.2) with a maximum of 9, and the mean size difference between superclique pairs from the two graphs is 1.3 (s.d. = 1.2) but has a maximum of 6. These values are not remarkable according to Wilson (Wilson 1995). Taking the view that no matches should be excluded *a priori*, it would be necessary to add 6 dummy labels to every dictionary item. In this case, $|C_j| \approx 6$ and $K = 6$. It follows from equation 3.17 that the size of the largest dictionary is $\begin{pmatrix} 13 \\ 6 \end{pmatrix} = 1716$. A separate dictionary is needed for each of the 70 nodes, so the grand total could be as high as $70 \times 1716 = 120120$, which is considerable. The dictionary sizes become even larger when the cyclic permutations of each superclique are taken into account.

In general for matching Delaunay triangulations, about 80% of the superclique-pairs have size differences of 2 or less. Given the underlying assumption of equation 3.4, that only one

error occurs per mapping, the probability of significant superclique corruption is small. It is therefore reasonable to follow Wilson in (Wilson 1995), and suppose that supercliques with size differences greater than 2 are not matchable anyway. In other words, limiting the maximum amount of padding to 2 gives manageable dictionaries and screens out some 20% of probably unmatchable superclique-pairs, even though it violates the assumption of statistical independence that underpins equation 3.3.

However, it will not necessarily be possible to appeal to geometric properties of the problem to reduce the amount of padding in every case, and the worst case of equation 3.18 will always be lurking. Furthermore, there is a more elegant way of handling size-differences which yields performance benefits even when the size-differences are small, which is the topic of the next section.

3.6 Edit Distance

There are two important weaknesses of the dictionary-based approach to inexact labelling problems just outlined. The first is that it entails potentially exponential complexity: this can be remedied heuristically. The second is that it relies on a rather artificial model in which dictionaries have to be padded so that they are the same size as the constraint neighbourhoods. A measure of the distance between lists of differing lengths has existed for many years: the Levenshtein or string edit distance (Levenshtein 1966; Wagner and Fischer 1974). This avoids the use of padding altogether, by considering insertions and deletions in addition to changes. The main novel contribution in this chapter is in basing the dictionary comparison on the edit distance. The rest of this section gives a brief overview of the edit distance, before applying it to the inexact labelling problem.

3.6.1 Overview

Let X and Y be two strings of symbols drawn from an alphabet, Σ. X is to be converted to Y via an ordered sequence of operations such that the cost associated with the sequence is minimal. The original string to string correction algorithm defined "elementary edit operations", $\delta = (a, b)$, where a and b are symbols from the two strings or the null symbol, ϵ (the operation (ϵ, ϵ) is illegal). Changing symbol x to symbol y is denoted (x, y), inserting y is denoted (ϵ, y), and deleting x is denoted (x, ϵ). Any operation (x, x) is an identity. A

sequence of elementary edit operations which transforms X into Y is known as an "edit transformation", and is denoted $\Delta =< \delta_1, ..., \delta_{|\Delta|} >$. Elementary costs are assigned by an elementary weighting function, $\gamma : (\Sigma \cup \epsilon) \times (\Sigma \cup \epsilon) \mapsto \Re$; the cost of an edit transformation, $W(\Delta)$, is the sum of its elementary costs. The edit distance between X and Y is defined as

$$\mathbf{d}(X, Y) = \min\{W(\Delta) | \Delta \text{ transforms } X \text{ to } Y\} \tag{3.19}$$

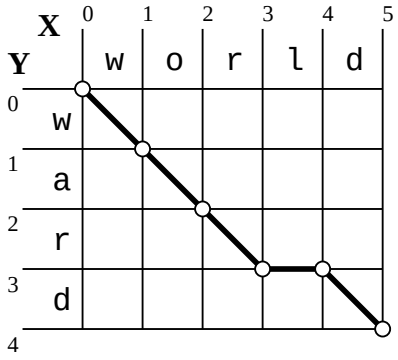
An interesting property of this quantity is that it is a metric if $\gamma > 0$ for all non-identity operations and γ is self-inverse (Marzal and Vidal 1993).

The raw edit distance, $\mathbf{d}$, may not always be useful, since correcting 2 errors in a neighbourhood of size 3 should be more expensive than correcting 5 errors in a neighbourhood of size 10. Furthermore, for the present purpose, it is not entirely clear that an ordered sequence of operations can be the result of a memoryless random error process. Marzal and Vidal's "normalised edit distance", presented in (Marzal and Vidal 1993), possesses the desired normalisation properties, and is consistent with statistical independence of labelling errors. They introduce the notion of an "edit path" which is a sequence of ordered pairs of positions in X and Y such that the path monotonically traverses the edit matrix of X and Y from $(0, 0)$ to $(|X|, |Y|)$, as shown in figure 3.3.

The transition from one point in the path to the next is equivalent to an elementary edit operation: $(i, j) \rightarrow (i + 1, j)$ corresponds to deletion of the symbol in X at position i. Similarly, $(i, j) \rightarrow (i, j + 1)$ corresponds to insertion of the symbol at position j in Y. $(i, j) \rightarrow (i + 1, j + 1)$ corresponds to changing $X(i)$ to $Y(j)$. Applying the elementary weighting function to each elementary edit operation implied by these transitions yields a weighted path such that

$$\mathbf{d}(X, Y) = \min\{W(P) | P \text{ is an edit path from } X \text{ to } Y\} \tag{3.20}$$

This quantity can be normalised by dividing by the length of the path, $L(P)$. Thus, the normalised edit distance is



(a) Edit Matrix

$$\begin{aligned}
\Delta &= (\mathrm{w},\mathrm{w}),(\mathrm{o},\mathrm{a}),(\mathrm{r},\mathrm{r}),(\mathrm{l},\epsilon),(\mathrm{d},\mathrm{d}) \\
\mathbf{d} &= \gamma(\mathrm{o},\mathrm{a}) + \gamma(\mathrm{l},\epsilon) = 2 \\
P &= <(0,0),(1,1),(2,2), \\
&\quad (3,3),(4,3),(5,4)> \\
W(P) &= \mathbf{d} = 2 \\
L(P) &= 6 \\
\Rightarrow \hat{\mathbf{d}} &= \frac{1}{3}
\end{aligned}$$

(b) Edit Distances

Figure 3.3: Example Edit Matrix from X to Y. The thick black line is the editing path. The relationship between the classical and normalised edit distances is shown in panel (b).

$$\begin{aligned}
\hat{\mathbf{d}}(X,Y) &= \min\left\{\frac{W(P)}{L(P)} | P \text{ is an edit path from } X \text{ to } Y\right\} \\
&= \frac{W(P^*)}{L(P^*)}
\end{aligned} \tag{3.21}$$

As Marzal and Vidal point out, it is clear from the form of $\hat{\mathbf{d}}$ that it cannot be determined by simply minimising $W(P)$ and then dividing by $L(\arg\min_P W(P))$. The complexity of computing $\hat{\mathbf{d}}$ is $\mathrm{O}(|X|.|Y|^2)$, $|X| \geq |Y|$.

3.6.2 Inexact Labelling

If X and Y in figure 3.3 are replaced by a dictionary item, S_i, and a labelled configuration, Γ_j, it can be seen that Γ_j could indeed have arisen from S_i through the action of a memoryless error process, statistically independent of position (since the errors that "transformed" S_i into Γ_j could have occurred in any order). This means that equation 3.3 can still be applied, except that it is now necessary factorise over the elementary edit operations implied by the transitions in the optimal edit path, P^*, to give

$$P(\Gamma_j|S_i) = \prod_{\delta \Leftarrow P^*_{\Gamma_j S_i}} p(\delta) \tag{3.22}$$

where δ is an insertion, a deletion, a change or an identity. It remains to define the elementary edit probabilities, $p(\delta)$, and to consider the weighting function, γ. Although γ does not appear in equation 3.22, it influences the choice of optimal edit path.

In (Bunke and Csirik 1995), Bunke and Csirik suggest parameterising the edit distance by fixing the weights of insertion and deletion, leaving the cost of a change as the only parameter. They give the following definition of γ:

$$\gamma(a,b) = \begin{cases} 1 & \text{if } a = \epsilon \vee b = \epsilon \\ r & \text{if } a \neq b \\ 0 & \text{otherwise} \end{cases} \tag{3.23}$$

where r is the remaining parameter, the cost of a change. Bunke and Csirik point out that r must be less than 2, if one observes the general restriction that $0 \leq \gamma(a,b) \leq (\gamma(a,\epsilon) + \gamma(\epsilon,b))$ for a single change to be preferable to a deletion followed by an insertion. Whether or not it is desirable to distinguish between changes and insertions or deletions, it is convenient here to set r to 1, so that all non-identity edit operations contribute equally to the edit distance. Any distinction to be made can be deferred to the distribution rule; this thesis makes no such distinction so the distribution rule is

$$p(\delta) = \begin{cases} (1 - P_e) & \text{if } \delta \text{ is an identity} \\ P_e & \text{otherwise} \end{cases} \tag{3.24}$$

Equation 3.5 can now be rewritten in terms of the number of non-identity transformations in the optimal edit path from Γ_j to S_i. Given our weighting function, this number is simply $W(P^*_{\Gamma_j S_i})$, so the exponential of equation 3.7 becomes

$$P(\Gamma_j) = \frac{1}{|\Theta_j|} \sum_{S_i \in \Theta_j} \exp\left[-\left(k_W W(P^*_{\Gamma_j S_i}) + k_L L(P^*_{\Gamma_j S_i})\right)\right] \tag{3.25}$$

where $k_W = \ln \frac{(1-P_e)}{P_e}$, and $k_L = \ln \frac{1}{(1-P_e)}$. If all edit operations have equal weights, the length of the optimal path will be equal to the length of the larger of $|\Gamma_j|$ and $|S_i|$.

The normalised edit distance is not used directly: the criterion merely counts the elementary operations that make up the optimal path. The items in Θ_j no longer need to be padded, so using the edit distance instead of dictionary padding reduces the worst case space requirements of the dictionaries from $\mathrm{O}(4^{|C_j|})$ to $\mathrm{O}(|C_j|)$.

3.7 Algorithm Complexity

The arguments in this and subsequent sections apply specifically to inexact relational matching, rather than to the consistent labelling problem in general.

Equations 3.9 and 3.13 can be rewritten in terms of a framework component over the matchable supercliques of both graphs and a kernel component over individual dictionary comparisons. This section explicitly considers equation 3.9, but the argument can be applied equally well to equation 3.13. For configuration, Γ_j, the dictionary, Θ_j, can be partitioned according to the model graph supercliques: $\Theta_j = \bigcup_{i \in \mathbf{V}_M} \Theta_{j,i}$, where $\Theta_{j,i} = \{S_i | S_i \in \Theta_j \wedge S_i(0) = i\}$. $P(\Gamma)$ can now be written as:

$$P(\Gamma) = \frac{1}{|\mathbf{V}_D|} \sum_{j \in \mathbf{V}_D} \frac{1}{|\Theta_j|} \sum_{i \in \mathbf{V}_M} P(\Gamma_j | \Theta_{j,i}) \tag{3.26}$$

Thus, it can be seen that the time-complexity of the evaluation of $P(\Gamma)$ is the quadratic framework component, $\mathrm{O}(|\mathbf{V}_D||\mathbf{V}_M|)$, multiplied by the kernel component, $\mathrm{T}[P(\Gamma_j|\Theta_{j,i})]$, where $\mathrm{T}[x]$ is the time taken to evaluate x. The evaluation of $P(\Gamma_j|\Theta_{j,i})$ is simply

$$P(\Gamma_j | \Theta_{j,i}) = \sum_{S_i \in \Theta_{j,i}} P(\Gamma_j | S_i) \tag{3.27}$$

So the kernel component, $P(\Gamma_j|\Theta_{j,i})$, can be computed in $\mathrm{O}(|\Theta_{j,i}|) \times \mathrm{T}[P(\Gamma_j|S_i)]$ time. For the sake of simplicity, observe that the sizes of supercliques, labelling configurations and dictionary items only differ by small constant factors, and denote them all by C, such that $\forall_{j \in \mathbf{V}_D, S_i \in \Theta_j, i \in \mathbf{V}_M} C = \mathrm{O}(|\Gamma_j|) = \mathrm{O}(|S_i|) = \mathrm{O}(|C_i|)$.

For exact matching, $\Theta_{j,i}$ consists of the cyclic permutations of those model graph supercliques which could match the j^{th} data graph superclique, and the Hamming distance is computed in linear time, so the time taken to evaluate the kernel component is cubic in

the configuration size and is $O(C^3)$. For inexact matching with padded dictionaries, the dictionary item comparison is still linear, but the worst case dictionary size is as in equation 3.18, so evaluation of the kernel component requires time exponential in the average superclique size and is $O(C.4^C)$. If the normalised edit distance is used, instead of the Hamming distance, as the distance measure, the distance evaluation is cubic but the dictionary size is now only quadratic, giving a quintic kernel, $O(C^5)$. Worst case complexities for these three cases are summarised in table 3.1. The worst case for exact matching is a quintic algorithm which is feasible. The worst case for inexact matching with dictionary padding is infeasible. The worst case for inexact matching with edit distance may or may not be feasible depending on how powerful the hardware is, and how large C is.

	Complexity			
Matching	Dictionary Size	T$[D(\Gamma_j, S_i)]$	T$[P(\Gamma_j \vert \Theta_{j,i})]$	Overall
Exact	$O(C^2)$	$O(C)$	$O(C^3)$	$O(\vert\mathbf{V}_D\vert\vert\mathbf{V}_M\vert C^3)$
Inexact (padding)	$O(4^C)$	$O(C)$	$O(C4^C)$	$O(\vert\mathbf{V}_D\vert\vert\mathbf{V}_M\vert C4^C)$
Inexact (edit)	$O(C^2)$	$O(C^3)$	$O(C^5)$	$O(\vert\mathbf{V}_D\vert\vert\mathbf{V}_M\vert C^5)$

Table 3.1: Worst Case Complexities.

It should be stressed that for exact matching and inexact matching with edit distance, these worst cases are also average cases. The average case for inexact matching with dictionary padding depends on the amount of padding.

3.7.1 Accelerating the Kernel Component

Using the edit distance trades large dictionaries for slow comparisons between configurations and dictionary items. The most important part of the dictionary padding approach used by Wilson was the screening out of mappings between supercliques with size differences greater than 1 or 2. This drastically reduces the size of the dictionaries and renders the whole calculation feasible. The heuristic is equally applicable to the edit distance based criterion because it is effectively a form of Waltz filtering: (probably) globally inconsistent labellings are removed from the dictionary at the outset. If the superclique size differences are small enough with respect to the superclique sizes, then in equation 3.25, the lengths of the editing paths will be roughly the same. This means that the accuracy lost by post-normalisation of the raw edit distance may not be very great, and it may be reasonable to use Wagner and Fischer's original algorithm, which is $O(C^2)$, instead of Marzal and

Vidal's.[3] This still leaves an algorithm which has time complexity $O(|\mathbf{V}_D||\mathbf{V}_M|C^4)$.

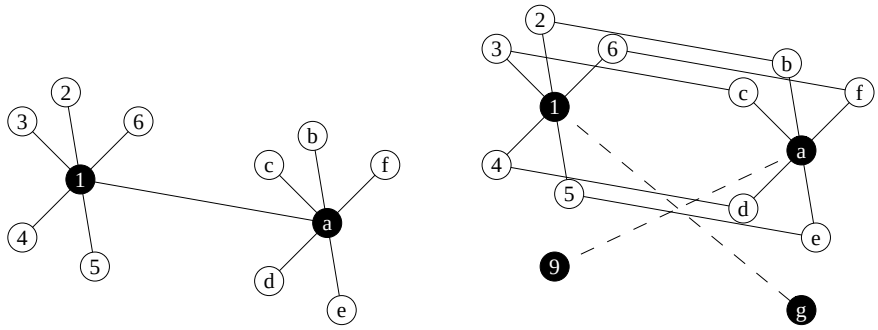
3.7.2 Accelerating the Framework Component

Neither the nature of the problem nor the choice of distance measure are really implementation choices. Graph matching problems are either exact or inexact, and for inexact matching problems the edit distance is the only generally appropriate distance measure. For exact problems, the edit distance is the same as the Hamming distance given our weighting function, so the kernel component of equation 3.26 can be evaluated in linear time. In other words, the kernel component of the labelling criterion is determined by the nature of the problem. Since one cannot change the dictionary comparison, $P(\Gamma_j|\Theta_{j,i})$, it must be regarded as the characteristic operation of the evaluation of the labelling criterion. The framework component requires $O(|\mathbf{V}_D||\mathbf{V}_M|)$ dictionary comparisons.

Although the consistency criterion in equation 3.9 is defined in terms of labellings of supercliques in the data graph, it is the labelling of the nodes themselves rather than the supercliques that is meaningful in the context of graph matching. The central idea behind Wilson's use of supercliques is that if two nodes match, their neighbours should also match (Wilson 1995). However, the summation in equation 3.26 is over all feasible mappings between supercliques, not just those with matching centre nodes. That is, all model graph supercliques are considered potential matches for each data graph superclique. It is possible, however, to take the view that only mappings between supercliques with matching centre nodes are feasible. This is shown in figure 3.4. Although there only appears to be one mismatch in panel (b), the labellings of the supercliques of nodes 2, 3, 4, 5 and 6 are also affected. Nevertheless, the criterion in equation 3.26 would possibly consider such a mislabelling tolerable in this particular context.

If only those mappings between supercliques with matching centre nodes were considered, it would suffice to compare the configuration, Γ_j, with the dictionary, $\Theta_{j,f(j)} \in \Theta_j$, where $f(j)$ is the label applied to j. Equation 3.26 can now be written without the sum over the model graph supercliques as

[3] Ukkonen describes an even faster edit distance algorithm in (Ukkonen 1985) which is $O(\mathbf{d}C)$. However this improvement is offset by more expensive initialisation. In practice, it has been found that Wagner and Fischer's algorithm is acceptably fast.



(a) Matched Centre Nodes (b) Mismatched Centre Nodes

Figure 3.4: Superclique Mappings. The relational image of data graph superclique $< 1, 2, 3, 4, 5, 6 >$ is to be compared to model graph superclique $< a, b, c, d, e, f >$. The framework of equation 3.26 would consider both (a) and (b) feasible. This may permit the mislabelling of the centre node in (b) (dashed line). The alternative is to consider (a) feasible but (b) infeasible.

$$P(\Gamma) = \frac{1}{|\mathbf{V}_D|} \sum_{j \in \mathbf{V}_D} \frac{1}{|\Theta_j|} P(\Gamma_j | \Theta_{j,f(j)}) \tag{3.28}$$

which only requires $O(|\mathbf{V}_D|)$ dictionary comparisons.

3.8 Experiments

Three relational matching criteria were evaluated. The first was the criterion in equation 3.9 with padded dictionaries. This is the one originally used by Wilson in (Wilson 1995). The second was the criterion in equation 3.28, dubbed the "neighbourhood approximation", and the third was the neighbourhood approximation with edit distance instead of dictionary padding. The aims of these experiments were (1) to establish the sensitivity of the criteria to relational corruption and initialisation errors, (2) to compare the times required to evaluate the criteria, and (3) to demonstrate the new criteria on an uncalibrated stereo matching task. These are discussed in sections 3.8.1, 3.8.2 and 3.8.3.

3.8.1 Sensitivity

Synthetic 50-node nearest-neighbour graphs with node attributes drawn from a uniform distribution were used. To simulate the effects of errors in feature detection, nodes were

randomly added and deleted, and the graphs re-triangulated. Gaussian noise was added to the node attributes until approximately 50% of the nodes would be misclassified by a simple Gaussian classifier. The results are shown in figure 3.5, from which it is clear that the neighbourhood approximation of equation 3.28 performs at least as well as Wilson's original criterion of equation 3.9. The slight improvement in matching accuracy is probably because the neighbourhood approximation is more directly concerned with matching individual nodes than whole supercliques. More importantly, the figure shows that the edit distance method comfortably outperforms the padded dictionary approach.

Sensitivity to initialisation error was examined by adding varying amounts of Gaussian noise to the node attributes of uncorrupted graphs, so that the proportion of correct mappings in an initial guess furnished by the Gaussian classifier ranged from 10% to 90%. The results are given in figure 3.6. The edit distance method is more sensitive to initialisation error than either of the padded dictionary approaches. This may be because when the graphs have the same numbers of nodes, the estimate of P_ϕ for the padded dictionary approach will be zero. Thus, null mappings will not be tolerated. However, the edit distance approach ignores P_ϕ and will treat a null mapping the same as an ordinary mismatch, insertion or deletion. In other words, the distribution rule in equation 3.24 is a little naïve in that it fails to distinguish null mappings. When a similarly naïve distribution rule is used for the padded dictionary criteria in place of equation 3.16, they are unable to handle either initialisation error or relational corruption.

3.8.2 Timing

To demonstrate the efficiency of the new matching criteria, random matches between graphs of differing sizes were evaluated. The theoretical complexities of the criteria have already been established in section 3.7. Panel (a) of figure 3.7 shows the advantage of removing a linear factor from the framework component. Panel (b) of the same figure shows that even when the permitted size difference between matching supercliques is as little as 2, the padded dictionary approach becomes more expensive than using edit distance.

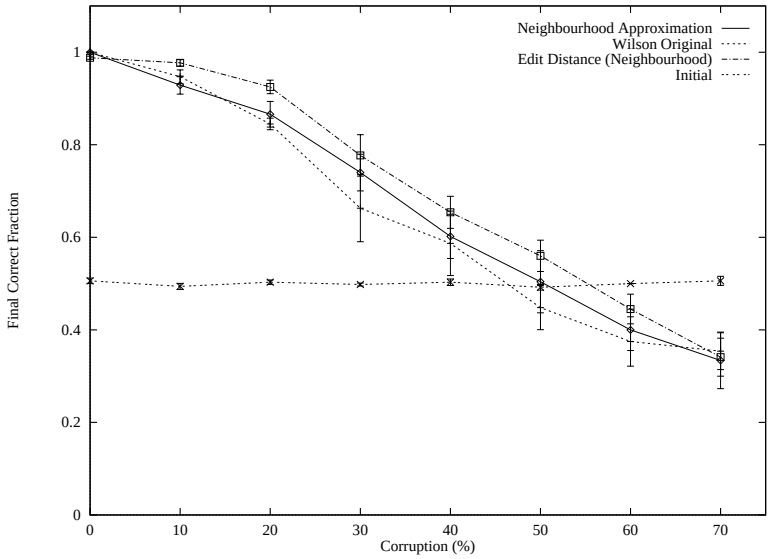


Figure 3.5: Sensitivity to Corruption. The neighbourhood approximation is no worse than the original criterion. The edit distance criterion has the best performance.

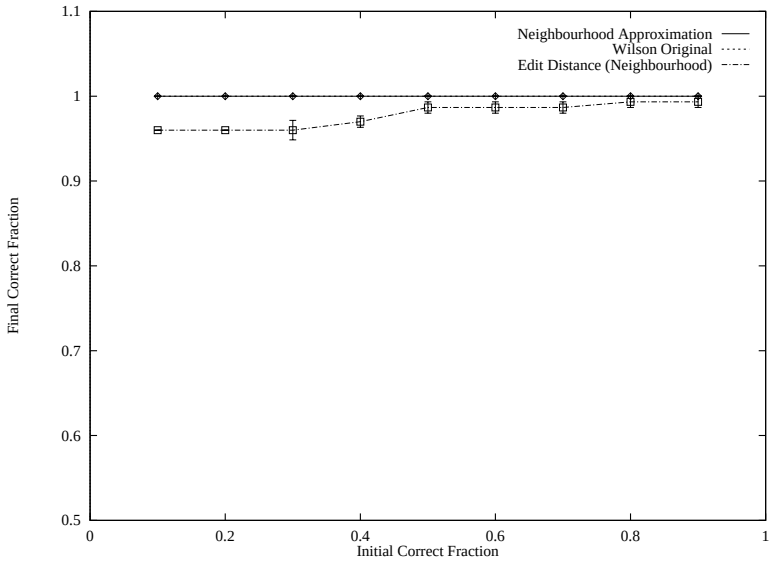
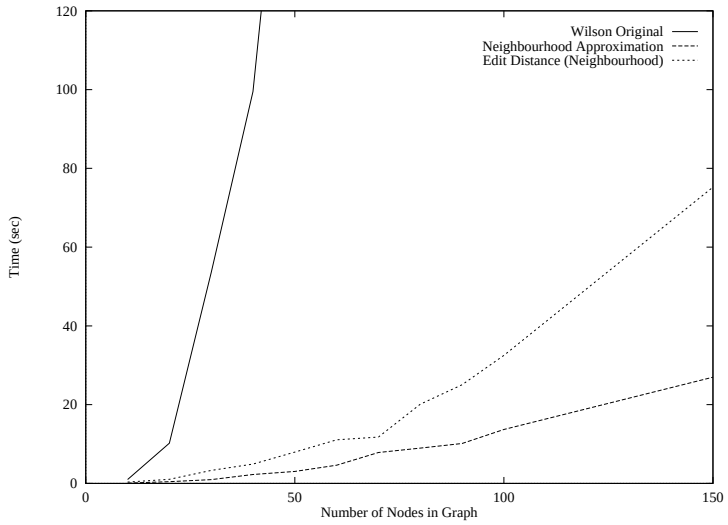
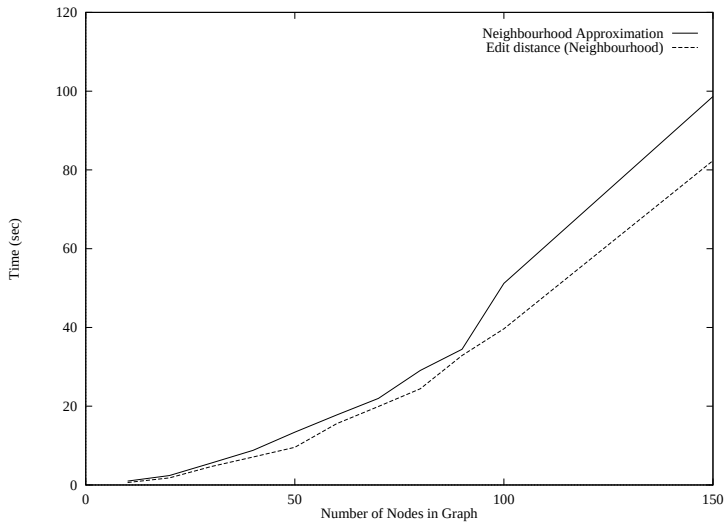


Figure 3.6: Sensitivity to Initialisation Error. The edit distance criterion is more sensitive to initialisation errors than the other two. It is nevertheless still capable of repairing significant initialisation error. Criteria were tested on equal-sized graphs.



(a) Superclique Size Difference ≤ 1



(b) Superclique Size Difference ≤ 2

Figure 3.7: Time Required for Evaluation of the Criteria. (a) Superclique size differences must be no greater than 1. The neighbourhood approximation is clearly worthwhile. (b) Size differences up to 2 are tolerated, and the edit distance criterion becomes more efficient than the padded dictionary approach.

3.8.3 Real Images

The practical applicability of these graph matching criteria was demonstrated on a wide baseline uncalibrated stereo matching problem. The lack of camera calibration makes this much more difficult than the standard stereo correspondence problem. Corners were extracted from a greyscale image pair (an office scene taken with an IndyCam) using the SUSAN corner detector (Smith and Brady 1995). Each image produced about 70 corners. The corner sets were Delaunay triangulated using Triangle (Shewchuk 1996). The grey level at each corner was used for the attribute information. This is hardly the most robust choice but the illumination is fairly constant between the two scenes. A better choice of feature is considered in the next chapter. The match was initialised using a naïve winner-take-all method. The only difference between the criteria was the time taken to reach the solution. The neighbourhood approximation took about 17 seconds on an SGI Indy workstation (133MHz MIPS R4600/4610 processor); equation 3.9 took around 19 minutes. Results are given in figure 3.8 which shows the two images with detected corners, the initial guess based on the attributes, and the results of applying the edit distance criterion. There are 70 corners in the left image of which 67 have feasible correspondences in the right image. The initial guess assigns 33 (47%) correctly. The edit distance criterion increases the number of correct assignments to 66 (99%).

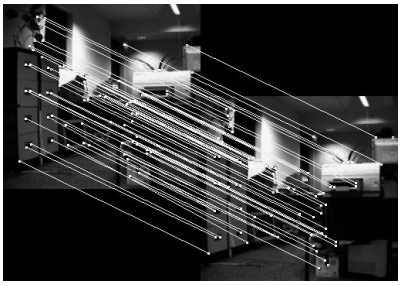
3.9 Summary

This chapter cast line labelling and attributed relational matching as consistent labelling problems which can be solved by optimisation of measures of labelling consistency. Two such measures were presented: first, Hancock and Kittler's original consistency criterion, which compares a labelling configuration against all dictionary items, and second, a linear sum of minimum distances, which considers only the dictionary item closest to the configuration. It was shown that where the Bayesian criterion is an arithmetic mean, approximating the corresponding geometric mean leads to the linear criterion. This enables control schedules established for the Bayesian criterion to be applied to the linear one. The linear criterion lends itself directly to stochastic optimisation, whereas the Bayesian one would better suit deterministic schemes.

Wilson's dictionary based approach to inexact relational matching was reformulated in

(a) Uncalibrated Stereo Pair

(b) Initial Guess



(c) Final Match

Figure 3.8: Performance on Uncalibrated Stereo Matching. The number of correct mappings is increased from an initial guess of 33/67 to 66/67.

terms of the edit distances between label configurations and dictionary items. This reduced the worst case time and space complexities of criterion evaluation by avoiding dictionary padding, and obviated the need for an extra parameter. Additionally, the neighbourhood approximation to the matching criterion was shown to be more efficient by a linear factor. Although primarily aimed at graph matching, this approach can in principle be applied to any inexact labelling problem.

It was shown experimentally that the edit distance based dictionary comparisons are more accurate with respect to relational corruption, and more efficient, than those involving padded dictionaries. The edit distance based method is, however, slightly more sensitive to initialisation error. The neighbourhood approximation was found to significantly improve algorithm running times without sacrificing accuracy.

Chapter 4

Genetic Algorithms and Ambiguous Labelling

This chapter explores the issue of ambiguity in labelling problems. Consistent labelling problems as posed in the previous chapter may have more than one solution. The novel aspect of the work reported here lies primarily in the use of genetic algorithms to provide a least commitment framework for labelling problems. Additionally, a new approach is taken to the use of measurement information in graph matching. Two novel, metric based crossover operators are also considered.

According to Marr's principle of least commitment (Marr 1982), the best way to handle ambiguous labelling problems is to simultaneously entertain several hypotheses until there is sufficient evidence to drop all but one. As a population based global optimiser, the genetic algorithm would seem to be the ideal framework for solving ambiguous labelling problems. When faced with an ambiguous problem, the algorithm must not only perform well as an optimiser, which is a *sine qua non*, but must also produce adequate solution yields. These two goals are different, and may even be in conflict. The main aim of this chapter (and the next) is to achieve good optimisation performance, consistent with reasonable solution yield.

The genetic algorithm will be applied to ambiguous line labelling and inexact relational matching problems. Line labelling is the easier of the two, and the genetic algorithm can be used directly to optimise the labelling consistency criterion. Graph matching is more realistic and more difficult than line labelling. The labelling criterion must be

augmented by incorporating measurement information, and the genetic algorithm requires the addition of a local search step.

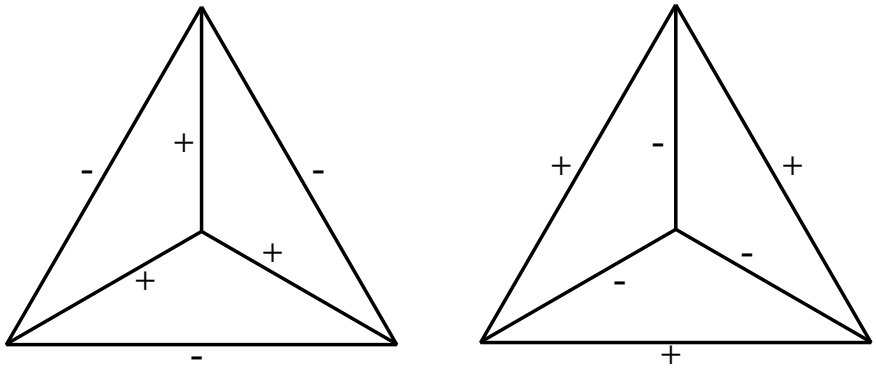
Genetic algorithms are complicated, and there are many algorithm variations, operators and parameters to choose from. The focus of this chapter is primarily on which operators are most useful for labelling problems. The main emphasis is on the local search step and the crossover operator. Consideration of algorithm parameters is limited to refuting the notion that "standard" parameter sets drawn from the literature (DeJong 1975; Grefenstette 1986) are suitable for the problems studied here. More detailed investigations of algorithm parameters and control strategies are made in later chapters.

The next section considers the ambiguities which may arise in practice with labelling problems. The use of the genetic algorithm for solving these problems is discussed in section 4.2 with emphasis on the crossover operator. Section 4.3 presents a preliminary study of the application of the algorithm to line labelling. Ambiguous relational matching is addressed in section 4.4, which adapts Wilson's original graph matching framework (Wilson 1995) to handle the ambiguous case. Non-standard crossover operators, including Cross's geometric crossover (Cross 1998) and several metric based ("optimal") crossovers, are compared with standard crossovers for graph matching.

4.1 Ambiguity

Ambiguities arise in labelling problems because the dictionaries specify constraints on local neighbourhoods without considering the global structure of the problem. Even Waltz's seminal algorithm, presented in (Waltz 1975), only takes local structure into account. Waltz recognised the issue of ambiguity but did not provide a strategy for handling it, relying on search to enumerate the possible solutions. Figure 4.1 gives an example of an ambiguous line drawing, in which the object could be a pyramid viewed from above or a cavity in a flat surface. The only way to resolve such ambiguities is to consider additional evidence, which could come either from the scene itself or from the global context. Such information may be present if the line drawing arose from a real image or scene. If the drawing is a sketch, however, no such information may be available.

Early disambiguation may be appropriate if there is compelling local evidence for a particular interpretation; but if not, the system will have to backtrack, which is generally



(a) Pyramid (b) Cavity

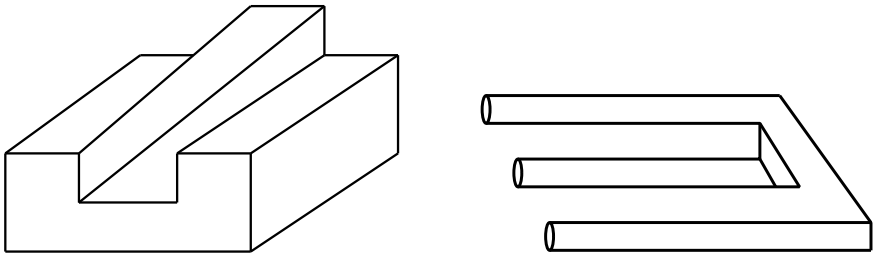
Figure 4.1: Ambiguous Line Drawing. Interpretations (a) and (b) are equally valid.

inefficient (Haralick and Elliott 1979). The use of global contextual information in scene interpretation is a major unsolved problem in machine vision but premature commitment to a particular interpretation may not be helpful. For hand-drawn sketches where there is no *a priori* evidence for a particular interpretation, or where the evidence is unconvincing, the situation could easily be worsened. Suppose that it is known that for the example of figure 4.1 the probability of interpretation (a) is 0.51. A labelling process that used such information to minimise the Bayes risk would always choose interpretation (a), and thus be wrong nearly half the time. In such cases, the initial stage of scene interpretation should follow the principle of least commitment by suggesting several plausible, and perhaps related, solutions from which some higher level process can choose without having to backtrack. In other words, where there is no convincing evidence in favour of any particular interpretation, a labelling process should not discriminate between competing consistent labellings, but should present as many as possible to higher level processes.

4.1.1 Inexact Problems

The principle of least commitment can also be applied to inexact problems in which there is no globally consistent interpretation. These problems are ambiguous in the sense that it is not always possible to identify a unique best labelling. In line labelling, such problems correspond to an interesting class of drawings of "impossible objects", which partly motivated Huffman's original investigation in (Huffman 1971). Figure 4.2 gives two examples of impossible line drawings. In such cases, any labelling is guaranteed to

be inconsistent. Even if the consistency criterion has a unique global optimum, this may not correspond to the best interpretation, since the syntactic constraints on which the dictionaries are based have been violated. It is therefore more important in this case than before, that the labelling process produce several alternative interpretations. Waltz filtering cannot be brought to bear on these problems because it would result in empty dictionaries for certain neighbourhoods, so one is left having to consider the entire search space.



(a) From (Hancock 1994) (b) "Devil's Pitchfork"

Figure 4.2: Impossible Objects. Such drawings are often disturbing to the eye.

Impossible objects are interesting curiosities, but it is difficult to see how drawings such as those in figure 4.2 could arise in practice. Segmentation errors in line-finders are more likely to fail to join lines to junctions, or join too many lines, than to construct objects such as those shown in the figure. Nevertheless, drawings of impossible objects could be input to a system designed to interpret sketches.

A more realistic inexact problem is matching point sets derived from image features. This problem was cast as attributed relational matching by Wilson in (Wilson 1995). Wilson's graphs were Delaunay triangulations of the point sets. The attributes were measurements made on the features. Wilson showed that combining the measurement information with the structural matching criterion from chapter 3 enables such point sets to be matched with considerable accuracy. Wilson used the orientation of extracted features as node attributes, and assumed that these measurements were sufficiently diverse to allow unambiguous assignments. One example concerned T-junctions, which were extracted from aerial infrared images of road networks: it is unlikely that any two such T-junctions will share the same orientation.

The problems considered in this thesis do not generally produce unambiguous measurements. Take for example the case of the stereogram in panels (a) and (b) of figure 4.3,

in which the images are to be matched by comparing the Delaunay triangulations of the centroids of regions segmented from the images. The surfaces in this example are sufficiently far from the camera for texture to be unimportant. Assuming that the light source is diffuse, it is reasonable to expect that the intensities of different parts of the image are insensitive to small changes in camera position. This is confirmed in figure 4.4. Panel (c) shows the average grey levels of each region. The overlap between left and right image measurements indicates that the images are matchable. However, the overlap between left and left, and right and right, image attributes suggests that unambiguous assignments will not be possible. This topic is considered in more detail in section 4.4.

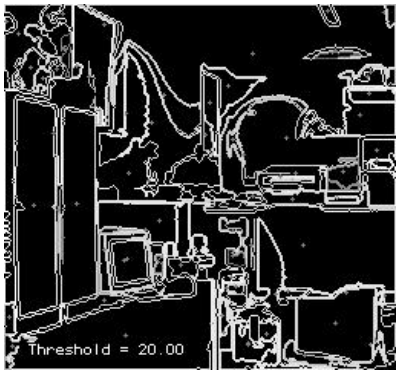
(a) Left Image (b) Right Image

Figure 4.3: Uncalibrated Stereogram. The camera positions are not known.

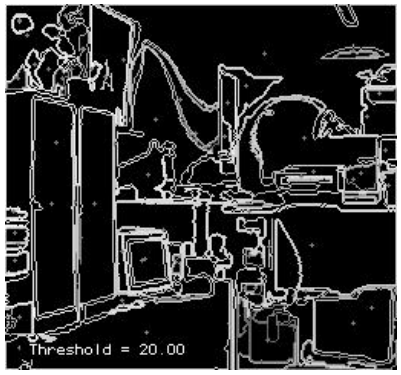
4.2 Genetic Algorithms

The genetic algorithm appears to be an attractive way of solving ambiguous labelling problems, since it maintains a population of candidate solutions, which are free to develop in any way as long as they satisfy the labelling constraints. It is hoped that the algorithm will produce several solutions to the same problem in a single program run, i.e. given the same starting point. Before considering the algorithm's ability to produce multiple optima, however, it is necessary to apply the algorithm to the problem, and then to verify that it gives adequate optimisation performance.

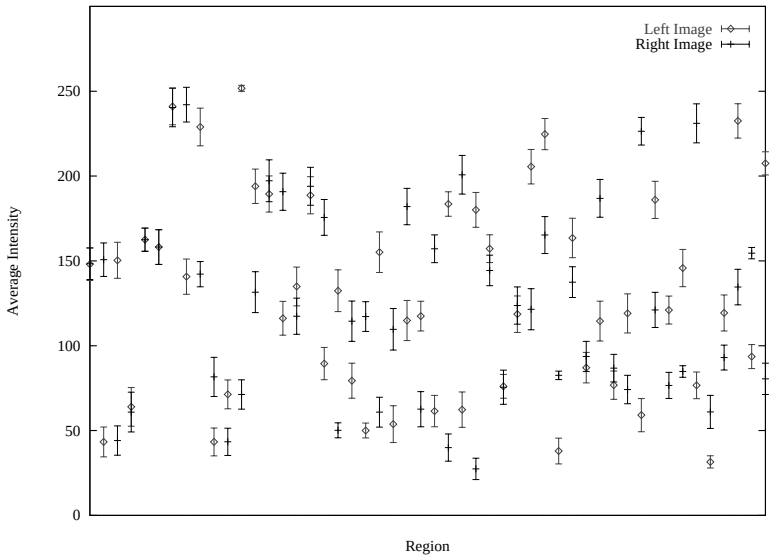
This section considers appropriate encodings and genetic operators for labelling problems. An attractive feature of genetic algorithms is that only the chromosome encoding and fitness evaluation are necessarily problem-specific: the crossover and mutation operators



(a) Left Image Regions



(b) Right Image Regions



(c) Average Region Intensities

Figure 4.4: Results of Region Segmentation. A simple segmentation technique was used.

are syntactic, and the phenotype required by selection is abstracted in the fitness function. In practice, however, it may be better to use crossover and mutation operators that take problem knowledge into account, such as the optimal crossover operators considered at the end of section 4.4.

4.2.1 Encoding

Recall from the introduction to chapter 3 that a labelling is a list of labels drawn from the label set, $\mathbf{\Lambda}$, applied to each of the objects in $\mathbf{V}$, i.e., $\Gamma =< \lambda_1, \lambda_2, ..., \lambda_{|\mathbf{V}|} >$. The standard encoding used in genetic algorithms is binary (Holland 1975; Goldberg 1989;

Mitchell 1996), in which each label, λ, would correspond to a sequence of $\lceil \log_2 |\mathbf{\Lambda}| \rceil$ bits in the chromosome. There are two problems with this standard scheme. The first is that, if $|\mathbf{\Lambda}|$ is not a multiple of 2, there will be certain bit patterns which do not correspond to labels. For example, suppose that there are five labels, and thus three bits per label in the encoding. Bit patterns `000`, `001`, `010`, `011`, and `100` will be meaningful, but patterns `101`, `110`, and `111` will not be. This creates significant difficulties in the implementation of the mutation and particularly the crossover operators. These operators should be unbiased, and yet must be constrained to produce only legal bit combinations. The second difficulty is that labels are usually semantically indivisible: it makes no sense to consider crossover between label pairs (e.g. what is the result of combining the labels "+" and "−" in line labelling? or nodes 10 and 5 in graph matching?). Thus, crossover should be constrained only to operate at the boundaries between labels, which is tantamount to using a non-binary encoding in the first place. The implementation is much more natural if the requirement for binary encoding is dropped, as Michalewicz among others suggests in (Michalewicz 1996). Each chromosome will be a labelling - i.e. it will consist of a string of labels. Unless otherwise stated, the ordering of the genes will be arbitrary. Mutation and crossover now act at a genic level and cannot produce illegal labellings, although they may still produce inconsistent ones.

4.2.2 Crossover and Local Search

The standard crossover operators are uniform and multi-point crossovers. Uniform crossover swaps parental genes with probability 0.5 at each locus; multi-point crossover exchanges sequences of genes between crossover points. These operators can be thought of as belonging to two classes, contiguous and non-contiguous. Contiguous crossovers preserve substantial sections of the parent chromosomes whereas non-contiguous crossovers do not. Figure 4.5 shows this for two point (contiguous) and uniform (non-contiguous) crossovers. It should be noted that for two-dimensional problems, there is good reason to assume that the chromosomes are circular rather than linear, because the "first" gene on the chromosome might be just as close to the "last" as it is to the "second" in the original data. These very different operators reflect differing views in the genetic algorithm literature as to whether it is better to proceed by slowly assembling partial solutions of high quality, in which case the more conservative contiguous crossover would be preferred, or whether it is better to aggressively disrupt chromosomes in a broader exploration of the search space.

In (Eshelman 1991), Eshelman defined "half-uniform" (HUX) crossover, in which exactly half of the differing parental genes are exchanged. This can be seen as a limit which uniform crossover approaches as the chromosomes get longer. Similarly, as the number of crossover points increases, multi-point crossover becomes more like uniform.

