# No Optimisation Without Representation: A Knowledge Based Systems View of Evolutionary/Neighbourhood Search Optimisation

**Andrew Laurence Tuson**

Ph.D.
University of Edinburgh
1999

# Abstract

In recent years, research into 'neighbourhood search' optimisation techniques such as simulated annealing, tabu search, and evolutionary algorithms has increased apace, resulting in a number of useful heuristic solution procedures for real-world and research combinatorial and function optimisation problems. Unfortunately, their selection and design remains a somewhat *ad hoc* procedure and very much an art. Needless to say, this shortcoming presents real difficulties for the future development and deployment of these methods.

This thesis presents work aimed at resolving this issue of principled optimiser design. Driven by the needs of both the end-user and designer, and their knowledge of the problem domain and the search dynamics of these techniques, a semi-formal, structured, design methodology that makes full use of the available knowledge will be proposed, justified, and evaluated. This methodology is centred around a **Knowledge Based System** (KBS) view of neighbourhood search with a number of well-defined knowledge sources that relate to specific hypotheses about the problem domain. This viewpoint is complemented by a number of **design heuristics** that suggest a structured series of hillclimbing experiments which allow these results to be empirically evaluated and then transferred to other optimisation techniques if desired.

First of all, this thesis reviews the techniques under consideration. The case for the exploitation of problem-specific knowledge in optimiser design is then made. Optimiser knowledge is shown to be derived from either the problem domain theory, or the optimiser search dynamics theory. From this, it will be argued that the design process should be primarily driven by the problem domain theory knowledge as this makes best use of the available knowledge and results in a system whose behaviour is more likely to be justifiable to the end-user.

The encoding and neighbourhood operators are shown to embody the main source of problem domain knowledge, and it will be shown how forma analysis can be used to formalise the hypotheses about the problem domain that they represent. Therefore it should be possible for the designer to experimentally evaluate hypotheses about the problem domain. To this end, proposed design heuristics that allow the transfer of results across optimisers based on a common hillclimbing class, and that can be used to inform the choice of evolutionary algorithm recombination operators, will be justified. In fact, the above approach bears some similarity to that of KBS design. Additional knowledge sources and roles will therefore be described and discussed, and it will be shown how forma analysis again plays a key part in their formalisation. Design heuristics for many of these knowledge sources will then be proposed and justified.

This methodology will be evaluated by testing the validity of the proposed design heuristics in the context of two sequencing case studies. The first case study is a well-studied problem from operational research, the flowshop sequencing problem, which will provide a through test of many of the design heuristics proposed here. Also, an idle-time move preference heuristic will be proposed and demonstrated on both directed mutation and candidate list methods.

The second case study applies the above methodology to design a prototype system for resource redistribution in the developing world, a problem that can be modelled as a very large transportation problem with non-linear constraints and objective function. The system, combining neighbourhood search with a constructive algorithm which reformulates the problem to one of sequencing, was able to produce feasible shipment plans for problems derived from data from the World Health Organisation's TB programme in China that are much larger than those problems tackled by the current 'state-of-the-art' for transportation problems.

## Acknowledgements

First of all, I must thank my supervisors Dr Peter Ross and Mr Tim Duncan for their support and advice during the course of the PhD. I would also like to thank Mr Richard Wheeler for providing the data and technical information for the resource redistribution problem, and my current employers (City University) for their assistance and understanding.

I should also especially thank my office-mate Josh Singer for his invaluable support and tolerance during my write-up and for knowing exactly when a coffee was required!

I thank my friends for their support, encouragement, beer, jokes, and for making me forget about work (!): Keith and Kaska Anderson, Peter Baron, Rob Briggs, Daren Croxford, Mike Fairbank, Dave Gibb, Damian Hackett, Mark Jones, Jeroen Keppens, Marion Klein, Spencer Lewis, Chris Raymond, Hanson Schmidt-Cornelius, Elaine Staniforth, and Tony Thompstone. I also thank my flatmates for their understanding and occasional hygiene!...:-)

There are a number of people I would like to thank on a professional level, and for their advice and support on both general and specific matters: Francesco Arci, Howard Beck, Dave Corne, Chris Gathercole, Andy Harrison, Emma Hart, Riccardo Poli, Steve Polyak, Derek Sleeman, Patrick Surry, Brian Turton, City's MOLE group (and Peter Smith in particular).

I would like to take this opportunity to thank the MSc and undergraduate students I supervised during my PhD and who proved to be among some of my most perceptive critics: Nick Altmann, Peter Baron, Javier Gomez-Marin-Blazque, Shigeaki Nakata, George Papadopoulos, Somnuk Phon-Amnuaisuk, Ketan Patel, Mike Ratford, Sergio Ramos, and Chi Kit Tsang. I should not also forget to include the MSc and undergraduate students I tutored while at DAI.

In the above context, gratitude is also due to the MSc/AI4 project co-supervisors from whom I learnt much about the process and management of research: Andrew Coulson, Chris Malcolm, Bob Fisher, Chris Mellish, Joel Malard, Frank Mill, Julian Miller, Qiang Shen, Andy Sherlock, Alan Smaill, Shane Sturrock, Henry Thompson, Peter Thompson, and Geraint Wiggins.

Special thanks is due in two cases. I am first in debt to Hugh Cartwright, my Chemistry Part II project supervisor at Oxford University, who first introduced me to AI, and genetic algorithms in particular. His high standard of supervision and willingness to discuss ideas were instrumental in forming my current interest in AI, and are qualities that I still strive to emulate. In short, if I hadn't have met Hugh it would be unlikely that I would be doing anything nearly so interesting, and instead I would be trapped in a lifetime of chartered accountancy!

Secondly, thanks are due to the DAI secretaries, for providing a helpful, efficient, and friendly atmosphere. Thanks especially to Jean Bunten, for providing me with such a detailed education of Edinburgh's pubs that I could probably write another PhD thesis about them!

Finally, I would like to thank the Engineering and Physical Sciences Research Council (EPSRC) for their financial support via a research studentship (95306458).


### Dedication

I would like to dedicate this thesis to my family: Margaret, Laurence, and Karen Tuson for their unceasing devotion, support, and encouragement over the years of my education and especially during the difficult times.

# Declaration

I hereby declare that I composed this thesis entirely myself and that it describes my own research.

A L Tuson
Edinburgh
November 19, 1999

# Contents

# List of Figures

# List of Algorithms

# Chapter 1

# Introduction

This thesis motivates, describes, justifies, and evaluates a methodology for the *principled* design of a class of heuristic optimisation techniques known as neighbourhood search.

This chapter will provide an brief overview of the search paradigm that is the subject of this thesis and its wider context, and then introduce the main topic of this research — namely *how* to design these optimisers effectively. From this, the requirements of a desired design methodology will be stated and expanded upon. Finally, an overview of the approach taken in this thesis will be given and the structure of the thesis outlined.

## 1.1 Introduction

Suppose that we are presented with a **Combinatorial Optimisation Problem** (COP) which has the form:

- A space of possible solutions, $S$;
- A description of the solutions in $S$ in terms of *symbolic* **decision variables**;
- Possibly constraints on valid solutions $S$ specified in terms of the decision variables;
- A 'quality' measure for each solution, $quality(s)$ where $s \in S$, again in terms of the decision variables;

where the objective is to find a solution $s$ such that $quality(s)$ is maximised and the constraints satisfied. Such problems are usually NP-hard [Garey & Johnson 79] and therefore, for a problem of large enough size, the time taken to find an optimal solution by exact methods will become prohibitive. Fortunately, in practice a ‘good enough’ answer in the time available is all that is required, which opens up the possibility of using **heuristic** methods that do not guarantee optimality, but work well in practice.

A class of combinatorial optimisation techniques often termed **meta-heuristics** have been developed to tackle difficult COPs. They are distinct from conventional heuristics in that they are not tied to a single problem, but are in fact *templates* from which problem-specific optimisers can be derived.

The early parts of this chapter will first motivate the need for heuristic methods, and then introduce the paradigm of the meta-heuristic techniques of interest (known as neighbourhood search) and outline links between it and other search techniques. After this, the subject of this thesis will be discussed and the thesis structure given.

## 1.2 A (Quick) Introduction to P and NP

In combinatorial optimisation, algorithms are classed as **efficient** if they can solve every instance of a COP to optimality in polynomial time. Following from this, a problem can termed **hard**, or **intractable** if efficient algorithms for solving it do not exist. Therefore a central question in combinatorial optimisation research has been to determine whether an efficient optimisation algorithm can be constructed for a given problem.