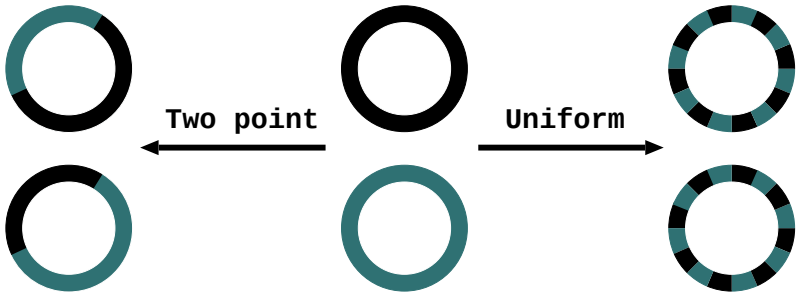
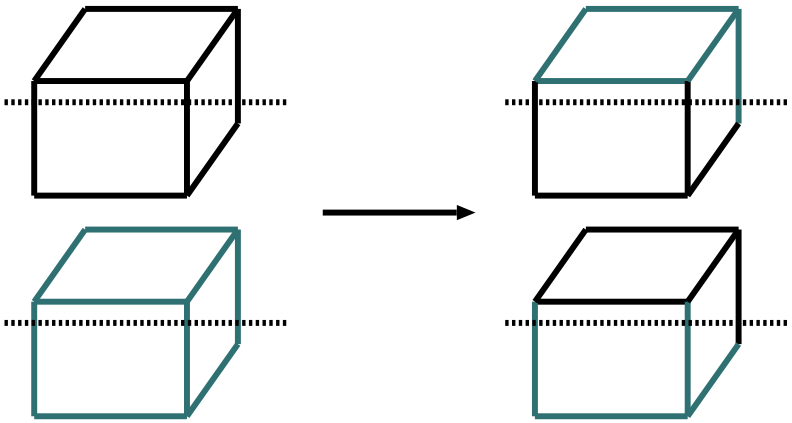


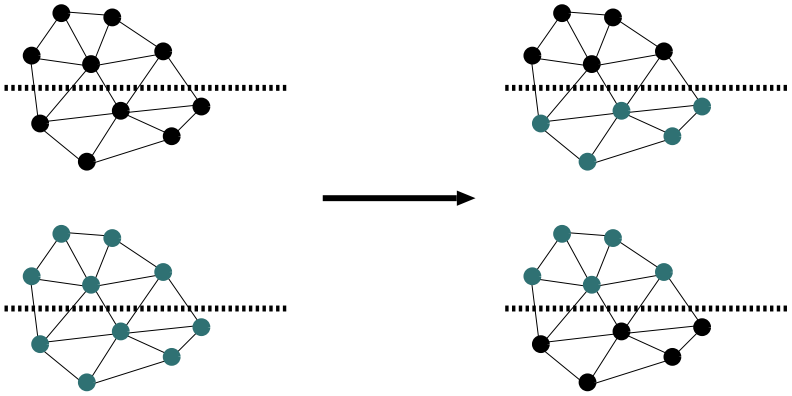
Figure 4.5: Standard Crossovers. Two point is contiguous, uniform is not.

In (Cross 1998), Cross addressed an important shortcoming in the standard crossover operators when applied to two-dimensional problems. Adjacent genes are strongly linked under two point crossover in the sense that they are highly likely to appear in the same offspring. Distant genes are weakly linked in this sense. However, in a problem with two-dimensional structure such as graph matching, there is no guarantee that nodes connected by an edge will be strongly linked in the chromosome. This means that the supposedly conservative two point crossover may well disrupt the labellings of supercliques, which is inconvenient. Cross proposed "geometric crossover", in which the labels of nodes lying on either side of a random line drawn through the centroid of the graph are exchanged. This crossover is easily generalised to line drawings by exchanging labels of lines whose centres fall on either side of the crossover line. Figure 4.6 gives an example of geometric crossover for line drawings and graphs. Thus, for two-dimensional problems, geometric crossover should be considered as an alternative to two point crossover in cases where a conservative operator is required.

Several authors have suggested the addition of a local search step to the algorithm (Goldberg 1989; Davis 1991; Whitley et al. 1995; Cross et al. 1997). Such an augmented algorithm is called a "hybrid genetic algorithm". The local search step improves individuals deterministically prior to selection and is seen as a more efficient alternative to stochastic exploration of the search space by crossover and mutation. Such a step would seem ideal for labelling problems since gradient ascent is a standard technique for solving them in an optimisation framework (Hummel and Zucker 1983; Faugeras and Berthod 1981; Hancock and Kittler 1990a; Wilson and Hancock 1997). The major drawback of gra-



(a) Line Drawing



(b) Graph

Figure 4.6: Geometric Crossover. Two-dimensional structure is preserved by this operator.

dient ascent, that it gets trapped in local optima, is no great disadvantage in the context of a global optimiser such as the genetic algorithm.

Because the constraints in labelling problems are local, gradient ascent should be expected to establish regions of local consistency. Cross took the view in (Cross 1998) that crossover should be conservative to avoid disrupting these regions, which motivated the development of geometric crossover. In this paradigm, the genetic algorithm is seen as assembling globally optimal solutions from the results of the gradient ascent step. In other words, gradient ascent provides "initial guesses" for the genetic algorithm, which serve as "building blocks" for future solutions. Holland originally proposed the building block hypothesis for the plain genetic algorithm in (Holland 1975). Although this hypothesis is plausible for the plain genetic algorithm, it may not be valid for the hybrid genetic

algorithm. In the population of a hybrid genetic algorithm, individuals with regions of local consistency are by definition in local optima. Therefore, it could be argued that they should be disrupted as much as possible in order to allow the algorithm to proceed to the global optimum. Crossover only affects regions of disagreement between parent chromosomes. If it is assumed that the number of local optima greatly exceeds the number of global optima, it is arguable that these regions of disagreement are more likely to represent local than global optima. This argument suggests a second paradigm, in which the gradient ascent step can be seen as improving on the "guesses" made by the genetic algorithm. These two paradigms represent opposing views of the relative importance of the genetic algorithm and the gradient ascent step. The first holds that the hybrid algorithm is really a genetic algorithm with modifications to improve efficiency; the second holds that the hybrid algorithm is gradient ascent in a framework that enables it to escape local optima. If the second view is more accurate, it may not be the case that contiguous crossovers should be preferred in hybrid algorithms. This consideration is explored further in sections 4.3 and 4.4, and in chapter 5.

4.2.3 Selection

The most common selection mechanism is fitness-proportionate selection, introduced by Goldberg in (Goldberg 1989), in which individuals are selected with replacement from the population with probabilities proportional to their fitness - i.e., in a population, $\mathbf{\Psi}$, the probability of selecting a labelling, Γ, is proportional to its fitness, f_Γ,

$$p_\Gamma = \frac{f_\Gamma}{\sum_{\gamma \in \mathbf{\Psi}} f_\gamma} \tag{4.1}$$

For labelling problems, the criterion in equation 3.9 from chapter 3,

$$P(\Gamma) = \frac{1}{|\mathbf{V}|} \sum_{j \in \mathbf{V}} \frac{K_{C_j}}{|\Theta_j|} \sum_{S_i \in \Theta_j} \exp\left[-k_e D(\Gamma_j, S_i)\right] \tag{4.2}$$

can be used directly for the fitness. Alternatively, the criterion from equation 3.13,

$$E(\Gamma) = \sum_{j \in \mathbf{V}} \min_{S_i \in \Theta_j} D(\Gamma_j, S_i) \tag{4.3}$$

could be used to measure the cost of the labelling, $E(\Gamma)$. This cost can be converted to a fitness in several ways, but the most appealing is to define $f_\Gamma = \exp[-\beta E(\Gamma)]$, as in Boltzmann selection (Goldberg 1990; Prügel-Bennett and Shapiro 1994). The parameter, β, is regarded as inverse temperature by Prügel-Bennett and Shapiro in (Prügel-Bennett and Shapiro 1994), but can equally well be considered a scale parameter which determines the strength with which labelling constraints are enforced. In the work reported in this chapter, β is fixed at 1.0.

In (Rudolph 1994), Rudolph showed that the genetic algorithm could only be guaranteed to converge if Grefenstette's "elitist" selection heuristic was used. Elitist selection simply guarantees that the best individual in the population is selected for future reproduction.

4.3 Line Labelling

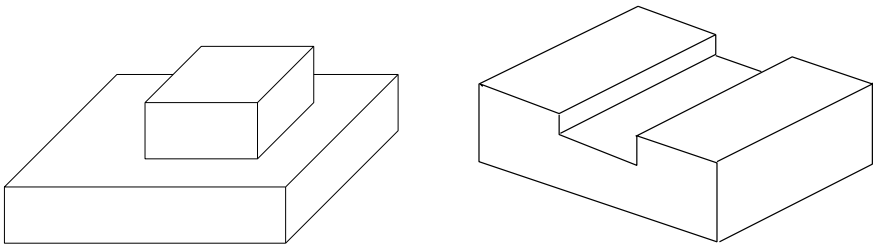
The primary aim of the preliminary study reported in this section was to investigate the suitability of the genetic algorithm as a framework for simultaneously obtaining multiple solutions to ambiguous labelling problems. A secondary aim was to consider suitable population sizes, crossover rates and mutation rates.

The algorithm was tested on two labelling problems with and without gradient ascent. Several different parameter sets were tried. The number of iterations required to find a solution and the solution yields were recorded. The algorithm was compared to multiple restarts of gradient ascent.

No timing data for the algorithm are given because first, such data are generally highly implementation-dependent, and second, the main concern here is not algorithm efficiency. Suffice to say that G generations of a genetic algorithm with population size P running on a single processor will require $O(PG)$ crossovers and mutations, both of which operations scale linearly with problem size. However, the characteristic operation of the algorithm will probably be fitness evaluation in this case, which is polynomial in problem size as shown in section 3.7 of chapter 3. In the case of the hybrid algorithm, the characteristic operation is the combined gradient ascent and evaluation step, which makes a number of fitness evaluations quadratic in the problem size.

4.3.1 Method

The algorithm used was a slight variation on the standard genetic algorithm. At every iteration, all individuals take part in recombination with probability equal to the crossover rate. This models the panmitic populations often observed in nature. Individuals taking part in recombination may also undergo mutation. This algorithm allows the population to expand transiently to up to three times its original size before the selection step, which reduces the population size to its base value. The initial population is created at random. The algorithm terminates after a set number of iterations regardless of whether any solutions have been found.



(a) "Wedding Cake" (b) "Groove 2"

Figure 4.7: Test Drawings.

The algorithm was tested using the problems shown in figure 4.7. These problems can be made arbitrarily larger by adding disconnected copies; this is reasonable because the algorithm does not "know" that the two drawings are identical: it just sees more lines. The local nature of the constraints means that disconnected copies are almost as difficult as connected copies. In the work reported here, two copies of each drawing had to be labelled.

This study uses multi-point crossover instead of geometric crossover, which raises the issue of how to maintain strong linkage between loci on the chromosomes which correspond to lines which occupy the same region of the drawing. This is not a problem with synthetic data, since humans have a natural tendency to segment line drawings and number junctions and arcs accordingly. Thus, synthetic data can be primed subconsciously to yield solutions. However, the same is not true of real world data, such as edge-detector output. This study uses the following heuristic to number the arcs. In general, TEE junctions represent occlusions of part of the scene by an overlying plane (Huffman 1971). A **crude** segmentation can be achieved by numbering the arcs depth-first, backtracking at TEE

junctions. For our drawings, this makes strongly linked loci in the chromosome map to broadly similar regions in the drawing. However, the inverse relation does not necessarily hold.

	Set L1 (DeJong 1975)	Set L2 (Grefenstette 1986)
Population size	100	30
Crossover type	2 point	Uniform
Crossover rate	0.6	0.9
Mutation rate	0.001	0.01

Table 4.1: Parameter Sets from the Literature.

	Set A	Set B	Set C	Set D	Set E
Population size	100	100	100	100	100
Crossover type	Uniform	Weighted	HUX	1 point	2 point
Crossover rate	0.9	0.9	0.9	0.9	0.9
Mutation rate	0.03	0.03	0.03	0.03	0.03

Table 4.2: Additional Parameter Sets.

Five different crossover operators were considered: uniform, half-uniform, one point and two point crossovers have already been mentioned. "Weighted" crossover is uniform crossover in which the exchange probability depends on the fitness of the parents. Thus, one offspring receives a preponderance of genes from the better parent, and vice-versa. As mentioned in section 4.2.2, HUX crossover deterministically exchanges exactly half the differing parental genes (Eshelman 1991). Population size, and crossover and mutation rates for the genetic algorithm are notoriously difficult to set (Schaffer et al. 1989). The literature recommends two alternative parameter sets as shown in table 4.1. These parameters are based on the standard suite of test problems for the genetic algorithm developed by DeJong in (DeJong 1975). Several other sets were also tried, as shown in table 4.2. The number of iterations was fixed at 1000 for the non-hybrid algorithm and 10 for the hybrid. Results were obtained over 1000 trials.

4.3.2 Performance

The results are summarised in tables 4.3 and 4.4. The most striking feature of these results is the superiority of the hybrid algorithm. The figures for average solution yield

suggest that about 20% of the population will consist of distinct optimal solutions, given a decent set of parameters. The plain genetic algorithm does not seem to be suitable for handling ambiguous problems. Two point crossover performed better than one point, which is as expected given circular chromosomes, because in this case one point is just a special case of two point, with one crossover point fixed at the start of the chromosome. Another interesting finding is that the parameter sets taken from the literature (L1 and L2) performed poorly. This is perhaps not surprising since these parameter sets were derived for numerical optimisation rather than labelling problems. There is clearly scope for optimising these parameter sets further.

		Set L1	Set L2	Set A	Set B	Set C	Set D	Set E
	c	2.30%	17.8%	29.3%	30.2%	30.4%	35.5%	38.8%
Plain	$\bar{y}$	0.06	0.54	2.10	2.27	1.87	3.17	3.45
	$\bar{g}$	595	528	281	269	305	237	245
	c	99.2%	76.1%	99.4%	97.8%	99.2%	100%	100%
Hybrid	$\bar{y}$	17.0	3.34	17.3	13.5	17.6	25.2	33.0
	$\bar{g}$	2.47	3.45	2.37	2.54	2.34	2.29	2.22

Table 4.3: Results for the Wedding Cake problem. c is the proportion of trials yielding consistent labellings, $\bar{y}$ is the average solution yield over all trials, and $\bar{g}$ is the average generation at which the first solutions are found.

		Set L1	Set L2	Set A	Set B	Set C	Set D	Set E
	c	3.80%	23.3%	38.3%	37.4%	33.3%	42.6%	42.9%
Plain	$\bar{y}$	0.04	0.34	1.02	0.99	0.80	1.11	1.10
	$\bar{g}$	687	508	230	270	250	244	224
	c	98.6%	75.9%	99.2%	99.4%	98.4%	99.9%	99.9%
Hybrid	$\bar{y}$	9.78	3.23	15.1	13.4	15.3	17.8	19.8
	$\bar{g}$	2.96	4.23	2.76	2.77	2.77	2.47	2.61

Table 4.4: Results for the Groove 2 problem. c is the proportion of trials yielding consistent labellings, $\bar{y}$ is the average solution yield over all trials, and $\bar{g}$ is the average generation at which the first solutions are found.

The fact that the most convincing results were produced when the algorithm was augmented by gradient ascent might suggest that the rôle of the genetic algorithm is not significant, since solutions were found on average within five iterations. However 100000

restarts of gradient ascent from the same (random) initial conditions only resulted in 84 (8%) and 59 (5%) consistent labellings for each problem. It is quite clear from this that gradient ascent is getting stuck in local optima, an escape route from which is provided by the genetic algorithm. The number of iterations required to find an optimal solution (about five) compares favourably with the 20 or so needed by the "multi-niche crowding" algorithm used by Cedeño et al. in (Cedeño et al. 1995).

4.3.3 Propagation of Interpretation

To investigate the evolution of solutions by the algorithm, the problem shown in figure 4.8 was used, together with an enhanced dictionary to permit "origami world" labellings (Kanade 1978) of the central object (thick lines) as a pyramid or a cavity.

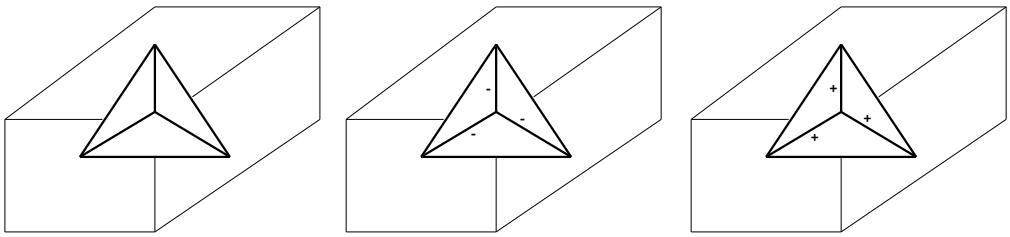


Figure 4.8: Additional Test Drawing. The thick lines could represent a pyramid or a cavity.

The solutions found tended to be invariant with respect to interpretation as pyramid or cavity. The results of a typical trial which found 11 distinct labellings are given in figure 4.9. The convex interpretation of the FORK predominates. This cannot be explained simply by the linkage of the arcs in the chromosome, since the arcs belonging to the FORK junction, (0,5,9), are less strongly linked than other groups of arcs, e.g. (2,3,4) and (13,14,15), which have variable interpretations. These observations can be understood by considering how many label changes accompany the transitions between consistent configurations for the different junction types.

It is likely that a random change in the labelling of a consistently labelled junction will yield a less good labelling. Consider an ELL junction: there are 16 combinatorial labelling possibilities, six have Hamming distances of zero from the Huffman dictionary (i.e. they are consistent), and ten have Hamming distances of one; none have Hamming distances of two. This means that a random replacement of a consistent labelling has a probability of $\frac{5}{15} = 0.\dot{3}$ of yielding another consistent labelling and a probability of $\frac{10}{15} = 0.\dot{6}$ of yielding a labelling with a single error. By contrast, a FORK junction has 64 combinatorial

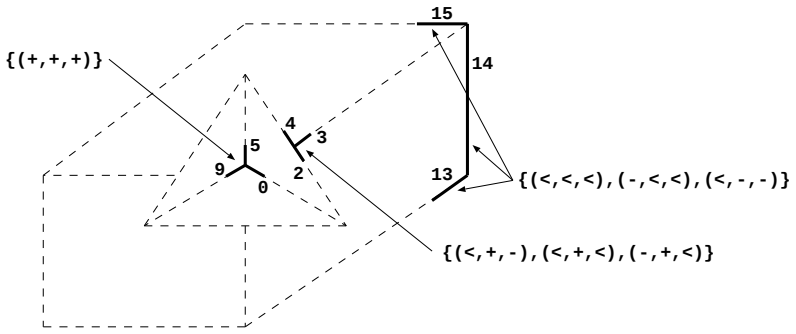


Figure 4.9: Related Labellings. Labellings of line-triples found in 11 distinct solutions.

possibilities of which five are consistent; the outcomes of a replacement of a consistent labelling are: another consistent labelling with probability $\frac{4}{63} = 0.06$, a labelling with Hamming distance one with probability $\frac{39}{63} = 0.62$, or a labelling with Hamming distance two with probability $\frac{20}{63} = 0.32$. Thus, the expectation of the Hamming distance from a consistent labelling following a labelling change is $0.\dot{6}$ for an ELL junction and 1.25 for a FORK junction, so FORKs can be said to be more strongly constrained than ELLs. We would therefore expect the labellings of FORK junctions to be relatively immune to the effects of gradient ascent, crossover and selection; and the final population will probably only contain individuals with one labelling for any particular FORK.

Figure 4.10 shows the evolution of the best labelling in the population over several algorithm iterations. Initially, all but one of the junctions are inconsistently labelled. Although consistent labellings for weakly constrained junctions such as ELLs are established early, they are subject to change in later iterations. However, once the central FORK junction is consistently labelled, its labelling never changes.

This observation is in general agreement with the findings of Trueswell and others (Trueswell et al. 1994; Kawabata 1978) with respect to the propagation of interpretations. Trueswell and coworkers have suggested that in language processing, rapid syntactic disambiguation occurs in regions of strong semantic constraint (Trueswell et al. 1994). Kawabata has suggested that a local interpretation tends to propagate when humans are faced with ambiguous scenes (Kawabata 1978). With this in mind, FORK junctions can be seen as models for strongly constrained localities which tend to dictate the interpretation of their immediate surroundings. This chimes with the notion that the alternative interpretations of a drawing should all be considered plausible *a priori*, and suggests that where evidence as to the labelling of strongly constrained regions of the scene is available, the initial

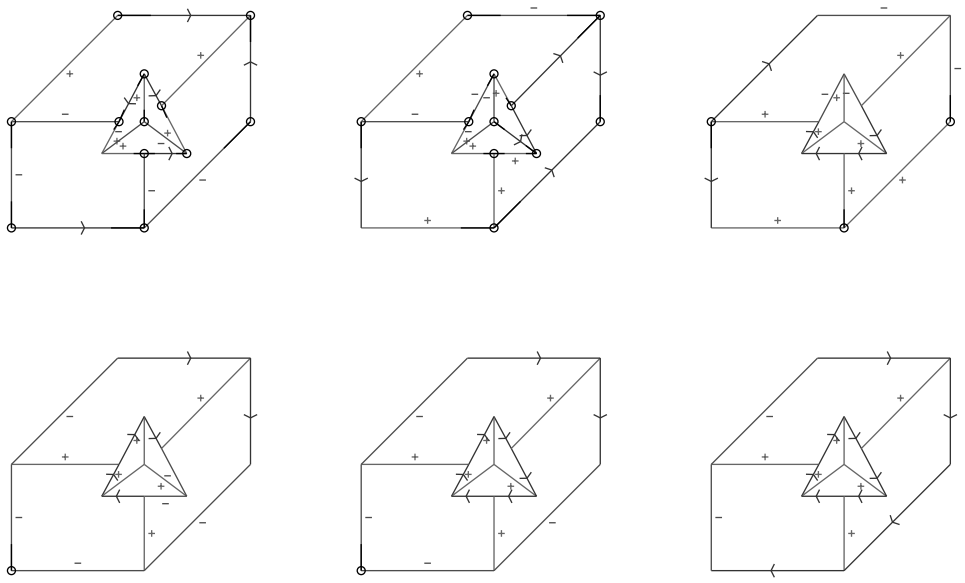
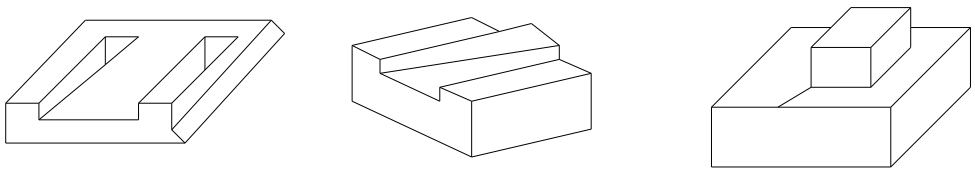


Figure 4.10: Evolution of a Labelling. Inconsistent junctions are marked with a circle.

population can be seeded accordingly.

4.3.4 Impossible Objects

To round off this study, the algorithm's behaviour for the three impossible objects in figure 4.11 was investigated. Although these objects cannot be consistently labelled, the algorithm often found maximally consistent labellings. Sample output is shown in figure 4.12.

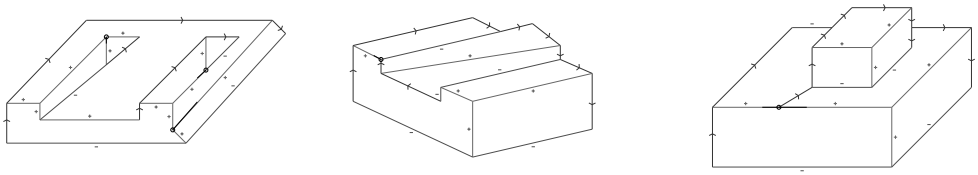


(a) From (Clowes 1971) (b) From (Hancock 1994) (c) Based on figure 4.7a

Figure 4.11: Impossible Test Drawings.

Usually, the algorithm correctly labelled all junctions, except those which prevent the drawings being realised in three dimensions. For example, panel (b) of figure 4.12 shows

that the interpretation of the drawing as some object lying in front of and partially occluding a cuboid is almost reasonable, and is only wrong in that the leftmost face of the cuboid must be simultaneously visible and occluded. Similarly in panel (c), the TEE junction must be inconsistent because no part of a scene can be occluded by a convex edge. In panel (a) the right hand pair of inconsistent junctions correspond to one's intuition about the problem with this object, but the one in the top left region of the drawing appears counter-intuitive. However, a little thought reveals that the inconsistency could indeed lie here, if one rejects the tempting hypothesis that the region in question is a pit.



(a) 3 inconsistencies (b) 1 inconsistency (c) 2 inconsistencies

Figure 4.12: Best Labellings. Inconsistent junctions are marked with a circle.

The algorithm often found more than one possible labelling for these objects. These labellings were not generally found to disagree as to where the inconsistency lay. The members of the final population of a particular run almost always agreed as to which junctions could not be labelled consistently. There was also a very high degree of agreement between program runs. In almost all cases, the differences between the distinct members of the final population were due to differing interpretations of the consistent part of the drawing; for example, whether the drawing was floating in space or anchored by one or more edges. Thus, it appears that the algorithm is capable of localising the problems in drawings of impossible objects.

4.4 Graph Matching

This section describes the application of genetic algorithms to ambiguous graph matching problems. It commences with a brief review of the unambiguous case. Section 4.4.2 gives a formulation of measurement based graph matching more suitable to ambiguous graphs. A preliminary experimental study is described in section 4.4.4, and some novel crossover operators are introduced in section 4.4.5.

4.4.1 Unambiguous Matching

Inexact matching with genetic algorithms in the unambiguous case was considered by Cross in (Cross 1998), following Wilson's work on the matching problem itself (Wilson 1995). For attributed relational matching, Wilson combined the labelling consistency criterion with measurement information to give a maximum *a posteriori* probability (MAP) update rule. The goal is to optimise the *a posteriori* probability of the labelling given the measurements:

$$P(f|\mathbf{A}_D, \mathbf{A}_M) = \frac{p(\mathbf{A}_D, \mathbf{A}_M|f)}{p(\mathbf{A}_D, \mathbf{A}_M)} P(f) \tag{4.4}$$

where f is the match between the two graphs corresponding to the labelling of $\mathbf{V}_D$, and $P(f)$ is the labelling consistency criterion from equation 3.9 in chapter 3. The joint measurement density, $p(\mathbf{A}_D, \mathbf{A}_M)$, only depends on the measurements and is thus a static property of the problem which can be ignored when comparing labellings. The conditional measurement density, $p(\mathbf{A}_D, \mathbf{A}_M|f)$, depends on both the current labelling and the measurements. Wilson showed that, assuming conditional independence of these measurements given the current labelling, the conditional measurement density can be factorised over the tuples in f to give

$$p(\mathbf{A}_D, \mathbf{A}_M|f) = \prod_{(u,v)\in f} P(u, v|x_u, x_v) \frac{p(x_u, x_v)}{P(u, v)} \tag{4.5}$$

where the posterior matching probability, $P(u, v|x_u, x_v)$, is the probability of node u from the data graph matching node v in the model graph given their measurements, x_u and x_v. Like $p(\mathbf{A}_D, \mathbf{A}_M)$, the unconditional density, $p(x_u, x_v)$, is independent of the current match, f. Assuming that the matching priors, $P(u, v)$, are uniformly distributed, equation 4.4 can be written

$$P(f|\mathbf{A}_D, \mathbf{A}_M) \propto \left[\prod_{(u,v)\in f} P(u, v|x_u, x_v) \right] P(f) \tag{4.6}$$

Recalling the relationship between the linear and Bayesian criteria from equation 3.13 in chapter 3, we could also write

$$P(f|\mathbf{A}_D, \mathbf{A}_M) \propto \left[\prod_{(u,v)\in f} P(u,v|x_u,x_v)\right] e^{-\beta E(f)} \tag{4.7}$$

Wilson used the relationship in equation 4.6 to formulate the MAP update rule given in (Wilson 1995) for iteratively improving the labelling, f. The mapping of data graph node u, was chosen from the union of the model graph node set, $\mathbf{V}_m$, with the null label, $\{\phi\}$, according to:

$$f(u) = \arg \max_{v\in \mathbf{V}_m \cup \{\phi\}} P(u,v|x_u,x_v)P(f) \tag{4.8}$$

In (Cross 1998), Cross used a genetic algorithm for graph matching, in which equation 4.6 became the fitness of the i^{th} labelling in the population, $\mathbf{\Psi}$, so that the selection probability under fitness-proportionate selection was:

$$p_i = \frac{\left[\prod_{(u,v)\in f_i} P(u,v|x_u,x_v)\right] P(f_i)}{\sum_{j\in\mathbf{\Psi}} \left\{\left[\prod_{(u,v)\in f_j} P(u,v|x_u,x_v)\right] P(f_j)\right\}} \tag{4.9}$$

In (Cross 1998), Cross also used the MAP update rule, equation 4.8, as the gradient ascent step in the hybrid genetic algorithm.

4.4.2 Measurement Ambiguity

The measurement information contributes to the matching criterion via the posterior matching probability, $P(u,v|x_u,x_v)$, which has yet to be defined. In (Wilson and Hancock 1997), Wilson and Hancock defined it in terms of the Euclidean distance between attribute pairs for non-null mappings:

$$P(u,v|x_u,x_v) = \begin{cases} P_\phi & \text{if } v = \phi \\ (1-P_\phi)\dfrac{\exp\left[\frac{-(x_u-x_v)^2}{2\hat{\sigma}_v^2}\right]}{\sum_{w\in V_M}\exp\left[\frac{-(x_u-x_w)^2}{2\hat{\sigma}_w^2}\right]} & \text{otherwise} \end{cases} \tag{4.10}$$

where P_ϕ is the prior probability of a null match, which may be taken as $2\left\|\frac{|\mathbf{V}_D|-|\mathbf{V}_M|}{|\mathbf{V}_D|+|\mathbf{V}_M|}\right\|$, and $\hat{\sigma}_v^2$ is the estimated variance of x_v. This effectively regards the model graph node measurement, x_v, as a mean about which the data graph node measurement, x_u, varies

with estimated variance $\hat{\sigma}_v^2$, under the null hypothesis that the two measurements are the same (because the nodes match). This approach requires the assumption that a data measurement is only likely to be statistically close to **one** of the model measurements. This is ideal when there is little overlap between classes, e.g. for possible angles of line-fragments segmented from a radar image. However, if there is significant overlap, e.g. in the average intensities of regions, such a scheme will not reflect these ambiguities in its classification of features.

The alternative is to compare the data measurements to the model measurements using an artificial scale. This can be done by considering the number of standard deviations separating the data measurement from its class mean under the null hypothesis that the nodes match. Table 4.5 gives an example of such a scale for the arbitrary classes "similar", "comparable", and "different".

Class	Range of standard deviations from x_v
Similar	[0,1.0]
Comparable	(1.0,2.0]
Different	(2.0,∞]

Table 4.5: Example Scale for Measurement Comparisons.

Consider the standardised distance, $\Delta_{uv} = \|x_u - x_v\|/\hat{\sigma}_v$. The probability that x_u lies within $[a, b]$ standard deviations on either side of x_v is twice the standard Normal integral from a to b:

$$\begin{aligned} P(a \leq \Delta_{uv} \leq b) &= \sqrt{\frac{2}{\pi}} \int_a^b e^{-\frac{1}{2}z^2} dz \\ &= \text{erf}\left(\frac{b}{\sqrt{2}}\right) - \text{erf}\left(\frac{a}{\sqrt{2}}\right) \end{aligned} \tag{4.11}$$

Each of the classes in table 4.5 corresponds to a separate interval which must be considered. Rather than introduce so many extra parameters, it is better to simplify the classification to "similar" if $\Delta_{uv} \in [0, a]$ and "dissimilar" otherwise. Thus, $P(u, v|x_u, x_v)$ for can be defined as follows

$$P(u,v|x_u,x_v) = \begin{cases} P_\phi & \text{if } v = \phi \\ (1-P_\phi)\frac{\exp\left[\frac{-(x_u-x_v)^2}{2\hat{\sigma}_v^2}\right]}{\sum_{w\in V_M}\exp\left[\frac{-(x_u-x_w)^2}{2\hat{\sigma}_w^2}\right]} & \text{if } a = 0 \\ (1-P_\phi)P[\Delta_{uv} \le a] & \text{if } \Delta_{uv} \le a \\ (1-P_\phi)(1-P[\Delta_{uv} \le a]) & \text{otherwise} \end{cases} \tag{4.12}$$

For convenience, the original unambiguous definition is used when $a = 0$. At the cost of an extra parameter, a, ambiguous measurements can now be handled. The important property of equation 4.12 is that when $a > 0$, it assigns the exact same probability to sets of mappings, thus enabling different alternatives to be considered.

The ambiguity parameter, a, has a direct interpretation as the number of standard units beyond which data measurements are considered dissimilar from the model. However, it has a more useful indirect interpretation in terms of "matching tolerance". The matching tolerance, T, is defined as the average proportion of model measurements similar to each data measurement, and is a function of a for any particular graph pair as shown in equation 4.13. The feasibility of mappings, feasible(u,v), is determined according to the superclique size difference constraint mentioned in section 3.7.1 of chapter 3.

$$T(a) = \frac{1}{|\mathbf{V}_D||\mathbf{V}_M|}|\{(u,v) \in \mathbf{V}_D \times \mathbf{V}_M | \text{feasible}(u,v) \wedge \Delta_{uv} \le a\}| \tag{4.13}$$

Figure 4.13 shows T as a function of a for several synthetic graphs. The tolerance reaches a plateau of between 0.4 and 0.7 at values of a higher than about 2. This plateau is the limit imposed by the feasibility constraint, feasible(u,v). These graphs suggest that it should be possible to determine an appropriate value of a by estimating the proportion of "similar" features in the data set. For example, if this proportion is estimated to be 10%, a should be about 0.5.

4.4.3 Variance Estimation

The origin of the estimated variance, $\hat{\sigma}_v^2$, of the measurement distribution for model graph node v has not yet been considered. The data graph measurements, x_u, are generally drawn directly from features extracted from the image - e.g. orientation, size, or intensity. The model will either be an abstract one such as a map, or another set of features as is the

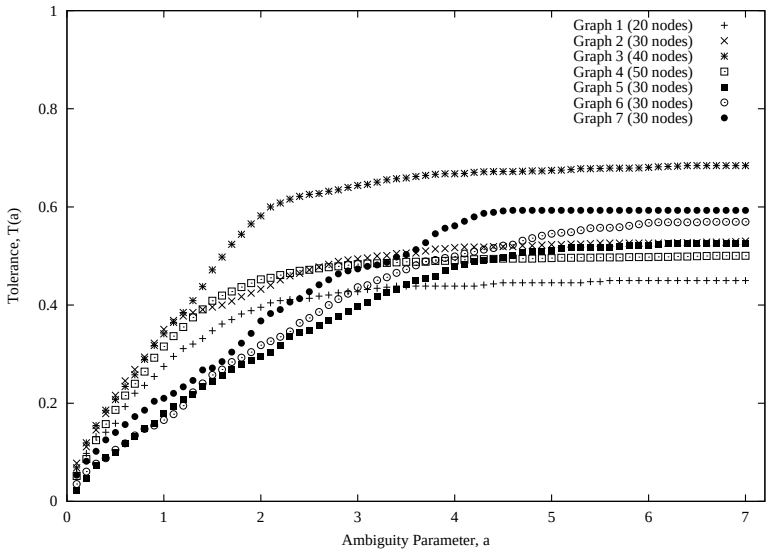


Figure 4.13: Matching Tolerance as a Function of the Ambiguity Parameter.

case with stereograms. For certain feature types, the measurement distributions will have very similar variances: for example, the angles of line segments recovered from a radar image are all subject to the same distortion process. Thus estimating σ_v^2 for each node is the same as estimating the overall variance of the noise which is the same for every node, as in (Wilson and Hancock 1997). Note that this noise variance is not the same as the measurement variance within the feature set.

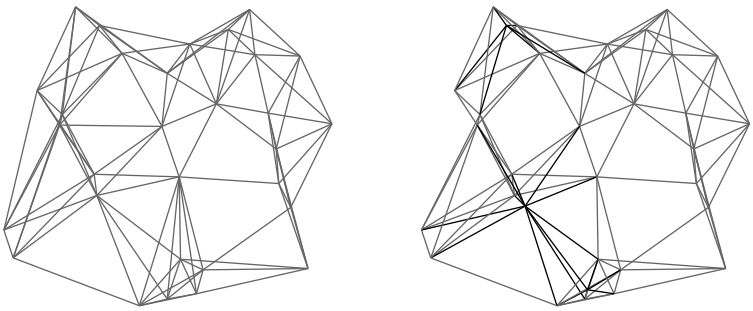
When matching pairs of extracted feature sets, both sets of measurements are subject to error, and there may be no identifiable ground truth. The classification should properly be made by comparing the measurement distributions at each node - the correct method is the t-test (see pages 321–323 of (Hays 1994)). However, it is possible to follow Cross in (Cross 1998) and regard one of the stereo pair as a model, estimating σ_v^2 from the extracted features. An important caveat of this approach is that the results will be different when each image in turn is regarded as the model. In the graph editing framework of Wilson and Hancock, it is natural to regard the smaller graph as the model (Wilson and Hancock 1997; Cross et al. 1997).

Matching regions segmented from a pair of images poses an additional problem, which is that the intensity of each region has its own distribution. For matching to work at all, it should first be verified that all the measurement distributions are unimodal. It is by no means certain that the variances will be sufficiently similar to justify using t-tests (see page 617 of (Press et al. 1992)). Again, this difficulty can be avoided by treating one

image as the model. A key assumption of the experimental work presented here, then, is that the error incurred by treating the distributions as if they were the same is negligible.

4.4.4 Initial Experiments

Like section 4.3, this preliminary study establishes the suitability of the genetic algorithm for ambiguous graph matching, and considers some control parameter sets. The algorithm was tested on four 30-node synthetic graphs like the one in figure 4.14. The point sets were generated at random, and then triangulated by connecting each point to six of its nearest neighbours. Data graphs were generated by randomly perturbing the node attributes, and then duplicating 10% of the nodes and perturbing their attributes. The intention was to simulate segmentation errors expected of region extraction, such as the splitting of one region into two similar ones. The triangulations used are generally rather more dense than Delaunay triangulations, and the addition of even 10% clutter causes more relational disruption than in the Delaunay graphs used by Cross in (Cross 1998).



(a) Original Graph (b) 10% Additional Clutter

Figure 4.14: Synthetic Graph with Corruption. Clutter is indicated by dark edges.

The same genetic algorithm as in section 4.3 was used. Since Cross found in (Cross 1998) that graph matching was not feasible with non-hybrid algorithms, this study only considers hybrid algorithms. The algorithms used were the hybrid genetic algorithm with and without mutation, crossover or both (hGA, hGA-m, hGA-x and hGA-xm),[1] a hybrid

[1]These should be regarded as different algorithms, not merely different parameter sets for a genetic algorithm, because a genetic algorithm with no crossover or mutation is fundamentally different from one which has these operators. For example, the hGA-xm algorithm is really just multiple restarts of gradient ascent with a selection step.

version of Eshelman's CHC algorithm (Eshelman 1991) (hCHC), and plain gradient ascent (HC). The experimental conditions are summarised in table 4.6. Simulated annealing was not considered: Cross found that the hybrid genetic algorithm outperformed simulated annealing; and the technique is very slow compared to the genetic algorithm.

	hGA	hGA-m	hGA-x	hGA-xm	hCHC	HC
Population size	50	50	120	120	100	1
Iteration limit	5	5	5	5	5	10
Crossover type	Uniform	Uniform	Uniform	Uniform	HUX	n/a
Crossover rate	0.9	0.9	0.0	0.0	1.0	n/a
Mutation rate	0.3	0.0	0.3	0.0	0.35	n/a

Table 4.6: Algorithms for Graph Matching. Each algorithm, apart from HC, made approximately 700,000 fitness evaluations. Abbreviations: hGA = hybrid genetic algorithm, hGA-m = hGA without mutation, hGA-x = hGA without crossover, hGA-xm = hGA with neither mutation nor crossover, hCHC = hybrid CHC, and HC = gradient ascent (hillclimbing).

Each of the algorithms listed in table 4.6, except HC, was run 100 times on the four graphs. Since HC is deterministic, it was only run once per graph. The results for the four graphs were pooled to give 400 observations per algorithm. Algorithm performance was assessed according to two criteria. The first was the average fraction of correct mappings in the final population. The second was the proportion of distinct individuals in the final population with more than 95% correct mappings. The results are reported in table 4.7.

Algorithm	Average Fraction Correct	Average Fraction Distinct
hGA	0.90 (0.0044)	0.078 (0.0019)
hGA-m	0.88 (0.0051)	0.040 (0.0012)
hGA-x	0.84 (0.0052)	0.044 (0.00094)
hGA-xm	0.76 (0.0068)	0.013 (0.00036)
hCHC	0.92 (0.0042)	0.012 (0.00033)
HC	0.97 (n/a)	n/a

Table 4.7: Graph Matching Results. Standard Errors are given in parentheses. Abbreviations: hGA = hybrid genetic algorithm, hGA-m = hGA without mutation, hGA-x = hGA without crossover, hGA-xm = hGA with neither mutation nor crossover, hCHC = hybrid CHC, and HC = gradient ascent (hillclimbing).

Pure gradient ascent appears to outperform all the other algorithms. The reason for

this is that the gradient ascent algorithm starts from an initial guess in which about 50% of the mappings are correct, whereas the other algorithms start with random initial guesses. In addition, the final population of a genetic algorithm typically contains solutions much better and worse than the average. Thus, this comparison is not really fair: a fairer comparison of optimisation performance comes from considering hGA-xm, which is multiple random restarts of gradient ascent. Furthermore, gradient ascent is deterministic, and therefore always gives the same result, but the genetic algorithm is stochastic and may do significantly better or worse than gradient ascent. Indeed, the genetic algorithm occasionally found matches with 100% correct mappings. However, the performance of gradient ascent alone suggests that for unambiguous problems, genetic algorithms may not necessarily be the method of choice. Apart from pure gradient ascent, the best optimiser was hCHC, which is only slightly better than hGA. The results for hGA-m and hGA-x indicate that crossover and mutation are playing an active part in the optimisation process. Turning to the fraction of distinct individuals with over 95% correct mappings, it is clear that pure gradient ascent is incapable of finding more than one solution. The hCHC algorithm appears to converge to fewer solutions than the hGA algorithm. In all, the hybrid genetic algorithm (hGA) combines strong optimisation performance with the highest solution yield, and it is this algorithm which will be the subject of the remainder of this study.

Table 4.8 shows the parameter sets for the preliminary study of the hybrid genetic algorithm. Sets L1 and L2 are drawn from the genetic algorithm literature (DeJong 1975; Grefenstette 1986); sets A to D were chosen to investigate the relative importance of population size and iteration limit. All sets required about 700000 fitness evaluations. The results are given in table 4.9.

	Set L1 (DeJong 1975)	Set L2 (Grefenstette 1986)	Set A	Set B	Set C	Set D
Population size	100	30	12	24	50	80
Iteration limit	3	8	20	10	5	3
Crossover type	Two point	Uniform	Uniform	Uniform	Uniform	Uniform
Crossover rate	0.6	0.9	0.9	0.9	0.9	0.9
Mutation rate	0.001	0.01	0.3	0.3	0.3	0.3

Table 4.8: Parameter Sets for Graph Matching. Each set required approximately 700,000 fitness evaluations.

Parameter Set	Average Fraction Correct	Average Fraction Distinct
L1 (DeJong 1975)	0.73 (0.0049)	0.030 (0.00082)
L2 (Grefenstette 1986)	0.91 (0.0047)	0.045 (0.0014)
A	0.93 (0.0041)	0.160 (0.0045)
B	0.92 (0.0043)	0.110 (0.0030)
C	0.90 (0.0044)	0.078 (0.0019)
D	0.80 (0.0046)	0.059 (0.0015)

Table 4.9: Effect of Parameter Sets on Algorithm Performance. Standard Errors are given in parentheses.

The differences in algorithm performance between sets L2, A, B and C are small. For these sets, optimisation is as well served by a large populations as it is by a large number of iterations. The poor performance of set D, and to some extent set L1, can be attributed to the small number of iterations. Presumably, the algorithm needs more than three iterations regardless of population size. For solution yield, the best sets appear to be sets A, B and C. However, the solution yield is only a proportion, and although set A produced an average yield of 16%, this corresponds to only 1.92 solutions on average. By contrast, sets B and C yielded 2.64 and 3.9 solutions on average. It would require at least two restarts of set A to achieve the same yield as set C, and yet both sets make the same number of fitness evaluations, and so have approximately equal running times. So it would appear that the small improvement in optimisation performance with more iterations is offset by a larger efficiency gain in solution yield with a larger population.

One possible additional reason for the poor performance of set L1 is that it used two point crossover, where all the other experiments so far used uniform or HUX crossovers. Table 4.10 gives the results of a study of different crossover types for the hybrid genetic algorithm using parameter set C from table 4.8.

It seems that the contiguous crossovers, one point, two point and geometric, all give roughly the same optimisation performance. The non-contiguous crossovers, uniform and HUX, appear almost identical, which is not surprising since for chromosomes containing 33 genes (one for each node in the data graph), the effect of uniform crossover on the chromosomes should be statistically indistinguishable from that of HUX. Although, the differences are small, there is a case to be made that uniform and HUX are the better operators. Nevertheless, this is a surprising result. Geometric crossover was designed

Crossover	Average Fraction Correct	Average Fraction Distinct
Uniform	0.90 (0.0044)	0.078 (0.0019)
HUX	0.90 (0.0042)	0.077 (0.0019)
One point	0.87 (0.0050)	0.073 (0.0016)
Two point	0.88 (0.0048)	0.069 (0.0017)
Geometric (Cross 1998)	0.87 (0.0055)	0.068 (0.0017)

Table 4.10: Effect of Crossover Type on Algorithm Performance. Standard Errors are given in parentheses.

specifically for two-dimensional problems (Cross 1998), and yet it performs no better than two point and worse than uniform. The most likely explanation of this is that the hybrid algorithm requires disruptive crossover. Indeed, the results obtained with uniform crossover are very good. However, the crossovers considered in this section have all been strictly syntactic operators. The next section considers whether the performance of crossover can be improved by incorporating problem knowledge.

4.4.5 Optimal Crossover

The purely syntactic crossover operators studied in the previous section are weak in the sense that they do not exploit the knowledge gained by evaluating the fitness of each individual, viz. which regions are consistently labelled and which are not. This section considers some "optimal" crossover operators, which attempt to use the information implicit in the consistency criterion to give better optimisation performance.

In (Williams and Hancock 1999), Williams and Hancock proposed "metric based crossover", in which half of the best node labels in one parent are exchanged with the other. They found that this operator led to faster convergence to a single solution; but that the quality of the final solution was no better than that obtained with uniform crossover. As Williams and Hancock pointed out, the drawback of this strategy is that it does not preserve structural consistency because the nodes with the best labels will not necessarily be connected in the data graph.

An additional (and probably minor) drawback of Williams and Hancock's study was that the consistency criterion used was equation 3.9 from chapter 3,

$$P(\Gamma) = \frac{1}{|\mathbf{V}|} \sum_{j \in \mathbf{V}} \frac{K_{C_j}}{|\Theta_j|} \sum_{S_i \in \Theta_j} \exp\left[-k_e D(\Gamma_j, S_i)\right] \tag{4.14}$$

which is expressed in terms of the supercliques, not the nodes, in the data graph. Although the labelling of the j^{th} superclique might be among the best, the labelling of node j itself might be wrong. Thus simply exchanging node labels rather than superclique configurations may not exploit the full power of metric based crossover. Fortunately this problem does not arise when the neighbourhood approximation derived in section 3.7.2 of chapter 3,

$$P(\Gamma) = \frac{1}{|\mathbf{V}_D|} \sum_{j \in \mathbf{V}_D} \frac{1}{|\Theta_j|} P(\Gamma_j | \Theta_{j,f(j)}) \tag{4.15}$$

is used. The fitness of the individual is now simply the mean of the "partial" fitnesses of the labellings of the data graph nodes, the j^{th} of which is $\frac{1}{|\Theta_j|} P(\Gamma_j | \Theta_{j,f(j)})$. Both the labelling criteria considered in section 4.2.3 can be used in this way, so optimal crossover can be applied when using either criterion.

A more subtle consideration arises from the two distinct views of the crossover operator mentioned in section 4.2.2. If crossover is required to be conservative, then greedy schemes such as Williams and Hancock's may be appropriate because they maximise the consistency of one of the offspring. On the other hand, if disruptive crossovers are preferred, then a more equal division of consistency between the two offspring might be better. Greedy operators which attempt to concentrate the best labellings in one offspring will be termed "unbalanced", while those which distribute consistent labels evenly between the offspring will be termed "balanced". Five optimal crossover operators are considered in the remainder of this section. They are:

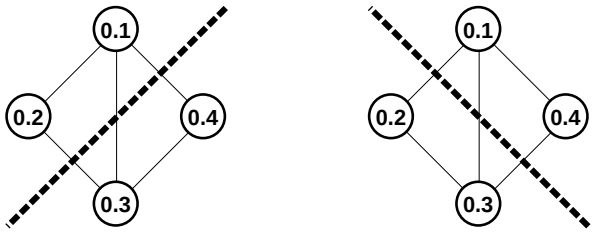
- Balanced optimal geometric
- Unbalanced optimal geometric
- Balanced optimal contiguous
- Unbalanced optimal contiguous
- Greedy uniform

The balanced counterpart of greedy uniform crossover is standard uniform crossover, which is balanced because a particular label has an equal probability of being transmitted to either offspring.

Optimal Geometric Crossover: Geometric crossover, as originally defined by Cross in (Cross 1998), exchanges the labels of nodes on either side of a random bisector of the point set. Consider the bisector at angle ω, drawn from a set of possible bisector angles, Ω, which partitions the node set of the data graph into $\mathbf{V}_D^{\omega,1}$ and $\mathbf{V}_D^{\omega,2}$. The difference between the contributions of each partition to the overall fitness is

$$\Delta f(\omega) = \frac{1}{|\mathbf{V}_D|} \left\| \sum_{j \in \mathbf{V}_D^{\omega,1}} P(\Gamma_j) - \sum_{j \in \mathbf{V}_D^{\omega,2}} P(\Gamma_j) \right\| \tag{4.16}$$

This is shown in figure 4.15 for a very simple graph. The nodes represent the images of data graph nodes under the current match. The nodes are labelled with hypothetical values of $P(\Gamma_j)$. Panel (a) shows the unbalanced bisector, which maximises the cost difference between the two partitions. Panel (b) shows the balanced bisector, which partitions the mappings equally.



(a) Unbalanced (b) Balanced

Figure 4.15: Optimal Geometric Bisectors. Each node represents the image of a data graph node under the match. The nodes are labelled with the contribution of each mapping, $P(\Gamma_j)$.

Equation 4.16 allows the angles, ω_b and ω_u, that the balanced and unbalanced bisectors make with the horizontal, to be determined:

$$\omega_\mathrm{b} = \arg\min_{\omega \in \Omega} \Delta f(\omega) \tag{4.17}$$

$$\omega_\mathrm{u} = \arg\max_{\omega \in \Omega} \Delta f(\omega) \tag{4.18}$$

The choice of possible bisectors, Ω, requires a little thought. There are four options. The first is to exhaustively enumerate Ω, which may be computationally expensive. The second is to construct Ω from bisectors at intervals $|\Omega|/2\pi$, which will lead to non-unique partitions. The third is to form Ω at random, which would only be worthwhile if the others were too computationally demanding. The fourth method is to construct Ω from the bisectors of the angles between nodes, which should approximate the first method reasonably well without generating redundant partitions. It is not difficult to see that once the fitness of a labelling has been evaluated, $\Delta f(\omega)$ can be calculated in $O(|\mathbf{V}_D|)$ additions, and that finding ω_b and ω_u requires $O(|\Omega|)$ comparisons. Neither of these is significant when compared to fitness evaluation, so optimal geometric crossover will not significantly slow the algorithm.

Optimal Contiguous Crossover: Optimal contiguous crossover is a variant of two point crossover in which the chromosome is seen as a circle, and an optimal semicircular partition is sought. If one of the crossover points is x, the other must be $(x + 0.5|\mathbf{V}_D|) \bmod |\mathbf{V}_D|$, and the data graph is partitioned into the sets $\mathbf{V}_D^{x,1}$ and $\mathbf{V}_D^{x,2}$. By analogy with equations 4.16 to 4.18, $\Delta f(x)$ and thus balanced and unbalanced contiguous crossover points, x_b and x_u, can be obtained. These can be determined by a linear search through the chromosome.

It is possible to define a more general operator in which the partitions need not be equal. This would require examination of the $O(|\mathbf{V}_D|^2)$ pairs of crossover points, and would also require a weighting function for $\Delta f(x, y)$ so that the partial fitnesses of unequal partitions could be compared.

Greedy Uniform Crossover: Greedy uniform crossover is similar to Williams and Hancock's metric based crossover (Williams and Hancock 1999). One offspring receives the best labels from each parent. For parents, Γ^A and Γ^B, the "better" offspring, Γ^C, is constructed by

$$\Gamma^C(j) = \begin{cases} \Gamma^A(j) & \text{if } P(\Gamma^A_j) > P(\Gamma^B_j) \\ \Gamma^B(j) & \text{otherwise} \end{cases} \tag{4.19}$$

and the other offspring may be constructed by the converse rule. Metric based crossover would assign the best half of Γ^A to Γ^C, and draw the other half of Γ^C from Γ^B.

Experiments

The different crossover operators were tested using a hybrid genetic algorithm with a population size of 40, an iteration limit of 5, and crossover and mutation rates of 0.9 and 0.3. The same four graphs as in section 4.4.4 were used as test problems, and 100 program runs were performed for each graph. Table 4.11 summarises the results for each graph and crossover type. The rows of the table give the average fraction correct in the final population for each of the seven crossover operators considered. The operators were tested on the four graphs, A, B, C and D, shown in the columns of the table. This arrangement gives a total of 400 observations per crossover operator. Since only 600000 fitness evaluations were made per program run, these figures should be expected to be lower than those reported earlier.

	Graph			
Crossover	A	B	C	D
Standard geometric	0.8126	0.7081	0.8724	0.7744
Balanced geometric	0.8403	0.7115	0.8758	0.7963
Unbalanced geometric	0.7127	0.6843	0.8372	0.7130
Standard two point	0.8147	0.7300	0.8788	0.7705
Balanced contiguous	0.8600	0.7554	0.9097	0.8132
Unbalanced contiguous	0.8639	0.7535	0.9008	0.7935
Standard uniform	0.8554	0.7519	0.9053	0.8138
Greedy uniform	0.7734	0.6902	0.8588	0.7490

Table 4.11: Comparison of Optimal Crossovers.

These results were analysed using analysis of variance (ANOVA), which showed that the balanced operators were superior in all cases. Table 4.12 shows the ANOVA for the best three crossovers. In this case, crossover accounts for only 3.8% of the total variation, and interaction for less than 1%. The interaction plot is given in figure 4.16, and shows that balanced contiguous and standard uniform crossovers consistently outperformed balanced geometric. The ANOVA indicates that the poorer performance of balanced geometric crossover is significant, suggesting the conclusion that either balanced contiguous or standard uniform crossovers are better than balanced geometric crossover. However, it was

Source	SS	df	MS	F-ratio	P
Crossover	0.1978	2	0.09890	120.0	0.00
Graph	4.045	3	1.348	1636.0	0.00
Interaction	0.03020	6	0.005034	6.108	0.00
Error	0.9792	1188	0.0008242		
TOTAL	5.253	1199			

Table 4.12: ANOVA for Balanced Crossovers. Abbreviations: SS=sum of squares, df=degrees of freedom, MS=mean square (MS=SS/df).

not possible to distinguish between the standard uniform and balanced contiguous operators. Balanced contiguous gave slightly better results, but these would only have been significant at the 10% level.

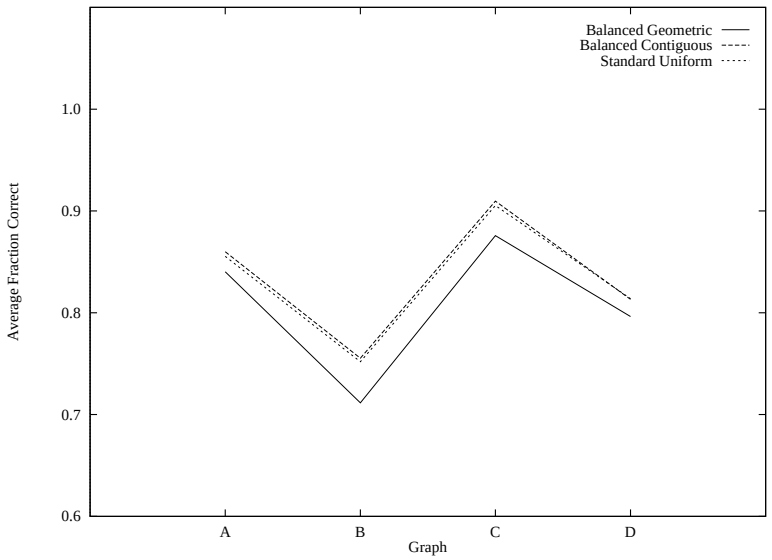


Figure 4.16: Interaction Between Crossover Type and Graph.

These results indicate that the ideal crossovers are disruptive rather than conservative. The explanation for this may lie with the gradient ascent step, which is guaranteed to move an individual labelling to its "closest" local optimum. Greedy or unbalanced operators will tend to make better partial labellings segregate together during recombination. Since these labellings specify the optimum located by the last gradient ascent step, it is unlikely that future gradient ascent will move the labelling to a different optimum. In other words, unbalanced operators only make minor changes to locally optimal labellings and may thus "stall" the algorithm. Gradient ascent itself is a greedy operator, the major drawback of which is convergence to local optima. The global optimisation properties of the genetic

algorithm should ameliorate this shortcoming, but the application of a greedy crossover would tend to worsen the situation.

According to this argument, the best crossovers should be those which cause maximal disruption. Those are standard uniform and balanced contiguous crossovers; geometric crossover does not cause as much disruption as two point crossover, as discussed in section 4.2.2. The algorithm benefits from such operators because local optima are severely disrupted - damage which can always be repaired by gradient ascent. The disruption caused by such crossovers is qualitatively different from that caused by mutation. Where the parents both agree as to the labelling of a node, no crossover can change that labelling: crossover only changes the regions of disagreement between the parents. Mutation, is not so discriminating, changing the labels of all nodes with equal probability. This may explain why genetic algorithms tolerate high crossover rates for disruptive crossovers, but do not tolerate high mutation rates.

4.5 Summary

This chapter has considered the problem of solving ambiguous labelling problems. Where there is significant ambiguity, the labelling process should not make arbitrary decisions as to the interpretation, but present as many possible interpretations as possible to some higher level module. Genetic algorithms were suggested as a framework for simultaneously finding alternative labellings. Two paradigms for understanding hybrid genetic algorithms were suggested: the first, that the hybrid algorithm is fundamentally a genetic algorithm with an extra optimisation step; and the second, that the algorithm is really gradient ascent with a global optimisation framework.

A preliminary study for line labelling showed that the hybrid genetic algorithm with a gradient ascent step could be suitable for ambiguous problems. The algorithm simultaneously produced several interpretations of line drawings, among which the labellings of strongly constrained junctions were much less subject to variation than the labellings of weakly constrained junctions. It was also found that the algorithm could localise the inconsistencies in drawings of impossible objects. Neither of the two sets of algorithm control parameters drawn from the literature were found to give adequate algorithm performance.

The graph matching framework developed by Wilson in (Wilson 1995) was adapted for the

ambiguous case by introducing a new measurement classification scheme. The estimated degree of ambiguity in the problem can be used to control the classifier. A preliminary experimental study showed that the hybrid genetic algorithm is a realistic method for obtaining multiple solutions to ambiguous graph matching problems. An initial study of the algorithm control parameters indicated that given a sufficient number of iterations, greater solution yields were obtained with larger populations. As with line labelling, the two sets of algorithm control parameters drawn from the literature were found to be inadequate.

In an attempt to improve the algorithm's optimisation performance, five different "optimal" crossover operators were proposed. It was found that the more disruptive, "balanced" operators, which distributed consistent labels equally between the offspring, outperformed their greedy counterparts; the best crossovers were found to be uniform and balanced contiguous crossover. These findings suggest that the paradigm in which the genetic algorithm provides initial guesses for gradient ascent is more likely to lead to better models of algorithm behaviour.

A common feature of the two experimental studies presented in this chapter has been the inadequacy of the "standard" genetic algorithm control parameter sets drawn from the literature. It is clear that there is much to be gained from choosing optimal algorithm control parameters, and it is to this which we turn our attention next.

Chapter 5

Optimal Genetic Algorithm Control Variables

This chapter addresses the problem of choosing an optimal set of control variables[1] for the genetic algorithm. Its major novel contribution is the development of empirical models of genetic algorithm performance for line labelling and inexact matching problems based on sets of factorial experiments. The experimental study reported is one of the most extensive to date in terms of the amount of data gathered. Furthermore, the models developed here are confined to a single problem domain, unlike previous attempts (Schaffer et al. 1989).

The control variables for the genetic algorithm are generally taken to be the population size, number of iterations,[2] crossover rate and mutation rate. There are also several other things which affect algorithm performance. These are selection strategy, crossover type, evaluation function, problem size and structure, and any additional parameters, for example the label error probability, P_e, in graph matching and line labelling. This study will focus on criterion, crossover type and problem size in addition to the standard four variables. It will also consider the utility of adding a gradient ascent step and the control

[1] I use the term "control variable" in preference to the more usual "control parameter" because these quantities are not parameters in the statistical sense.

[2] It is more usual to use number of fitness evaluations instead of iteration limit to provide fairer comparisons between genetic algorithms and other optimisation techniques. This is justifiable because in many real problems, the evaluation step will be the critical operation of the optimisation algorithm. However, this study is concerned solely with the genetic algorithm, and the number of iterations may be important irrespective of the population size.

of the error probability, P_e.

The next section introduces and motivates the experimental study, then the experimental design is described in section 5.2. Results for line labelling and graph matching are given and discussed in sections 5.3 and 5.4 respectively. Finally, section 5.5 summarises the main points of this chapter.

5.1 Introduction

There are two aspects of the genetic algorithm's behaviour which are of interest here, the performance (i.e. how good the final population is), and the solution yield (i.e. how many different good solutions the algorithm finds). This study will attempt to find analytical empirical models relating algorithm performance and solution yield to the control variables and other factors. Unfortunately, one is generally more interested in inverting these models to determine what control variable settings will lead to a given outcome. This is by no means an easy problem in this case, but given an analytical model, numerical optimisation techniques can be brought to bear. The goals of this study are:

- **Practical:** to establish an optimal combination of genetic algorithm control variables.
- **Theoretical:** to examine the relationships between algorithm performance and solution yield, and the control variables.
- **Technical:** to establish how useful the model is in setting control variables - i.e. how robust it is with respect to interpolation and extrapolation.

There are two criteria to choose from, three to four crossover types and any number of problems in addition to the four standard control variables. This represents a very large space of possibilities to explore. The literature provides four experimental alternatives.

First, treat the choice of optimal parameters as a combinatoric optimisation problem in itself and use a genetic algorithm or other technique to determine the optimal parameter set. This approach was adopted by Grefenstette in (Grefenstette 1986). It begs two questions. The first is what control variables would be appropriate for such a "control variable meta-optimiser". The second is how can the results be generalised so that they

apply to more than one problem instance. Grefenstette's study suffered from the additional weakness of comparing the outcomes of several random processes (the genetic algorithms) on the basis of very small sample sizes (1 in most cases, 5 at best).

Second, look at the effects of the different variables one or two at a time while keeping the others constant. For example, in (DeJong and Spears 1990), DeJong and Spears compared various crossover types and population sizes, but the other control variables such as mutation rate and crossover rate were held constant. These results will not necessarily generalise well. For example, it is plausible that crossover type and mutation rate may interact, since highly disruptive crossovers act like mutation in some respects: uniform crossovers change labellings at random whereas contiguous crossovers preserve long subsequences. So the crossover type may have some influence over which mutation rate is appropriate. Ultimately, it is impossible to simultaneously examine every combination of variable settings, but this sort of compromise is best postponed as long as possible.

Third, extrapolate from either Grefenstette's or DeJong's recommendations (or somebody else's). This may be a reasonable way to start out, but is really just an exercise in trial and error.

Fourth, follow the example of Schaffer and co-workers in (Schaffer et al. 1989) and use a factorial experimental design, including all variables which are likely to have an effect on the outcome. This is the best alternative, since it allows the experiment to be analysed with standard statistical techniques. Paradoxically, less work is required for this approach than to follow the second approach for each variable in isolation (see pages 150 and 151 of (Cochran and Cox 1957)), and to extrapolate based on the results of such a study can be no worse than any other extrapolation. Schaffer et al.'s study was conducted mostly on numerical optimisation problems, so it can not be assumed that their results will generalise to other domains. Indeed, they found significant interactions between the test problems and the control variables. However, a major flaw in their experimental design was to mix problems from different domains, resulting in a model in which the problem may be treated only as a qualitative factor. In (Mühlenbein 1994), Mühlenbein showed that Schaffer et al.'s model does not extrapolate well. Nevertheless, their approach was fundamentally correct, and it is this paradigm which will be adopted in this study.

5.1.1 Terminology

In statistical terminology, one is interested in an "outcome" or "response" variable which depends on three additive terms: systematic effects from known, independent "explanatory variables", systematic effects of unknown origin, and random effects or errors. Experiments are usually randomised so that the unknown systematic effects are absorbed into the random component. However, if one is confident that the unknown component is zero, randomisation is not necessary.

When explanatory variables may only take certain values, they are referred to as "quantitative factors", and the values as "levels" of the factor. Explanatory variables may also be "class variables" or "qualitative factors" when they represent categories such as gender or the presence or absence of some attribute. It is possible to treat any explanatory variable as a class variable by arbitrarily dividing its domain, for example into age-groups. Quantitative factors have a dual existence, being treatable as arithmetic quantities or classes.

5.2 Experimental Design

This study uses factorial experiments in which all relevant explanatory variables are examined simultaneously. Factorial designs have two major advantages over other designs (see pages 150 and 151 of (Cochran and Cox 1957)). First, they allow one to investigate quantitatively the interactions between explanatory variables. Second, they permit the quantification of the relationship between the outcome and the explanatory variables with a high degree of precision in a single experiment.

To enjoy these advantages, each possible combination of variable levels must appear once. Such a design is said to be fully crossed and balanced. If each explanatory variable has l levels, the number of combinations to be tested is $\prod_{j=1}^{J} l_j$ for J explanatory variables, which can rapidly become large, making both the conduct and analysis of the experiment difficult. Some statistical texts recommend that the number of factors be reduced using preliminary experiments, e.g. Cochran and Cox (Cochran and Cox 1957); others that a factorial design can serve as a summary of the data, e.g. Hays (Hays 1994). This chapter

takes the view that it is worth using the largest feasible design and then simplifying.[3] However, the fact remains that for a large number of factors, the number of levels per factor must be kept small for the experiment to be feasible. For example, a design with five factors each having two levels, requires that 32 combinations be tested. If there are four levels per factor, there will be 1024 combinations to consider. This is a problem because the genetic algorithm control variables and problem size can take very many different values and it would be best to look at as many of these as possible. It would also be convenient to do some preliminary experiments to determine the interesting ranges of these variables.

A cheaper alternative to a fully crossed balanced factorial design is a fractionally replicated factorial or graeco-latin square design, such as that shown in figure 5.1. This design has the property that no level of any explanatory variable appears more than once with any level of any other explanatory variable: there are no pairwise correlations between the explanatory variables. This allows the investigation of individual effects of explanatory variables using fewer combinations. For example, four factors of nine levels each require $9^4 = 6561$ combinations in a full factorial design, but only $9^2 = 81$ combinations in a graeco-latin square.

	1	2	3
I	A_α	B_γ	C_β
II	B_β	C_α	A_γ
III	C_γ	A_β	B_α

Figure 5.1: A 3x3 Graeco-Latin Square.

The price to be paid for reducing the dimensionality in this way is an inability to consider interactions since the co-occurrence of, say, 2 and A cannot be distinguished from the co-occurrences of 2 and III, 2 and β, III and β, or A and β. However, the graeco-latin square allows one to consider more levels per factor than the full factorial design for the same computational effort.

In these experiments, the genetic algorithm control variables will be in either graeco-latin

[3]For example, consider a design with 7 explanatory variables. When analysing such an experiment, very complex interactions involving more than three variables at a time could be ignored. However, at a later date, the data could be re-examined and these interactions modelled if necessary. This option would not be available had 5 out of the 7 variables been eliminated at the outset.

square or fully crossed configurations, and this arrangement will in turn be embedded in a fully crossed design for criterion, crossover type and problem size. These arrangements are shown in figure 5.2.

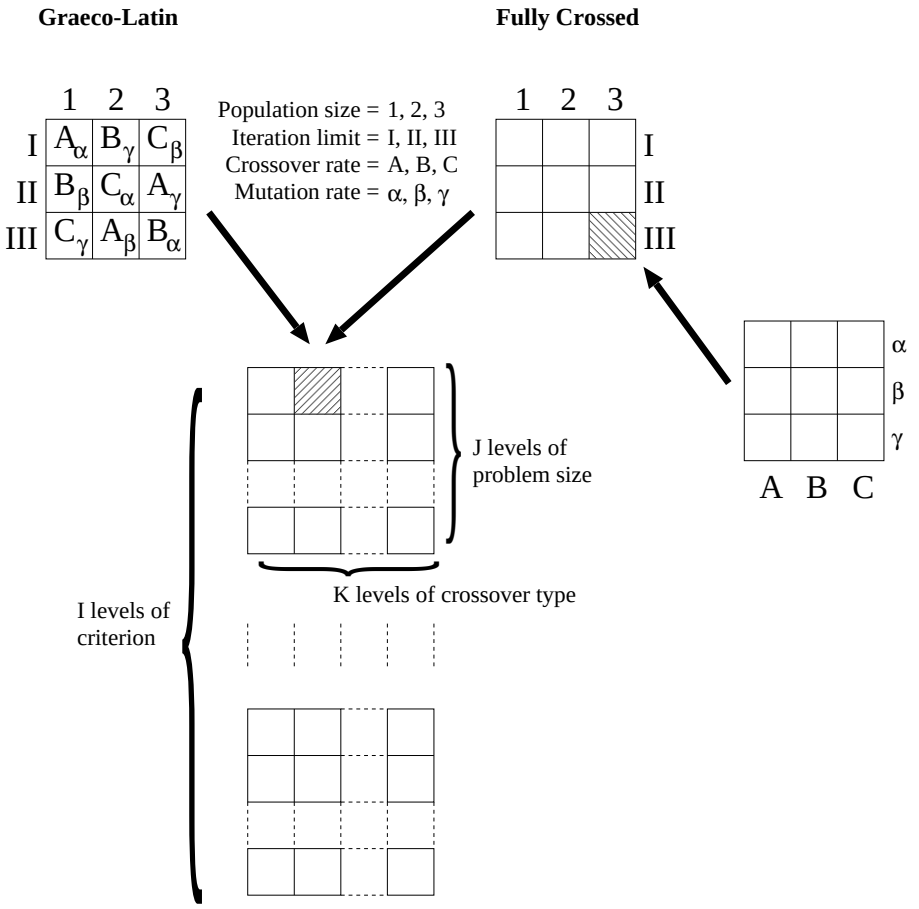


Figure 5.2: Experimental Design. The genetic algorithm control variables are shown with 3 levels each for simplicity. In practice, there were 9 levels for the graeco-latin square configuration (81 combinations) and 4 levels for the fully crossed configuration (256 combinations).

To maximise the savings, the explanatory variables with the largest numbers of levels are placed in the graeco-latin square. It is worth emphasising that the purpose of the graeco-latin square arrangement is to explore wider ranges of factor levels. The analysis cannot be based on the graeco-latin square experiments because interactions between the genetic algorithm control variables are highly likely to occur. However, this method can be used to find suitable ranges of the control variables and check that the response is continuous over the domain of interest.

Since this study will be conducted on a computer, there are no unknown systematic effects so there will be no randomisation. The only source of uncontrolled variation in the experiments is the pseudo-random number generator.[4]

5.3 Line Labelling

This section compares the effects of

- linear vs. Bayesian labelling criteria (equations 3.13 and 3.9 from chapter 3),
- annealing vs. no annealing of the label error probability, P_e,
- four crossovers: uniform, one point, two point and half-uniform crossover (Eshelman 1991),
- the addition of a gradient ascent step,

in addition to the four control variables on algorithm performance and solution yield for several different line labelling problems. To render the analysis of the experiments feasible, gradient ascent and annealing of P_e are not varied within each experiment.

Since the line labelling problem is exact, it is possible to identify whether or not the algorithm has located a global optimum. Thus, algorithm performance can be measured by a binary variable which takes the value 1 if the algorithm has located a global optimum (the algorithm is stopped when this happens, so the number of iterations is really an upper limit). Analysing binary outcome variables is inconvenient when attempting to estimate the probability that the algorithm will find a global optimum. So 20 program runs per factor level combination will be performed, and the number of successful runs will be treated as having a binomial distribution with denominator 20.

Similarly, in this exact case, it is also possible to count the number of distinct globally optimal solutions located by the algorithm. This quantity may range from zero to the pop-

[4]The assumption of no unknown systematic effects is almost always made when using computers in general: without it computers would not be much use. In contrast to the real world, it is the assumption of randomness which may cause problems. The pseudo-random number generator used in these experiments has been tested to confirm that it has a suitably long period and produces statistically uncorrelated sequences of numbers.

ulation size, and is therefore binomially distributed on the population size. Ten program runs (replications) were performed for each combination.

The experiments were conducted on a SGI Origin2000 computer with 32 180MHz MIPS R10000/R10010 processors. Timings vary, but each experiment required between four and eight days of computer time (the graph matching experiments described later could take up to a fortnight of computer time). The actual time taken for each experiment was reduced to about a day by running parts of the experiment on separate processors. Beyond this, no advantage was taken of the parallel architecture of the machine. Parallel implementation, although an exciting possibility, is beyond the scope of this thesis.

The next section gives some preliminary results, then section 5.3.2 describes the full factorial experiments and their analysis. Algorithm performance is considered in section 5.3.3, and solution yield in section 5.3.4.

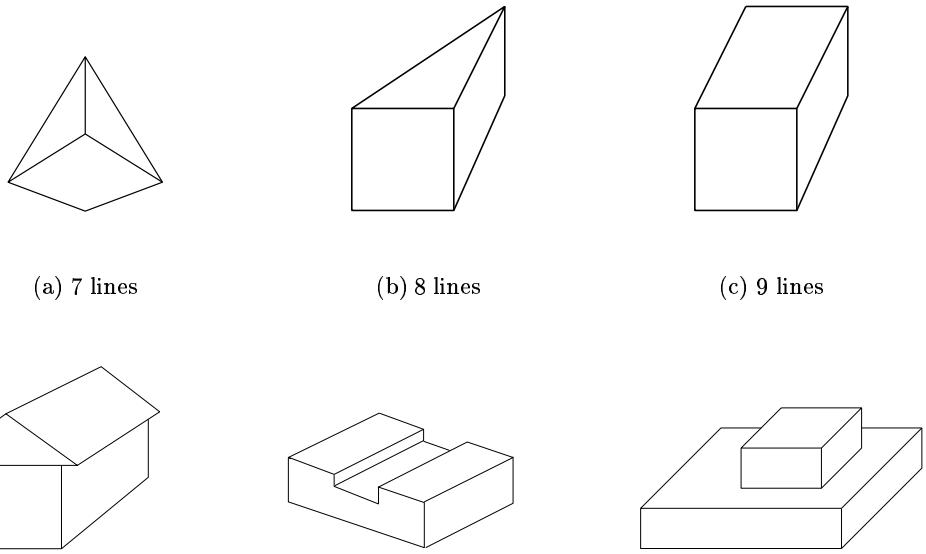
5.3.1 Preliminary Study

This preliminary study used only the linear criterion with no annealing of P_e or gradient ascent and only uniform or two point crossovers. The algorithm was tested on the nine synthetic line drawings shown in figure 5.3. The sizes of the search spaces for these drawings are 16384, 65536, 262144, 7.2×10^{16}, 1.2×10^{24} and 1.9×10^{25} respectively. The control variables were arranged according to a 9x9 graeco-latin square with values shown in table 5.1.

Variable	Values
Population Size	10, 20, 30, 40, 50, 60, 70, 80, 90
Iteration Limit	10, 20, 100, 200, 250, 300, 350, 400, 500
Crossover Rate	0.10, 0.20, 0.35, 0.24, 0.60, 0.65 , 0.75 , 0.85 , 0.90
Mutation Rate	0.001, 0.005, 0.010, 0.020, 0.050, 0.075, 0.100, 0.200, 0.300

Table 5.1: Factor Levels in the 9x9 Graeco-Latin Square.

Combined plots of the effect of varying the mutation and crossover rates and the population size are given in figures 5.4 and 5.5. Six line drawings and two crossover operators were considered. Table 5.2 gives a key to the figure legends. These plots give a broad picture of the relationships between parameter values and success rate. The iteration limit appeared



(a) 7 lines (b) 8 lines (c) 9 lines

(d) "House" (e) "Groove 2" (f) "Wedding Cake"

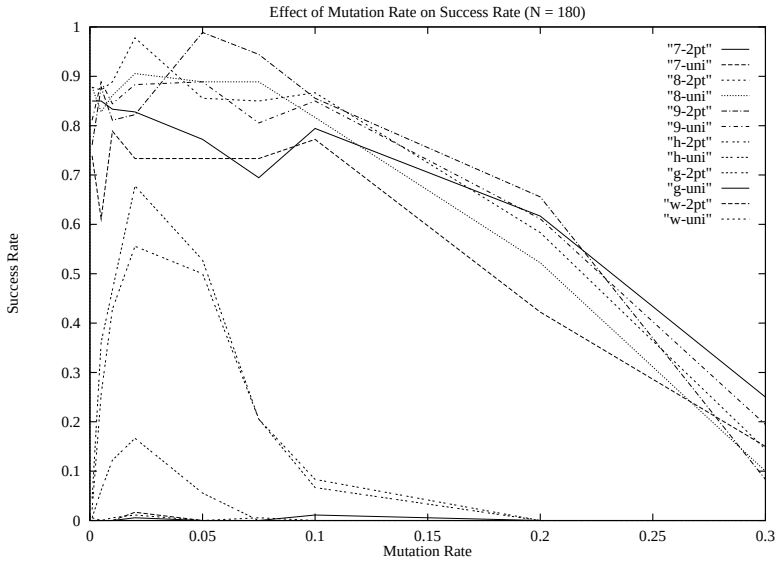
Figure 5.3: Test Drawings. Top row: 7, 8 and 9-line figures. Bottom row: 14, 20 and 21-line figures. 28, 40 and 42-line problems can be constructed using disconnected copies of the drawings in the bottom row.

less important in these experiments. Two point crossover was found to be better than uniform for most problems.

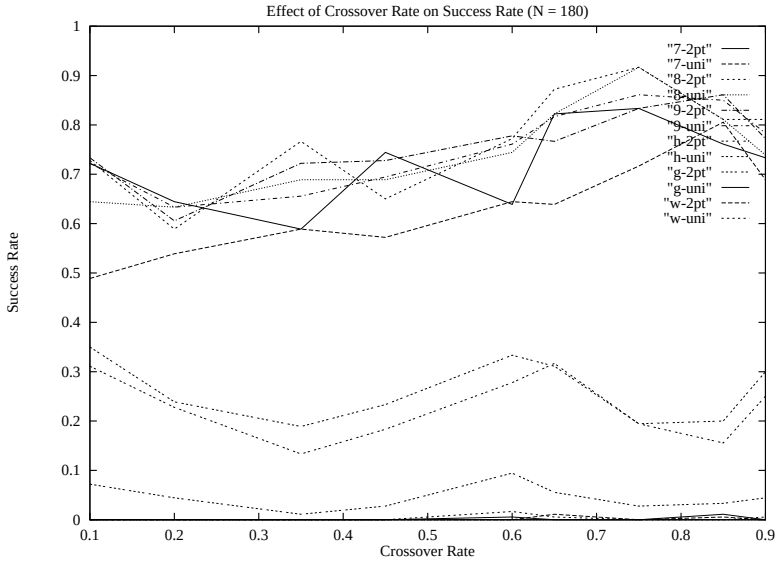
Problem		Crossover	
7	7 line drawing (figure 5.3 (a))	2pt	Two point
8	8 line drawing (figure 5.3 (b))	uni	Uniform
9	9 line drawing (figure 5.3 (c))		
h	"House", 28-line version (figure 5.3 (d))		
g	"Groove 2", 40-line version (figure 5.3 (e))		
w	"Wedding Cake", 42-line version (figure 5.3 (f))		

Table 5.2: Key to Figure Legends. For figures 5.4 and 5.5

The most striking feature of these plots is the sensitivity to mutation rate (panel (a) in figure 5.4): almost independently of problem size, the optimal mutation rate seems to be less than 0.1. The scalability of mutation rate effects is unsurprising since mutation is genic (i.e. more genes, more mutations). The fact that low but non-zero values of the mutation rate are beneficial agrees with a general view in the genetic algorithm literature that mutation is a necessary source of background noise, allowing the exploration of new regions



(a) Mutation Rate

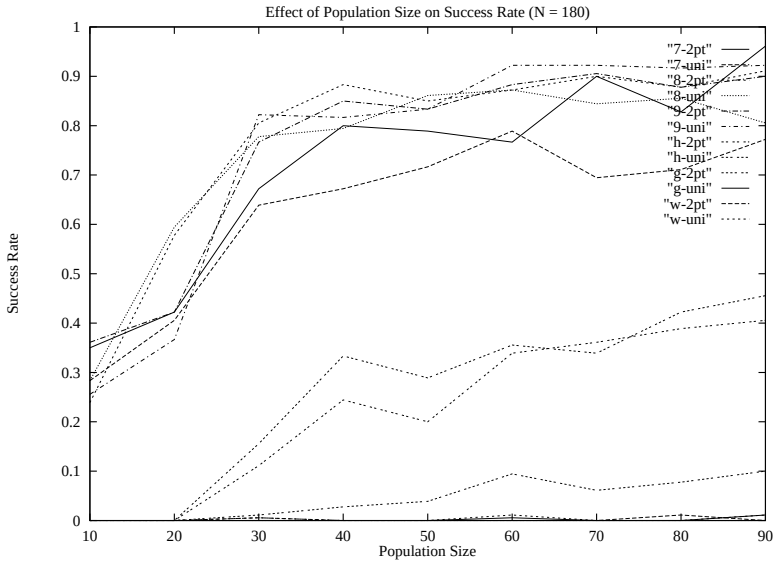


(b) Crossover Rate

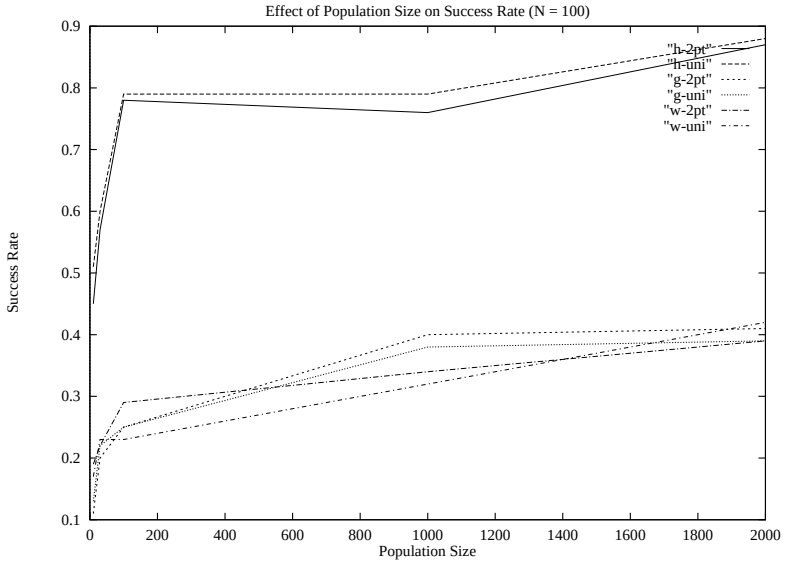
Figure 5.4: Effect of Varying the Mutation and Crossover Rates. Note the marked improvement in success rate at lower mutation rates, especially around 0.02. For each problem there is a peak in success rate for some crossover rate in the range [0.6,0.9]. There is also an upswing at lower crossover rates.

of the search space, but that it is not the primary mechanism of algorithm convergence.

The crossover rate does not appear to be as important as one might have expected (panel (b) in figure 5.4). Higher crossover rates do seem to improve success rate up to some upper



(a) Smaller Populations



(b) Larger Populations

Figure 5.5: Effect of Varying the Population Size. There appears to be a point beyond which relatively little is gained by increasing population size. Additional experiments (panel b) indicate that it is hard to solve the large problems even with a population size of 2000.

limit beyond which there is a falloff. The effects appear slight and it is possible say only that the optimal crossover rate will probably occur in the range [0.6,0.9] for each problem. Since like mutation rate, crossover rate is defined per gene, its effects might be expected to scale with the problem size.

The plots in figure 5.5 suggest that a population size of at least 30 is necessary for optimal success rates with the smaller problems, and that in these cases there is little to be gained from increasing the population size above this threshold. However, panel (b) of the same plot shows that the plain genetic algorithm does not scale at all well with problem size.

5.3.2 Full Factorial Experiments

In all, seven factorial experiments were done for line labelling: one without gradient ascent, three with gradient ascent but no annealing of P_e, and three with both gradient ascent and annealing of P_e. Annealing P_e was found to have little effect, so these experiments will not be discussed in detail. Suffice to say that P_e was originally used by Wilson and Hancock in (Wilson 1995) and (Hancock 1994) to iteratively harden labelling constraints, and hence avoid local optima. The genetic algorithm is a global optimiser, and so presumably has no need for such constraint hardening.

This leaves three experiments involving gradient ascent. Two of these consider the success probability as the outcome variable, and have a single replication of 8192 cells for a total of 8192 observations over 163840 program runs. The third considers the number of optimal solutions found and has 2048 cells with ten replications to give 20480 observations from 20480 program runs. The explanatory variables are summarised in table 5.3: for clarity, factor names, e.g. "LINES", will be used in the text, and variable names, e.g. "x_L", will be used in formulae. Statistical models were fitted using NAg's GLIM Version 4 update 8 (Francis et al. 1993). There is insufficient space to give full details of the statistical modelling involved.

Linear Models

The analysis will consist of fitting generalised linear models (McCullagh and Nelder 1989) to the data. These models are of the form:

$$y = g^{-1}(\eta + \epsilon) \tag{5.1}$$

where y is the outcome variable, g is the "link function" (q.v.), which transforms y so that it matches the range of the "linear predictor", η, and ϵ is the effect of random errors.

Factor, Variable	Description	Values
COST, w_C	Criterion type	linear, Bayesian
CROSS, w_X	Crossover type	uniform, two point, half-uniform, geometric
LINES, x_L	Problem size	9, 21, 28, 40
POP, x_P	Population size	50, 100, 150, 200 (without gradient ascent)
		10, 20, 30, 40 (with gradient ascent)
		fixed at 40 (for solution yield experiments)
GEN, x_G	Iteration limit	50, 150, 250, 350 (without gradient ascent)
		2, 4, 6, 8 (with gradient ascent)
P_x, x_X	Crossover rate	0.2, 0.4, 0.6, 0.8
P_m, x_M	Mutation rate	0.01, 0.02, 0.03, 0.04

Table 5.3: Factors Included in Line Labelling Experiments.

Since the error term, ϵ, is random, its mean is assumed to be zero and the expected value of $g(y)$ depends only on the linear predictor, η, which in turn depends on the explanatory variables as follows:

$$\eta = \sum_{0 \leq j \leq J} \beta_j x_j \tag{5.2}$$

where J is the number of explanatory variables, and β_j is the coefficient of the j^{th} explanatory variable, x_j. It is conventional to take $x_0 = 1$ so that β_0 represents a constant offset. Such models are known as "generalised linear models" (McCullagh and Nelder 1989) because they are linear in the parameters β, but not necessarily in the variables x. Indeed, any transformation of x may be substituted, such as its logarithm or reciprocal, or even K polynomials of degrees K, $K-1$, and so-on down to 1.

The link function, g, is used to map the domain of the response variable, y, onto the range of the linear predictor, η, which is taken to be $(-\infty, +\infty)$. In the particular case when y is a proportion the link function transforms $[0, 1]$ onto $(-\infty, +\infty)$. There are four such link functions of interest; they are:

- The logistic or logit link, $\text{logit}(y) = \ln\left(\frac{y}{1-y}\right)$, which has a natural interpretation as the log-odds of the event associated with y.

- The probit link, $\text{probit}(y) = \mathbf{\Phi}^{-1}(y)$, where $\mathbf{\Phi}(a)$ is the standard Normal integral from $-\infty$ to a.

- The complementary log-log link, $\text{cll}(y) = \ln[-\ln(1-y)]$

- The parameterised link function introduced by Aranda-Ordaz in (Aranda-Ordaz 1981) which is $g(y;\alpha) = \ln\left[\frac{(1-y)^{-\alpha}-1}{\alpha}\right]$ for $\alpha > 0$. This has two important special cases: when $\alpha = 1$, the function is equivalent to the logistic link, $g(y;1) \equiv \text{logit}(y)$; and $\lim_{\alpha\to 0} g(y;\alpha) = \text{cll}(y)$, the complementary log-log link.

The model algebra developed by Wilkinson and Rogers (Wilkinson and Rogers 1973) will be used to describe the form of the linear predictor. The notation is shown in table 5.4.

`A+B`	denotes the main effects	$\beta_A A + \beta_B B$
`A.B`	denotes the interaction	$\beta_{AB} AB$
`A*B`	is equivalent to	`A + B + A.B`
`A<k>`	denotes the polynomials	$\beta_{A1} X_A^1 + ... + \beta_{Ak} X_A^k$
M`**n`	represents the cartesian product	$\mathbf{M}^n$

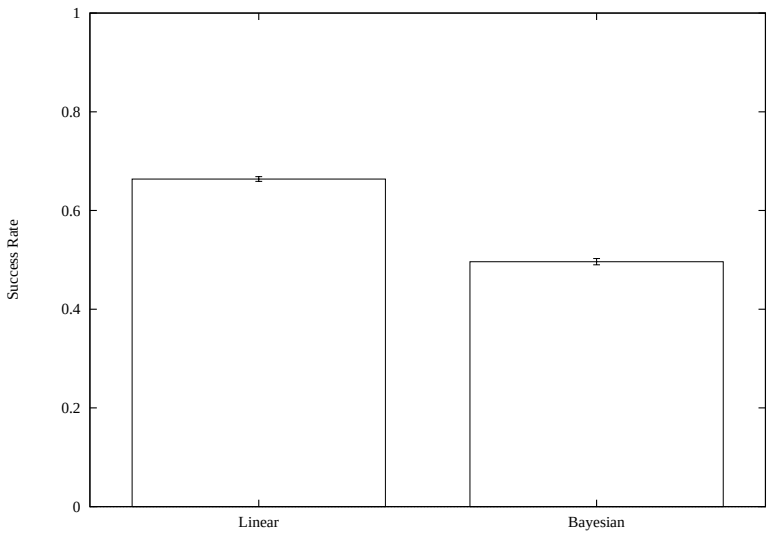
Table 5.4: Summary of Model Algebra. `A` and `B` are individual terms and **M** is a set of terms.