The answer to this question lies in the fact that combinatorial optimisation problems can be viewed as being derived from underlying **decision problems**. For example, the optimisation version of the travelling salesman problem (TSP) which is to find a tour such that the distance travelled is minimised, can be cast as the following decision problem ‘does a tour exist that has length less than $L$?’. Therefore the question of whether an efficient optimiser can be constructed rests upon whether a worst-case polynomial-time recogniser can be constructed for its corresponding decision problem. This is because a polynomial-time optimiser can be used to solve the decision problem in polynomial time (and vice-versa).

Work in the 1970’s on computational complexity by [Cook 71] showed that such decision problems can be solved in (worst-case) polynomial time with a **non-deterministic Turing**

**Machine**. Such machines can be thought of as standard Turing Machines, with the addition of an **oracle** that, when given a choice point, takes the correct path. These decision problems are said to be in the complexity class $\mathbf{NP}$. Problems that can also be solved in polynomial time on a standard Turing Machine (i.e. those problems that we can build an oracle for) are said to be in the complexity class $\mathbf{P}$. In other words, decision problems in the set $\mathbf{NP} \setminus \mathbf{P}$ will have exponential-time worst case performance.

Now given a problem $P$, if we can **reduce** (convert) *every* problem in $\mathbf{NP}$ to $P$ in *polynomial* time, then it is said to be **NP-hard** — in other words, it is as hard as a problem in $\mathbf{NP}$. If the problem $P$ is itself also in $\mathbf{NP}$, then it is said to be **NP-complete**. It turns out that many optimisation problems that are of real-world relevance, such as the TSP, are NP-hard, and therefore if the set $\mathbf{NP} \setminus \mathbf{P}$ is non-empty, then no efficient optimiser is possible for these problems on a conventional Turing Machine.

Now the question is whether the set $\mathbf{NP} \setminus \mathbf{P}$ is empty, or in other words whether $\mathbf{P} = \mathbf{NP}$? Though this question has not been unequivocally answered, the common belief is that $\mathbf{P} \neq \mathbf{NP}$, as the vast amounts of man-effort spent in finding a polynomial-time algorithm for these problems have so far failed. Therefore in, practical terms, it is considered unlikely that an efficient exact method for solving many optimisation problems of interest to optimality will ever be found, and a change of approach is therefore required.

Interestingly, given the above, one consequence of the reducibility requirement is that if a polynomial-time algorithm could be constructed to solve one NP-complete problem, then it could be used to solve all of the other NP-complete problems. Therefore in order to show that $\mathbf{P} = \mathbf{NP}$ all that is required is to devise a *provably* polynomial-time recogniser for *one* NP-complete problem.

The discussion above is only a brief and informal introduction to these issues. For a more formal and detailed coverage of these issues, the reader is directed to [Garey & Johnson 79, Papadimitriou & Steiglitz 82] for further reading.

## 1.3 The Case for Heuristics

It should be noted, however, that the above definition of efficiency is not entirely appropriate to the demands of the real world. First of all, it concerns itself only with worst-case performance, whereas it is usually the average-case performance that practitioners are interested in. Second, it does not necessarily follow from the above that *all* instances of an NP-hard problem have to take exponential time to solve. Third, the optimal solution is not necessarily required — a 'good enough' solution in the time available will often suffice.

Therefore *one* approach to dealing with this problem is to use heuristic methods which work well in practice[1]. This is sensible because it is usually possible to construct, using our knowledge of the structure of the problem, an algorithm that can give high-quality and possibly even optimal solutions to an NP-hard problem in a reasonable time in most cases. In fact, classical heuristics have a long and distinguished history (for example see [Polya 57]), and the success and size of this subfield of optimisation can be judged by reviews such as that of [Stewart *et al.* 94], which list over 900 published papers on heuristic search up till 1992 in the AI literature alone! The definition of a heuristic search method used here is taken from [Reeves 93b]:

> "A heuristic is a technique which seeks good (i.e. near optimal) solutions at a reasonable computational cost without being able to guarantee either feasibility or optimality, or even in many cases to state how close to optimality a particular feasible solution is."

Despite the lack of guaranteed optimality, there are a number of reasons why heuristic methods are applicable, and have been the basis of useful applications in practice. The reasons given below are adapted from [Osman 95].