5.3.3 Success Rate

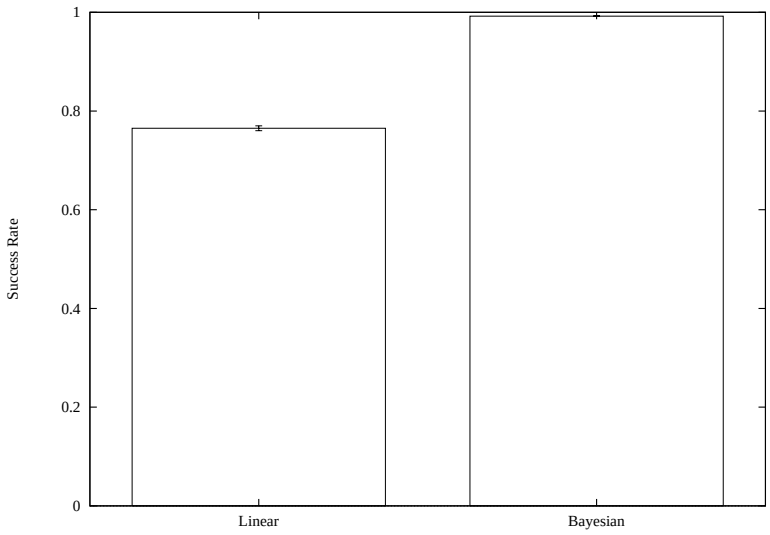
Plain Algorithm and Gradient Ascent Hybrid

In these experiments, the addition of a gradient ascent step was found to greatly improve algorithm performance: it achieved a higher success rate with one fifth of the population size and one twenty-fifth of the iterations when compared to the plain algorithm. This section does not dwell on the plain algorithm, but considers some contrasts between it and the hybrid. Perhaps the most interesting differences between the hybrid and non-hybrid algorithms concern the criterion type and the mutation rate.

Figure 5.6 shows the average success rates for each criterion with the plain and hybrid algorithms. The obvious interpretation is that gradient ascent does better with the Bayesian criterion since it is less likely to get stuck in local optima than the linear one (Hancock 1994).



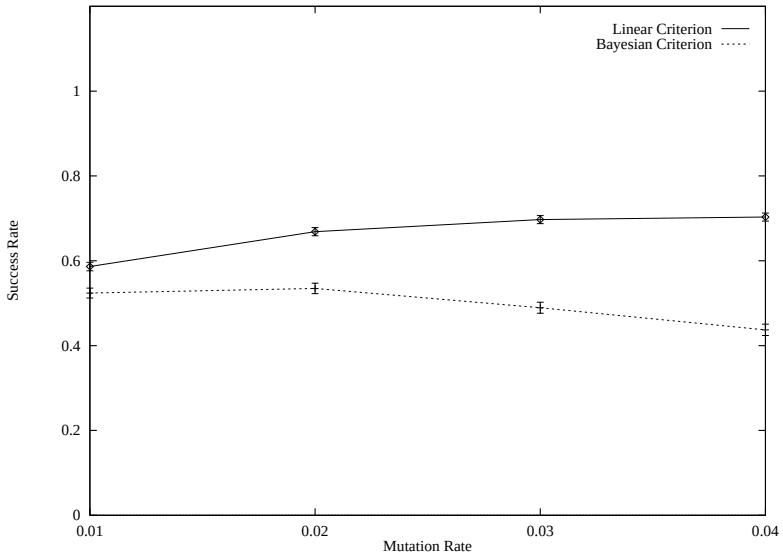
(a) Plain Algorithm



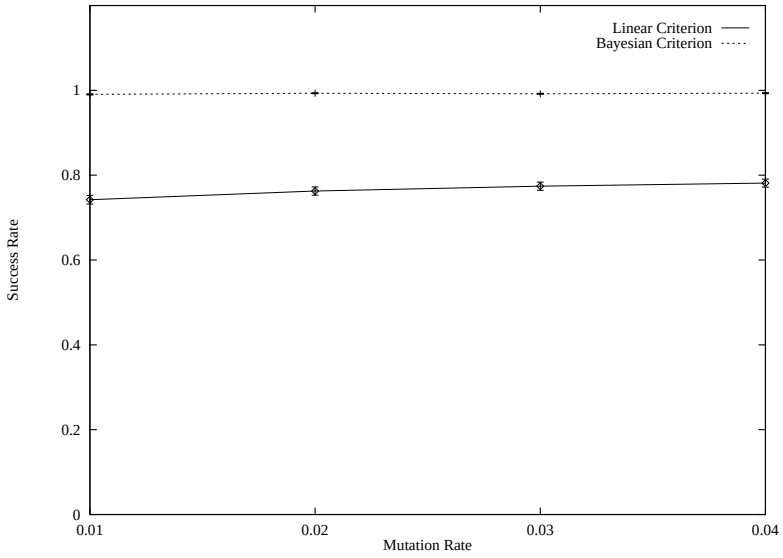
(b) Hybrid Algorithm

Figure 5.6: Effect of Criterion on Success Rate. The improvement for the Bayesian criterion very greatly outstrips that for the linear one. The error bars in this and subsequent figures reflect the precision of the mean, *not* the underlying variation in the data.

Another important difference is the effect of mutation rate on success rate, shown in figure 5.7, where the hybrid algorithm appears much less sensitive to changes in mutation rate than the plain one. This is presumably because gradient ascent will tend to correct small disturbances caused by mutation, hence the hybrid algorithm will tolerate higher mutation rates.



(a) Plain Algorithm



(b) Hybrid Algorithm

Figure 5.7: Effect of Mutation Rate on Success Rate. With gradient ascent, the algorithm appears relatively insensitive to mutation rate.

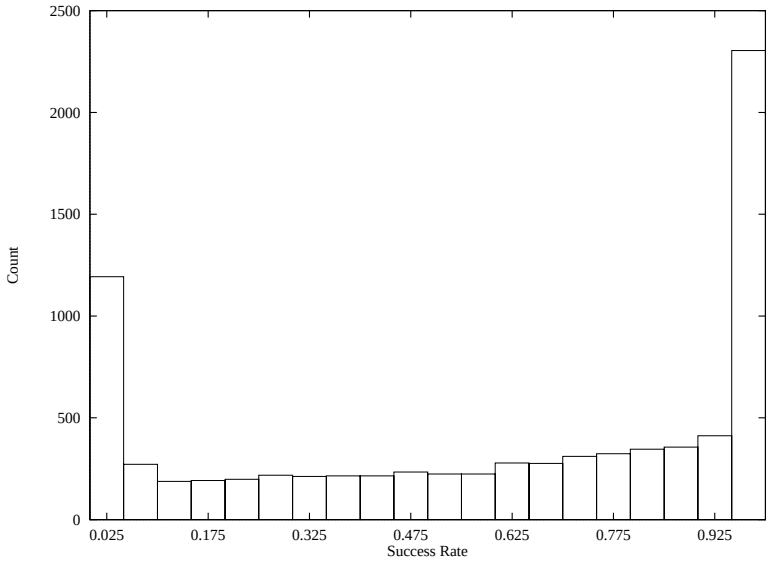
The histograms of success rate, shown in figure 5.8, are also very different. The histogram for the plain algorithm is bimodal, indicating that some problems were very hard for this algorithm while others were very easy. The histogram for the hybrid algorithm has only one mode, suggesting that all the problems were quite easy for the algorithm. For the hybrid algorithm, about 74% of the observations coincide with the mode, indicating that the algorithm located the global optimum in almost all cases. Since the hybrid algorithm almost never failed to solve a problem in all 20 trials, it would appear that this algorithm scales better with population size than the plain one. The hybrid algorithm is now considered in some detail.

Gradient Ascent Hybrid

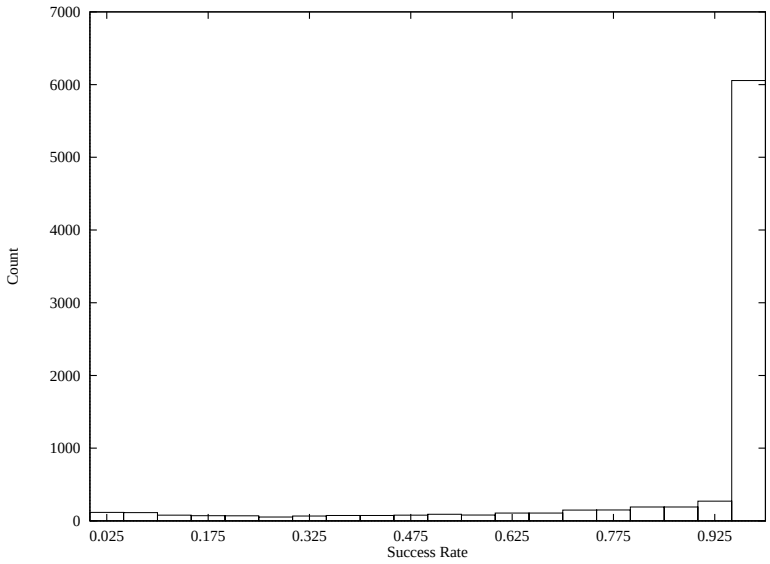
As the histogram in panel (b) of figure 5.8 indicates, three quarters of the program runs succeeded in each of the 20 trials. A salient question at this point is whether there is any variation at all over some of the explanatory variables. Table 5.5 shows the proportion of cells for which the observed success rate was 100% for each level of the explanatory variables. Tabulating the double interactions between each pair of explanatory variables (COST.LINES is shown in table 5.6 as an example) shows that only the following combinations were saturated: COST = Bayesian and LINES = 9 or 28; POP = 30 or 40 and LINES = 9; GEN = 8 and LINES = 9. There may still, therefore, be scope for modelling, especially with the linear criterion.

Factor / Level	1	2	3	4
COST	1794 (0.44)	3831 (0.94)	-	-
CROSS	1369 (0.67)	1422 (0.69)	1405 (0.69)	1429 (0.70)
LINES	2011 (0.98)	1358 (0.66)	1492 (0.73)	764 (0.37)
POP	1105 (0.54)	1350 (0.66)	1530 (0.75)	1640 (0.80)
GEN	1123 (0.55)	1422 (0.69)	1524 (0.74)	1556 (0.76)
P_x	1311 (0.64)	1384 (0.68)	1449 (0.71)	1481 (0.72)
P_m	1373 (0.67)	1395 (0.68)	1420 (0.69)	1437 (0.70)

Table 5.5: Saturated Factor Levels for Hybrid Algorithm. The total number of cells at each level is 4096 for COST and 2048 for the other factors. The table shows counts (proportions) of cells which had 20 out of 20 successes for each factor level.



(a) Plain Algorithm



(b) Hybrid Algorithm

Figure 5.8: Histograms of Success Rate. None of the test problems seemed hard for the hybrid algorithm. The plain algorithm clearly did not find the global optimum in some cases.

The initial statistical model fitted was the quadratic response surface,

$$\begin{aligned}&\texttt{1+(COST+CROSS+LINES+POP+GEN+}P_x\texttt{+}P_m\texttt{)**2}\\&\texttt{+(COST+CROSS)*(LINES<2>+POP<2>+GEN<2>+}P_x\texttt{<2>+}P_m\texttt{<2>)}\end{aligned} \tag{5.3}$$

which has a deviance[5] of 6776.9 on 8124 degrees of freedom (d.f.). The model contains 68 terms and appears to be somewhat overfitted. The standard method for simplifying such models is to observe that for binomial data, if two models **M1** and **M2** are nested - i.e. **M2** is a subset of **M1** - the difference in their deviances is approximately distributed as χ^2 on the difference in their d.f. This approximation is good when the total number of binary observations is high (as it is here being 163840). Thus, terms may removed or added to a model, their significance being judged on the basis of χ^2-tests in a procedure analogous to the analysis of variance. A good introduction to this topic is given by Collett in (Collett 1991).

	LINES			
COST	9	21	28	40
Linear	987	336	468	3
Bayesian	1024	1022	1024	761

Table 5.6: Saturated Cells Between COST and LINES. The maximum possible number of saturated cells is 1024. Only the 40-line problem presents problems to the Bayesian criterion.

The analysis of deviance for the interactions of COST in model 5.3 is given in table 5.7 as an example. There is insufficient space to give full analyses, so only the results will be quoted. Table 5.7 shows that in addition to an obvious main effect, four out of six double interactions are significant. This means that there are really two different models, which would be cumbersome to consider simultaneously. The modelling will therefore be done separately for each criterion.

[5]The deviance is a summary measure of goodness of fit derived from the likelihood ratio of the current model to an ideal one with a parameter for every observation. It is approximately distributed as χ^2 on the residual degrees of freedom of the current model.

Interaction	Deviance change	d.f. change	p-value
COST.CROSS	24.05	3	0.00
COST.LINES	136.7	2	0.00
COST.POP	132.2	2	0.00
COST.GEN	0.1460	2	0.93
COST.P_x	19.58	2	0.00
COST.P_m	0.9683	2	0.62

Table 5.7: Analysis of Deviance for Interactions of COST. The p-value indicates the probability that such a large deviance change could have arisen by chance if the term in question really were insignificant.

Bayesian Criterion

As shown in table 5.5, 94% of cells with the Bayesian criterion have 100% success rates. The interaction tabulated in table 5.6 indicates that this criterion virtually guarantees success for the 9 and 28-line problems given the other explanatory variables. Because of this, it is only sensible to consider models which are linear in the explanatory variables: there is insufficient variation in the data set to support more complex models. The first model is:

$$\texttt{1+CROSS+LINES+POP+GEN+}P_x\texttt{+}P_m \tag{5.4}$$

which has a deviance of just 675.72 on 4087 d.f. The part of the linear predictor involving CROSS is shown in table 5.8. For binomial data, the t-value has a standard Normal distribution under the null hypothesis, $\beta = 0$.

Term	$\hat{\beta}$	s.e.($\hat{\beta}$)	t-value	p-value
CROSS = two point	-0.2850	0.1214	-2.348	0.02
CROSS = geometric	-0.1964	0.1231	-1.595	0.11
CROSS = half-uniform	$4.700\text{x}10^{-14}$	0.1275	$3.687\text{x}10^{-13}$	1.00

Table 5.8: Parameter Estimates for CROSS. The t-value is the ratio of the estimate to its standard error, $\hat{\beta}$/s.e.($\hat{\beta}$). The significance test is one-tailed.

By this criterion, only two point crossover has a parameter significantly different from zero, so uniform, geometric and half-uniform crossovers may be amalgamated, and an offset for two point crossover included, in future models. An analysis of the link function, given in table 5.9, showed that the probit link gave the best fit.

Link function	Deviance
Logit	679.15
Ideal Aranda-Ordaz	675.64
Complementary log-log	689.47
Probit	674.11

Table 5.9: Goodness of Link Test. There is no formal significance test: the link function with the lowest deviance gives the best fit, which in this case is the probit link. For the Aranda-Ordaz link, the constructed variable technique described in (Collett 1991) was used to estimate the parameter of the link function.

It was also found that taking the logarithms of POP and GEN improved the fit. This leads to the model

$$\left.\begin{aligned} r &= \text{probit}^{-1}(\eta) \\ \eta &= 3.659 - 0.1786 x_L + 1.176 \ln x_P + 1.080 \ln x_G + \\ &\quad 1.010 x_X + 6.066 x_M + 0.1243 w_X \\ w_X &= \begin{cases} 0 & \text{for two point crossover} \\ 1 & \text{otherwise} \end{cases} \end{aligned}\right\} \tag{5.5}$$

This model will help achieve the primary goal of finding optimal conditions for the algorithm since the Bayesian is the better of the two criteria.

Linear Criterion

This was investigated in pursuit of the second goal, to shed some light on the relationships between the explanatory variables and the outcome. The Bayesian criterion worked so well that there was little scope for such investigation in the previous section, so the linear criterion was used for this purpose. Initial analysis suggested that the complementary log-log link function was the most appropriate. Again, it appears on the basis of t-values that uniform, geometric and half-uniform crossovers could not be distinguished, so CROSS

will be redefined with only two levels: two point and non-two point.

Since there are only four levels of LINES, POP, GEN, P_x and P_m, the most complex response surface supported is cubic:

$$\begin{aligned}&1+\texttt{CROSS*(LINES<3>+ln(POP)<3>+ln(GEN)<3>+}P_x\texttt{<3>+}P_m\texttt{<3>}\\&\texttt{+(LINES+ln(POP)+ln(GEN)+}P_x\texttt{+}P_m\texttt{)**2)}\\&\texttt{+(LINES+ln(POP)+ln(GEN)+}P_x\texttt{+}P_m\texttt{)**3}\end{aligned} \tag{5.6}$$

which has a deviance of 3140.6 on 4034 d.f. This model can be simplified using repeated analysis of deviance to

$$\begin{aligned}&1+\texttt{CROSS*(LINES<3>+ln(GEN)<2>)+ln(POP)+}P_x\texttt{<2>+}P_m\texttt{<2>}\\&\texttt{+(LINES+ln(POP)+ln(GEN)+}P_x\texttt{+}P_m\texttt{)**2)-ln(POP).}P_x\\&\texttt{+LINES.ln(POP).ln(GEN)+LINES.ln(GEN).}P_m\end{aligned} \tag{5.7}$$

which has deviance 3179.2 on 4068 d.f. Since these models are nested, the deviance change of 38.6 can be compared to χ^2 on 34 d.f., and found to be insignificant. This is the simplest model which adequately describes the data. A plot of the Anscombe residuals given in figure 5.9 for this model shows no clear pattern apart from some overfitting to the smallest problem, and only about 2% of the residuals are significantly large at the 5% level.[6] Although a significantly large residual implies a poor fit, one would expect about 5% of the residuals to be large purely by chance.

The functional form of model 5.7 contains 28 terms and is given in equation 5.8. This is a very complex six-dimensional function. The only way to visualise it is to hold some of the explanatory variables constant. Figure 5.10 shows the success rates for the factor levels of LINES, POP, GEN, P_x and P_m. The least important variables appear to be P_x and P_m. The coefficients in the model in equation 5.8 for interactions involving P_x and P_m are smaller than those for the main effects, suggesting that the interactions are not as important as the main effects. If three of POP, GEN P_x and P_m are fixed, the success rate can be plotted as a function of LINES and the fourth variable. These are shown in figures 5.11 and 5.12 for POP and GEN.

[6] Anscombe residuals (Anscombe 1953) are constructed so as to have a standard Normal distribution for binomial data. A residual of magnitude 1.96 is therefore significantly large at the 5% level.

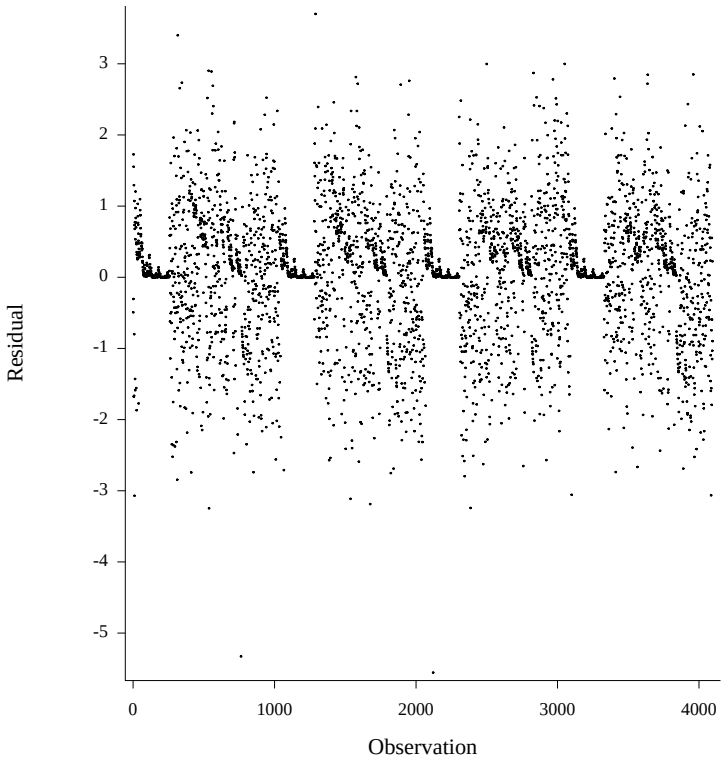


Figure 5.9: Anscombe Residuals for Model 5.7. The plot shows that the model overfits the 9-line problem. Apart from that there is no pattern to the residuals.

$$
\left.
\begin{aligned}
r &= \mathrm{cll}^{-1}(\eta) \\
\eta &= 2.401 - 1.213x_L + 0.05144x_L^2 - 0.0007183x_L^3 + 1.943\ln x_P + \\
&\quad 2.811\ln x_G - 0.4411(\ln x_G)^2 + 2.325x_X - 0.9407x_X^2 + \\
&\quad 42.46x_M - 184.9x_M^2 - 0.01940x_L\ln x_P - 0.6367\ln x_P\ln x_G - \\
&\quad 0.003289x_L\ln x_G + 0.01239x_Lx_X - 0.3246x_X\ln x_G - \\
&\quad 5.948x_M\ln x_P - 0.3023x_Lx_M - 9.499x_M\ln x_G - \\
&\quad 10.03x_Xx_M + 0.01970x_L\ln x_P\ln x_G + 0.4675x_Lx_M\ln x_G \\
&\quad w_X(-1.116 + 0.1529x_L - 0.007309x_L^2 + 0.0001018x_L^3 + \\
&\quad 0.7089\ln x_G - 0.2670(\ln x_G)^2) \\
w_X &= \begin{cases} 0 & \text{for two point crossover} \\ 1 & \text{otherwise} \end{cases}
\end{aligned}
\right\} \qquad (5.8)
$$

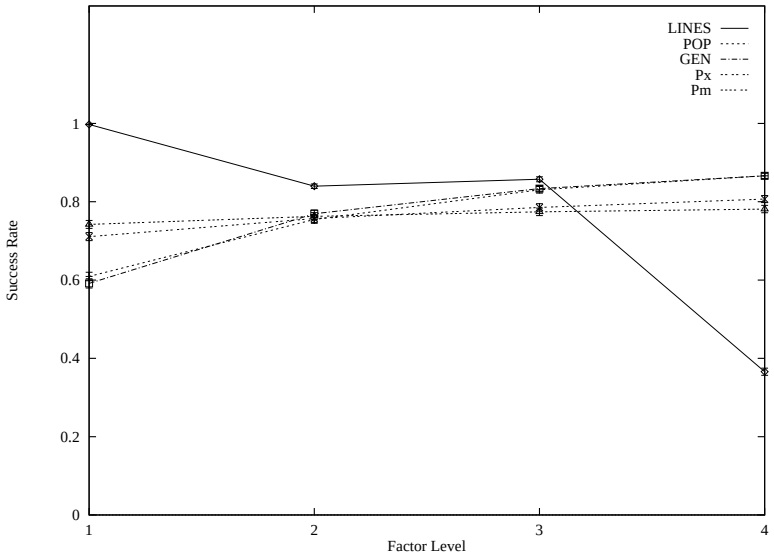


Figure 5.10: Success Rate Plots. The values for each factor level are given in table 5.3. A larger slope indicates that a particular variable has a greater effect on the success rate. The ranges of the variables must also be taken into account - for example, only $\frac{1}{25}^{\text{th}}$ of the range of mutation rate is explored.

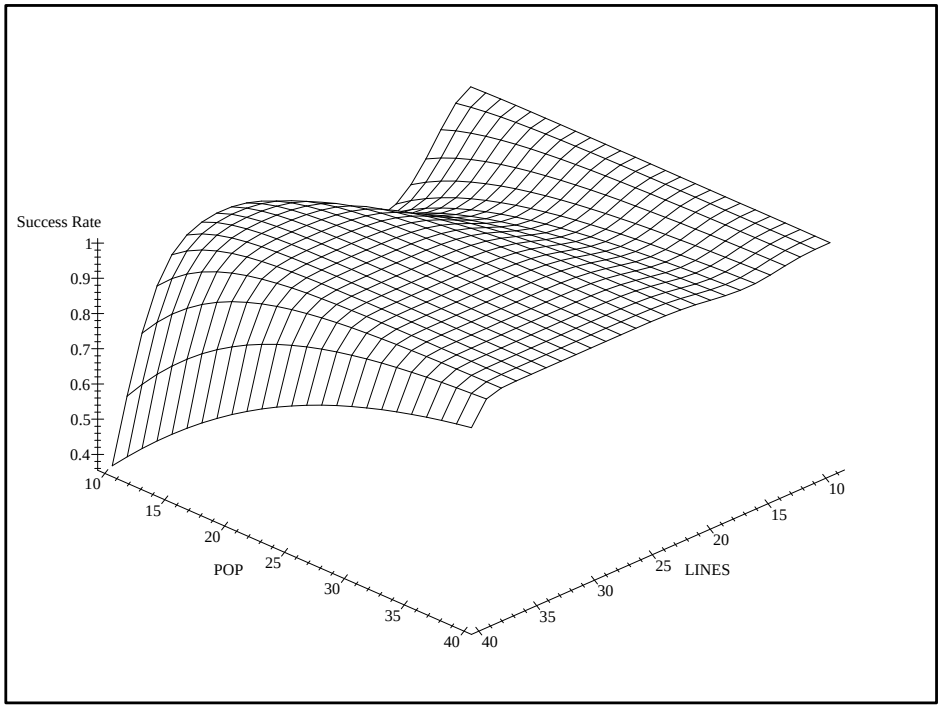


Figure 5.11: Plot of Model 5.8. Success rate is plotted as a function of LINES and POP, with GEN fixed at 8, P_x fixed at 0.8 and P_m fixed at 0.04.

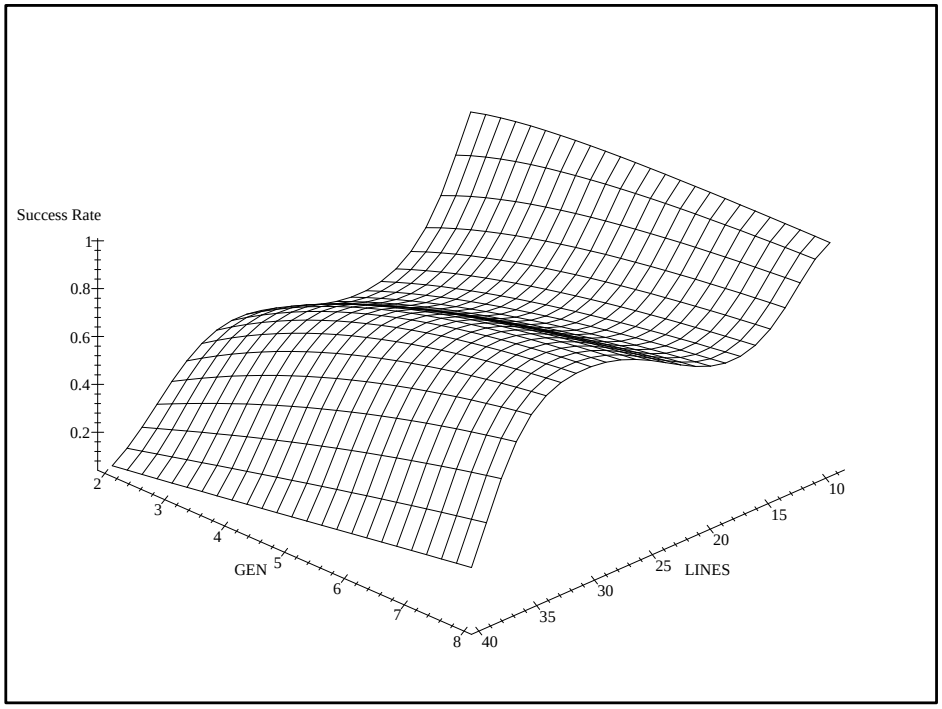


Figure 5.12: Plot of Model 5.8. Success rate is plotted against LINES and GEN with POP fixed at 40, $P_x = 0.8$ and $P_m = 0.04$.

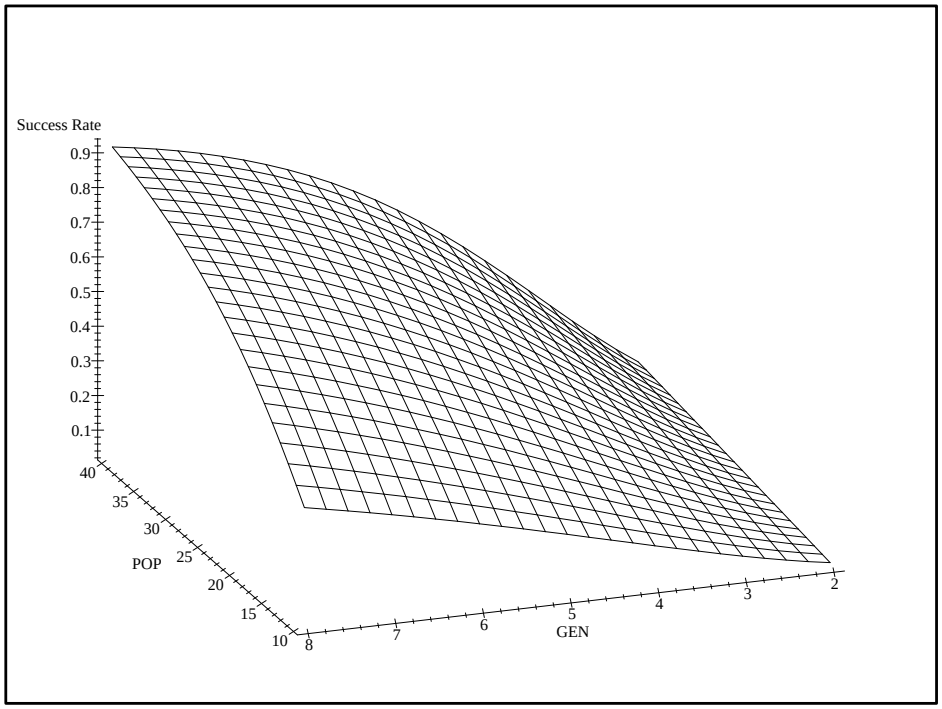


Figure 5.13: Plot of Model 5.8. Success rate is plotted as a function of POP and GEN for the 40-line problem, P_x fixed at 0.8 and P_m fixed at 0.04.

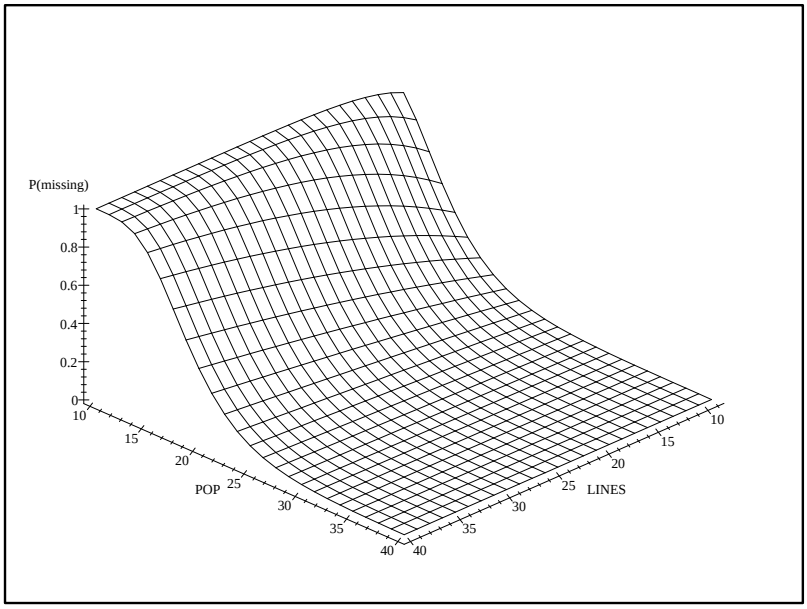
Although this model explains the data well by statistical criteria, there remains some question as to its validity. The 28-line problem seems to be easier for the algorithm than the 21-line problem. This difference is not very great but it does account for the cubic polynomial in LINES: any curve fitted to these points must have two local optima and must therefore be cubic or of higher order. There must be additional aspects of the problems which make them easy or hard, apart from the number of lines in the drawing. These aspects are probably structural - neither the numbers nor ratios of the different junction types are any better at explaining the data than drawing size. These structural differences between line drawings mean that one must be careful about fitting models polynomial in LINES since they are likely to be responding as much to structural variations among problems as to problem size. In view of this, attention should perhaps be paid to the relationships to POP and GEN, shown in figure 5.13.

The success rate increases with the logarithm of the population size confirming the observation made at the end of section 5.3.1 that there is a limit to the benefit of increasing the population size. This is an interesting phenomenon because the size of the search spaces for these problems are so large that it is very unlikely that a global optimum will be found in the initial population (which is generated at random). Of course, all that is really necessary is that a sufficiently good initial guess for gradient ascent is present in the population: this often occurs with the 9-line problem but rarely with the other problems.

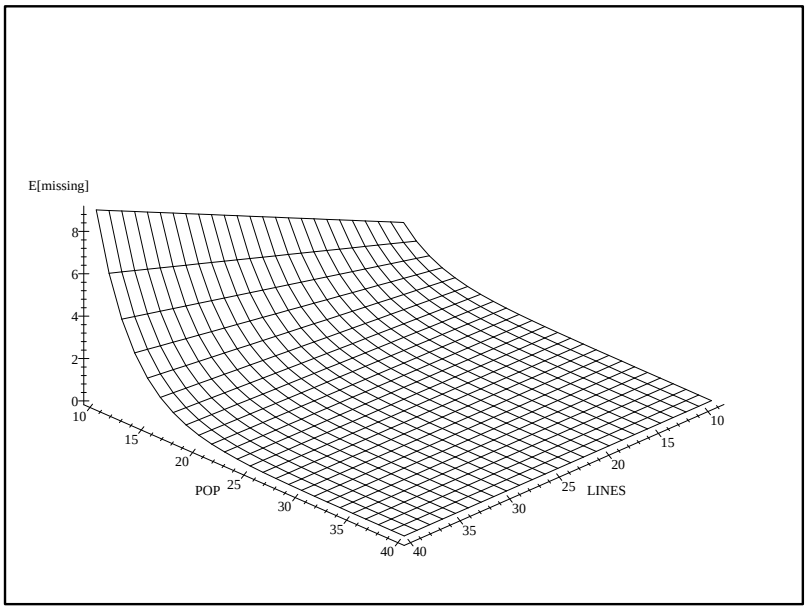
It is, however, virtually certain that all labels will appear at all loci. There are only four labels, which are equiprobable, so the probability of a label not appearing at a particular locus is $p_a = 4 \times (3/4)^{x_P}$. The probability of this happening at least once is

$$\begin{aligned} P(\text{at least 1 missing label}) &= 1 - P(\text{no missing labels}) \\ &= 1 - (1 - p_a)^{x_L} \end{aligned} \tag{5.9}$$

This is shown in figure 5.14, together with the expected number of missing labels (LINES $\times$ p_a), as a function of LINES and POP. For smaller problems, these are effectively zero for population sizes larger than 30. This agrees with the results of the preliminary study and suggests that POP should increase with LINES so that the probability of missing labels in the initial population is very small. If this probability is to be no greater than some threshold, P^*, manipulation of equation 5.9 gives:



(a) Probability



(b) Expectation

Figure 5.14: Missing Labels vs. LINES and POP. (a) Probability that at least one label is missing over all loci. (b) Expected number of missing labels over all loci.

$$x_P \geq \frac{\ln\left[1-(1-P^*)^{\frac{1}{x_L}}\right] - \ln|\mathbf{\Lambda}|}{\ln(|\mathbf{\Lambda}|-1) - \ln|\mathbf{\Lambda}|} \tag{5.10}$$

where $|\mathbf{\Lambda}|$ is the size of the label set (4 in this case). The optimal population size clearly increases with LINES with no upper bound. This formula is not amenable to direct manipulation, but it is shown graphically in figure 5.15 that this lower bound on population size increases at a lower rate for larger problems.

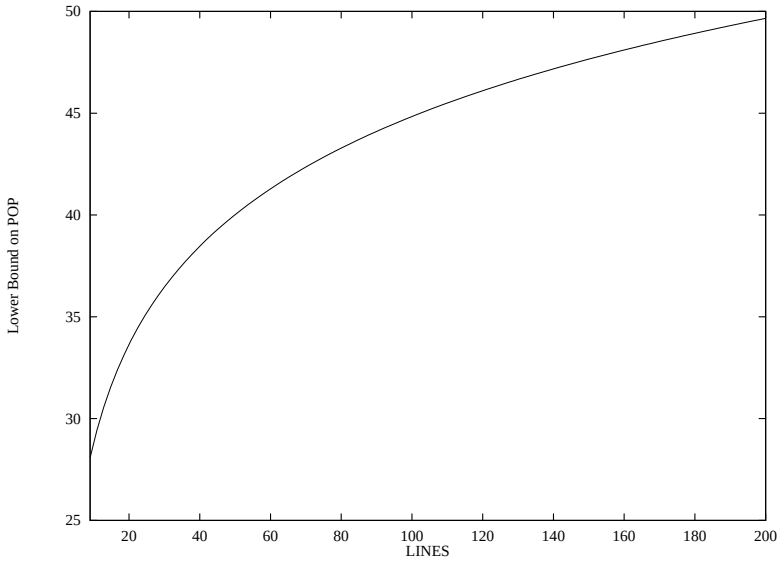


Figure 5.15: Lower Bound on Population Size. Equation 5.10 is plotted as a function of LINES with P^* set to 0.9.

Under the assumption that the population is large enough to "guarantee" that all labels appear at all loci, all that remains is for the genetic algorithm to assemble a good initial guess for gradient ascent. Following this argument, it would seem that high mutation and crossover rates are grist to the mill for gradient ascent. There must be some point beyond which increasing the mutation rate is counterproductive, since mutation indiscriminately perturbs the population. Such a danger does not exist for crossover, since it has its disruptive effect only at those loci where there is disagreement as to the labelling.

To summarise, the behaviour of the hybrid genetic algorithm can be understood in terms of gradient ascent transforming a good initial guess into an optimal solution. This happens over the course of a single algorithm iteration. So, provided the population is large enough to furnish a good selection of labels for all loci, crossover and mutation can stitch together a suitable initial guess within a few iterations. Clearly, the more iterations, the larger the chance of this occurring, provided selection pressure is not too great.

The third goal of this study is to establish how robust the models are for the purpose of setting control variables. Only the optimal conditions for the algorithm described in section 5.3.3 are relevant here. If the crossover type is fixed at non-two point (uniform), equation 5.5 becomes

$$\left.\begin{aligned} r &= \text{probit}^{-1}(\eta) \\ \eta &= 3.783 - 0.1786x_L + 1.176\ln x_P + 1.080\ln x_G + 1.010x_X + 6.066x_M \end{aligned}\right\} \quad (5.11)$$

The task is now to solve the inverse problem: for a given value of LINES, what values of POP, GEN, P_x and P_m will give an acceptably high success probability, p? This is generally not an easy problem to solve; however if the surface is simple enough, numerical optimisation techniques such as the downhill simplex method can be used (see pages 408-412 of (Press et al. 1992) for a good description and implementation). Table 5.10 shows how well the model predicts control variables for (1) the four problems in the data set, (2) three new problems requiring interpolation, and (3) four problems requiring extrapolation.

The model seems reasonable for interpolation, which is not really surprising since the process of model-fitting is essentially one of interpolation. However, the model's performance degrades very rapidly on extrapolation, giving excessively large estimates for POP, GEN, and in the last case P_m. The required population sizes are at odds with both the predictions of equation 5.10 and one's experience with genetic algorithms. On the other hand, the predictions for crossover and mutation rates appear quite reasonable, except in the last case. This suggests that the problem is with the form of the model. Although there are good reasons to believe that logarithmic terms in POP and GEN explain the algorithm's performance well, a little re-arrangement of equation 5.11 will indicate that, all other things being equal, the model is of the form: $\ln x_P \propto x_L + k$. This suggests that the population size should increase exponentially with the number of lines to be labelled, exactly the opposite of the previous conclusion.

The implication of equation 5.10 and figure 5.15, is that a model should be used which is logarithmic in LINES rather than in POP and GEN. Such a model is shown in equation 5.12, and also includes interaction terms which further increase the estimated success rate. Since it is generally inadvisable to extrapolate with polynomial models, only linear terms

LINES	Simplex Optimum					Predicted 95%	Actual p
	POP	GEN	P_x	P_m	$\hat{p}$	Conf. Int. for p	(N=100)
9	5	2	0.008	0.0003	1.00	[1.00, 1.00]	1.00
21	6	2	0.005	0.001	0.99	[0.99, 1.00]	1.00
28	6	3	0.38	0.050	0.99	[0.99, 1.00]	1.00
40	7	13	0.77	0.040	0.99	[0.99, 1.00]	1.00
26	6	3	0.29	0.048	0.99	[0.99, 1.00]	0.94
33	6	5	0.69	0.043	0.99	[0.99, 1.00]	1.00
36	6	8	0.79	0.041	0.99	[0.99, 1.00]	1.00
41	8	14	0.69	0.039	0.99	[0.99, 1.00]	1.00
53	22	32	0.78	0.025	0.99	[0.98, 1.00]	1.00
65	61	59	0.71	0.076	0.99	[0.96, 1.00]	1.00
130*	1039	793	0.98	0.93	0.99	[0.07, 1.00]	1.00

Table 5.10: Performance of Model 5.11. The upper rows are the original test problems, the middle rows are interpolation problems and the bottom rows are extrapolation problems. The simplex optimum vector is shown, together with a 95% confidence interval for the outcome based on the model, and the actual outcome over 100 trials of the algorithm. The initial guess for the simplex method was POP=40, GEN=8, P_x=0.8 and P_m=0.03, except for the row marked (*), which was POP=800, GEN=800, P_x=0.9 and P_m=0.3.

are included. Table 5.11 shows the results for this model. It still appears to overestimate the requirements for POP and GEN but nowhere near as badly as model 5.11. On the other hand, it is not very good for small problems.

$$\left.\begin{aligned} r &= \text{probit}^{-1}(\eta) \\ \eta &= 20.92 - 5.597 \ln x_L + 0.02215 x_P - 0.1781 x_G + 0.6277 x_X - 11.64 x_M + \\ &\quad 0.01597 x_P x_G + 0.1229 x_g x_X + 5.506 x_G x_M \end{aligned}\right\} \quad (5.12)$$

The surface can be visualised by fixing P_x and P_m as before, and requiring that $r = 0.99$, hence probit$(r) = 2.326$: it is shown in figure 5.16. The figure shows that the model extrapolates stably as LINES and POP increase well beyond the range of the data set. Whether or not the predicted value of GEN is correct is another matter: it is decreasingly likely to be accurate as one departs further from the original data.

LINES	Simplex Optimum					Predicted 95%	Actual p
	POP	GEN	P_x	P_m	$\hat{p}$	Conf. Int. for p	(N=100)
9	6	2	0.006	0.001	1.00	[1.00, 1.00]	0.99
21	5	1	0.005	0.001	1.00	[1.00, 1.00]	0.57
28	5	1	0.19	0.00	1.00	[0.97, 1.00]	1.00
40	6	6	0.74	0.05	0.99	[0.98, 1.00]	0.90
26	5	1	0.00	0.00	1.00	[0.98, 1.00]	0.82
33	6	3	0.72	0.038	0.99	[0.98, 0.99]	1.00
36	6	5	0.52	0.044	0.99	[0.98, 0.99]	1.00
41	6	7	0.60	0.042	0.99	[0.98, 0.99]	0.98
53	7	10	0.72	0.044	0.99	[0.96, 1.00]	1.00
65	16	8	0.79	0.048	0.99	[0.93, 1.00]	1.00
130	32	11	0.74	0.044	0.99	[0.88, 1.00]	1.00

Table 5.11: Performance of Model 5.12. The upper rows are the original test problems, the middle rows are interpolation problems and the bottom rows are extrapolation problems. The simplex optimum vector is shown, together with a 95% confidence interval for the outcome based on the model, and the actual outcome over 100 trials of the algorithm. Again, the initial guess for the simplex method was POP=40, GEN=8, P_x=0.8 and P_m=0.03.

5.3.4 Solution Yield

The simplest model found to explain the data well is given in equation 5.13. This is shown under four conditions, 2 each for COST and CROSS, in figures 5.17 and 5.18, from which it is clear that these models will only be usable for interpolation. The models in figure 5.17 have a moderately large quadratic component in GEN which suggests that solution yield will decrease as GEN is increased. This is plausible since by genetic drift the population will eventually become saturated and the number of distinct optimal solutions will fall to 1. The models for the Bayesian criterion in figure 5.18, however, suggest that as LINES is increased beyond 40, the solution yield should increase. Unsurprisingly, extrapolation of equation 5.13 gives poor results.

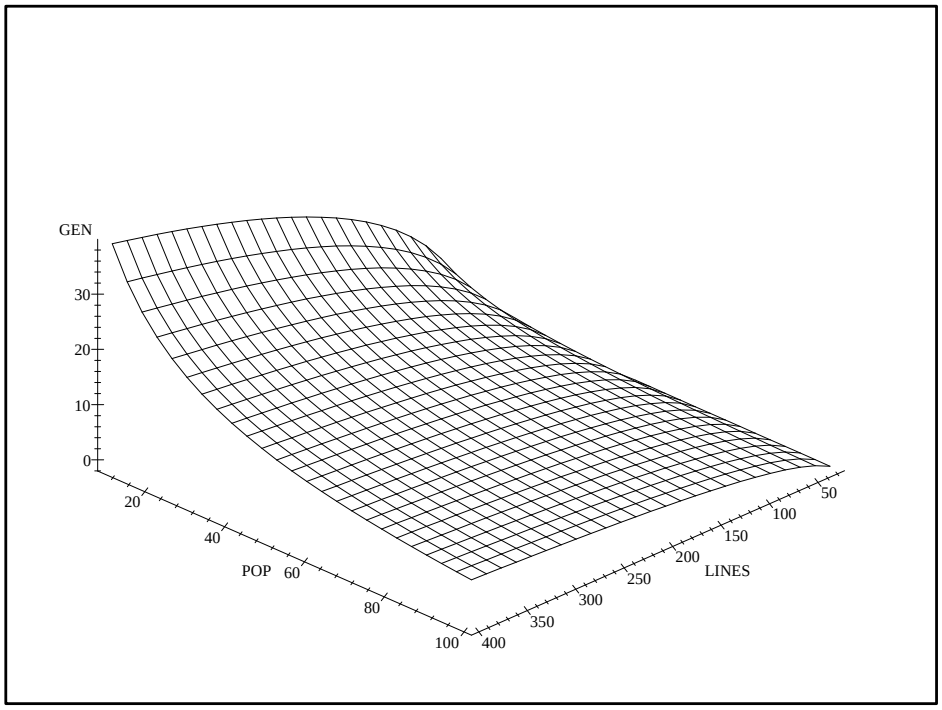
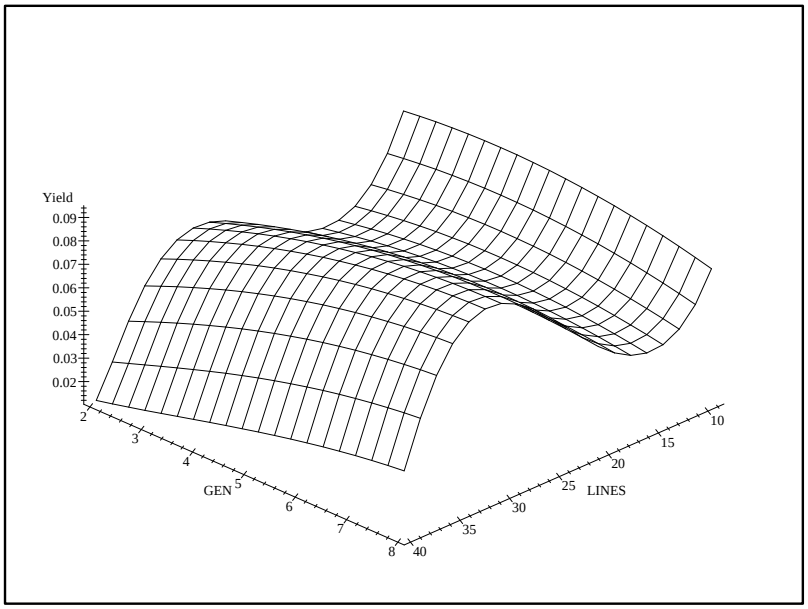
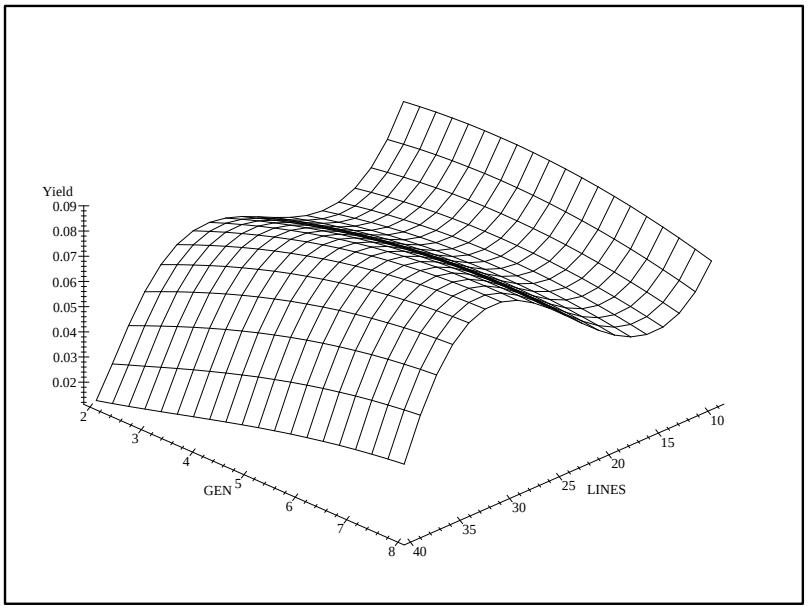


Figure 5.16: Plot of Model 5.12. Number of iterations (GEN) required to achieve a success rate of 0.99 is plotted against LINES and POP. P_x and P_m are fixed at 0.8 and 0.03.

$$
\left.
\begin{aligned}
y &= g^{-1}(\eta; 39.51) \\
\eta &= 6.819 - 1.553x_L + 0.07095x_L^2 - 0.001019x_L^3 + 0.1428x_G - \\
&\quad 0.03260x_G^2 + 1.140x_X + 7.055x_M + 0.01235x_Lx_G + 0.01738x_Lx_X + \\
&\quad w_C(-17.85 + 3.221x_L - 0.1431x_L^2 + 0.001919x_L^3 - 0.09142x_G + \\
&\quad 0.02629x_G^2 - 0.7607x_X - 7.077x_M - 0.01154x_Lx_G - \\
&\quad 0.02247x_Lx_X) + \\
&\quad w_X(-1.714 + 0.3029x_L - 0.01422x_L^2 + 0.0001942x_L^3) \\
w_C &= \begin{cases} 1 & \text{for the Bayesian criterion} \\ 0 & \text{for the linear criterion} \end{cases} \\
w_X &= \begin{cases} 1 & \text{for two point or geometric crossover} \\ 0 & \text{for uniform or half-uniform crossover} \end{cases}
\end{aligned}
\right\} \quad (5.13)
$$

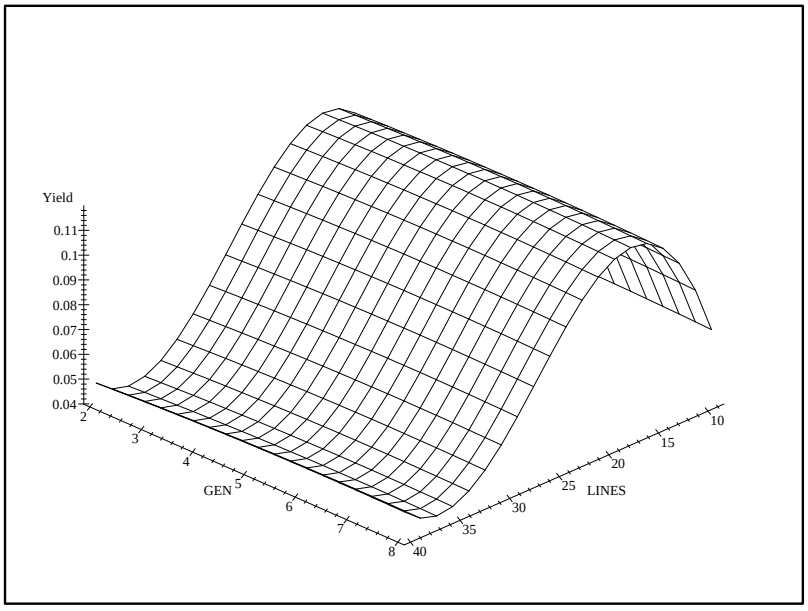


(a) Two point and geometric crossovers

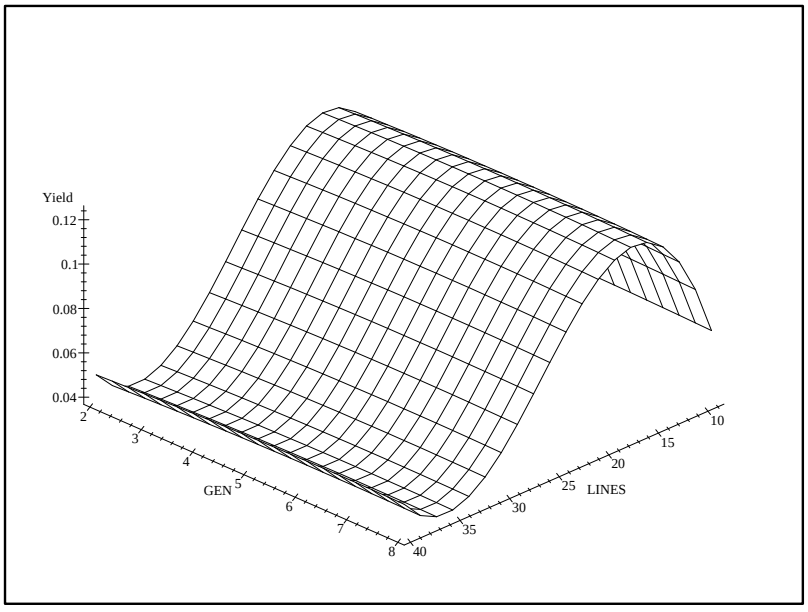


(b) Uniform and half-uniform crossovers

Figure 5.17: Model 5.13 with linear criterion. Solution yield as a proportion of population size. P_x is fixed at 0.8, P_m at 0.03.



(a) Two point and geometric crossovers



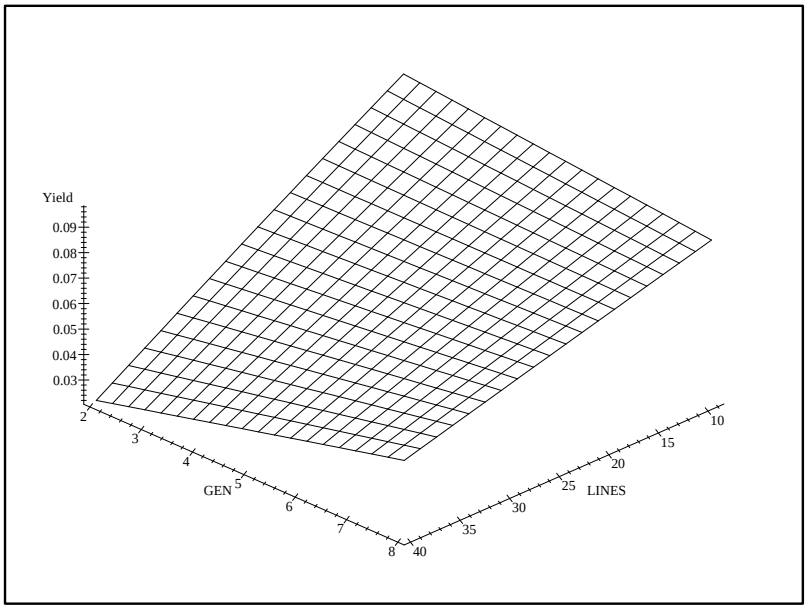
(b) Uniform and half-uniform crossovers

Figure 5.18: Model 5.13 with Bayesian criterion. Solution yield as a proportion of population size. P_x is fixed at 0.8, P_m at 0.03.

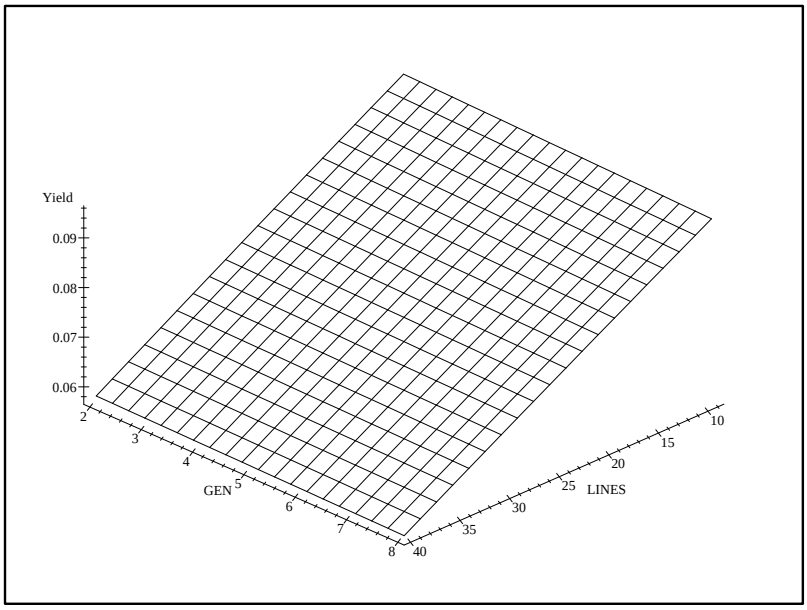
A simpler, first-order model is given in equation 5.14. Figure 5.19 shows the surfaces for the linear and Bayesian criteria, both with uniform crossover. Unfortunately, this model is not particularly useful since it always recommends that GEN be set as low as possible and is not much more enlightening about P_x or P_m.

$$\left.\begin{aligned} y &= g^{-1}(\eta; 173.5) \\ \eta &= 10.49 - 0.4774x_L - 0.7547x_G + 6.832x_X + \\ &\quad 24.75x_M + 0.0411x_L x_G - 0.05175x_L x_X + \\ &\quad w_C(1.431 + 0.2992x_L + 0.6844x_G - 4.255x_X - \\ &\quad 25.06x_M - 0.04004x_L x_G) \\ &\quad +0.07401w_X \\ w_C &= \begin{cases} 1 & \text{for the Bayesian criterion} \\ 0 & \text{for the linear criterion} \end{cases} \\ w_X &= \begin{cases} 1 & \text{for two point or geometric crossover} \\ 0 & \text{for uniform or half-uniform crossover} \end{cases} \end{aligned}\right\} \quad (5.14)$$

The difficulty appears to be that if the algorithm is allowed to iterate, the population will tend to converge to a single solution. This means that the best solution yield occurs when initial guesses leading to different optimal solutions under gradient ascent are formed from the starting population. However, reducing the iteration limit will tend to compromise the search for global optima. So the fundamental problem with using control variables to influence solution yield appears to be the loss of diversity over algorithm iterations. This loss of diversity is mediated by the selection operator, which is not directly affected by the control variables, and which shall be investigated in chapter 6.



(a) Linear Criterion



(b) Bayesian Criterion

Figure 5.19: Model 5.14. Solution yield as a proportion of population size. Uniform crossover was used for both. P_x is fixed at 0.8, P_m at 0.03.

5.4 Graph Matching

This section repeats the analysis of section 5.3 for graph matching, which is a more realistic (and difficult) problem than line labelling. The knowledge of line labelling from the previous section can be used to a limited extent to guide the exploration of this problem.

Unlike line labelling, inexact graph matching does not permit the identification of global optima purely on the basis of the matching criterion. The global optimum of the criterion need not coincide with the best possible match since it is an indirect measure. For synthetic graphs it is possible to calculate the fraction of correct mappings, since the ground truth is known. This quantity, Fc, is a proportion, $\mathrm{Nc}/|\mathbf{V}_D|$, where the number of correct mappings, Nc, is binomially distributed on the size of the data graph, $|\mathbf{V}_D|$. The average fraction of correct mappings in the final population will be used to measure algorithm performance. By the central limit theorem, this quantity is asymptotically Normally distributed as the population size increases. The advantage of this approximation is that it allows the use of ANOVA instead of logistic regression. ANOVA is simpler because model terms may be considered independent in orthogonal designs. Model-fitting is also faster, so it is feasible to look at all interactions between the explanatory variables. According to Snedecor and Cochran (page 53 of (Snedecor and Cochran 1980)), the smallest population size for which this approximation is valid is about 20. Ten replications are performed for each combination.

Since it is not possible count the number of distinct global optima, the measure of solution yield will be based on the number of distinct solutions with fractions correct above a certain threshold. For 10% relational corruption the empirical lower bound of matching accuracy is about 95%, so this will be taken as the threshold (see figure 3.5 in section 3.8 of chapter 3). This quantity is binomially distributed on the population size. Ten replications are performed for each combination.

The next section discusses some previous work by Cross on graph matching with a genetic algorithm (Cross 1998), then algorithm performance is considered in section 5.4.2, and solution yield in section 5.4.3.

5.4.1 Preliminaries

In (Cross 1998), Cross found that the gradient ascent step was essential if the algorithm was to find an acceptable solution in a reasonable amount of time. For example, without gradient ascent, a 10-node graph required a population size of 1000 and thousands of iterations. Cross also used a somewhat *ad hoc* argument which suggested that the population size should be made approximately equal to the number of nodes in the data graph, so that the average Hamming distance between members of the population would be less than some threshold. The crossover rate was fixed at 1.0 and the mutation rate was set equal to the label error probability, P_e.

Fixing the crossover rate at 1.0 seems sensible in view of Cross's findings and the results for line labelling. However, tying the mutation rate to P_e is more contentious. The label error probability refers to a theoretical process of corruption which turns a consistent match into an inconsistent one (Wilson and Hancock 1997; Hancock 1994), and is viewed as a control variable which progressively tightens constraints over successive iterations of gradient ascent. Mutation, on the other hand, is a process by which the genetic algorithm recovers labels that have been lost through selection. Simultaneously reducing the mutation rate and tightening the matching constraints will certainly lead to algorithm convergence, but not necessarily to the best possible match. Indeed, it could be argued that as the matching criterion progressively enforces constraints, the mutation rate should be increased: gradient ascent is more likely to correct noise when constraints are tight (up to a point). Cross found that when the mutation rate was decoupled from P_e, the algorithm appeared to be relatively insensitive to mutation for $P_m \leq 0.6$.

5.4.2 Final Correct Fraction

This study compared algorithm performance over Bayesian and linear criteria, and three different crossovers (uniform, balanced geometric and balanced contiguous) for four matching problems. The graphs had 20, 30, 40 and 50 nodes with 10% additive relational corruption. According to Cross's recommendations, population sizes of 30, 40, 50 and 60 were used. The crossover rate was fixed at 1.0 and the mutation rate took values of 0.2, 0.4, 0.6 and 0.8. The iteration limits were 4, 6, 8 and 10. These explanatory variables have the same names as in table 5.3, except that problem size is denoted NODES, x_N in formulae, instead of LINES. The main effects are summarised in figure 5.20. The most

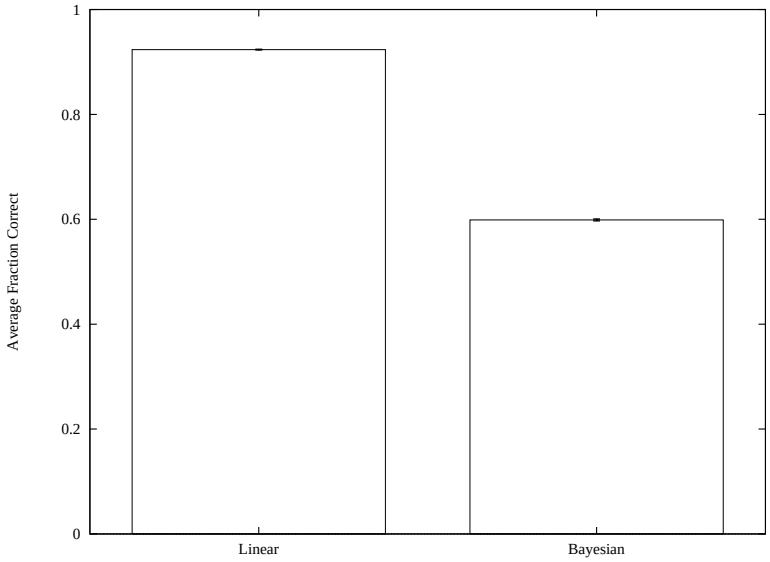
surprising findings are that the linear criterion outperformed the Bayesian one, and that the population size appeared to have no main effect. There also appear to be structural effects at play as well as graph size. The best mutation rate is somewhere between 0.4 and 0.6 which accords with Cross's predictions and (unreported) observations for line labelling. As with line labelling, annealing the label error probability, P_e, was not found to significantly affect the outcome.

The superiority of the linear criterion may be due to the more direct way in which it measures consistency: it is based on the sum of minimum edit distances to dictionary items. It is axiomatic that a mapping with a low minimum edit distance to its dictionary is better than one with a high minimum distance. The Bayesian criterion measures consistency by summing the edit distance over all dictionary items. The smoother optimisation surface thus provided reduces the risk that gradient ascent will stop in a poor local optimum. However, the genetic algorithm is a global optimiser, so this benefit of the Bayesian criterion disappears, leaving only the cost associated with considering distances between labelling configurations and inappropriate dictionary items. The fact that this trade-off appears to favour the linear criterion for graph matching but the Bayesian one for line labelling can be explained by recalling how algorithm performance for line labelling is measured. For line labelling, the outcome is all-or-nothing: the algorithm either locates a global optimum or it does not. Hence, any correlation between the criterion and the fraction of correct labellings is immaterial. However, the global optima of both criteria correspond to 100% correct labellings because the problem is exact. This is not the case with inexact graph matching, where a fraction correct of 1.0 may not coincide with the global optimum of the Bayesian criterion, but must still coincide with the global optimum of the linear one.

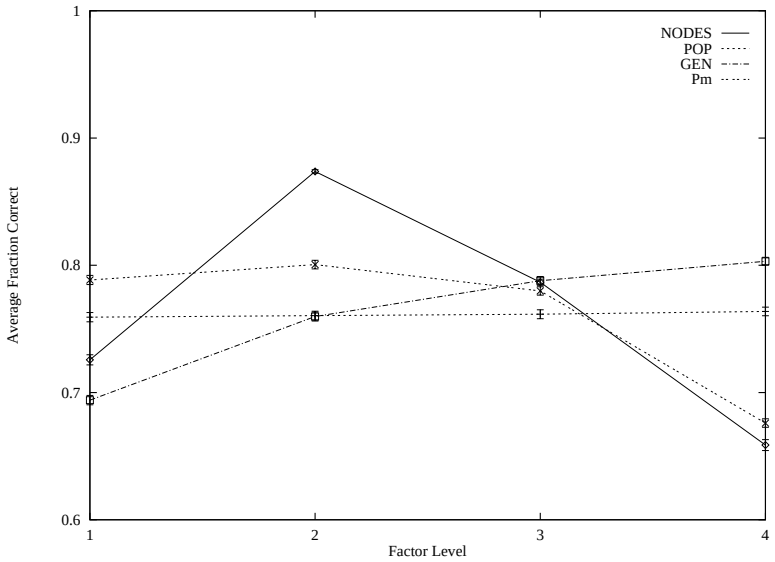
Population Size

The apparent lack of main effect for population size does not mean that this is not an important consideration. First, the possibility of there being interactions with other variables, which equalise the marginal totals for POP, must be eliminated (this is not likely). The ANOVA given in table 5.12 showed that POP does actually have a significant main effect together with a significant interaction with COST. This is shown in figure 5.21.

The interaction is much smaller than the main effect. The conclusion is that POP, although



(a) Criterion



(b) Other Variables

Figure 5.20: Average Fraction Correct. (a) Effect of criterion. The linear criterion appears to perform better. (b) NODES, POP, GEN and P_m. A larger slope indicates that a particular variable has more influence on the outcome.

statistically significant, has a very small effect indeed: it accounts for less than 1% of the total variation in the data. The reason for this is almost certainly that all of these population sizes are sufficiently large for all problems to be solved by the algorithm.

At first sight, the missing label argument from section 5.3.3 and equation 5.10 suggests that

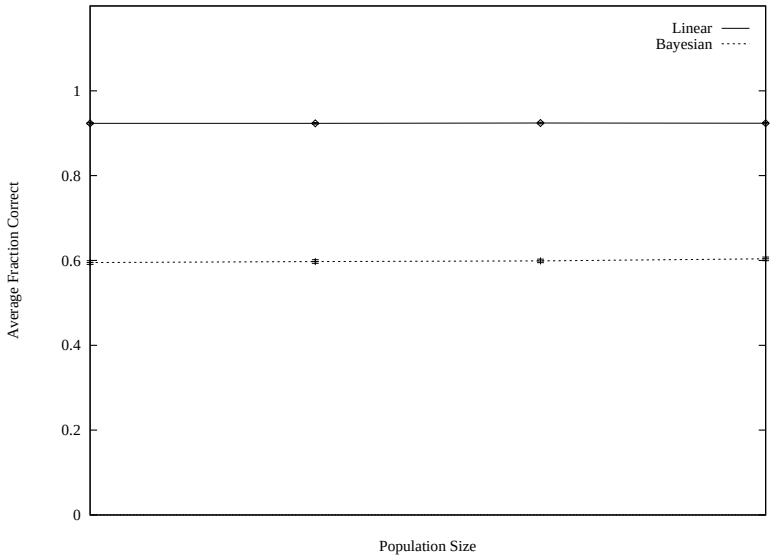


Figure 5.21: Interaction of COST with POP. The plots for fraction correct versus population size are nearly parallel, indicating a very small interaction.

even larger population sizes are needed, e.g. about 250 for a 20-node graph for a missing label probability of 0.001. However, for graph matching an inferior initial guess suffices. It has already been established in chapter 3 that a poor initial guess is good enough for gradient ascent when matching graphs. It is capable of correcting 90% initialisation error to around 10% (see figure 3.6 in section 3.8 of chapter 3). So the genetic algorithm need only assemble an initial guess with about 10% of the mappings correct for gradient ascent to have a good chance of success within a few iterations. Panel (b) of figure 5.20 supports this idea since the algorithm seems to need at least one iteration to produce a reasonable solution.

Under the hypothesis that, for a graph with 10% corruption, there are on average 1.1 correct labels per node, the probability of at least one correct label appearing at a given locus at least once in the initial population is $pc = 1 - (1 - \frac{1.1}{|\mathbf{V}_M|+1})^{x_P}$. So the probability that at least 10% the loci will have at least one correct label is

$$P(\geq 1 \text{ correct label at } \geq 10\% \text{ loci}) = \sum_{\frac{|\mathbf{V}_D|}{10} \leq k \leq |\mathbf{V}_D|} \left(\begin{array}{c} |\mathbf{V}_D| \\ k \end{array} \right) pc^k (1 - pc)^{|\mathbf{V}_D| - k} \quad (5.15)$$

For a 50-node problem and a population size of 10, this probability is 0.98; with a population size of 5, the probability is 0.60. So even for the largest graph, a population of about 10 should be adequate.

Analysis of Variance

Part of the analysis of variance table is given in table 5.12. Main effects account for about 80% of the variation, double interactions for another 15%, and triple interactions for 1%, with error accounting for the remaining 5%. The most important effects together with their contributions to the total variation are given in table 5.13.

These contributions suggest the appropriate model should probably just include these terms. The best model found was a cubic response surface which is virtually guaranteed to break down under extrapolation. Taking the partial derivative with respect to NODES showed that, all other things being equal, there was a maximum at NODES=15 and a minimum at NODES=50, beyond which Fc continued to rise (steeply). The implication

Source	SS	df	MS	F-ratio	P
Main Effects:					
COST	405.0	1	405.0	16300.0	0.00
CROSS	3.64	2	1.82	731.0	0.00
LINES	96.2	3	32.1	12900.0	0.00
POP	0.0431	3	0.0144	5.76	0.00
GEN	26.8	3	8.94	3590.0	0.00
PM	38.0	3	12.7	5080.0	0.00
2-way Interactions:					
COST.CROSS	0.426	2	0.213	85.6	0.00
COST.LINES	94.7	3	31.6	12700.0	0.00
COST.POP	0.0371	3	0.0124	4.97	0.00
COST.GEN	2.36	3	0.785	315.0	0.00
COST.PM	0.0917	3	0.0306	12.3	0.00
CROSS.LINES	0.981	6	0.163	65.6	0.00
CROSS.POP	0.0402	6	0.00671	2.69	0.01
CROSS.GEN	0.0481	6	0.00801	3.22	0.00
CROSS.PM	0.0982	6	0.0164	6.57	0.00
LINES.POP	0.0287	9	0.00319	1.28	0.24
LINES.GEN	5.36	9	0.596	239.0	0.00
LINES.PM	1.08	9	0.120	48.3	0.00
POP.GEN	0.0369	9	0.00410	1.65	0.10
POP.PM	0.0138	9	0.00154	0.617	n/s
GEN.PM	1.92	9	0.213	85.5	0.00
...	...	...	...	...	...
Error	34.4	13824	0.002491		
TOTAL	719.0	15359			

Table 5.12: Analysis of Variance. For brevity, only main effects and double interactions are shown. Abbreviations: SS=sum of squares, df=degrees of freedom, MS=mean square (MS=SS/df), n/s=not significant.

Effect	Contribution
COST	0.56
LINES	0.13
COST.LINES	0.13
PM	0.05
GEN	0.04
LINES.GEN	0.01
CROSS	0.01
	0.93
Error	0.05
TOTAL	0.98

Table 5.13: Contributions of Major Terms to Total Variation. The additional 2% is distributed among 22 terms which individually account for less than 0.4% of the total variation.

was that the larger the graph, the larger Fc will be for the same set of control variables: this is clearly implausible. Recalling panel (b) of figure 5.20, one can see that structural differences between the graphs account for the cubic polynomial. Since it is the smallest of these graphs that seems anomalous, a quadratic surface would also probably be incorrect; the best that can be done is linear. Fitting a quadratic model for GEN is also guaranteed to give trouble sooner or later because the maximum will be to the right of GEN=4 - i.e. the predicted Fc will eventually start to fall as GEN is increased. It is also possible to simplify P_m by regarding values below 0.6 as reasonable and those above as unreasonable (this is supported by an analysis of the t-values for the levels of P_m). The final model obtained is given in equation 5.16.

$$
\left.
\begin{aligned}
\mathrm{Fc} &= 1.083 - 0.006526x_N - 0.007803x_G + 0.0006489x_Nx_G + \\
&\quad w_C(-0.3506 - 0.001249x_N + 0.01099x_G) + \\
&\quad w_X(0.007154 - 0.0008139x_N) - 0.02214w_Cw_X + \\
&\quad w_M(-0.03976 - 0.01054x_G) \\
w_C &= \begin{cases} 1 & \text{for the Bayesian criterion} \\ 0 & \text{for the linear criterion} \end{cases} \\
w_X &= \begin{cases} 1 & \text{for balanced geometric crossover} \\ 0 & \text{otherwise} \end{cases} \\
w_M &= \begin{cases} 1 & \text{if } P_m > 0.6 \\ 0 & \text{otherwise} \end{cases}
\end{aligned}
\right\} \qquad (5.16)
$$

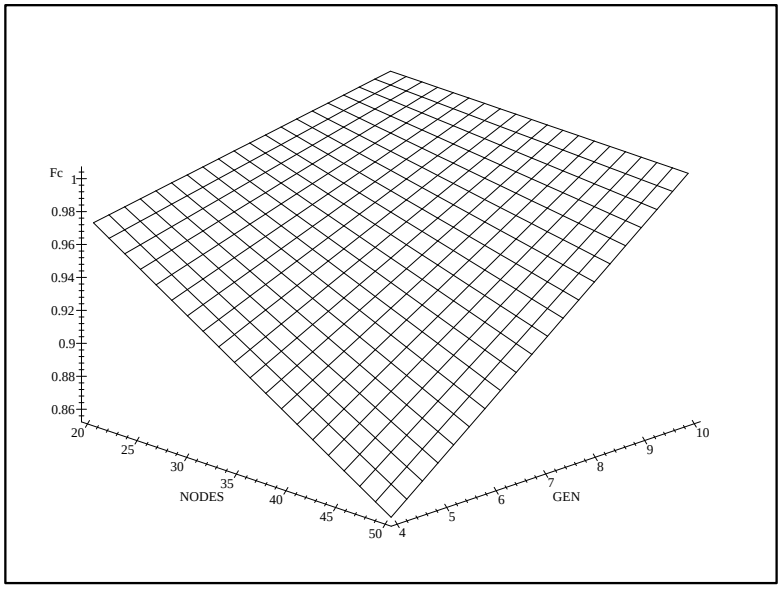
Figure 5.22 shows the optimal condition of this model together with the results of extrapolation of NODES and GEN. Extrapolation seems possible with respect to NODES but not GEN: beyond about 14 iterations the slope for NODES becomes positive - i.e. the model suggests that larger problems will be easier to solve when the number of iterations is also large, which is implausible.

Extrapolation Study

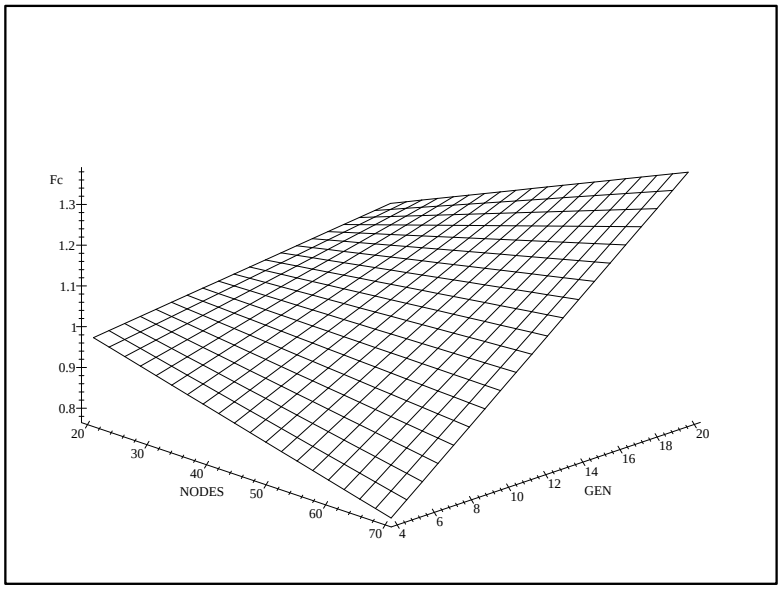
Fortunately, it will not generally be necessary to extrapolate quantitative components of model 5.16 since NODES is a feature of the problem under study and, given that POP will be made large enough to furnish a reasonable initial guess for gradient ascent, GEN need not be set too high. The element of extrapolation comes from the fact that for larger graphs, the linear criterion will be used with uniform or balanced contiguous crossover, together with a mutation rate below 0.6. If GEN is fixed at 10, model 5.16 becomes linear in NODES, as shown in equation 5.17 and figure 5.23.

$$
\mathrm{Fc} = 1.005 - 0.000037x_N \qquad (5.17)
$$

Extrapolation of this model can only show how well the algorithm should be expected to perform under the conditions just mentioned. Table 5.14 compares the predicted and actual outcomes: the good agreement between them shows that the model is reasonable.



(a) Fitted Model



(b) Extrapolation

Figure 5.22: Model 5.16 for linear criterion with balanced contiguous or uniform crossover and $P_m \leq 0.6$: (a) Fitted model and (b) Extrapolation. The model is not particularly good under extrapolation of GEN.

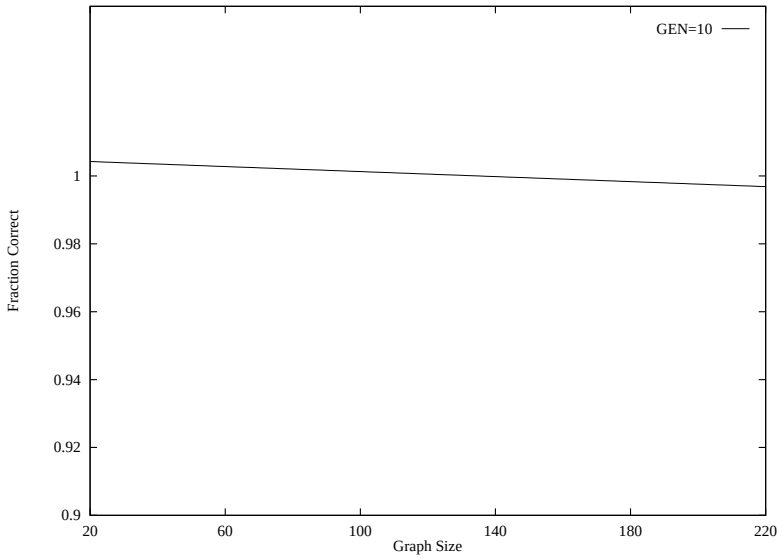


Figure 5.23: Model 5.17. The estimated final correct fraction is a slowly decreasing linear function of NODES.

Graph Size	Population Size	Predicted 95% Conf. Int. for Fc	Actual Fc (N = 40)	Actual 95% Conf. Int. for Fc
70	10	[0.99, 1.01]	0.98	[0.87, 1.07]
100	15	[0.98, 1.03]	0.98	[0.97, 0.99]
150	20	[0.96, 1.04]	0.97	[0.93, 1.01]
200	25	[0.94, 1.06]	0.98	[0.96, 1.00]

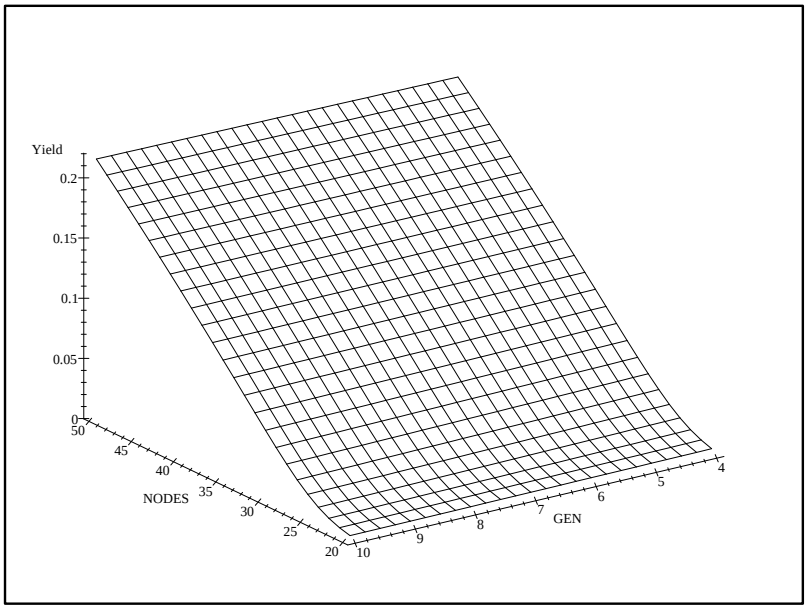
Table 5.14: Extrapolation of Model 5.17. These experiments used the linear criterion, uniform crossover, and had GEN set to 10 and P_m set to 0.4. Population sizes were chosen so that the probability of there being a correct label for at least 10% of nodes in the initial population was around 0.9 (equation 5.15). Predicted and actual 95% confidence intervals for Fc are given: overlap indicates good model performance.

5.4.3 Number of Distinct Solutions

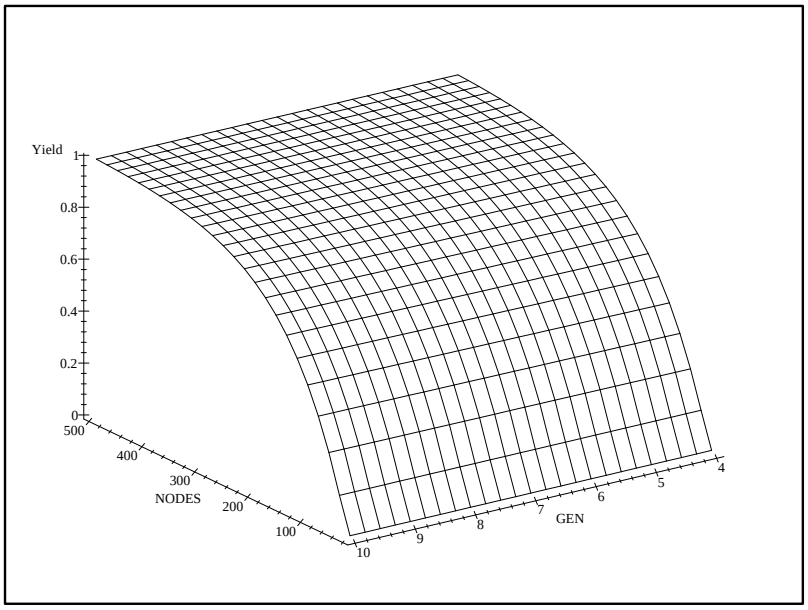
Solution yield was measured by the number of distinct matches in the final population with at least 95% of the mappings correct. The number of distinct matches in the final population is inversely correlated with the average final fraction correct, which merely reflects the fact that the population is more diverse when the algorithm fails to find good solutions. Only the linear criterion was considered. As with previous models, the model which best fitted the data had a cubic term, presumably as a result of structural differences between the graphs. A simple linear model, given in equation 5.18, was fitted. This model is shown in figures 5.24 and 5.25 as fitted and under extrapolation with respect to NODES.

$$\left.\begin{aligned} y &= g^{-1}(\eta; 46.6) \\ \eta &= -12.68 + 0.4076x_N - 0.2401x_G - 0.4074x_M + \\ &\quad 0.002940x_N x_G - 0.01310x_N x_M + 0.2919x_G x_M + \\ &\quad w_X(4.463 - 0.1780x_N - 0.6729x_G + 4.076x_M + \\ &\quad 0.02737x_N x_G - 0.1661x_N x_M) \\ w_X &= \begin{cases} 1 & \text{for balanced geometric crossover} \\ 0 & \text{otherwise} \end{cases} \end{aligned}\right\} \quad (5.18)$$

The model appears to be relatively insensitive to mutation rate and iteration limit. However, it suggests that solution yield should increase as the graphs become larger. This result is plausible to the extent that larger graphs may have more good matches than smaller ones, but it should be treated with caution. As with line labelling, it seems that attempting to use the genetic algorithm control variables to influence the solution yield is inappropriate: this is the province of the selection operator to which we turn our attention in chapter 6.

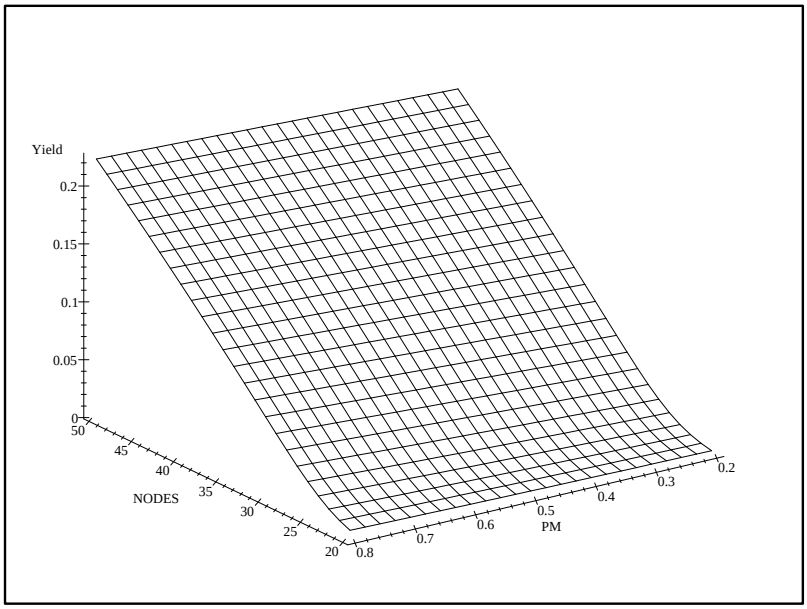


(a) Interpolation

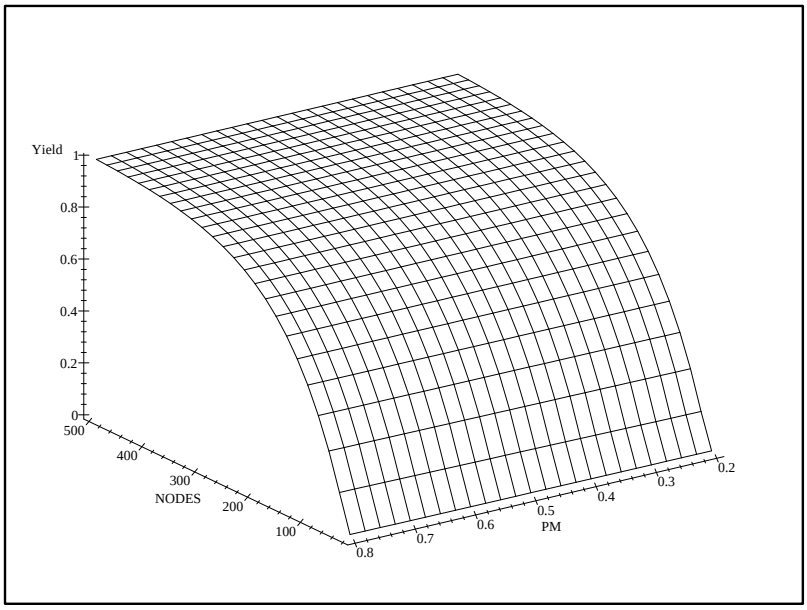


(b) Extrapolation

Figure 5.24: Model 5.18. Solution yield as a proportion of population size vs. graph size and iteration limit. P_m is fixed at 0.4; crossover is assumed to be uniform or balanced contiguous.



(a) Interpolation



(b) Extrapolation

Figure 5.25: Model 5.18. Solution yield as a proportion of population size vs. graph size and mutation rate. GEN is fixed at 10; crossover is assumed to be uniform or balanced contiguous.

5.5 Summary

This chapter has described a series of factorial experiments which attempted to

- find optimal genetic algorithm control variable settings
- model the relationship between control variables and algorithm performance
- establish the stability of those models under extrapolation

for line labelling and graph matching problems. Two outcomes were considered: quality of the solution found, and solution yield.

Optimal conditions common to both problems were: the addition of a gradient ascent step, the use of disruptive crossovers at a rate of 1.0, and setting the mutation rate below 0.6. It was also found that annealing P_e had no effect on the outcome. The Bayesian criterion outperformed the linear one for line labelling, but the linear criterion was better for graph matching. Only a few algorithm iterations were needed.

In general, the most important variable determining algorithm performance was the problem to be solved, with structural effects rather than problem size responsible for much of the variation in the data sets. Second to problem size was population size, which can be minimised by observing that the genetic operators need only assemble a good initial guess for gradient ascent, which is responsible for most of the optimisation. These population sizes are considerably smaller than those suggested by Cross in (Cross 1998).

The empirical models for solution quality were found to be reasonably stable under moderate extrapolation with respect to problem size, predicting performance with reasonable accuracy for problems at least three times larger than those initially studied.

It was not possible adequately to control solution yield by the methods described in this chapter. Solution yield is closely coupled to diversity in the genetic algorithm population, and it is to this which we turn our attention next.