- **Tractability**. As noted above, many problems cannot be solved in a reasonable time by exact methods due to excessive computational requirements (ie. time and memory).
- **Applicability**. Optimisation is in fact performed upon *models* of problems, which may have constraints and objectives that cannot be modelled effectively by exact methods. In

[1] Though it should be noted that exact methods can also be made efficient enough in many cases as well.

the worst case, this could lead to an exact method providing an optimal solution to the wrong problem! Heuristics make fewer assumptions about the form that the problem model can take and are therefore applicable to a wider range of problems.

- **Accessibility**. Heuristics are often simpler for the decision maker to understand, when compared to exact methods which are often more mathematically involved. This increases their chances of implementation in the field.

- **Flexibility**. Heuristics are flexible, both in terms of their generality and their ability to be modified when problem conditions change in the external environment.

- **Rapid Development**. As a result of the above, less effort is often required, *in practice*, to develop heuristic optimisers that can obtain high-quality solutions, than to develop a corresponding exact method.

It should be noted that [Osman 95] may have possibly overstated some of the above advantages somewhat; for example exact methods can be devised that are easily accessible to decision makers in the field. So it is somewhat unclear as to the extent that the above advantages of heuristics listed above are due to their inherent nature, or the emphasis on empirical design that they engender. For instance, [Osman 95] notes that there are two additional drawbacks to heuristic methods: first that they are hard to analyse mathematically, and second that there is little guidance on their design.

Even so [Osman 95] goes on to note that (in this case) operational research practitioners have been ‘voting with their feet’ and their use has become common — in fact [Reeves 93b] remarks that these methods are of increasing importance, and not just ‘methods of last resort’ or an ‘admission of defeat’. In addition, heuristics can be used in conjunction with exact methods — an example would be to provide a good upper bound with which to prune the search for a branch and bound procedure.

## 1.4 The Neighbourhood Search Paradigm

An applicable and simple-minded, although sometimes effective, heuristic approach for any given combinatorial optimisation problem would be to employ some form of hillclimbing. This requires the following decisions to be made:

1. Find a suitable **encoding scheme** for the candidate solutions in $S$.

2. A **quality measure** for each solution (ie. $quality(s)$).

3. A method of modifying an encoded solution to give another solution (a **move operator**).

There is usually more than one possible solution (valid or invalid) that applying the move operator on a given solution can produce; therefore we define a **neighbourhood**, $N(s, m)$, as the set of solutions in $S$ that are produced by applying the move operator, $m$, on a solution $s$. Once the above have been defined, the general algorithm is quite straightforward and is described below:

1. Generate an initial solution (usually at random) and evaluate it;

2. Apply and evaluate move(s) in the neighbourhood of the current solution and apply an **acceptance criterion** to decide whether to use the new solution(s) thus generated;

3. Go back to step 2 until a **termination criterion** is reached.

The termination condition is merely when the user would like to stop the search, examples would include when a certain amount of CPU time has elapsed, or when a solution of a certain quality has been found. The acceptance criterion determines whether a new solution generated by a move operator replaces the current solution, and it introduces a bias towards better solutions in the search. Acceptance criteria, and thus hillclimbers, can be broadly classified as follows.

- **Any-Ascent** hillclimbers (AHC) accept moves that give solutions, $s_{new}$, that have better or equal quality than the current solution, $s_{curr}$ (ie. accept when $quality(s_{new}) \geq quality(s_{curr})$). One common variant of any-ascent hillclimbing is **stochastic** hillclimbing where moves are tried in a randomised order (rather than, say, a fixed lexicographic order).

- **First-ascent** (FAHC) hillclimbers, operate similarly, but take the first *improvement* in quality found (ie. accept if $quality(s_{new}) > quality(s_{curr})$).

- **Steepest-ascent** (SAHC) hillclimbers, systematically evaluate the entire neighbourhood of the current solution and accept the most improving move available (ie. accept if $quality(s_{best}) \geq quality(s_{curr})$ with the condition that $s_{best}$ is the neighbourhood's best solution, $quality(s_{best}) \geq quality(s_{any})$ $(s_{any} \in N(s_{curr}, m))$).

The bias introduced by the acceptance criterion, though necessary, can lead to problems. An implicit assumption is made that the **landscape** [Jones 95] or **cost surface** [Boese 96], which is a graph induced by the move operator and quality function that connects solutions that are accessible to each other by a move operator, is correlated in such a way as to lead the hillclimber into a region of the landscape with high-quality solutions. Deviations from this ideal can lead to the hillclimber being **deceived** (the correlations lead the hillclimber away from the optimum), or stuck in **local optima**.