Chapter 6

Maintenance of Diversity in the Genetic Algorithm

This chapter considers more sophisticated control strategies for the genetic algorithm, with a view to enhancing the solution yield by maintaining greater diversity in the population. Previous chapters have considered the crossover and local search operators, and the choice of algorithm parameters. This chapter concentrates on mutation and selection. The novel contributions in this chapter are first, the definitions of three different yet complementary measures of diversity in the population, and second, the development of non-standard mutation and selection operators for the algorithm.

Section 6.2 defines the three quantitative diversity measures and considers the likely effects of the standard genetic algorithm operators on them. The evolution of these diversity measures is tracked: a common feature is that most of the diversity is lost within the first few algorithm iterations. Section 6.3 identifies the mutation operator as a potential cause of early loss of diversity, and considers two methods of varying the mutation rate as the algorithm proceeds. This is distinguished from other reported attempts at using variable mutation rates by the fact that the mutation rate generally increases as the algorithm proceeds. Section 6.4 considers ways of modifying the selection operator to avoid unnecessary loss of diversity. Finally, section 6.5 presents two examples of the algorithm applied to real data.

6.1 Introduction

The maintenance of diversity is crucial to the genetic algorithm, because what differentiates it from parallel generate-and-test is the ability to combine information from several sources in the quest for global optima. However, the action of the algorithm is to reduce diversity at every iteration. This happens because the selection operator can only remove particular genotypes from a finite population: it cannot create new ones. This effect of selection, known as "genetic drift", cannot be counteracted by mutation, since the required rate would be so high as to compromise the search. Neither can it be corrected by crossover, which can only recombine genetic material currently in the population. Like selection, crossover cannot introduce new information. Genetic drift is therefore a major problem with genetic algorithms, which has attracted considerable attention in the literature over the past 25 years (DeJong 1975; Baker 1987; Mauldin 1984; Louis and Rawlins 1993). The problem is even more pressing when more than one optimum is sought.

Strategies for maintaining diversity with a view to finding multiple optima tend to focus on the selection operator, either by changing the selection algorithm or by modifying the fitness function. In (Cedeño et al. 1995), Cedeño et al. used DeJong's crowding heuristic (DeJong 1975) twice per algorithm iteration. Although they obtained good results for the algorithm on a DNA sequencing problem, the method requires two additional parameters (the two "crowding factors").

In (Goldberg and Richardson 1987), Goldberg and Richardson suggested "fitness sharing", in which an individual's fitness was degraded if it was too similar to other individuals in the population. In (Deb and Goldberg 1989), Deb and Goldberg used Hamming distance as the similarity measure. There are three principal disadvantages to this scheme. First, it is necessary to estimate a "sharing parameter", which depends on a prior knowledge of the number of optima. Second, some function by which the fitness should be degraded must be defined. Third, sharing implicitly assumes that the optima are evenly spaced. In (Beasley et al. 1993), Beasley and co-workers developed the "sequential niche" technique, which combines the idea of sharing with Glover's concept of "tabu search" (Glover 1989). Sequential niching dynamically changes the fitness criterion so that previously located optima are ignored. This has the advantage of being applicable to any global optimiser, and does not require the optima to be equally spaced. However, it requires the estimation of a "niche radius", which is the minimum distance the algorithm must keep from previously

located optima. Sequential niching may not be useful for labelling problems, because global optima might be adjacent, in which case the niche radius would have to be 0, since 1 would be too large. However, a niche radius of 0 would create new local optima around the old solution, making the fitness landscape rougher in that region. This is contrary to the basic idea of sequential niching, which is to make the fitness landscape smoother around previously visited optima. An additional disadvantage of sharing and sequential niching when applied to discrete problems is that the fitness de-rating function would be difficult to implement efficiently, because every fitness evaluation would require an exhaustive check through the list of known optima.

Smith and co-workers showed in (Smith et al. 1993) that no significant modifications of the algorithm are necessary for certain problems. They demonstrated that in a simulation of the immune system, in which fitness was assigned by a bidding process, stable subpopulations developed in a standard genetic algorithm.

The view taken in this chapter is that the genetic algorithm is already over-parameterised, so the introduction of new parameters is to be avoided. The work of Smith et al. (Smith et al. 1993) showed that in certain circumstances, diversity can be maintained without significant alterations to the algorithm. The preliminary studies in chapter 4 suggest that the hybrid genetic algorithm applied to labelling problems may be such a case. However, before attempting to maintain diversity, it is necessary to define and then measure it.

6.2 Diversity Measures

Taken loosely, the term "diversity" expresses the concept of differences between population members. Many authors define these differences in terms of Hamming distance (e.g. Louis and Rawlins in (Louis and Rawlins 1993), and Deb and Goldberg in (Deb and Goldberg 1989)). Whilst Hamming distance may be appropriate for populations of bit strings, it may not be useful when the encoding is non-binary. To see the difficulty, consider a population of symbol strings. Suppose that there are only two types of string in the population, and that the Hamming distance between them is 20. If the population contains equal numbers of each type of string, the average Hamming distance between pairs of strings will be 10. Now suppose that each string in the population is unique at 10 positions. The average pairwise Hamming distance is still 10, but this second population is surely more diverse than the first. Nevertheless, the classes in the first population are more different than

those in the second. This section quantifies three aspects of population diversity. First, the degree of clustering, or how uniform the population is; second, how far apart the population members are in the search space; and third, the size of the pool of information available to the crossover operator.

6.2.1 Clustering

From an information-theoretic point of view, the genetic algorithm's search space is an alphabet from which a population of symbols is drawn. The Shannon entropy (Shannon 1948) is a natural measure of how much information about the search space is contained in the population, and corresponds to the degree of clustering (a "cluster" is a bag of identical strings in this case). The Shannon entropy is defined as follows for a bag of strings, $\mathbf{\Psi}$. Let pr_i be the proportion of the i^{th} distinct string in $\mathbf{\Psi}$. Then the Shannon entropy, S, is given by:

$$S = -\sum_{i \in \mathbf{\Psi}} \mathrm{pr}_i \log_r \mathrm{pr}_i \tag{6.1}$$

The base of the logarithm, r, depends on the number of possible values of each element in a string. For a binary encoding, this would be 2, but since the encoding need not be binary, it seems more sensible to use the natural logarithm, $r = e$, and measure the information in "natural units" (Shannon 1948).

The entropy measures clustering. It is 0 when all the strings are identical. Otherwise, S is greater than 0, and maximal when all the strings are distinct, in which case $S = \ln |\mathbf{\Psi}|$. Consider replacing some string, x, with a new string, y, and the effects of this on the entropy, S, and the average Hamming distance, $\bar{\mathrm{H}}$. There are three cases shown in table 6.1, where $N_t(x)$ denotes the number of strings x in the population at time t. According to Shannon's observation that any averaging operation will monotonically increase the entropy (Shannon 1948), if $N_t(x) > N_{t+1}(y)$, S must increase when an x is replaced by a y.

The entropy monotonically increases as new information is introduced into the population (first two cases), and monotonically decreases as information is removed from the population (last two cases). Thus, crossover and mutation should monotonically increase

Case	Entropy, S	Hamming, $\bar{\mathrm{H}}$
$N_t(x) > N_{t+1}(y)$	increased	unknown
$N_t(x) = N_{t+1}(y)$	unchanged	unknown
$N_t(x) < N_{t+1}(y)$	decreased	unknown

Table 6.1: Properties of Entropy, S, and Average Hamming Distance, $\bar{\mathrm{H}}$.

S in most cases, and selection should monotonically decrease it. An increase in entropy corresponds to exploration of the search space by the genetic algorithm; a decrease to convergence of the algorithm (in the sense of loss of diversity, as opposed to finding a global optimum). Even when no distinct string has been added or removed, changes in S are predictable. By contrast, $\bar{\mathrm{H}}$ is unpredictable in all cases and furthermore tells us nothing about the homogeneity of the population.

6.2.2 Span

The degree of clustering gives no information about the region of the search space occupied by the population. If a coordinate system could be applied to the search space, the volume of the search space (or a subspace) enclosed by the population could be used for this purpose. However, in labelling problems, there is no ordering of the labels, so no coordinate system can be imposed. The only useful measure of the distance between two points in the search space is the Hamming distance.

The extent to which the population spans the search space can be measured by the total inter-cluster Hamming distance, H_T, which is defined by rewriting $\mathbf{\Psi}$ as $\bigcup \psi_i$, where ψ_i is the i^{th} cluster in $\mathbf{\Psi}$. H_T is given by:

$$\mathrm{H}_T = \frac{1}{2} \sum_{\psi_i \in \mathbf{\Psi}} \sum_{\psi_j \in \mathbf{\Psi}} \mathrm{H}(\psi_i, \psi_j) \tag{6.2}$$

The factor of $\frac{1}{2}$ is needed because every pair of clusters is compared twice in the above sum. H_T compares favourably with the average Hamming distance, because a population with a low average Hamming distance may be composed of a few large, different clusters, in which case it will have a low total inter-cluster Hamming distance. Alternatively, it may consist of many small, similar clusters, in which the total inter-cluster Hamming distance will be

high. In general, crossover and mutation should monotonically increase H_T, since they are unlikely to produce genotypes already present in the population. Selection cannot produce new genotypes, and may remove a cluster entirely from the population, so its effect should be to reduce H_T monotonically.

6.2.3 Gene Pool

The genetic algorithm's population is only one of very many different possible combinations of the genetic material contained in its constituent individuals. An individual chromosome is a string of alleles, each allele being a smallest indivisible part of a solution to the problem in hand. For example, in labelling problems, the chromosome is a string of labels. For simplicity, assume that all chromosomes are of equal length, and that the number of possible alleles at each locus on a chromosome is constant. Denoting a chromosome by χ and the set of alleles (labels) by $\mathbf{\Lambda}$, the number of possible chromosomes is $|\mathbf{\Lambda}|^{|\chi|}$, which should be the same as the size of the search space. The mutation operator draws from this search space to create new individuals. However, neither crossover nor selection has access to the search space directly. These operate on a subspace defined by the current population. This "gene pool" is the set of alleles available at each locus across the whole population, i.e. $\mathbf{G} =< \mathbf{\Lambda}'_1, \mathbf{\Lambda}'_2, ..., \mathbf{\Lambda}'_{|\chi|} >$, where $\mathbf{\Lambda}'_i$ is the set of alleles available for locus i, $\mathbf{\Lambda}'_i = \{\lambda \in \mathbf{\Lambda} | \exists_{\chi \in \mathbf{\Psi}} \chi(i) = \lambda\}$. In (Mühlenbein 1994), Mühlenbein generated offspring by sampling the gene pool directly; however, the interest here is in the size of the gene pool as an indicator of population diversity rather than in the gene pool itself as a source of genetic material. Clearly, the cardinality of $\mathbf{G}$ is just the length of the chromosomes, $|\mathbf{G}| = |\chi|$. It is more useful to define the "size" of the gene pool, g, as

$$g = \sum_{0<i\leq|\chi|} |\mathbf{\Lambda}'_i| \tag{6.3}$$

This quantity varies with the number of alleles in the population. In the degenerate case, when the population is saturated with a single individual, there is only one allele per locus, so $g = |\chi|$. The size of the gene pool is maximal when either all alleles appear in the population, or every individual differs at every locus, depending on which of $|\mathbf{\Psi}|$ and $|\mathbf{\Lambda}|$ is the larger. Thus,

$$|\chi| \leq g \leq |\chi| \min(|\mathbf{\Psi}|, |\mathbf{\Lambda}|) \tag{6.4}$$

The size of the gene pool, g, is the number of alleles in the population which would be available to the crossover operator. The number of different individuals that could be produced by crossover is the "potential", ρ, of the gene pool,

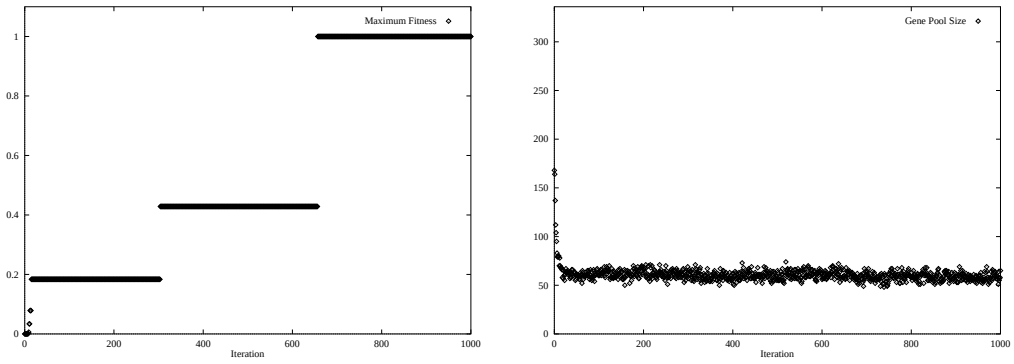
$$\rho = \prod_{0<i\leq|\chi|} |\mathbf{\Lambda}'_i| \tag{6.5}$$

which increases monotonically with g since both $|\chi|$ and $|\mathbf{\Psi}|$ are greater than 1. In the degenerate case, $\rho = 1$; its maximum is $[\min(|\mathbf{\Psi}|, |\mathbf{\Lambda}|)]^{|\chi|}$. Crossover never changes $\mathbf{G}$, g or ρ, since it only redistributes alleles already in the population. Mutation may change the gene pool, either by introducing a new allele, which would increase g and ρ, or by deleting the last copy of particular allele, which would decrease g and ρ. The first case is more likely, so on the whole, one would expect mutation to increase the size of the gene pool. Selection is incapable of adding alleles to the population, but it can remove them, so it will monotonically reduce g and ρ.

6.2.4 Evolution of Diversity

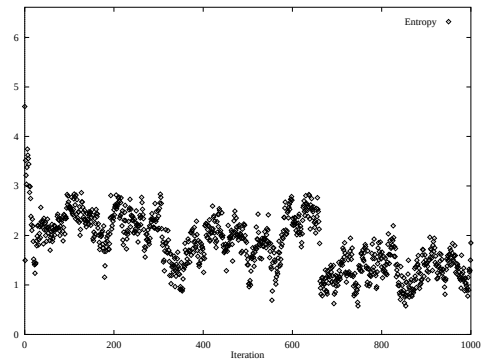
Figures 6.1 and 6.2 show the evolution of the three diversity measures in two typical runs of a plain genetic algorithm run on a 42-line Huffman labelling problem. The first run, figure 6.1, located an optimal solution at iteration 657; the second run failed to find a global optimum. In both cases, the Shannon entropy started at its maximum value, $\ln 100 = 4.6$. As the population became saturated, the entropy fell to some minimum below about 2, but the variations in entropy and total inter-cluster Hamming distance after saturation indicate that the algorithm was still attempting to explore the search space. The presence of a set of relatively fit individuals reduces the likelihood that new chromosomes will persist, and this may explain why the final entropy for successful runs was often slightly lower than for unsuccessful runs.

Some, but not all, of the major peaks in entropy coincided with the jumps in the maximum fitness, which occurred when the algorithm located a new optimum; those which did not presumably represent unsuccessful forays in the search space. The peaks which coincide

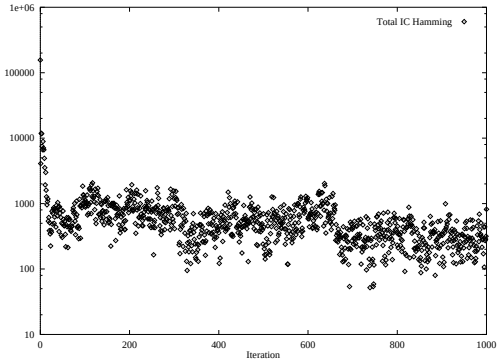


(a) Maximum Fitness

(b) Gene Pool Size



(c) Entropy

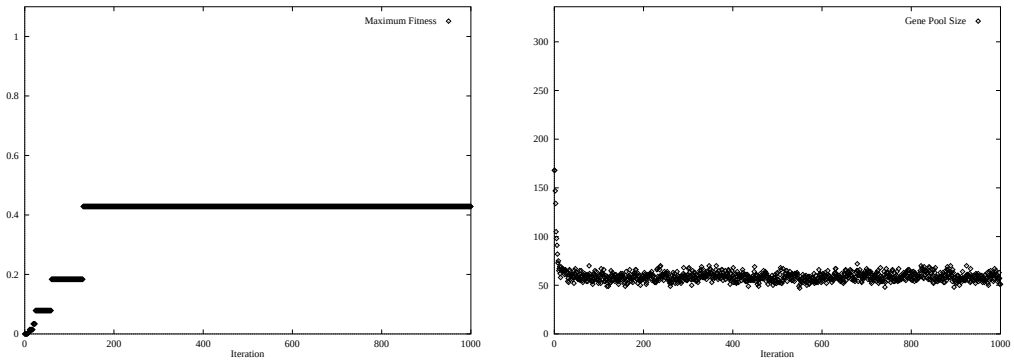


(d) Inter-Cluster Hamming Distance

Figure 6.1: Successful Run of Plain Algorithm for Huffman Labelling. Population size was 100.

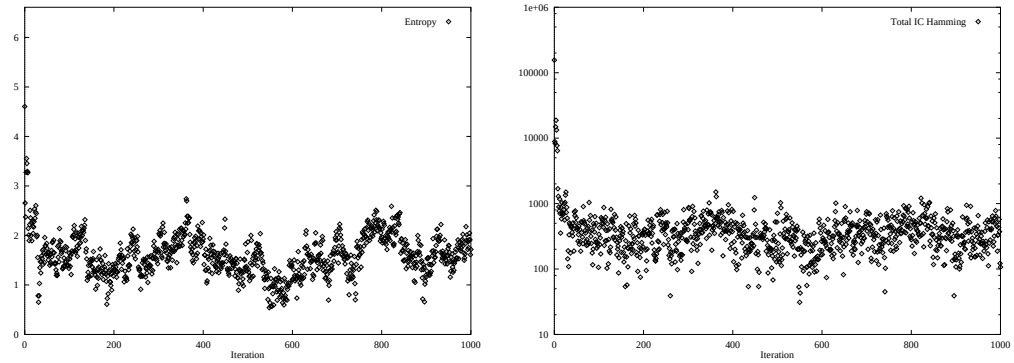
with jumps in maximum fitness may either precede or follow them. This can be explained by proposing two mechanisms by which new optimal solutions can arise. The algorithm may explore some fruitful avenue in the search space, causing an increase in entropy, then an optimal solution may be found following a crossover or mutation. Thus an entropy peak can precede a fitness jump. Alternatively, a new solution may arise spontaneously, without extensive search. There will be a fitness jump with no entropy peak. However, if the copy number of the new solution increases over the next few generations, the entropy peak will succeed the fitness jump. A peak occurs because the initial copy number is 1. Replacing a string from a large cluster with one from a smaller one will increase the entropy, but at some point, the cluster containing the new string becomes sufficiently large that adding to it reduces the entropy, hence the peak.

Figures 6.1 and 6.2 only show a single algorithm run, but the evolution of the diversity measures is remarkably consistent. Figures 6.3 and 6.4 show the change in diversity over



(a) Maximum Fitness

(b) Gene Pool Size

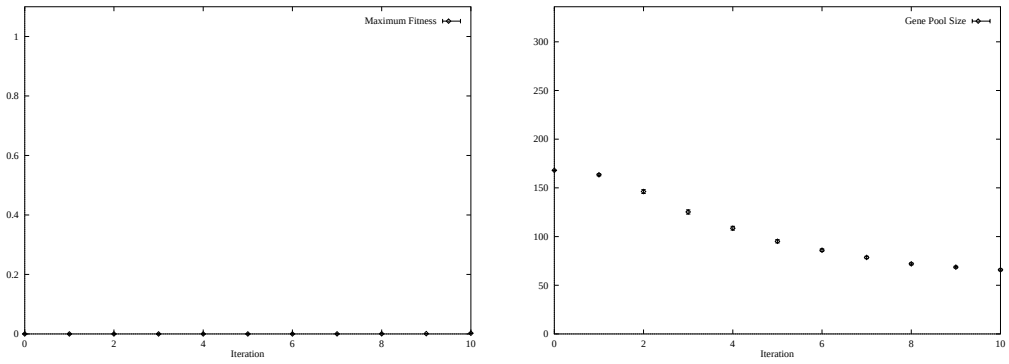


(c) Entropy

(d) Inter-Cluster Hamming Distance

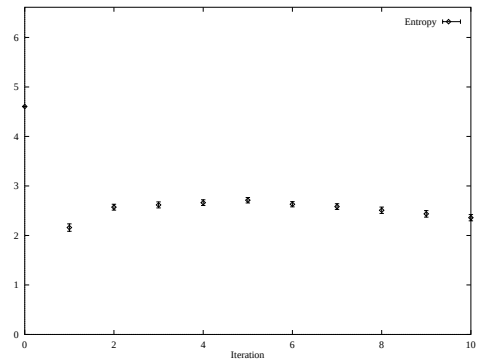
Figure 6.2: Unsuccessful Run of Plain Algorithm for Huffman Labelling. Population size was 100.

the course of the first 10 iterations of 100 runs of plain and hybrid genetic algorithms for the same Huffman labelling problem. Both algorithms showed large initial drops in all three diversity measures, followed by small recoveries in entropy and inter-cluster Hamming distance. Most of the diversity in the population was lost in the first few iterations. The total inter-cluster Hamming distance fell from around 100,000 to around 1,000, the size of the gene pool fell from 170 to between 60 and 80, and the entropy lost over a third of its initial value. Since the test problem used to generate these figures had 42 lines, a gene pool size less than 84 means that some loci must be degenerate. The hybrid algorithm appears to have sustained greater diversity than the plain one. This may be because the hybrid located several optima. Indeed, by the time that the inevitable decrease in diversity was under way, the hybrid algorithm had already found optimal solutions. The hybrid algorithm did more searching in 10 iterations than the plain algorithm. Thus, in addition to being more efficient, the hybrid conducts its most fruitful search when the population is still relatively diverse.

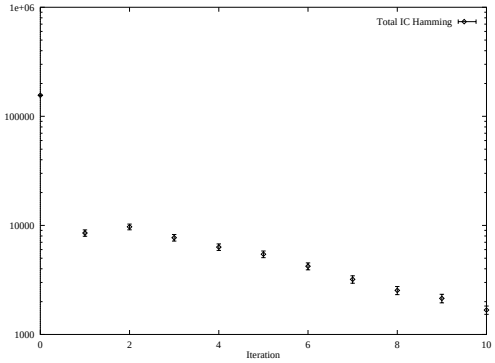


(a) Maximum Fitness

(b) Gene Pool Size



(c) Entropy

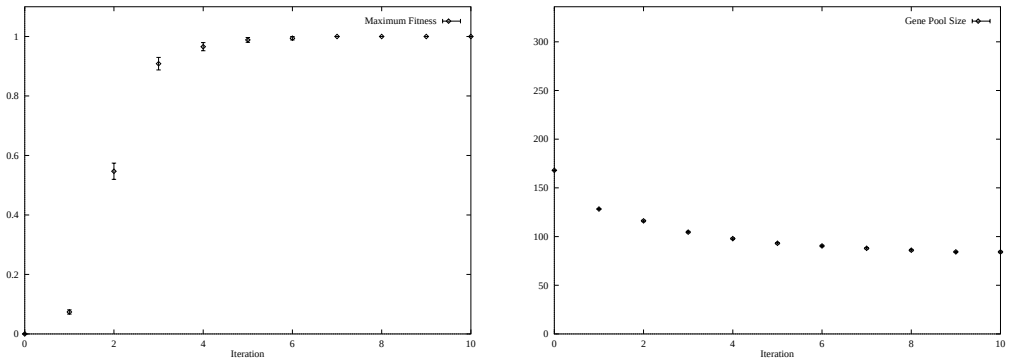


(d) Inter-Cluster Hamming Distance

Figure 6.3: 100 Runs of Plain Algorithm for Huffman Labelling. Population size was 100, there were 10 iterations.

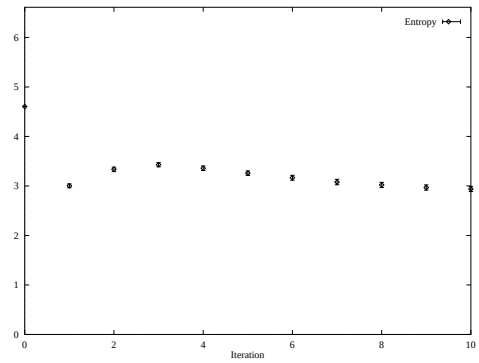
Figures 6.5 and 6.6 show the change in diversity over the course of 10 iterations of hybrid genetic and CHC algorithms for a 30-node graph matching problem, averaged over 100 program runs. Both algorithms located optimal matches within three iterations on average. However, the loss of diversity in the hybrid CHC algorithm was catastrophic compared to the loss of diversity with the hybrid genetic algorithm. This confirms the observation from chapter 4 that the hybrid genetic algorithm gives a higher solution yield than CHC, although CHC is the better optimiser.[1] The main reason for the lack of diversity with CHC would appear to be that CHC does not use mutation at every step, relying on restarts whenever the population members become sufficiently similar. However, the hybrid CHC

[1] In fact, the CHC algorithm is capable of matching 30-node graphs even without a gradient ascent step. However, this is less efficient than the hybrid version, and not as reliable, requiring a population of over 500 and about 150 iterations. Nevertheless, this is a vast improvement on the very poor performance of the plain genetic algorithm reported by Cross in (Cross 1998).

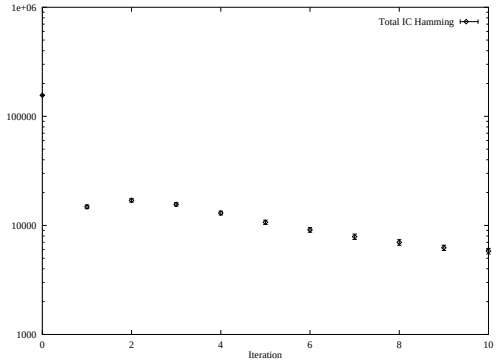


(a) Maximum Fitness

(b) Gene Pool Size



(c) Entropy

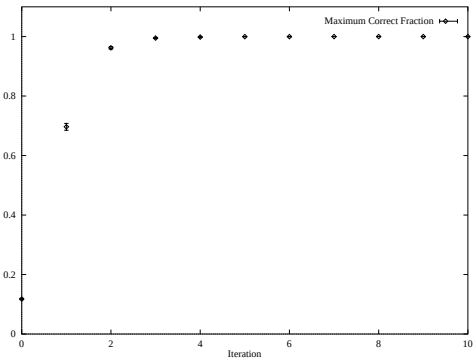


(d) Inter-Cluster Hamming Distance

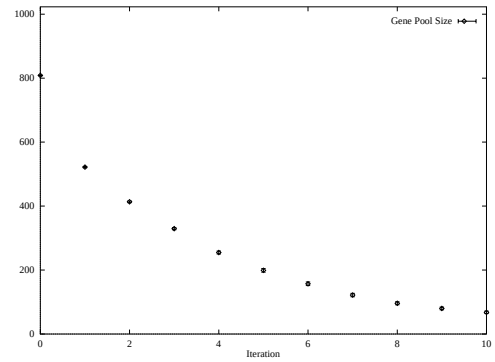
Figure 6.4: 100 Runs of Hybrid Algorithm for Huffman Labelling. Population size was 100, there were 10 iterations.

algorithm never loses enough diversity to force a restart, with the result that hybrid CHC effectively does not use mutation at all.

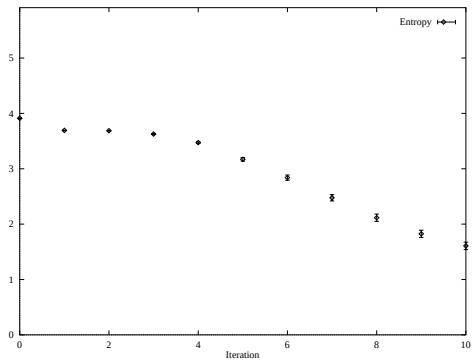
The common feature of all these experiments is a substantial, irreversible loss of diversity over the first few algorithm iterations. This is especially marked in the line labelling experiments. The reduction in diversity is almost certainly the result of genetic drift. However, the initial decreases are much more pronounced for line labelling. A natural explanation of this might be that genetic drift is greater for line labelling, but the initial fitness distributions are very similar in both cases. So the selection operator alone cannot be responsible for the decrease in diversity, since it should affect each problem equally. The next section considers the mutation operator and its effect on diversity.



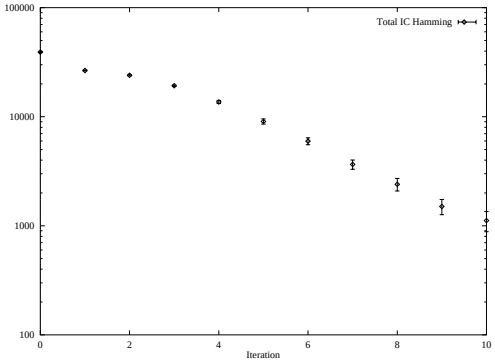
(a) Maximum Fraction Correct



(b) Gene Pool Size

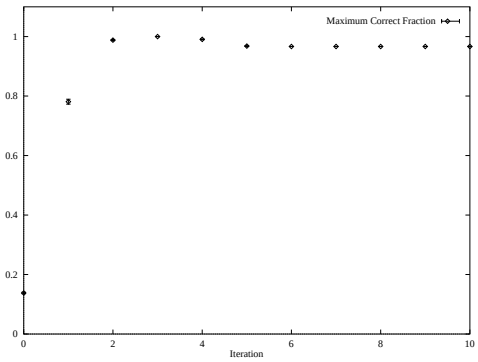


(c) Entropy

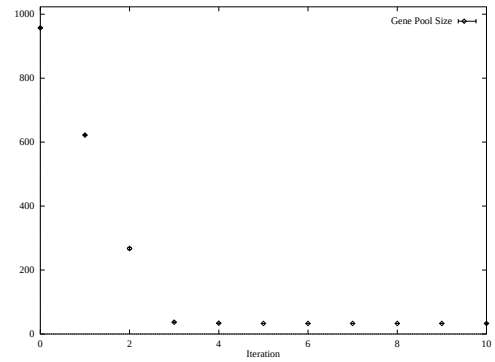


(d) Inter-Cluster Hamming Distance

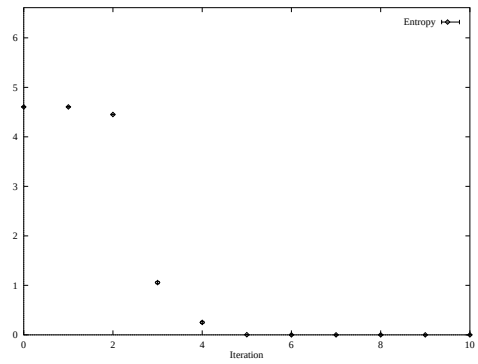
Figure 6.5: 100 Runs of Hybrid Genetic Algorithm for Graph Matching. Population size was 50, there were 10 iterations.



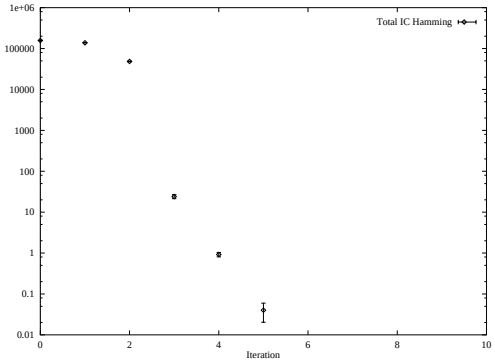
(a) Maximum Fraction Correct



(b) Gene Pool Size



(c) Entropy



(d) Inter-Cluster Hamming Distance

Figure 6.6: 100 Runs of Hybrid CHC Algorithm for Graph Matching. Population size was 100, there were 10 iterations.

6.3 Mutation

Consider the effect of mutation on the size of the gene pool. In section 6.2 it was argued informally that mutation would generally act to increase the Shannon entropy, total inter-cluster Hamming distance and gene pool size. As for the entropy, we begin by considering what happens to the size of the gene pool when some allele, x, is changed to a y. This depends on the numbers of x and y at time t, $N_t(x)$ and $N_t(y)$. There are six possibilities, shown in table 6.2, together with the corresponding changes in the gene pool size, Δg.

Before mutation		After mutation		
$N_t(x)$	$N_t(y)$	$N_{t+1}(x)$	$N_{t+1}(y)$	Δg
1	0	0	1	0
> 1	0	≥ 1	1	$+1$
1	1	0	2	-1
> 1	1	≥ 1	2	0
1	> 1	0	> 2	-1
> 1	> 1	≥ 1	> 2	0

Table 6.2: Effect of a Single Mutation of Gene Pool Size.

In terms of the change in gene pool size, Δg, cases 1, 4 and 6 are "don't cares", 3 and 5 are reductions, and only in case 2 does the gene pool size increase. The probabilities of these cases are governed by the distribution of alleles in the population, $\mathbf{\Psi}$. Without loss of generality, consider the alleles at a particular locus drawn from the label set, $\mathbf{\Lambda}$. The bag of alleles at this locus over every individual in the population constitutes a sample of size $|\mathbf{\Psi}|$ with replacement from $\mathbf{\Lambda}$. The number, $N(\lambda)$, of a single allele λ is binomially distributed:

$$P(N(\lambda) = n) = \left(\begin{array}{c} |\mathbf{\Psi}| \\ n \end{array} \right) p_\lambda^n (1 - p_\lambda)^{|\mathbf{\Psi}| - n} \tag{6.6}$$

where p_λ is the probability of finding the label λ at this locus. It is now possible to assign probabilities to the outcomes for Δg in table 6.2, according to the numbers of x and y at time t.

$$\Delta g = \begin{cases} 1 & \text{with probability } P(N_t(x) > 1, N_t(y) = 0) \\ -1 & \text{with probability } P(N_t(x) = 1, N_t(y) > 0) \\ 0 & \text{otherwise} \end{cases} \tag{6.7}$$

It can be seen that Δg will be $+1$ only when the proportion of x is greater than $\frac{1}{|\mathbf{\Psi}|}$ and the proportion of y is zero. Similarly, Δg will be -1 when the proportion of x is exactly $\frac{1}{|\mathbf{\Psi}|}$ and the proportion of y is non-zero. This gives rise to two extreme cases:

1. **Saturated population.** In this case, which corresponds to convergence of the algorithm, the population is saturated with a single allele, x, such that the $N(i) = |\mathbf{\Psi}|$ and $N(y) = 0$. In this case, Δg is inevitably $+1$.

2. **Random population.** This is the case in the initial iteration of the algorithm. Assuming that all alleles are equiprobable, i.e. for each label, λ, $p_\lambda = \frac{1}{|\mathbf{\Lambda}|}$. In this case, the expectation of the change in gene pool size following a single mutation is given by

$$\begin{aligned} E(\Delta g) &= P(N_t(x) > 1, N_t(y) = 0) - P(N_t(x) = 1, N_t(y) > 0) \\ &= \left(1 - \frac{1}{|\mathbf{\Lambda}|}\right)^{|\mathbf{\Psi}|} - \left(1 - \frac{1}{|\mathbf{\Lambda}|}\right)^{2|\mathbf{\Psi}|} - \frac{|\mathbf{\Psi}|}{|\mathbf{\Lambda}|}\left(1 - \frac{1}{|\mathbf{\Lambda}|}\right)^{|\mathbf{\Psi}|-1} \end{aligned} \tag{6.8}$$

The second case, of a random population, is shown in figure 6.7 for population sizes between 2 and 100, and between 2 and 100 alleles. This figure shows that the expectation of Δg is generally negative for a random population, only approaching zero when one of $|\mathbf{\Psi}|$ or $|\mathbf{\Lambda}|$ is small. The minimum value is about -0.3, when both are small, and there is a valley at about -0.15, where $|\mathbf{\Psi}| \approx 2|\mathbf{\Lambda}|$.

The surprising conclusion, then, is that mutation actually *reduces* the diversity in the initial population, at least in terms of the size of the gene pool. It appears that the efficiency of mutation as a "diversity inducer" increases as the population becomes saturated. This suggests that the mutation rate should be low initially, and then increased as the algorithm proceeds, quite the opposite of the annealing strategies suggested in (Cross et al. 1997; Davis and Principe 1991) where the mutation rate is steadily decreased.

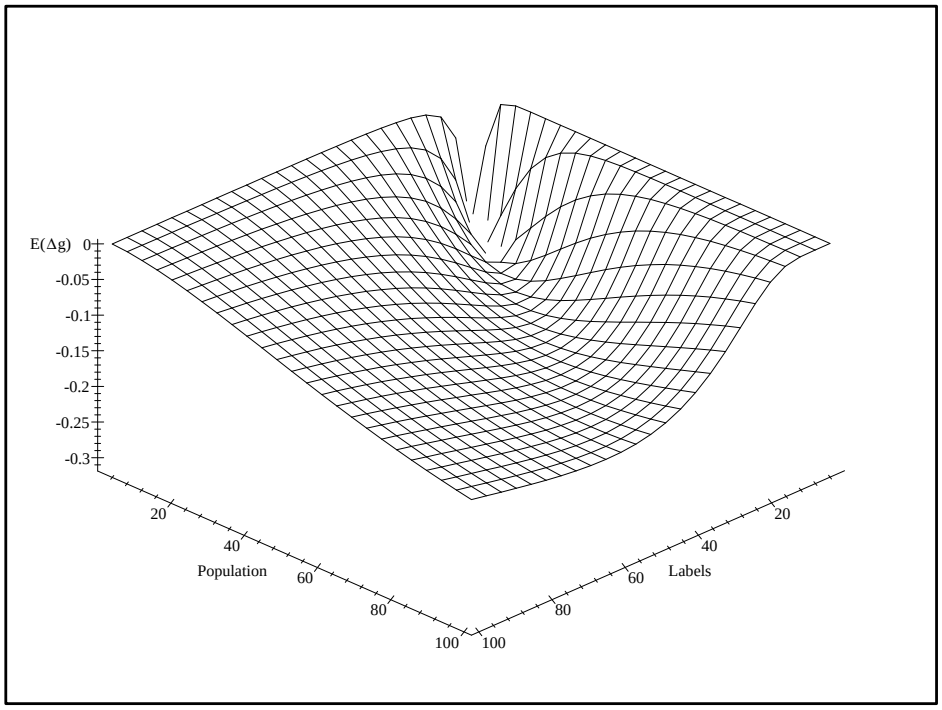


Figure 6.7: Expectation of Δg versus population size and size of label set.

6.3.1 Dynamic Mutation Rate

The idea of varying the mutation rate at each iteration of the genetic algorithm has appeared several times over the years. Previous studies have all tended to reduce the mutation rate as the algorithm proceeds (Fogarty 1989; Davis 1989; Cross et al. 1997; Davis and Principe 1991). Only Whitley in (Whitley and Starkweather 1990) and Srinivas and Patnaik in (Srinivas and Patnaik 1994a) suggest that the mutation rate should actually increase as the algorithm converges. Srinivas and Patnaik scaled the mutation rate for individuals based on their fitness and the current state of convergence, which was estimated by the difference between the maximum and average fitness in the population. To avoid destroying solutions, the mutation rate was lower for highly fit individuals. Srinivas and Patnaik's formula for mutation probability was

$$P_m = \begin{cases} k_1(f_{\text{MAX}} - f)/(f_{\text{MAX}} - \bar{f}) & \text{if } f \geq \bar{f} \\ k_2 & \text{otherwise} \end{cases} \tag{6.9}$$

where f_{MAX} and $\bar{f}$ are the maximum and average fitness in the population, and f is

the fitness of the individual under consideration. There are three disadvantages to this approach. The first is that it requires the addition of two parameters, k_1 and k_2. Srinivas and Patnaik set both of these to 0.5 in order to "completely disrupt" solutions with average or below average fitness. It is not clear how a mutation rate of 0.5 "completely" disrupts an individual. The second disadvantage is that when the population is saturated, the mutation rate is actually undefined according to this criterion. The third problem is that fit individuals will remain immune from the effects of mutation, which would defeat the purpose of finding distinct optimal solutions.

If mutation is intended to maintain diversity, the mutation rate should vary with one of the diversity measures in section 6.2. For example, the Shannon entropy could be used to define the mutation rate:

$$P_m = 1 - \frac{S}{\ln |\mathbf{\Psi}|} \tag{6.10}$$

where P_m is the mutation rate, and S is the Shannon entropy of the population, $\mathbf{\Psi}$. Alternatively, the gene pool size could be used instead of the entropy. The difficulty with defining the mutation rate in this way is that its maximum value depends on the minimum value of the diversity measure, which is not predictable, and may therefore force the mutation rate so high that the search is compromised. For example, from panel (b) of figure 6.5, one might expect that tying mutation rate to the gene pool size in a manner similar to equation 6.10 could result in assigning an inappropriately large value to P_m. At iteration 10, the gene pool size is about 80, but the maximum is 800, so P_m would be $1 - \frac{80}{800} = 0.9$. Furthermore, the mutation rate is now explicitly tied to only one out of the three diversity measures.

A more general control strategy for the mutation rate might be to vary it with time according to a pre-defined schedule. Since the number of iterations, $t_{\max}$, is fixed in advance, it is possible to define the following simple schedule for the mutation rate:

$$P_m^t = P_m^f \left(\frac{t}{t_{\max}} \right)^s \tag{6.11}$$

for $s > 0$, where t is the iteration number. The "base mutation rate", P_m^f, sets a ceiling above which P_m cannot rise. The schedule parameter, s, controls the profile of the mu-

tation rate throughout the program run. If $s < 1$, the mutation rate rises quickly and stays high over the course of the algorithm. If $s > 1$, the mutation rate stays low before rising to P_m^f at the end of the run. Naturally, there are many other ways of increasing the mutation rate as the algorithm proceeds, but this one has the advantage that it controls the mutation rate in a natural way. It guarantees that the mutation rate is non-zero, and that it smoothly increases to a known maximum. The base mutation rate should be set to the maximum sustainable mutation rate above which the algorithm ceases to perform well as an optimiser. This leaves only the shape of the schedule, s. It is almost certain that the most important property of the schedule will be whether it is convex ($s < 1$), linear ($s = 1$) or concave ($s > 1$). High curvature in either sense corresponds to more or less constant mutation rate, so in practice, finding a suitable value of s should not be difficult.

Experiments

Two sets of experiments were conducted to evaluate the performance of the different mutation rate schedules. These experiments were restricted to graph matching. Line labelling was not considered. In each case, a hybrid genetic algorithm was used, with a population size of 20, and balanced contiguous crossover at a rate of 1.0, and run to 10 iterations. The selection method was fitness-proportionate with elitism, and the fitness criterion was the linear one, equation 3.13 from chapter 3. In each set four different graphs were used, and 50 replications performed for each graph. The first set of experiments used 20, 30, 40 and 50 node graphs; the second set used four 30 node graphs. The schedules for mutation rate are shown in table 6.3. The first condition was a control in which the mutation rate is fixed. The second condition used equation 6.10 to set the mutation rate from the entropy. The other six conditions all used equation 6.11: "reasonable" (0.6 or 0.4) and "unreasonable" (0.9) base mutation rates were used, each with convex ($s = 0.5$), linear ($s = 1.0$), or concave ($s = 2.0$) schedules. The resulting data matrix for each experiment had 32 cells, and was drawn from 1600 program runs.

The results for the two experiments are summarised in figures 6.8 and 6.9. Both graph and condition scales are categorical, i.e. there is no order along the x and y axes. The z-axis is the solution yield, which in this case is binomially distributed on the population size (20). From these figures, it can be seen that too high a base mutation rate (conditions 6, 7 and 8) almost always leads to bad performance. However, it is not at all clear whether basing the mutation rate on entropy or iteration is any better than using a fixed

Condition	Experiment 1		Experiment 2	
	P_m^f	s	P_m^f	s
1	P_m fixed at 0.4			
2	P_m depends on entropy			
3	0.6	0.5	0.4	0.5
4	0.6	1.0	0.4	1.0
5	0.6	2.0	0.4	2.0
6	0.9	0.5	0.9	0.5
7	0.9	1.0	0.9	1.0
8	0.9	2.0	0.9	2.0

Table 6.3: Conditions for Mutation Rate Experiments.

mutation rate. A logistic regression analysis of the data from the first experiment showed that the entropy based schedule (condition 2), was better than the "reasonable" iteration based schedules (conditions 3, 4 and 5). However, this conclusion could only be reached by assuming that the very high yield obtained with the 50 node graph and fixed mutation rate was anomalous. Unfortunately, there is no reason to believe this assumption. A similar analysis of the second experiment showed that varying the mutation rate had no significant effect, as long as P_m^f had a "reasonable" value. Thus it would seem that mutation plays a relatively minor part in the initial massive loss of diversity for these problems. This finding is not necessarily at variance with the theoretical account of the effect of mutation on gene pool size presented in the previous section. Figure 6.7 shows that for a population size of 20, mutation would cause a relatively small initial loss of diversity for 30 to 50 node graphs. It would appear, however, that selection has a greater bearing on population diversity than mutation. Since diversity is lost both with and without the gradient ascent step, the selection operator must be responsible for most of the initial diversity loss, and it to is this which we now turn our attention.

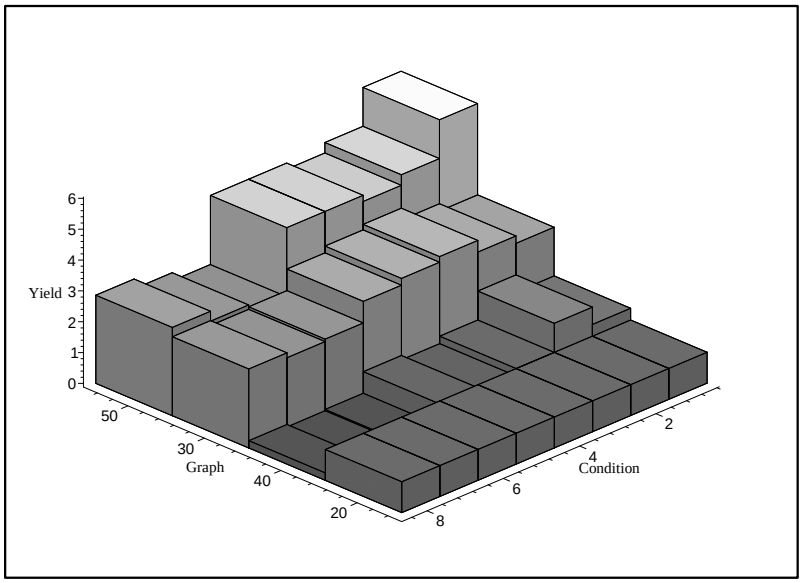


Figure 6.8: Average Yields with Dynamic Mutation Rates I. Note that the graphs have been arranged to make the plots easier to read.

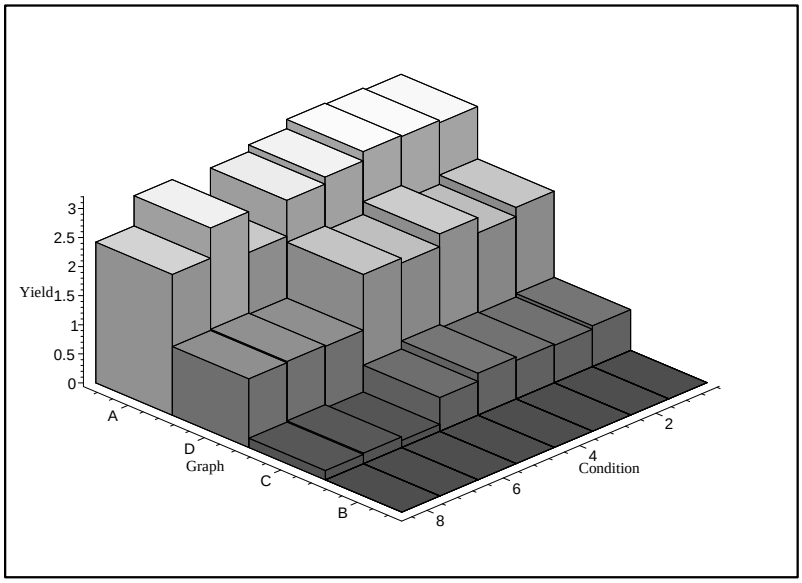


Figure 6.9: Average Yields with Dynamic Mutation Rates II. Note that the graphs have been arranged to make the plots easier to read.

6.4 Selection

This section considers the selection operator itself. After a brief analysis of some standard selection methods, alternatives for the hybrid genetic algorithm are discussed. The idea of selection without replacement is introduced, and some strategies for elitist selection are considered. The section ends with an experimental study.

6.4.1 Standard Methods

In a standard genetic algorithm, selection is crucial to the algorithm's search performance. Whereas mutation, crossover and local search are all "next-state" operators, selection imposes a stochastic acceptance criterion. The standard "roulette" selection algorithm, described by Goldberg in (Goldberg 1989), assigns each individual a probability of selection, p_i, proportional to its fitness, f_i. Recall from section 4.3 of chapter 4 that the genetic algorithm used here allows the population to grow transiently and then selects a population of size $|\mathbf{\Psi}|$ to form the new generation. Denoting the expanded population by $\mathbf{\Psi}_e$, the selection probability is given by

$$p_i = \frac{f_i}{\sum_{j \in \mathbf{\Psi}_e} f_j} \tag{6.12}$$

The algorithm then holds selection trials for each "slot" in the new population, for a total of $|\mathbf{\Psi}|$ trials. Since selection is with replacement, the constitution of the new population is governed by the multinomial distribution, and the copy number of a particular individual, $N(i)$, is distributed binomially:

$$P(N(i) = r) = \left(\begin{array}{c} |\mathbf{\Psi}| \\ r \end{array} \right) p_i^r (1 - p_i)^{|\mathbf{\Psi}| - r} \tag{6.13}$$

and so the expectation of $N(i)$, is $\mathrm{E}[N(i)] = |\mathbf{\Psi}| p_i$, and its variance is $\mathrm{Var}[N(i)] = |\mathbf{\Psi}| p_i (1 - p_i)$.

The search power of the standard genetic algorithm arises from the fact that if the individual in question is highly fit, p_i will be much larger than the average, and hence the expectation will be that the copy number will increase. This approach has two disadvan-

tages. The first is that for small populations, sampling errors may lead to copy numbers very much higher or lower than the expected values, which is the phenomenon of premature convergence. In (Baker 1987), Baker proposed "stochastic remainder sampling", which guarantees that the copy number will not be much different from the expectation by stipulating that $\lfloor \mathrm{E}[N(i)] \rfloor \leq N(i) \leq \lceil \mathrm{E}[N(i)] \rceil$. However, the larger the population, the less need there is for Baker's algorithm (Mitchell 1996). The second disadvantage is that less fit individuals have lower expectations, and that the lower the fitness, the lower the variance of the copy number. In other words, less fit individuals are increasingly likely to have lower copy numbers. When $E[N(i)]$ falls below 1, the individual will probably disappear from the population.

In general, the copy number variance decreases with decreasing fitness. Only when $p_i > 0.5$ does the variance decrease with increasing fitness. This occurs when the fitness of one individual accounts for at least half the total fitness of the population, i.e. when it is at least $|\mathbf{\Psi}_e| - 1$ times as fit as any other individual. This occurred very rarely in the genetic algorithms tested in section 6.2. However, for line labelling, some individuals in the initial population did meet this criterion. This may account for the larger initial drops in diversity for line labelling compared with graph matching.

In summary, the problem with roulette selection is that it imposes too strict an acceptance criterion on individuals with below average fitness. Several alternative strategies have been proposed to avoid this problem. "Sigma truncation", rank selection and tournament selection (Goldberg 1989; Baker 1985) all seek to maintain constant selection pressure by requiring individuals not to compete on the basis of their fitness, but on some indirect figure of merit such as the rank of their fitness, or the distance between their fitness and the average in standard units. Taking rank selection as a typical example of these strategies, the selection probabilities are assigned based on rank, with the best individual having the highest rank:

$$p_i = \frac{\mathrm{rank}_i}{\sum_{j \in \mathbf{\Psi}_e} \mathrm{rank}_j} \tag{6.14}$$

Assume without loss of generality that ties in ranks are resolved by random assignment of ranks. With this assumption, the denominator of equation 6.14 is simply the sum of the first $|\mathbf{\Psi}_e|$ integers, so the equation becomes

$$p_i = \frac{\text{rank}_i}{0.5|\mathbf{\Psi}_e|(|\mathbf{\Psi}_e| - 1)} \tag{6.15}$$

The fittest individual has rank $|\mathbf{\Psi}|_e$ and the least fit has rank 1. The expected copy numbers of the best and worst individuals are given by:

$$\left.\begin{array}{rcl} \mathrm{E}[N(\text{best})] & = & \frac{2|\mathbf{\Psi}|}{(|\mathbf{\Psi}_e|-1)} \\ \mathrm{E}[N(\text{worst})] & = & \frac{2|\mathbf{\Psi}|}{|\mathbf{\Psi}_e|(|\mathbf{\Psi}_e|-1)} \end{array}\right\} \tag{6.16}$$

So, the expected copy number of the fittest individual differs from that of the least fit by a factor of $|\mathbf{\Psi}_e|$. Moreover, if $|\mathbf{\Psi}_e|$ is even moderately large, $\mathrm{E}[N(\text{worst})]$ will be much less than 1. Indeed, $\mathrm{E}[N(i)]$ will be less than 1 for about half the population. Thus, under rank selection, less fit individuals are highly likely to disappear, even if they are quite good.

A second alternative to roulette selection is Boltzmann selection (Goldberg 1990; Prügel-Bennett and Shapiro 1994), in which an additional parameter is introduced. The selection probability is assigned as follows:

$$p_i = \frac{\exp\left[\beta f_i\right]}{\sum_{j \in \mathbf{\Psi}_e} \exp\left[\beta f_i\right]} \tag{6.17}$$

where β is the inverse temperature. This strategy borrows the idea from simulated annealing that at thermal equilibrium the probability of a system being in a particular state depends on the temperature, and the system's energy. The idea is that as β is raised, high energy (low fitness) states are less likely. The difficulty with this analogy is that it requires the system to have reached thermal equilibrium. In simulated annealing, this is achieved after very many updates at a particular temperature (see chapter 10, section 9 of (Press et al. 1992) for details and an implementation). In a genetic algorithm this would require many iterations at each temperature level to achieve equilibrium, coupled with a slow increase of β. Within the 10 or so iterations allowed for hybrid genetic algorithms, equilibrium cannot even be attained, let alone annealing occur. In this case, β should be interpreted as a scaling parameter which adjusts the selection pressure imposed by the algorithm, rather than as inverse temperature. Unfortunately, the experimental studies in chapter 5 indicated that there was little to be gained from annealing the selection pressure.

6.4.2 Alternative Methods

It would appear, then, that there is a tradeoff between premature convergence and the strength of the selection operator. The problem arises from the fact that expected copy numbers of fit individuals may be greater than one, while those of unfit individuals may be less than one. One way of preventing the increase in copy number of highly fit individuals is to use "truncation selection", as used in Rechenberg and Schwefel's evolution strategies (Rechenberg 1973; Schwefel 1981). Truncation selection would simply take the best $|\mathbf{\Psi}|$ individuals from the expanded population, $\mathbf{\Psi}_e$, to form the new population. The copy number of each individual is simply

$$N(i) = \begin{cases} 1 & \text{if } \text{rank}_i \geq |\mathbf{\Psi}_e| - |\mathbf{\Psi}| \\ 0 & \text{otherwise} \end{cases} \tag{6.18}$$

Although no individual may increase its copy number, the selection pressure might still be quite severe, since for the algorithm used in this thesis, $|\mathbf{\Psi}_e|$ can be as large as $3|\mathbf{\Psi}|$. In other words, less fit individuals still disappear at an early stage. The fact that individuals never increase their copy number makes this a relatively weak search operator, and probably unsuitable for a standard genetic algorithm. However, chapter 5 suggested that the local search step is mostly responsible for the optimisation performance of the algorithm. If this is so, selection is a much less important search operator for hybrid algorithms than for standard genetic algorithms. It may therefore be beneficial to trade search performance for greater diversity.

The benefits of stochastic selection can be combined with the evenness of truncation selection by selecting without replacement. Strictly speaking, selection without replacement should be done by recomputing the p_i after every selection event, because the population, $\mathbf{\Psi}_e$, is changed by selection. However, this extra computation can be avoided by increasing the probability of the next fittest individual. Suppose individual i is selected, and that the next fittest individual is j. The selection probability of j is increased by p_i: $p_j^{\text{new}} = p_j^{\text{old}} + p_i$. If there is no next fittest individual, the least fit individual which is fitter than i has its probability increased. This strategy can be called "biased selection without replacement", since it is biased first in favour of fitter individuals, but may also favour less fit ones.

The alternative is to abandon fitness based selection altogether, and rely on the local search step to do all the optimisation. If the genetic algorithm's rôle is explicitly limited to assembling a good initial guess for the local search operator, the selection probabilities can be assigned uniformly, i.e. $\forall_{i \in \mathbf{\Psi}_e} p_i = \frac{1}{|\mathbf{\Psi}|}$. This operator is called "neutral selection". Neutral selection without replacement can be implemented very efficiently by shuffling $\mathbf{\Psi}_e$ and choosing the "top" $|\mathbf{\Psi}|$ individuals. This strategy shares the advantage with truncation selection, that the minimum number of individuals are excluded from the new population, but also maintains the global stochastic acceptance properties of standard selection operators.

Elitist selection guarantees that at least one copy of the best individual so far found is selected for the new population. This heuristic is very widely used in genetic algorithms. In (Rudolph 1994), Rudolph showed that the algorithm's eventual convergence cannot be guaranteed without it. The elitist heuristic can be modified in two ways to help maintain diversity. First, it seems natural that if the goal is to simultaneously obtain several solutions to the problem in hand, several of the fittest individuals should be guaranteed in this way. This is called "multiple elitism". Second, if one wishes to avoid losing too many unfit individuals, the *worst* individual can also be granted free passage to the new population. This is called "anti-elitism". These heuristics, together with the selection strategies discussed earlier, are evaluated in the next section.

6.4.3 Experiments

As for the dynamic mutation rate strategies, two sets of experiments were conducted, both of which concerned graph matching. Both sets used the hybrid genetic algorithm with the linear fitness criterion. Optimal contiguous crossover was used at a rate of 1.0; the mutation rate was fixed at 0.4. The first set of experiments used 20, 30, 40 and 50 node graphs, and for these the population size was set to 10, and the algorithm run for 5 iterations. The second set of experiments used four 30 node graphs, with a population size of 20 and 10 iterations. Five different selection strategies were compared: they were standard roulette, rank, and truncation selection, and neutral and biased selection without replacement. Five combinations of elitist heuristics were considered: they were no elitism, single elitism, multiple elitism, anti-elitism, and a combination of multiple and anti-elitism. The experimental design was therefore a 5x5x4 factorial with 100 cells, which is summarised in table 6.4. The first set of experiments had 40 replications for a total of

Key	Selection	Elitism
1	Roulette	No elitism
2	Rank	Elitism
3	Truncation	Multiple Elitism
4	Neutral	Anti-elitism
5	Biased	Multiple and Anti-elitism

Table 6.4: Conditions for Selection Experiments. The keys in this table refer to the axis labels in figures 6.10 and 6.11.

4000 observations; and the second set had 50 replications for 5000 observations. Figures 6.10 and 6.11 summarise the results.

Both plots show that neutral selection without replacement produced the best yields, and that truncation selection produced the worst. Biased and roulette selection strategies gave similar results, and were both outperformed by rank selection. Linear logistic regression analysis of both data sets confirmed this ranking of selection strategies.

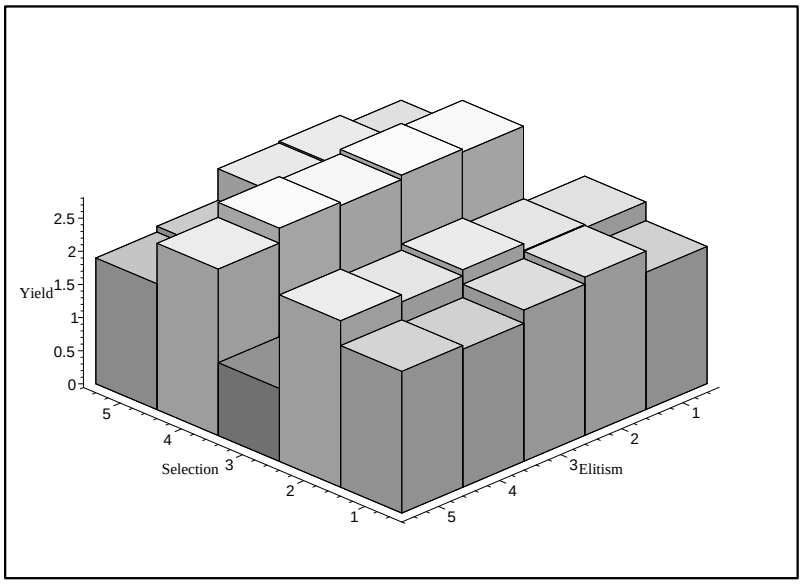


Figure 6.10: Average Yields versus Selection and Elitism I. Data from all four graphs has been pooled.

The results for elitism heuristic were not so convincing. It is questionable whether elitism has any overall effect: the regression analysis of the second data set found no significant effect of varying the elitism strategy. The analysis of the first data set did show that either

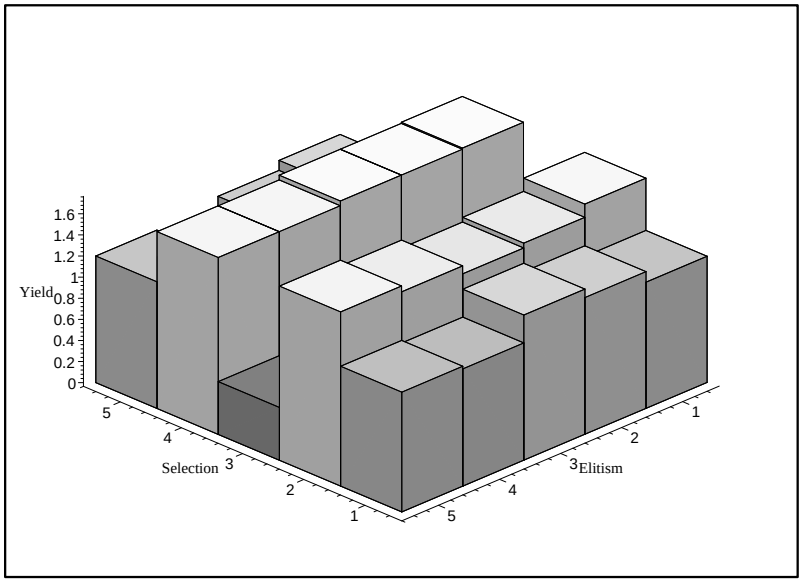


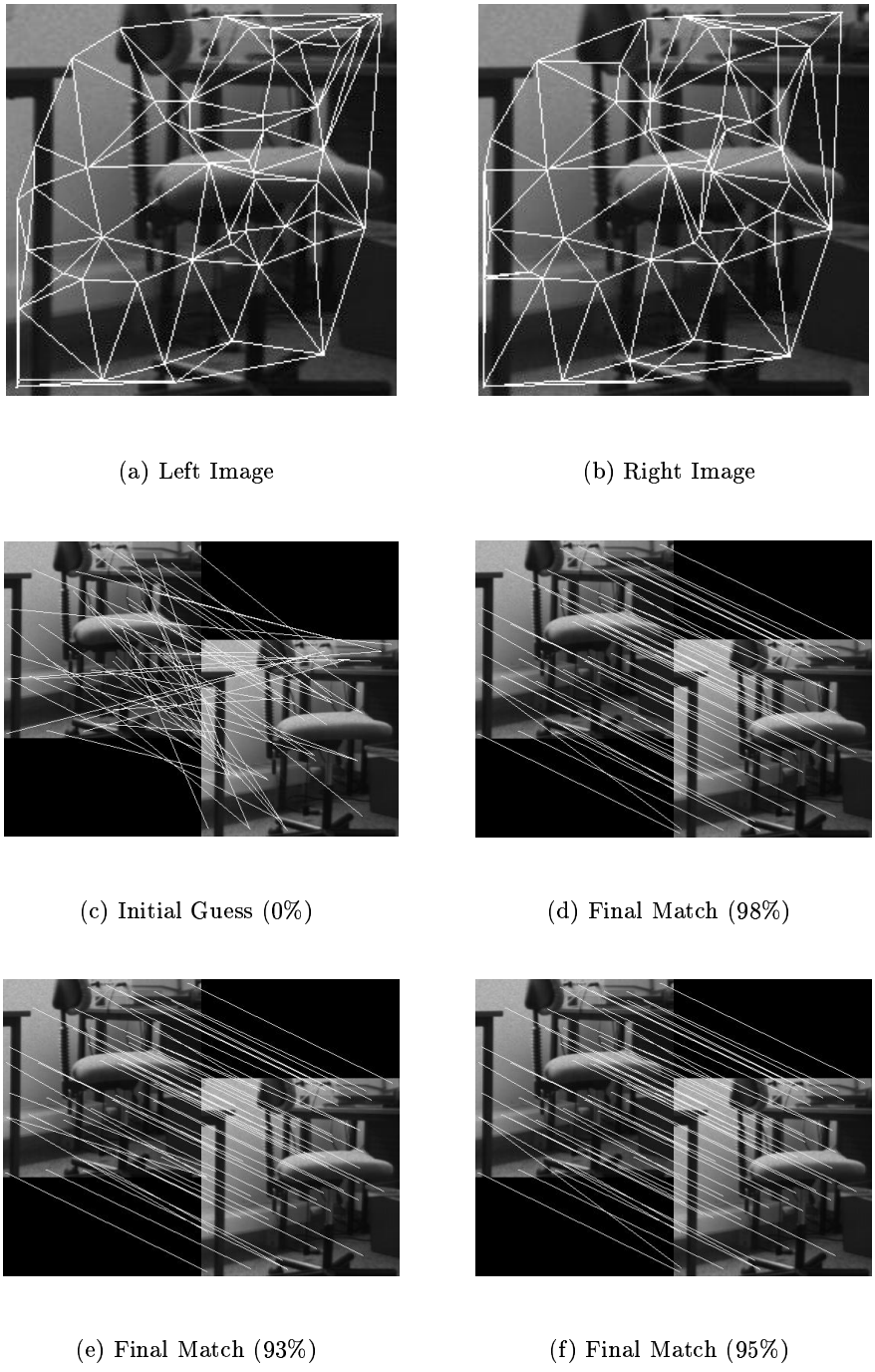
Figure 6.11: Average Yields versus Selection and Elitism II. Data from all four graphs has been pooled.

standard (single) or multiple elitism gave significantly better yields, but that the effect was small.

6.5 Real Images

This chapter closes with two examples showing the behaviour of the algorithm on real data. An in-depth study involving real images is beyond the scope of the thesis. However, these examples give some idea of the algorithm's capabilities. The sample problem is uncalibrated stereo matching. Panels (a) and (b) of figure 6.12 show an uncalibrated stereogram taken with a low-quality camera (an IndyCam). Regions were extracted from the greyscale image pair using a simple thresholding technique. Each image contained 50 regions. The region centroids were Delaunay triangulated using Triangle (Shewchuk 1996): the Delaunay graphs are shown superimposed on the original images. The average grey level over each region was used for the attribute information. As pointed out in section 4.1.1, such features do not generally permit unambiguous assignments. The Delaunay triangulations were matched using a hybrid genetic algorithm with neutral selection. The population size was set to 5, and 5 iterations were allowed. The crossover and mutation rates were 1.0 and 0.5 respectively. Panel (c) of figure 6.12 shows an initial guess in which none of the mappings is correct. Panels (d) to (f) show the three distinct solutions found.

There were 50 regions in the left image of which 42 had feasible correspondences in the right. The amount of relational corruption between the two triangulations was estimated at around 35% by counting the number of inconsistent supercliques given the ground truth match. Despite the significant relational corruption, the three solutions had 98%, 93% and 95% correct mappings.



(a) Left Image

(b) Right Image

(c) Initial Guess (0%)

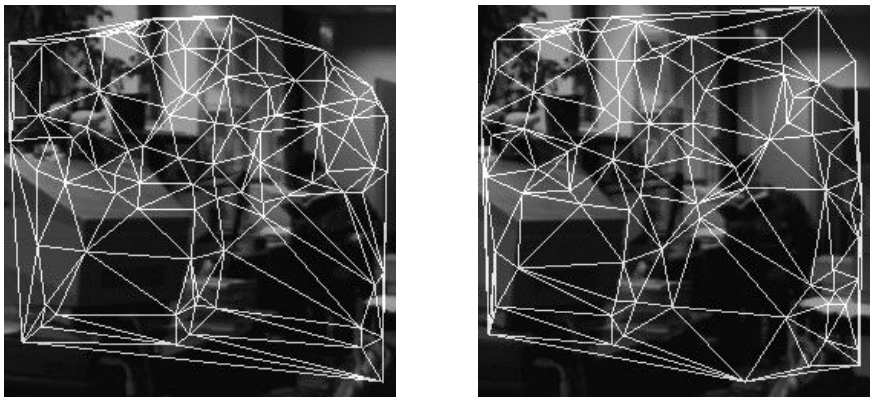
(d) Final Match (98%)

(e) Final Match (93%)

(f) Final Match (95%)

Figure 6.12: Uncalibrated Stereogram 1. The camera positions are not known.

A more challenging example is given in panels (a) and (b) of figure 6.13. Each of these images yielded about 100 regions. The region centroids were Delaunay triangulated as before. It can be seen from the shapes of the graphs that there is considerable relational corruption between the two images. The two graphs were again matched using a hybrid genetic algorithm with neutral selection. The population size was set to 10, and 10 iterations were allowed. The crossover and mutation rates were 1.0 and 0.5 respectively. Panel (a) of figure 6.14 shows an initial guess. Panels (b) to (f) show five out of the nine distinct solutions found. There was no ground truth for this example, but it can be seen that the initial matching errors have been largely corrected.



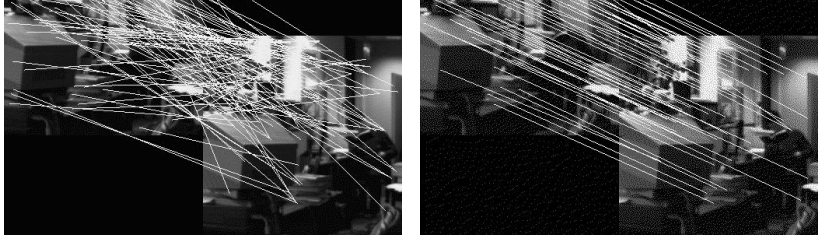
(a) Left Image (b) Right Image

Figure 6.13: Uncalibrated Stereogram 2. The camera positions are not known.

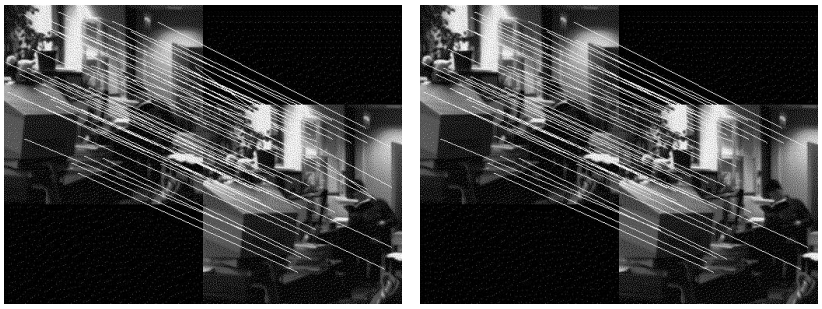
6.6 Summary

This chapter has considered ways of improving the solution yield of the genetic algorithm by promoting greater diversity in the population. The methods of maintaining diversity which have appeared in the literature almost always involve the addition of parameters to the algorithm, which is already over-parameterised. This chapter has shown that, for labelling problems, it is possible to maintain diversity in the genetic algorithm's population without the introduction of additional parameters.

Three distinct measures of population diversity were introduced. First, the Shannon entropy (Shannon 1948) measures the amount of clustering in the population. Second, the total inter-cluster Hamming distance seeks to estimate the volume of the search space



(a) Initial Guess (b) Final Match



(c) Final Match (d) Final Match

(e) Final Match (f) Final Match

Figure 6.14: Matching Results 2.

occupied by the populations. Third, the size of the gene pool represents the amount of genetic material available to the crossover operator, from which to synthesise new solutions. It was shown empirically that all three of these quantities decline inexorably over the course of an algorithm run, and that this decline is particularly marked in the first few iterations. With the CHC algorithm, the decline was catastrophic. It was argued that both selection and mutation played a part in the initial loss of diversity.

The mutation operator was shown to reduce the size of the gene pool in the initial population, but to increase it in saturated populations. This lead to the idea that the mutation rate should be set dynamically as the algorithm proceeds and that it should progressively

increase, in contrast to previous ideas about dynamic mutation rates (Fogarty 1989; Davis 1989; Cross et al. 1997; Davis and Principe 1991). Two methods of setting mutation rate were proposed. The first cast the mutation rate as a decreasing function of the entropy, while the second assigned the mutation rate according to a specific schedule. It was found that dynamically setting the mutation rate had no significant effect on the solution yield. This finding suggests that the loss of diversity due to mutation is not a major factor affecting solution yield.

An analysis of standard selection methods showed that a fundamental problem with this operator is the tradeoff between good search and premature convergence. Since the hybrid genetic algorithm contains a powerful local search mechanism, this tradeoff could be abandoned in favour of selection strategies which minimise selection pressure. Stochastic selection methods involving sampling with replacement may make several copies of fit individuals, but they also permit unfit individuals to be selected. Deterministic strategies, such as truncation, avoid this duplication at the cost of excluding unfit individuals. Selection without replacement was proposed in order to combine the advantages of limited copy number and stochastic acceptance. An experimental study showed that simple random sampling without replacement gave the best solution yield in a hybrid genetic algorithm.

Chapter 7

Conclusion

This thesis has shown that the genetic algorithm can be used successfully as part of a least commitment approach to ambiguous and inexact consistent labelling problems. The algorithm outperforms deterministic methods both in terms of optimisation performance, and in terms of the ease with which alternate solutions to a problem can be obtained. The major contribution of the thesis has been the least commitment framework for labelling. However, two other contributions are worthy of note. First, a substantial experimental study was conducted which resulted in empirical models for the algorithm's behaviour on labelling problems. These models can be used to set algorithm parameters for given problems. The models have been shown to extrapolate reasonably well. Second, it has been suggested that hybrid genetic algorithms work by assembling an initial guess for the local search step. Rather than the local search being an adjunct to the genetic algorithm, which is the more conventional view (Cross 1998; Whitley 1996), it actually seems to be responsible for most of the optimisation.

7.1 Summary of Contribution

Chapter 3 reviewed and gave a unified presentation of the Bayesian consistent labelling framework of Hancock and Kittler (Hancock and Kittler 1990a). This criterion was modified in three ways. First, defining the consistency criterion as a geometric rather than an arithmetic mean yielded a direct measure of consistency which is linear in the minimum distances between configurations and dictionary items. Second, the framework was

adapted to handle inexact labelling problems in terms of the string edit distance between labelled configurations and dictionary items. This is a more theoretically sound way of making the comparisons than Wilson's dictionary padding approach (Wilson 1995), and avoids the worst case exponential time and space complexity associated with dictionary padding. Third, the "neighbourhood approximation" was developed which requires a linear factor less time (or fewer processors) than the criterion given in (Wilson 1995). This approximation is specific to graph matching. An experimental study showed that the edit distance based dictionary comparisons are more accurate with respect to relational corruption, and more efficient, than those involving padded dictionaries. The edit distance based method was, however, found to be slightly more sensitive to initialisation error. The neighbourhood approximation was found to significantly improve algorithm running times without sacrificing accuracy.

The literature survey in chapter 2 identified the inability to apply the principle of least commitment to current labelling schemes as an unsolved problem in computer vision. Chapter 4 described this problem in the context of genuinely ambiguous line labelling problems. Labelling drawings of impossible objects and inexact matching problems are ambiguous in the sense that there may be several good, but inconsistent, solutions to the problem. A new method of combining evidence from the scene with the labelling constraints was proposed for graph matching problems, which takes into account the potential ambiguities in the measurement information. The genetic algorithm was applied to these problems: of particular interest were the crossover and local search operators. The hybrid genetic algorithm with a gradient ascent step was found to produce good optimisation performance and tolerable solution yields. This was in stark contrast to pure gradient ascent, which was inadequate for line labelling. For graph matching, gradient ascent performed well because the attribute information allowed a good initial guess to be made. However, this method is deterministic, and therefore only produces a single possible solution. Similarly, Eshelman's CHC algorithm (Eshelman 1991), although a better optimiser than the standard genetic algorithm, produced poor solution yields. A study of metric based crossovers found that the hybrid genetic algorithm performed better with highly disruptive crossovers. The standard uniform and novel "balanced optimal contiguous" crossovers outperformed both two point and Cross's geometric operator (Cross 1998). This suggests that the gradient ascent step took the greater part in the optimisation, since disruptive operators might be expected to compromise the search in a standard genetic algorithm applied to labelling problems (Cross 1998).

Having established the feasibility of using a genetic algorithm in this way, it was necessary to find a set of control parameters (population size, iteration limit, crossover and mutation rates) which gave the best algorithm performance. The literature only "recommends" three sets of control parameters for the genetic algorithm (DeJong 1975; Grefenstette 1986; Schaffer et al. 1989). The experimental study in chapter 4 demonstrated that two out of these three sets were inappropriate for hybrid genetic algorithms applied to labelling problems. The first part of chapter 5 argued against extrapolation from any of the three parameter sets. Any empirical model of genetic algorithm behaviour must be restricted to a particular problem domain. Results for line labelling do not always generalise to graph matching, even though those two problems are both instances of the consistent labelling problem. It is therefore even less sound to expect algorithm behaviour on continuous numerical optimisation problems to generalise to discrete optimisation problems. Chapter 5 presented a substantial experimental study which led to quantitative empirical models for labelling problems. These models allowed optimal conditions for the algorithm to be established. An interesting feature of the models is that very much lower population sizes than expected (Cross 1998) were needed. This can be explained by postulating that the main function of the genetic algorithm is to assemble an initial guess for the local search step. A probabilistic lower bound on the population size in terms of the size of the label set was established. As for the other parameters, only 10 iterations of search are required; the crossover rate should be set to 1.0, and the mutation rate should be high but below 0.6. The Bayesian criterion outperformed the linear one for line labelling, but the linear criterion was better for graph matching. The empirical models were found to behave well under extrapolation to problems up to three times as large as those used for the experiments.

The solution yield seemed relatively insensitive to algorithm control parameter choice, provided there was good search. Chapter 6 addressed the question of maintenance of diversity in the algorithm's population. Previous attempts invariably required the introduction of additional parameters (Cedeño et al. 1995; Goldberg and Richardson 1987; Beasley et al. 1993). The view was taken that the algorithm is already over-parameterised, and that attempts to maintain diversity should in the first instance avoid introducing new parameters. First, three different diversity measures were presented. These were the Shannon entropy, the total inter-cluster Hamming distance and the size of the gene pool. Experiments showed that all three of these measures declined as the algorithm proceeds, and that the most substantial decline occurs at a very early stage in the search. A theoretical analysis

showed that mutation would cause a decrease in the gene pool size at first, and suggested that it might therefore be helpful to steadily increase the mutation rate from some low initial value. Experiments showed that the initial effects of mutation were not sufficiently severe to necessitate this approach. The initial loss of diversity can be attributed mostly to the selection operator, since this phenomenon occurs both in the presence and in the absence of a gradient ascent step. The idea, from chapters 4 and 5, that in the hybrid genetic algorithm the standard operators mainly exist to serve initial guesses to the local search step, suggested that the rôle of the selection operator was relatively limited in terms of search. Analysis of selection showed that the tradeoff between good search and premature convergence could be avoided for hybrid genetic algorithms by applying no selection pressure at all. "Neutral selection" was proposed, in which individuals are selected from the current population at random without replacement. This operator guarantees that individuals will not saturate the population at an early stage. An experimental study showed that neutral selection produces large solution yields without compromising search.

7.2 Future Work

There are several issues raised by this thesis which merit further attention. The treatment of edit operation weights in chapter 3 is unsophisticated. It is possible to learn edit weights for particular problems (Ristad and Yianilos 1998). Rather than arbitrarily setting all the weights to 1, it might be possible to learn weights from a training set of labelling problems. Similarly, the ambiguous measurement framework for graph matching, introduced in chapter 4, was somewhat *ad hoc*. In particular, the control of the "ambiguity parameter" was not properly addressed: it was set according to an estimate of the amount of overlap between the two graphs. However, it is arguable that the degree of ambiguity need not remain static during the labelling process. Further work should be done to determine optimal values for this parameter. Another shortcoming of chapter 4 is the lack of a full comparative study between different optimisation schemes and genetic algorithm variants. Although comparison was made to gradient ascent and the CHC algorithm (Eshelman 1991), no attempt was made to compare the algorithm to simulated annealing, and no investigation of alternate genetic algorithms such as GENITOR (Whitley 1989) or the steady-state algorithm (Syswerda 1989) was made.

There is also considerable scope for extending the experiments in chapter 5. A limita-

tion of the experimental study was the relatively small number of graphs used as test material, especially in view of the finding that structural effects were a dominant factor in determining algorithm outcome. Any subsequent experiments along these lines should use several different graphs for each problem size, in the hope that the structural effects will be absorbed into random error. This would provide a more accurate picture of the true relationship between problem size and outcome. Because of limited computer time, it was not possible to study very many levels of each factor. This is less of a weakness than might be thought, because just four levels of each factor permit cubic models to be fitted with accuracy, and chapter 5 showed that cubic models were in general too complex to be of much use when extrapolating. A more subtle limitation relating to computational resources is that the analysis of factorial experiments is itself subject to a combinatoric explosion as the number of factors increases. It was not feasible to consider higher than triple interactions in these experiments, although they may well exist. However, the same data set can be re-analysed at a later date if higher order interactions are to be studied. The arguments concerning the minimum population size necessary to enable the algorithm to assemble a good initial guess need further development. At present, the expressions in equations 5.10 and 5.15 require the experimenter to choose a population size which raises the probability of good initial guesses to an arbitrary threshold. It would be better to have the population size expressed directly in terms of the threshold.

Chapter 6 attempted to control diversity in the genetic algorithm's population without introducing many new parameters to the algorithm. However, considerable success has been reported for niching techniques and distributed algorithms in terms of maintaining diversity (Goldberg and Richardson 1987; Beasley et al. 1993; Jelasity and Dombi 1998; Davidor 1991). Even if the distributed approach is only a temporary solution to the diversity problem, niches would only need to persist for a very few algorithm iterations, since hybrid algorithms usually locate their optimal solutions within 10 iterations. It is therefore necessary to extend this study to niching strategies and distributed arrangements.

One of the criticisms which is easiest to level at the genetic algorithm is its hunger for computational resources. A hybrid genetic algorithm with a population size of 100 will require more than 100 times the resources used by gradient ascent. In terms of improvement of solution quality, this may not be a price worth paying unless one has another requirement, such as simultaneously obtaining several solutions to the problem in hand. However, genetic algorithms are naturally parallel (Holland 1975). It is not clear how one might use a

parallel machine to perform optimisation tasks using other algorithms. It has been found that methods such as simulated annealing are not easy to implement in parallel (Aarts and Korst 1989; Goldberg 1990). In other words, a serial implementation of a genetic algorithm might require substantial computational resources, but a parallel implementation might make optimal use of the available resources. Mahfoud and Goldberg describe an interesting parallel implementation of a combination of simulated annealing with a genetic algorithm in (Mahfoud and Goldberg 1995). A parallel genetic algorithm might require O(1) fitness evaluations given enough processors. However, these fitness evaluations themselves could be performed in parallel since the criteria of equations 3.9 and 3.13 in chapter 3 are parallel iterative. The availability of parallel and hardware implementations which perform edit distance calculations (Egecioglu and Ibel 1996; Sastry et al. 1995) raises the possibility of real time implementation of the labelling process. Naturally, parallelisation of all three of the genetic algorithm, fitness evaluation and dictionary comparison would be a major undertaking. The designer would, however, have the choice of which level of the process to accelerate. It is quite possible, then, for the genetic algorithm to operate as a hypothesis server in a multilevel vision system. The algorithm would be capable of adapting to the dynamic changes in input and labelling constraints which would arise in real-time vision. The final population would have some of the perceptual richness of a primal sketch (Marr 1982), presenting many alternatives for a higher level process to choose from.

A final thought is that the consistent labelling problem encompasses problems ranging from the trivial to the intractable. Ackley's ONEMAX problem (Ackley 1987) would be solved in a single iteration of a hybrid genetic algorithm with a population size of 1. Line labelling is also relatively easy for the algorithm; graph matching requires a gradient ascent step. However, satisfiability would perhaps present a serious challenge to the algorithm. It is therefore not unreasonable to suppose that, because labelling problems have been so extensively studied (Mackworth 1977; Haralick and Shapiro 1979; Haralick and Shapiro 1980; Haralick et al. 1978; Nudel 1983; Jeavons 1998), they might provide a realistic framework in which to further investigate genetic algorithm behaviour.

Appendix A

Genetic Algorithms for Structural Editing

This appendix is a copy of a paper which was published as (Myers and Hancock 1998). It proposed a genetic algorithm framework for imposing structure on input data. The idea was that the genetic algorithm's operators could be seen as a solution editing process. The example used was a simple natural language processing problem. This was chosen because the "tree adjoining grammar" formalism developed by Joshi in (Joshi 1985) provides a structure and a set of editing rules which seemed readily adaptable to a genetic algorithm framework. Rather than generating parse trees by the application of rewrite rules, the algorithm constructed parse trees and attempted to fit them to the input. The quality (fitness) of the fit was measured by the edit distance between the terminal symbols in the parse tree and the lexical categories of the input.

Viewed as a labelling problem, the language processing example is not amenable to the dictionary based approach presented in chapter 3. This is because the constraints are applied globally rather than locally. Each word in the input has a dictionary of terminal symbols associated with it: these are the unary constraints. However, the binary constraints between word pairs depend in many cases on the syntactic structure of the sentence. In this paper, the algorithm effectively labelled the input string with an entire tree, as opposed to labelling the nodes of a graph as in chapter 4. The problem as posed was thus considerably harder than relational matching. To make matters worse, no *a priori* evidence was used to guide the labelling process. Expecting a genetic algorithm with no local search step to solve this problem was perhaps a little optimistic. Nevertheless, this study serves as an

example of how the algorithm can be used to generate interpretations with quite complex structures given relatively simple constraints.

A.1 Introduction

Graphical models have recently attracted considerable interest in the connectionist literature where they have been used to embed causal relations into network structures. Broadly speaking, the available models can be divided into those that draw on undirected graphs and those that draw on directed graphs. According to this taxonomy, Markov models belong to the former category while causal networks fall into the latter category. In fact directed graph structures are of particular importance since they can be used to represent subsumption or part-whole hierarchies. Such representational structures are of pivotal importance in language understanding and vision where they are used to control the vertical flow of perceptual inference. Despite this interest in the modelling of probabilistic interactions in network structures, the issue of how to control the structure itself has attracted less attention. This is an important omission since in practice the task of extracting hierarchical relations from real-world data is invariably one of extreme fragility. In the machine vision domain, it was Sanfeliu and Fu who first illustrated the importance of graph-edit operations in matching noise corrupted relational structures (Sanfeliu and Fu 1983). More recently, Meila and Jordan have used graph-edit operations to moralize intractable graph-structures (Meila and Jordan 1997).

It is the editing of directed graph structures to which we turn our attention in this paper. Specifically we focus on the issue of how to extract the most consistent tree-structure from imperfectly formed input. This can be viewed as an optimisation problem. Our goal is to recover hierarchical structure inherent in input, based on low-level information, according to predefined structural rules. Essentially, the structure which best fits the low-level data is the preferred interpretation. In order to do this we require a hierarchical representation of the domain of interest and a means of measuring the accuracy with which a model fits the data. It is clearly inappropriate to enumerate every possible interpretation in the model database. Rather, a natural means of extending some initial small database is required.

The obvious hierarchical representation is a graph - for a subsumption hierarchy, a tree. Models will need to undergo vertical edit operations to adapt their structure to the data.

Unfortunately, most graphical editing operations described in the literature are rather naïve, being single-node (or -edge) insertion, deletion or relabelling (Sanfeliu and Fu 1983; Eshera and Fu 1984; Shasha and Zhang 1989; Zhang 1996; Skillicorn 1996) (to be fair, these are generally intended as measures of distance in matching algorithms rather than as true editing operations). The geometric crossover operator described by Cross and Hancock in (Cross et al. 1997) is somewhat closer to the mark: it involves bisecting two Delaunay graphs with a random line in order to edit a match relation which exists between them. However, this operation does not necessarily generalise to other types of graph, and is in any case global. What is required is a local structural edit operation which modifies intermediate subgraphs, ranging from single nodes to the entire graph.

The *tree adjoining grammar* formalism developed by Joshi in (Joshi 1985) provides an interesting starting point. Tree adjoining grammars were originally designed for natural language processing: rather than use a set of rewrite rules, these grammars provide sets of minimal parse trees which may be extended by adjunction (section A.2). Thus parsing with a tree adjoining grammar can be seen as fitting a hierarchical model to the input and iteratively modifying that model until it adequately describes the data. Our interest is primarily in the structural editing process, rather than computational linguistics. However, language processing is the natural application of tree adjoining grammars and provides a convenient non-trivial hierarchical structure recovery problem with which to explore our framework.

A.2 Tree Adjoining Grammars

A tree adjoining grammar, G, is a pair, $(\mathbf{I}, \mathbf{A})$, of sets of ordered labelled trees, where $\mathbf{I}$ is the set of *initial trees*, which correspond to minimal sentences in the string language of G, and $\mathbf{A}$ is the set of *auxiliary trees*, which extend the trees in $\mathbf{I}$ via adjunction to give new initial trees. By definition, all the external nodes of an initial tree are preterminal symbols (lexical categories) and all of its internal nodes are nonterminals. Auxiliary trees are similar to initial trees, except that exactly one of the external nodes has the same label as the root: this is the *foot node*. Tree adjoining grammars are moderately context-sensitive, a property they owe to the manner in which recursion and dependencies are expressed (Joshi 1985).

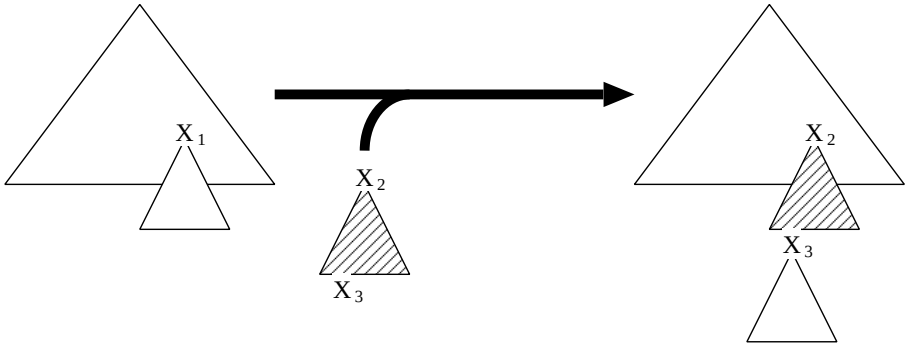


Figure A.1: Tree Adjunction.