The most common way of visualising the above the explanatory purposes is to make an analogy with **line search** [Woodruff 94] — a representation of a discrete combinatorial function as a continuous function in one dimension.

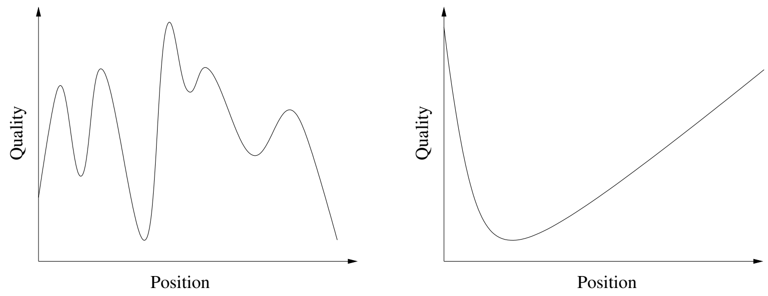


Figure 1.1: Examples of Local Optima (left) and Deception (right)

Therefore, Figure 1.1 above shows a largely uncorrelated landscape with many local optima on the left-hand diagram — for most starting positions a hillclimber would be lead into a local optimum. The diagram on the right of Figure 1.1 shows an example of a deceptive landscape. There are two optima (one global, on the left, and one local), however for most points on the line, the search will be lead *away* from the global optimum as although the landscape is correlated, the basin of attraction for the local optimum is much larger. In general, each of the different techniques described later in this thesis (Chapter 2) is resistant to different forms of these deviations from the ideal landscape (no local optima or deception) to different degrees, the art is thus in selecting the most suitable optimiser for the problem at hand.

Finally, it is worth noting that the potential difficulties outlined above do not necessarily arise in practice, as the assumptions made by hillclimbing are often not too far from the reality if a suitable move operator is used. This is reflected in the interest generated by GSAT [Selman *et al.* 92] (a contraction of Greedy SATisfiability), which is a steepest-ascent hillclimber, which restarts randomly if a local optimum is found, for the boolean 3-CNF satisfiability problem. In [Selman *et al.* 92], it was found that GSAT was able to find optimal solutions (ie. valid truth assignments to the 3-CNF satisfiability problem) for large hard (ie. phase transition) problems which proved too computationally demanding for exact methods to solve.

## 1.5 So What's The Problem?

> "If you really know me, you would know that all your assumptions are wrong."
> — Ann Muir Thomas

Even a casual reading of Chapter 2 later should suffice to convince the reader that the designer of a neighbourhood search optimiser has quite a range of techniques being presented before him, which raises the question of how to design an effective optimiser for a given problem. Until rather recently, a somewhat 'technique-centric' view has been adopted in some quarters in the optimisation community, with the emphasis of research being placed upon search algorithms and their dynamics as 'black-box' (general purpose) optimisers. Recently, theoretical work [Wolpert & Macready 95] has argued convincingly that no optimiser can outperform another over the entire space of problems, thus viewing black-box optimisers as the optimisation communities' philosopher's stone.

Unfortunately, it is not enough just to say that domain knowledge is important. Although such a statement is correct, it is also of little practical use as little or no guidelines are given on *how* to approach a problem, extract the salient features, and to map these onto the optimisation algorithm. For instance [Papadimitriou & Steiglitz 82] notes that:

> "... the design of effective local search algorithms has been, and remains, very much an art."

The above is a real problem for the future acceptance for these techniques as it not only reduces the likelihood of an effective optimiser being constructed, but without a principled **design rationale** it is difficult to persuade potential users to commission, let alone adopt, neighbourhood search based optimisation solutions to their problems. Furthermore, practical computer systems are not set in stone and may need modification — without a principled optimiser design, it may be more difficult to identify and implement the required changes.

In addition, each technique is often considered separately, which raises concerns over whether there is an efficient transfer of ideas between the techniques, and whether issues common to all (where they may arise) are being re-investigated by their separate communities. Therefore the need for clear design guidelines should be obvious. Unfortunately, such design guidelines that currently exist, with representative citations, are along the lines of the following.