A.2.1 Tree Adjunction

Adjunction (figure A.1) is a composition operation between initial and auxiliary trees: given an auxiliary tree, A, and an initial tree, I, which contains a node with the same label, X, as the root of A, the adjunction of I and A is achieved by removing the subtree in I rooted at X, inserting A in its place, then attaching the children of X to the foot node of A. Thus, adjunction inserts an extra layer of structure into an initial tree. It is usual to place constraints on adjunction at some of the internal nodes of trees. These constraints are discussed in detail in (Becker et al. 1995), where they are used to implement features via unification. The only constraint of interest to us at present is the null-adjunction constraint which forbids adjunction at any node bearing it.

A.2.2 Subtree Crossover

We define an additional operation, subtree crossover (figure A.2), in which subtrees with a common root label are exchanged between two initial trees. The subtrees should be distinct, as should the remainders of the parent trees. This is a special case of Koza's operator (Koza 1992), but a more general case of the *substitution* operator used in (Becker et al. 1995), which is essentially a notational shorthand; a non-terminal node in the frontier of an initial tree must be replaced by another initial tree to yield a tree in which the frontier nodes are all terminal symbols. Such trees represent sets of sentences with a certain type of constituent in a particular position.

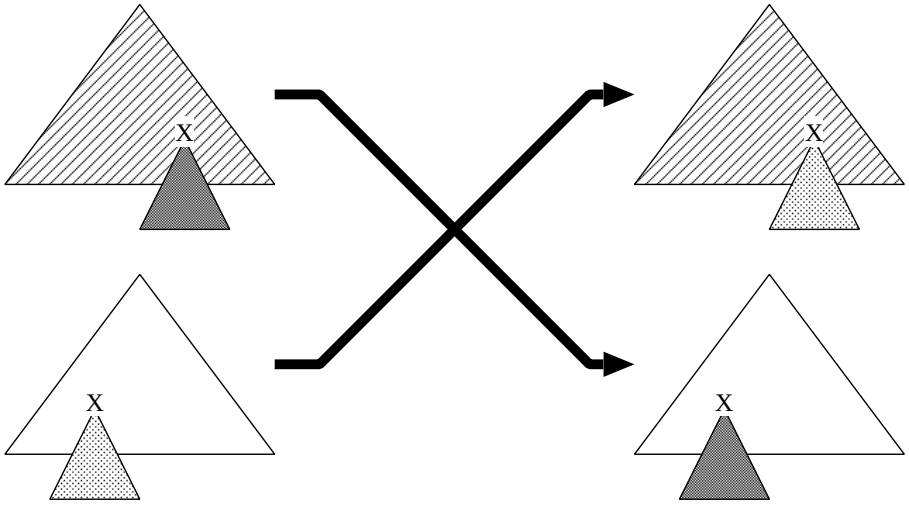
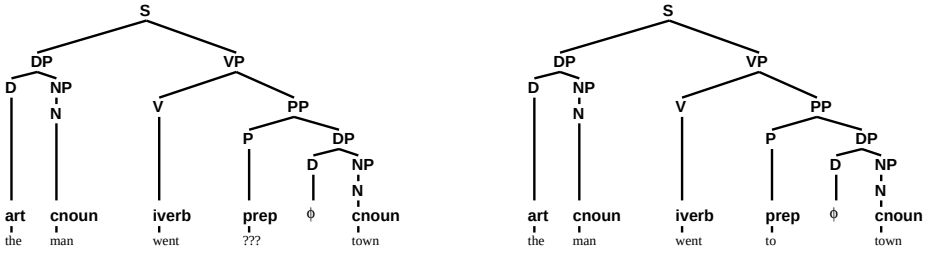


Figure A.2: Subtree Crossover.



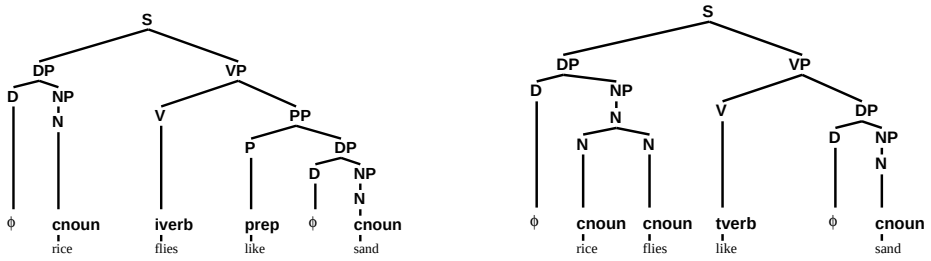
(a) Incomplete Sentence

(b) Complete Sentence

Figure A.3: Handling of Incomplete Sentences. The parser should recognise that a preposition is missing from the sentence *"the man went town", possibly enabling it to correct the error.

A.3 Natural Language Processing

This section describes our approach to natural language parsing. We use this as a test problem for the structural editing framework, and therefore give a much simpler formulation than the state of the art for language processing. Specifically, we concentrate on the structure of the parse tree and ignore augmentations such as features. The problem can be simply stated as that of finding the best parse of an input sentence. We would like to be able to gracefully handle the case where a word in the input is unknown, misspelt or missing as illustrated in figure A.3. We would also like ambiguous cases (see figure A.4) to be handled neutrally in the absence of *a priori* evidence, i.e. the ordering of rules in the rulebase should not bias the parser in favour of any particular interpretation.



(a) "Rice" is the subject

(b) "Flies" is the subject

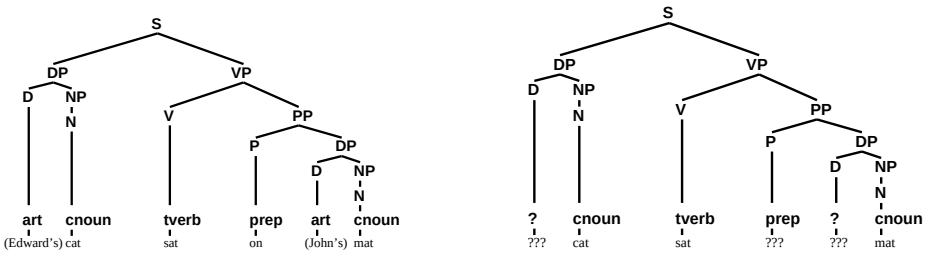
Figure A.4: Handling of Ambiguous Sentences. The parser should not discriminate arbitrarily between the available interpretations. This example is a modification of the one in (Allen 1994).

A.3.1 The Grammar

Our grammar is based on the lexicalised tree adjoining grammar given in (Becker et al. 1995). We do not consider features, and have simplified the grammar considerably to suit our experimental purposes. Nevertheless, we retain the null-adjunction constraint, and have also added a few trees to make the grammar more consistent with X-bar theory. The grammar covers sentences involving intransitive, transitive and ditransitive verbs, relative and subordinate clauses, adjuncts, conjunction, and a number of simple constructions; however, we make no claim as to its completeness or accuracy which are not of primary interest in this paper. As is standard, the nonterminal symbols are typically phrasal constituents and are written in uppercase letters: noun phrases (NP), verb phrases (VP), clauses (SC), sentences (S), adverbials (ADV) and so-on. The terminal symbols are the lexical categories for which we use a non-standard notation to partly compensate for the lack of features. These are written in lowercase and include articles (art), common nouns (cnoun), transitive verbs (tverb) and so-on.

A.3.2 Accuracy of Parses

Section A.2 describes how parse trees for a particular sentence may be generated; it remains to define some measure of how accurate a parse is, and whether or not it contains missing constituents. In (Becker et al. 1995), a lexicalised grammar is used: each word in the input selects a set of trees which are then combined to form a parse; the final parses are



(a) Unknown Words

(b) Missing Words

Figure A.5: Extreme Examples. In (a), words in parentheses are unknown, and may match anything (they must match something); the edit-distance is 2 without wildcards, 0 with. In (b) the essential structure of the utterance is recovered even though it is badly formed; the edit distance is 3. In both cases, the edit-distance algorithm allows the parser to make maximal use of the available information, yielding "correct" parses of the input.

assessed manually. In our scheme, each position in the sentence is assigned a dictionary of lexical entries based on the lexicon for the word at that position. For example, for the sentence "rice flies like sand", the dictionary at position 3 would be {tverb, prep}, hence the ambiguity (figure A.4). It is now possible to evaluate the accuracy of a parse tree by considering how well its frontier matches the dictionaries assigned to the word-positions in the input. This is done using a modified version of the classic edit-distance algorithm given by Wagner and Fischer in (Wagner and Fischer 1974): where they used equality as a match condition, we use set-membership. This algorithm is known to find the minimum number of insertions and deletions necessary to make the two strings identical and thus provides a natural measure of how well a particular parse tree describes a given sentence (since the hierarchical structure of a sentence depends only on the grammatical rules used to parse it). We use a wildcard dictionary entry for unknown words: this allows sentences with missing or misspelt words to be parsed successfully whilst preserving the structure of the known parts of the sentence.

Using the string edit-distance in this way imparts a measure of "intelligent" behavior to the algorithm. Suppose the input is "Edward's cat sat on John's mat" and that neither "Edward's" nor "John's" is in the lexicon. The best parse is still that shown in figure A.5 (a). Similarly, the best parse of *"cat sat mat" is that given in figure A.5 (b).

A.4 Genetic Algorithms

The genetic algorithm (and variants thereof) is a well-known population-based global optimisation strategy (Holland 1975; Rudolph 1994) - reviews may be found in (Fogel 1994) and (Srinivas and Patnaik 1994b). Briefly, a population of candidate solutions to a problem (the *individuals*), usually encoded as binary strings, is iteratively subjected to crossover in which parts of two individuals are mixed to yield two offspring, mutation in which one individual is subject to random change, and selection in which individuals are stochastically chosen to form the next generation. A measure of the quality of the solution represented by the individual is its *fitness*, which is translated into a probability of its survival into the next generation. The algorithm intuitively lends itself to problems which may be decomposed or partially decomposed and which have many local and global optima, for example line labelling (Huffman 1971; Myers and Hancock 1997a). The algorithm composes a good solution by mixing sub-solutions with the crossover operator. Mutation operates at a low level as a source of background variation which allows new information to enter the population. The stochastic nature of selection allows the population to escape local optima.

Natural language processing is related to line labelling, both being instances of the consistent labelling problem first formulated by Haralick and Shapiro in the 1970s (Haralick and Shapiro 1979; Haralick and Shapiro 1980). In natural language processing, the goal of the parser is to label the words in the input sentence with their lexical categories. Unlike line labelling, however, it is also desirable to construct a hierarchical representation of the sentence, its parse tree. We have demonstrated elsewhere that genetic algorithms are suitable for line labelling (Myers and Hancock 1997a), especially when it is necessary to obtain several closely related solutions simultaneously (Myers and Hancock 1997b). Our interest here is the algorithm's solution-editing framework (crossover and mutation) rather than its optimisation properties which are not certain - indeed for medium (40 to 60 lines) line labelling problems, exhaustive search comfortably outperforms the algorithm. The rest of this section formulates our version of natural language processing with a tree adjoining grammar for the genetic algorithm.

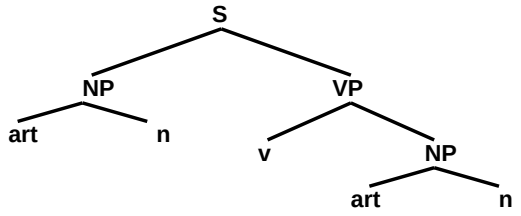


Figure A.6: S-Tree Example. This is the graphical representation of the S-tree `(S((NP((art)(n)))(VP((v)(NP((art)(n)))))))`.

A.4.1 Solution Encoding

The population in our genetic algorithm consists of a set of initial trees drawn (initially at random) from the set of initial trees in the grammar. A natural way to encode trees is as strings. These are technically Lisp-like S-expressions, but we refer to them as S-trees to emphasise their structure. The grammar for constructing S-trees is given below.

```
   STREE --> '(' label ['(' SUBTREES ')'] ')'
SUBTREES --> STREE | STREE SUBTREES
```

Label is the label of a node in the tree which must be a string of characters. As an example, figure A.6 gives a graphical representation of the S-tree:

```
(S((NP((art)(n)))(VP((v)(NP((art)(n)))))))
```

A null-label may be applied to any symbol in the tree, for example `(D:-)` is an empty D(eterminer). Other assignments are made by the edit-distance algorithm: when an assignment is inconsistent or involves an unknown word, it is denoted with an asterisk, '*'. Thus, the trees in figure A.5 would be denoted as shown in table A.1.

In principle, any tree can be represented in this manner, although the representation becomes clumsy when the node-labels are complex or the nodes contain a large amount of information. Simple features such as the null-adjunction constraint can be added by prefixing the node label with special characters according to some simple regular grammar. The advantage of this representation is its transparency, the ease of extracting the frontier while computing the edit-distance from the input, and the fact that the operations described in sections A.2 and A.4.3 can be implemented by matching and copying substrings. This representation is also compact.

(a) Unknown Words	(b) Missing Words
(S(	(S(
(DP(	(DP(
(D((*art:Edward's)))	(*D:-)
(NP((N((cnoun:cat)))))	(NP((N((cnoun:cat)))))
))	))
(VP(	(VP(
(V((tverb:sat)))	(V((tverb:sat)))
(PP(	(PP(
(P((prep:on)))	(P((*prep:-)))
(DP(	(DP(
(D((*art:John's)))	(*D:-)
(NP((N((cnoun:mat)))))))	(NP((N((cnoun:mat)))))))
))	))
))	))
))	))

Table A.1: String Representation of Parse Trees. Inconsistent assignments are marked with asterisks ('*').

A.4.2 Fitness Measure

The edit-distance cost function described in section A.3.2 is a discrete measure which takes values from the set $\{0, 1, ..., N\}$, where N is the larger of the number of words in the input and the number of nodes in the frontier of the tree. To convert this into a fitness measure suitable for use with a genetic algorithm we exponentiate. In equation A.1, ED_i is the edit-distance of the i^{th} S-tree, F_i is its fitness and β is an arbitrary scaling constant which defaults to 1.

$$F_i = \exp\left(-\beta.ED_i\right) \tag{A.1}$$

To convert this into a survival probability for the selection step, we divide by the total fitness of the n members of the genetic algorithm population.

$$P_i = \frac{F_i}{\sum_{j=1}^{n} F_j} \tag{A.2}$$

A.4.3 Genetic Algorithm Operators

Crossover is simply implemented using subtree crossover defined in section A.2. Given two parent S-trees, the algorithm selects a node from each subject to the conditions that (1) the nodes have identical labels and are not subject to null-adjunction, (2) the subtrees rooted at the nodes are different, (3) the remainders of the parent trees following excision of the subtrees are different, (4) at least one of the nodes is not the root node, and (5) at least one of the subtrees is not empty. In practice, several pairs of nodes may satisfy these criteria: in this case the crossover sites are selected at random. A drawback of subtree crossover is that not all pairs of trees can be crossed, and only a subset of the nodes may be part of a crossover site in a particular crossing, both of which make the term "crossover rate" a little harder to define. However, since our choice of interpretation is restricted by the input data, in practice only a relatively small number of S-trees will survive the first few iterations of the algorithm, and these are likely to be of similar types. This crossover will thus be less disruptive and explore the search space less well than more traditional operators.

Mutation does not have such a natural implementation. It does not make sense to simply relabel nodes in a parse tree: internal nodes must never be relabelled since they describe permitted phrase structures in the grammar, and frontier nodes cannot generally be relabelled because only certain classes of words can form particular constituents. The adjunction operator seems a reasonable choice since this makes a point-modification to a single initial tree. This is implemented by forming a set of adjoinable auxiliary trees, selecting one and then finding a suitable adjunction site in the initial tree (i.e. a node with the same label as the root of the auxiliary tree, and not subject to the null-adjunction constraint). A major disadvantage of this implementation is that mutation can no longer necessarily be considered a background operator: adjunction is a fundamental means by which novel parse trees are constructed. One way around this limitation is to adjoin every tree with every adjoinable auxiliary tree at every possible site in a preprocessing step. This step, which we call "expansion" may be repeated as many times as desired; however it is computationally expensive[1] and is therefore only suitable when the sets of trees are

[1]We do not give a formal complexity result, but the number of adjunctions is linear in the numbers

small. Adjunction may have far-reaching effects on the structure of a tree which is in sharp contrast to the local nature of the standard mutation operator.

A.5 Experiments

In a preliminary study, our algorithm was tested on artificially constructed sentences; for added realism, we also used 16 sentences drawn from a letter from a funding body. We increased the complexity of some sentences by adding adverbs, adjectives and additional words and clauses. Sample sentences are given below.

```
1) All awards are available.
2) All studentship awards are available.
3) Our awards are clearly helpful.
4) Please remember to include your award reference number.
5) Please remember to include the award reference number in the top
   right hand corner of the address label.
```

The grammar consisted of 49 initial and 45 auxiliary trees (370 and 1138 with expansion). We tested the algorithm on several different parameter settings with and without a single pass of expansion. 25 trials were conducted for each sentence with a variety of population sizes and iteration limits; crossover and mutation rates were fixed at 0.9. Lower values of crossover and mutation rates were also tried: these yielded uniformly poor performance and are not reported here. The results are given in table A.2.

A.5.1 Discussion

Elsewhere we have reported that for line labelling, mutation rate is the most accurate predictor of success rate, followed by population size and crossover rate with iteration limit playing little part (Myers and Hancock 1997a). Our initial results for language parsing agree to some extent with this: all the best runs had population sizes of 4000, and

of initial and auxiliary trees. The number of initial trees increases with the squares of the numbers of auxiliary trees and adjunction sites, which respectively increase quadratically and linearly with iteration number

Population Size	Iteration Limit	Accurate Parses	
		expansion	no expansion
100	1000	0%	13%
200	500	0%	13%
500	200	0%	13%
1000	100	6%	25%
2000	50	6%	25%
		-	17%
2000	100	0%	19%
4000	25	6%	13%
4000	50	6%	31%
		-	22%
4000	100	6%	25%
		-	27%

Table A.2: Experimental Results. 25 runs were performed for each sentence to give a total of 400 runs. The crossover and mutation rates were fixed at 0.9. Some combinations were tried more than once.

the limit on iterations does not seem to be particularly relevant. However, low values of the mutation and crossover rates tended to give poor performance. This is unsurprising since both operators are of fundamental importance in the parsing process.

The need for high population size is remarkable, since the best population size we tested was about 100 times the cardinality of the unexpanded initial tree set. This is probably due to the well-known problem of premature convergence - we have shown previously that the diversity of the population decreases sharply in the first few iterations (Myers and Hancock 1997b). Thus for complex sentences there may be relatively few avenues open for search after the initial phase of the algorithm.

It is clear that grammar expansion does not help. This can perhaps be explained in terms of the structure of the input sentence. A "deep" sentence is one which requires many adjunctions to generate a correct parse: it has a lot of structure and will typically contain many modifiers. It appears that the deep structure of sentences is relatively inaccessible to the algorithm, since many specific adjunctions are required to generate a correct parse. Thus, a single pass of the expansion step is unlikely to substantially simplify the task of

parsing complex sentences: only those sentences with one or two levels of complexity are made significantly easier. However, the expansion process does create a large number of "spurious" initial trees: it is likely that without expansion all 49 initial trees would be represented in a population of 1000, with few auxiliary trees to choose from for adjunction. Blindly increasing the numbers of initial and auxiliary trees appears to effectively increase the probability of making an incorrect choice.

A.6 Conclusion

The main contribution of this paper has been to investigate the use of discrete graphical editing operations within an optimisation framework. We have adapted the genetic algorithm for use with a tree adjoining grammar, and demonstrated its utility with a simple natural language processing example.

It is clear that there are several directions in which this work can be developed. It is worth investing some time fine-tuning the genetic algorithm's control parameters; these are notoriously difficult to set *a priori* (Grefenstette 1986; Schaffer et al. 1989; DeJong and Spears 1990) . To use our framework for serious language processing would require considerable work on augmenting the grammar and lexicon. We intend to develop the work by exploring more complex hierarchical problems furnished by vision, for example multilevel scene analysis.

Appendix B

Publication List

The following is a list of publications which resulted from the work undertaken in this thesis.

R. Myers and E. R. Hancock. Genetic algorithms for ambiguous labelling problems. *Lecture Notes in Computer Science (EMMCVPR 97)*, 1223:345–360, 1997.

R. Myers and E. R. Hancock. Genetic algorithm parameters for line labelling. *Pattern Recognition Letters*, 18:1363–1371, 1997.

R. Myers and E. R. Hancock. Genetic algorithms for structural editing. *Lecture Notes in Computer Science (SSPR 98)*, 1451:159–168, 1998.

R. C. Wilson, R. Myers and E. R. Hancock. Efficient relational matching with local edit distance. In *Proceedings of the* 14^{th} *International Conference on Pattern Recognition*, pages 1711–1714, 1998.

Bibliography

E. Aarts and J. Korst (1989). *Simulated Annealing and Boltzmann Machines.* Wiley.

D. H. Ackley (1987). *A Connectionist Machine for Genetic Hillclimbing.* Kluwer Academic.

J. Allen (1994). *Natural Language Understanding.* Benjamin-Cummings.

A. P. Ambler, H. G. Barrow, C. M. Brown, R. M. Burstall, and R. J. Popplestone (1975). A versatile system for computer-controlled assembly. *Artificial Intelligence 6*, 129–156.

P. Andrey and P. Tarroux (1994). Unsupervised image segmentation using a distributed genetic algorithm. *Pattern Recognition 27*, 659–673.

F. J. Anscombe (1953). Contribution to discussion of a paper by H. Hotelling. *Journal of the Royal Statistical Society B15*, 229–230.

F. J. Aranda-Ordaz (1981). On two families of transformations to additivity for binary response data. *Biometrika 68*, 357–363.

J. E. Baker (1985). Adaptive selection methods for genetic algorithms. In J. J. Grefenstette (Ed.), *Proceedings of the* 1st *International Conference on Genetic Algorithms.*

J. E. Baker (1987). Reducing bias and inefficiency in the selection algorithm. In *Proceedings of the* 2nd *International Conference on Genetic Algorithms*, pp. 14–21.

D. H. Ballard and C. M. Brown (1982). *Computer Vision.* Prentice-Hall.

H. G. Barrow and R. J. Popplestone (1971). Relational descriptions in picture processing. In B. Meltzer and D. Michie (Eds.), *Machine Intelligence*, Volume 6. Edinburgh University Press.

F. C. Bartlett (1932). *Remembering: A Study in Experimental and Social Psychology.* Cambridge University Press.

D. Beasley, D. R. Bull, and R. R. Martin (1993). A sequential niche technique for multimodal function optimisation. *Evolutionary Computation 1*, 101–125.

T. Becker, C. Doran, D. Egedi, B. A. Hockey, S. Kulick, and B. Srinivas (1995). A lexicalised tree adjoining grammar for english. Technical Report IRCS Report 95-03, University of Pennsylvania.

M. Bedau (1995). Three illustrations of artificial life's working hypothesis. *Lecture Notes in Computer Science 899*, 53–68.

S. M. Bhandarkar, Y. Zhang, and W. D. Potter (1994). An edge extraction technique using genetic algorithm-based optimization. *Pattern Recognition 27*, 1159–1180.

W. Bialek and M. Deweese (1995). Random switching and optimal processing in the perception of ambiguous signals. *Physical Review Letters 74*, 3077–3080.

H. J. Bremermann (1958). The evolution of intelligence. The nervous system as a model of its environment. Technical Report 477(17), Deparment of Mathematics, University of Washington.

H. Bunke and J. Csirik (1995). Parametric string edit distance and its application to pattern recognition. *IEEE Transactions on Systems, Man and Cybernetics 25*, 202–206.

F. G. Callari and F. P. Ferrie (1996). Active recognition: Using uncertainty to reduce ambiguity. In *Proceedings of the* 13$^{\text{th}}$ *International Conference on Pattern Recognition*, pp. 925–929.

W. Cedeño, V. R. Vemuri, and T. Slezak (1995). Multiniche crowding in genetic algorithms and its application to the assembly of DNA restriction-fragments. *Evolutionary Computation 2*, 321–345.

M. B. Clowes (1971). On seeing things. *Artificial Intelligence 2*, 79–116.

W. G. Cochran and G. M. Cox (1957). *Experimental Designs* (2$^{\text{nd}}$ ed.). Wiley.

D. Collett (1991). *Modelling Binary Data*. Chapman and Hall.

S. A. Cook (1971). The complexity of therem proving procedures. In *Proceedings of the* 3$^{\text{rd}}$ *ACM Symposium on the Theory of Computing*, pp. 151–158.

A. D. J. Cross (1998). *Optimization Methods for Relational Matching*. Ph. D. thesis, Department of Computer Science, University of York.

A. D. J. Cross, R. C. Wilson, and E. R. Hancock (1997). Inexact graph matching using genetic search. *Pattern Recognition 30*, 953–970.

Y. Davidor (1991). A naturally occurring niche and species phenomenon: The model and first results. In *Proceedings of the* 4th *International Conference on Genetic Algorithms*, pp. 257–263.

L. Davis (1989). Adapting operator probabilities in genetic algorithms. In J. D. Schaffer (Ed.), *Proceedings of the* 3rd *International Conference on Genetic Algorithms*, pp. 61–69.

L. S. Davis (1991). *A Handbook of Genetic Algorithms*. Van Nostrand Reinhold.

T. E. Davis and J. C. Principe (1991). A simulated annealing like convergence theory for the simple genetic algorithm. In *Proceedings of the* 4th *International Conference on Genetic Algorithms*, pp. 174–181.

K. Deb and D. E. Goldberg (1989). An investigation of niche and species formation in genetic function optimization. In *Proceedings of the* 3rd *International Conference on Genetic Algorithms*, pp. 42–50.

K. A. DeJong (1975). *An Analysis of the Behaviour of a Class of Genetic Adaptive Systems*. Ph. D. thesis, Department of Computer and Communication Sciences, University of Michigan.

K. A. DeJong and W. M. Spears (1990). An analysis of the interacting roles of population size and crossover in genetic algorithms. In *Proceedings of the* 1st *Workshop on Parallel Problem Solving from Nature*, pp. 38–47. Springer Verlag.

S. J. Dickinson, A. P. Pentland, and A. Rosenfeld (1992). 3-D shape recovery using distributed aspect matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence 14*, 174–198.

A. C. M. Dumay, R. J. van der Geest, J. J. Gerbrands, E. Jansen, and J. H. C. Reiber (1992). Consistent inexact graph matching applied to labeling coronary segments in arteriograms. In *Proceedings of the* 11th *International Conference on Pattern Recognition*, Volume C, pp. 439–442.

Ö. Egecioglu and M. Ibel (1996). Parallel algorithms for fast computation of normalized edit distances. In *Proceedings of the IEEE Symposium on Parallel and Distributed Processing*, pp. 496–503.

L. J. Eshelman (1991). The CHC adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination. In G. J. E. Rawlins (Ed.), *Foundations of Genetic Algorithms*, Volume 1, pp. 265–283. Morgan Kaufmann.

M. A. Eshera and K. S. Fu (1984). A graph distance measure for image analysis. *IEEE Transactions on Systems, Man and Cybernetics 14*, 398–407.

N. Ezquerra, S. Capell, L. Klein, and P. Duijves (1998). Model-guided labeling of coronary structure. *IEEE Transactions on Medical Imaging 17*, 429–441.

O. D. Faugeras and M. Berthod (1981). Improving consistency and reducing ambiguity in stochastic labeling: An optimisation approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence 3*, 412–424.

J. A. Feldman and D. H. Ballard (1982). Connectionist models and their properties. *Cognitive Science 6*, 205–254.

A. M. Finch, R. C. Wilson, and E. R. Hancock (1998). An energy function and continuous edit process for graph matching. *Neural Computation 10*, 1873–1894.

M. A. Fischler and R. A. Elschlager (1973). The representation and matching of pictorial structures. *IEEE Transactions on Computers 22*, 67–92.

C. Fleurent and J. A. Ferland (1996). Genetic and hybrid algorithms for graph coloring. *Annals of Operations Research 63*, 437–461.

T. C. Fogarty (1989). Varying the probability of mutation in the genetic algorithm. In J. D. Schaffer (Ed.), *Proceedings of the* 3rd *International Conference on Genetic Algorithms*, pp. 104–109.

D. B. Fogel (1994). An introduction to simulated evolutionary optimisation. *IEEE Transactions on Neural Networks 5*, 3–14.

B. Francis, M. Green, and C. Payne (Eds.) (1993). *The GLIM System Release 4 Manual.* Oxford University Press.

A. S. Fraser (1957). Simulation of genetic systems by automatic digital computers. *Australian Journal of Biological Science 10*, 484–491.

K. S. Fu (1983). A step towards unification of syntactic and statistical pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence 5*, 200–205.

M. R. Garey and D. S. Johnson (1979). *Computers and Intractability.* Freeman.

D. Geiger and F. Girosi (1991). Parallel and deterministic algorithms from MRFs: Surface reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence 13*, 401–412.

S. Geman and D. Geman (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine*

Intelligence 6, 721–741.

F. Glover (1989). Tabu search. *ORSA Journal on Computing 1*, 190–206.

D. Goldberg (1989). *Genetic Algorithms in Search, Optimisation and Machine Learning.* Addison-Wesley.

D. E. Goldberg (1987). Simple genetic algorithms and the minimal deceptive problem. In L. Davis (Ed.), *Genetic Algorithms and Simulated Annealing.* Morgan Kaufmann.

D. E. Goldberg (1990). A note on Boltzmann tournament selection for genetic algorithms and population-based simulated annealing. *Complex Systems 4*, 445–460.

D. E. Goldberg and J. Richardson (1987). Genetic algorithms with sharing for multimodal function optimization. In *Proceedings of the* 2^{nd} *International Conference on Genetic Algorithms*, pp. 41–49.

M. Gorges-Schleuter (1991). ASPARAGOS: A parallel genetic algorithm for population genetics. In *Parallelism, Learning, Evolution. Workshop on Evolutionary Models and Strategies*, pp. 407–418.

J. J. Grefenstette (1986). Optimisation of control parameters for genetic algorithms. *IEEE Transactions on Systems, Man and Cybernetics 16*, 122–128.

E. R. Hancock (1994). An optimisation approach to line labelling. In S. Impedovo (Ed.), *Progress in Image Analysis and Processing*, Volume 3, pp. 159–165. World Scientific.

E. R. Hancock and J. Kittler (1990a). Discrete relaxation. *Pattern Recognition 23*, 711–733.

E. R. Hancock and J. Kittler (1990b). Edge labelling using dictionary-based relaxation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 12*, 165–181.

R. M. Haralick, L. S. Davis, and A. Rosenfeld (1978). Reduction operations for constraint satisfaction. *Information Science 14*, 199–219.

R. M. Haralick and G. L. Elliott (1979). Increasing search tree efficiency for constraint satisfaction problems. In *Proceedings of the* 6^{th} *International Joint Conference on Artificial Intelligence*, pp. 356–364.

R. M. Haralick and L. G. Shapiro (1979). The consistent labelling problem: Part 1. *IEEE Transactions on Pattern Analysis and Machine Intelligence 1*, 173–184.

R. M. Haralick and L. G. Shapiro (1980). The consistent labelling problem: Part 2. *IEEE Transactions on Pattern Analysis and Machine Intelligence 2*, 193–203.

W. L. Hays (1994). *Statistics* (5^{th} ed.). Harcourt Brace.

J. H. Holland (1975). *Adaptation in Natural and Artificial Systems.* MIT Press.

R. Horaud and T. Skordas (1989). Stereo correspondence through feature grouping and maximal cliques. *IEEE Transactions on Pattern Analysis and Machine Intelligence 11*, 1168–1180.

K. L. Horlitz and A. O'Leary (1993). Satiation or availability - effects of attention, memory and imagery on the perception of ambiguous figures. *Perception and Psychophysics 53*, 668–681.

D. A. Huffman (1971). Impossible objects as nonsense sentences. In B. Meltzer and D. Michie (Eds.), *Machine Intelligence*, Volume 6, pp. 295–323. Edinburgh University Press.

R. A. Hummel and S. W. Zucker (1983). On the foundations of relaxation labeling processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence 5*, 267–287.

P. Jeavons (1998). On the algebraic structure of combinatorial problems. *Theoretical Computer Science 200*, 185–204.

D. Jefferson, R. Collins, C. Cooper, M. Dyer, M. Flowers, R. Korf, C. Taylor, and A. Wang (1991). Evolution as a theme in artificial life: The genesys/tracker system. In C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen (Eds.), *Artificial Life II.* Addison-Wesley.

M. Jelasity and J. Dombi (1998). GAS, a concept on modeling species in genetic algorithms. *Artificial Intelligence 99*, 1–19.

A. K. Joshi (1985). Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions? In D. R. Dowty, L. Karttunen, and A. M. Zwicky (Eds.), *Natural Language Parsing.* Cambridge University Press.

T. Kanade (1978). A theory of Origami world. Technical Report CMU-CS-78-144, Computer Science Department, Carnegie Mellon University.

N. Kawabata (1978). Visual fixation points and depth perception. *Vision Research 18*, 853–854.

N. Kawabata and T. Mori (1992). Disambiguating ambiguous figures by a model of selective attention. *Biological Cybernetics 67*, 417–425.

A. H. Kawamoto (1993). Nonlinear dynamics in the resolution of lexical ambiguity: A parallel distributed processing account. *Journal of Memory and Language 32*, 474–516.

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi (1983). Optimisation by simulated annealing. *Science 220*, 671–680.

L. M. Kirousis (1990). Effectively labeling planar projections of polyhedra. *IEEE Transactions on Pattern Analysis and Machine Intelligence 12*, 123–130.

J. Kittler and J. Illingworth (1985). Relaxation labelling algorithms - a review. *Image and Vision Computing 3*, 206–216.

J. Koenderink and A. van Doorn (1979). The internal representation of solid shape with respect to vision. *Biological Cybernetics 32*, 211–216.

J. R. Koza (1992). *Genetic Programming.* MIT Press.

V. Levenshtein (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics – Doklady 10*, 707–710.

H. Lipson and M. Shpitalni (1996). Optimization-based reconstruction of a 3D object from a single freehand line drawing. *CAD 28*, 651–663.

S. A. Lloyd (1983). An optimisation approach to relaxation labelling algorithms. *Image and Vision Computing 1*, 85–91.

S. J. Louis and G. J. E. Rawlins (1993). Syntactic analysis of convergence in genetic algorithms. In D. Whitley (Ed.), *Foundations of Genetic Algorithms*, Volume 2, pp. 141–151. Morgan Kaufmann.

A. K. Mackworth (1977). Consistency in networks of relations. *Artificial Intelligence 8*, 99–118.

A. K. Mackworth and E. C. Freuder (1985). The complexity of some polynomial network consistency algorithms. *Artificial Intelligence 25*, 65–74.

S. W. Mahfoud and D. E. Goldberg (1995). Parallel recombinative simulated annealing: A genetic algorithm. *Parallel Computing 21*, 1–28.

J. Malik (1987). Interpreting line drawings of curved objects. *International Journal of Computer Vision 1*, 73–103.

D. Marr (1982). *Vision.* Freeman.

A. Marzal and E. Vidal (1993). Computation of normalized edit distance and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence 15*, 926–932.

F. Masulli, M. Riani, and E. Simonotto (1990). Neural network models of perceptual alternation of ambiguous patterns. In V. Cantoni, L. P. Cordella, S. Levialdi, and G. S. di Baja (Eds.), *Progress in Image Analysis*, pp. 751–758. World Scientific.

M. L. Mauldin (1984). Maintaining diversity in genetic search. In *Proceedings of the National Conference on Artificial Intelligence*, pp. 247–250.

P. McCullagh and J. A. Nelder (1989). *Generalised Linear Models* (2nd ed.). Chapman and Hall.

M. Meila and M. I. Jordan (1997). Triangulation by continuous embedding. In *Advances in Neural Information Processing Systems 9*, pp. 557–563.

N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller (1953). Equation of state calculations by fast computing machines. *Journal of Chemical Physics 21*, 1087–1092.

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

B. L. Miller and D. E. Goldberg (1997). Genetic algorithms, selection schemes, and the varying effects of noise. *Evolutionary Computation 4*, 113–131.

M. Minsky (1975). A framework for representing knowledge. In P. H. Winston (Ed.), *The Psychology of Computer Vision*. McGraw-Hill.

M. Mitchell (1996). *An Introduction to Genetic Algorithms*. MIT Press.

J. L. Mohammed, R. A. Hummel, and S. W. Zucker (1983). A gradient projection algorithm for relaxation methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence 5*, 330–332.

H. Mühlenbein (1994). Genetic algorithms. In E. Aarts and J. K. Lenstra (Eds.), *Local Search in Combinatorial Optimisation*. Wiley.

H. Mühlenbein and D. Schlierkamp-Voosen (1995). Analysis of selection, mutation and recombination in genetic algorithms. *Lecture Notes in Computer Science 899*, 142–168.

R. Myers and E. R. Hancock (1997a). Genetic algorithm parameters for line labelling. *Pattern Recognition Letters 18*, 1363–1371.

R. Myers and E. R. Hancock (1997b). Genetic algorithms for ambiguous labelling problems. *Lecture Notes in Computer Science (EMMCVPR 97) 1223*, 345–360.

R. Myers and E. R. Hancock (1998). Genetic algorithms for structural editing. *Lecture Notes in Computer Science (SSPR 98) 1451*, 159–168.

B. Nudel (1983). Consistent labelling problems and their algorithms. *Artificial Intelligence 21*, 135–178.

P. Parodi and G. Piccioli (1996). 3D shape reconstruction by using vanishing points. *IEEE Transactions on Pattern Analysis and Machine Intelligence 18*, 211–217.

W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery (1992). *Numerical Recipes in C* (2^{nd} ed.). Cambridge University Press.

A. Prügel-Bennett and J. L. Shapiro (1994). An analysis of genetic algorithms using statistical physics. *Physical Review Letters 72*, 1305–1309.

X. Qi and F. Palmieri (1994a). Theoretical analysis of evolutionary algorithms with an infinite population size in continuous space, part 1: Basic properties of selection and mutation. *IEEE Transactions on Neural Networks 5*, 102–119.

X. Qi and F. Palmieri (1994b). Theoretical analysis of evolutionary algorithms with an infinite population size in continuous space, part 2: Analysis of the diversification rôle of the crossover. *IEEE Transactions on Neural Networks 5*, 120–129.

C. Qin and J. Y. S. Luh (1994). Ambiguity reduction by relaxation labelling. *Pattern Recognition 27*, 165–180.

I. Rechenberg (1973). *Evolutionsstrategie - Optimierung Technischer Systeme nach Prinzipien der biologischen Information.* Fromman Verlag.

J. Reed, R. Toombs, and N. A. Barricelli (1967). Simulation of biological evolution and machine learning. *Journal of Theoretical Biology 17*, 319–342.

M. Riani and E. Simonotto (1994). Stochastic resonance in the perceptyal interpretation of ambiguous figures - a neural network model. *Physical Review Letters 72*, 3120–3123.

E. S. Ristad and P. N. Yianilos (1998). Learning string edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence 20*, 522–532.

A. Rosenfeld, R. A. Hummel, and S. W. Zucker (1976). Scene labelling by relaxation operations. *IEEE Transactions on Systems, Man and Cybernetics 6*, 420–433.

G. Rudolph (1994). Convergence analysis of canonical genetic algorithms. *IEEE Transactions on Neural Networks 5*, 96–101.

H. Saito and M. Mori (1995). Application of genetic algorithms to stereo matching of images. *Pattern Recognition Letters 16*, 815–822.

A. Sanfeliu and K. S. Fu (1983). A distance measure between attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man and Cybernetics 13*, 353–362.

R. Sastry, N. Ranganathan, and K. Remedios (1995). CASM: A VLSI chip for approximate string matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence 17*, 824–830.

J. D. Schaffer, R. A. Caruna, L. J. Eshelman, and R. Das (1989). A study of control parameters affecting online performance of genetic algorithms for function optimisation. In *Proceedings of the* 3rd *International Conference on Genetic Algorithms*, pp. 51–60.

N. N. Schraudolf and R. K. Belew (1992). Dynamic parameter encoding for genetic algorithms. *Machine Learning 9*, 9–21.

H. Schwefel (1981). *Numerical Optimization of Computer Models*. Wiley.

C. E. Shannon (1948). A mathematical theory of communication. *Bell System Techincal Journal 27*, 379–423.

L. G. Shaprio and R. M. Haralick (1981). Structural descriptions and inexact matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence 3*, 504–519.

L. G. Shaprio and R. M. Haralick (1985). A metric for comparing relational descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence 7*, 90–94.

D. Shasha and K. Zhang (1989). Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal of Computing 18*, 1245–1262.

J. R. Shewchuk (1996). Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator. In *Proceedings of the* 1st *Workshop on Applied Computational Geometry*, pp. 124–133.

D. B. Skillicorn (1996). A parallel tree difference algorithm. *Information Processing Letters 60*, 231–235.

R. E. Smith, S. Forrest, and A. S. Perelson (1993). Searching for diverse, cooperative populations with genetic algorithms. *Evolutionary Computation 1*, 127–149.

S. M. Smith and J. M. Brady (1995). SUSAN - a new approach to low level image processing. Technical Report TR95SMS1b, Defence Research Agency, United Kingdom.

G. W. Snedecor and W. G. Cochran (1980). *Statistical Methods* (7th ed.). Iowa State University Press.

J. F. Sowa (1984). *Conceptual Structures*. Addison-Wesley.

M. Srinivas and L. M. Patnaik (1994a). Adaptive probabilities of crossover and mutation

in genetic algorithms. *IEEE Transactions on Systems, Man and Cybernetics 4*, 656–667.

M. Srinivas and L. M. Patnaik (1994b). Genetic algorithms: A survey. *IEEE Computer 27*, 17–26.

M. Srinivas and L. M. Patnaik (1996). On modeling genetic algorithms for functions of unitation. *IEEE Transactions on Systems, Man and Cybernetics 26*, 809–821.

K. Sugihara (1978). Picture language for skeletal polyhedra. *Computer Graphics and Image Processing 8*, 382–405.

G. Syswerda (1989). Uniform crossover in genetic algorithms. In *Proceedings of the* 3rd *International Conference on Genetic Algorithms*, pp. 2–9.

J. C. Trueswell, M. K. Tanenhaus, and S. M. Garnsey (1994). Semantic influences on parsing: Use of thematic rôle information in syntactic disambiguation. *Journal of Memory and Language 33*, 285–318.

P. W. M. Tsang (1997). A genetic algorithm for affine invariant recognition of object shapes from broken boundaries. *Pattern Recognition Letters 18*, 631–639.

M. Tuceryan and T. Chorzempa (1991). Relative sensitivity of a family of closest-point graphs in computer vision applications. *Pattern Recognition 24*, 361–373.

E. Ukkonen (1985). Algorithms for approximate string matching. *Information and Control 64*, 100–118.

J. R. Ullman (1976). An algorithm for subgraph isomorphism. *Journal of the ACM 23*, 31–42.

M. D. Vose (1995). Modelling simple genetic algorithms. *Evolutionary Computation 3*, 453–472.

R. A. Wagner and M. J. Fischer (1974). The string-to-string correction problem. *Journal of the ACM 21*, 168–173.

D. Waltz (1975). Understanding line drawings of scenes with shadows. In P. H. Winston (Ed.), *The Psychology of Computer Vision*, pp. 19–91. McGraw-Hill.

D. Whitley (1989). The GENITOR algorithm and selection pressure: Why rank-based allocation of reproductive trials is best. In *Proceedings of the* 3rd *International Conference on Genetic Algorithms*, pp. 116–121.

D. Whitley (1996). Modeling hybrid genetic algorithms. In G. Winter, J. Periaux, M. Galan, and P. Cuesta (Eds.), *Genetic Algorithms in Engineering and Computer*

Science. Wiley.

D. Whitley, R. Beveridge, C. Graves, and K. Mathias (1995). Test driving three 1995 genetic algorithms: New test functions and geometric matching. *Journal of Heuristics 1*, 77–104.

D. Whitley and D. Starkweather (1990). GENITOR-II: A distributed genetic algorithm. *Journal of Experimental and Theoretical Artificial Intelligence 2*, 189–214.

G. N. Wilkinson and C. E. Rogers (1973). Symbolic description of factorial models for analysis of variance. *Applied Statistics 22*, 392–399.

L. R. Williams (1992). Topological reconstruction of a smooth manifold-solid from its occluding contour. In *Proceedings of the* 2nd *European Conference on Computer Vision*, pp. 36–47.

M. L. Williams and E. R. Hancock (1999). Recombination strategies for evolutionary graph matching. To appear.

R. C. Wilson (1995). *Inexact Graph Matching Using Symbolic Constraints*. Ph. D. thesis, Department of Computer Science, University of York.

R. C. Wilson, A. D. J. Cross, and E. R. Hancock (1998). Graph matching with active triangulations. *Computer Vision and Image Understanding 72*, 21–38.

R. C. Wilson and E. R. Hancock (1997). Structural matching by discrete relaxation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 19*, 634–648.

P. P. C. Yip and Y. H. Pao (1995). Combinatorial optimisation with use of guided evolutionary simulated annealing. *IEEE Transactions on Neural Networks 6*, 290–295.

A. L. Yuille and J. J. Kosowsky (1994). Statistical physics algorithms that converge. *Neural Computation 6*, 341–356.

K. Zhang (1996). A constrained edit distance between unordered labeled trees. *Algorithmica 15*, 205–222.