- Understand the structure of the problem being solved.
- Use encodings and neighbourhood that are 'natural', ie. appropriate to the problem [Davis 89, Michalewicz 92].
- Use large neighbourhoods [Papadimitriou & Steiglitz 82].
- Use small neighbourhoods [Glover & Laguna 97].
- Hybridise with other techniques [Davis 89].
- Exploit biological metaphors [Goldberg 95, Schwefel 97]

Therefore, though *most* of the points above are essentially correct (in that they have some basis in fact), they can be said to be of little use in practice. This can be seen by taking each of the above guidelines in turn. The first is simply obvious (why would anyone do anything else?). The second guideline suffers from two objections: the first is that at face value the idea of a natural encoding can be considered to be a tautology, as any encoding that works well on a problem can be considered appropriate (in short, the concept of 'appropriate' needs to be defined more usefully); the second is that it does not address the more useful issue of *what* it is about a particular encoding or operator that makes it effective.

The next two guidelines are contradictory, which suggests that either one of them is wrong or the issue has been oversimplified somewhat; though to be fair the studies were each conducted with differing problems, aims, and approaches which may account for this. The next guideline concerning hybridisation is laudable and has been shown to work on a number of occasions. However, at face value, this again suffers from similar objections to the natural encoding guideline in that it does not address the issue of *how* the designer usefully combines the techniques at his disposal (remember that hybridisation will inevitably increase the number of design options).

Finally, though evolutionary algorithms have proved to be useful it should be emphasised that the processes of natural evolution and optimisation have different aims. Biological evolution is a behaviour **emergent** from the action of individual replicators (organisms) in an environment, and therefore to say that such systems have a definite idea of 'quality' or 'progress' is simply wrong. In fact the use of evolution to justify racism and eugenics is based on this fallacy. However, this is not the case in optimisation processes as in this case we want to *impose* a well-specified idea of solution quality. Therefore though the two processes are similar enough for biological metaphors to suggest useful ideas, they are *not* equivalent — so creating a detailed simulation of natural evolutionary processes will not necessarily lead to an effective optimiser.

Moreover, from the above arguments, it can be seen that there is a 'methodological gap' between the research literature, textbooks, and the actual application of these techniques, which urgently needs to be addressed if neighbourhood search techniques are to achieve their full potential for tackling difficult real-world problems. It is this need for a principled design methodology that is the focus of this thesis.

### 1.5.1 The Desired Methodology

Before such a methodology is proposed, it is sensible to first outline the desirable features that such a methodology should have/provide:

- **Accessablity to non-specialists in the field**. From an engineering viewpoint, it would be highly desirable that such an approach can easily be mapped onto concepts that the engineer (or end-user) who is tackling the problem, who will not necessarily be an optimisation expert, can understand — *this is motivated by the view that optimisation is far too useful to be left solely to the optimisation experts!*

- **Emphasis of the common features of the techniques**. To allow a free transfer of ideas between techniques and to identify which issues can be considered independently of a particular optimiser.

- **A systematic procedure for tackling real problems**. In practice, it is desirable that the engineer has some guidelines on how to approach a problem and for which features of the problem are likely to be useful.

- **A good grounding in the current theory**. As a principled methodology is a good methodology.

- **A useful organisation of the literature**. A useful methodology should provide a organisation of the literature that assists in the engineer's task of designing a useful optimiser.

- **Pedagogical utility**. This is related to the above point, as an organised body of knowledge is often easier to teach.

- **Directions for further research in both the theory and practice of these techniques**. Once a body of knowledge has been organised, then it becomes much simpler to both identify outstanding issues, link the theoretical work to actual practice, and to formulate a research programme to investigate these issues.

The task is now to design such a methodology. This task is the subject of this thesis.

## 1.6 Thesis Overview: The Approach Taken Here

To address the above requirements, a structured methodology for achieving them will be proposed. However, first of all it will be necessary to both introduce the techniques of interest, and show their diversity. Chapter 2 (Optimisation with Hillclimbing on Steroids) will therefore provide the detailed review required. This will give the reader the background for the remainder of the thesis.

**NOTE**: In writing this thesis I have attempted to take into account its potential relevance to the Operational Research (OR), Evolutionary Computation (EC), and Artificial Intelligence (AI) communities. To this end, I have assumed that the reader possesses the minimum common background of the three communities. Therefore, I hope that the more knowledgeable reader will understand if they come across sections that cover material they are familiar with.

### 1.6.1 Chapter 3: Paying for Lunch in Neighbourhood Search

Chapter 3, commences the search for a design methodology proper. First, the arguments against general purpose, or 'black-box', optimisers will be reviewed and discussed — a view long held by a significant number of practitioners involved with real-world problems [Davis 89, Michalewicz 92]. The case for a problem (as opposed to a technique) orientated approach to optimiser design will then be advocated. In other words, the problem should be first examined, and domain knowledge provided to the optimiser. This is strongly related to the important practical question of which choices should be made when designing a neighbourhood search optimiser. As noted above, we have many choices to make such as the solution encoding scheme, the neighbourhood operator, as well as the choice of optimisation algorithm and its parameters. This process can be thought of as a search through the space of algorithm behaviours.

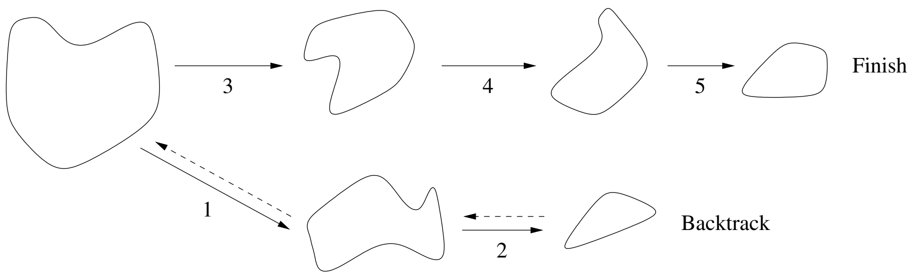


Figure 1.2: Search Through Algorithm Behaviour Space

This is illustrated by Figure 1.2. The behaviour of an optimiser can be characterised, for our purposes, as all of the possible sequences of points in the search space that the optimiser can produce (this will be formalised later in this thesis). Given that we are starting from enumeration/random search (which contains all of the search sequences), modifications of the optimiser can be seen as producing various subsets of the above. What the designer is doing, in effect, is finding an optimiser behaviour that gives satisfactory *expected* performance for the problem at hand.

Of course we could exhaustively search through the entire space of optimiser choices — however it would be easier to just search the problem's search space exhaustively (as that is much smaller that the space of algorithm behaviours). The question should therefore be formulated as one of *how* to conduct the above search. As will be noted later, due to the fact that optimiser design is itself a search problem, knowledge is also required. Therefore the first question to answer is what knowledge does the designer possess? It will be shown later that the designer does have knowledge, albeit incomplete, of both the problem domain, and the search dynamics of the optimisation techniques of interest (neighbourhood search). The remainder of this thesis devotes itself to the task of how to exploit this knowledge most effectively.

From this, the general aspects of neighbourhood search will be used to specify the main way in which domain knowledge can be represented to these optimisers (in the solution encoding and neighbourhood operators). The choice of optimisation technique will be shown, in a practical sense, to be of secondary importance — this forms the basis of the first of a series of 'design heuristics' which aim to guide the designer's search through the possible design options.

The discussion will then turn to formalising both the idea of a search through an 'algorithm behaviour space', and the roles of the solution encoding and neighbourhood operators. Given that the choice of solution encoding and operators represent an hypothesis about the nature of the problem being tackled, the effectiveness of the optimiser is a measure of the validity of these hypotheses. Therefore the available metrics for problem representation suitability with respect to neighbourhood search optimisers will be reviewed and found wanting.

Finally, this chapter will advocate an experimental programming approach and propose and justify a series of design heuristics. These allow hillclimbing experiments to test hypotheses about the nature of the problem in such a way that they are transferable across different neighbourhood search optimisers (including evolutionary algorithms). They can be used to derive suitable recombination operators for evolutionary algorithms. Therefore this will address the objective that the designer's experimentation is made as focused and effective as possible.

### 1.6.2 Chapter 4: No Optimisation Without Representation

In Chapter 4, a change in viewpoint will be advocated in order to arrive at a more useful methodology. At the moment, two paradigms are common: one based upon the study of dynamical and adaptive systems, which is dominant in the Evolutionary Computation community; the other based on the Operations Research approach of making a mathematical model of the problem and then solving it. Instead, it will be proposed in this thesis that an approach from Artificial Intelligence be taken, and that such optimisers be viewed as **Knowledge Based Systems** (KBSs). The key to this approach is to classify the domain knowledge by its role, and then to consider each item of domain knowledge as a 'knowledge source' about how to solve the problem.

In this chapter, the thesis will suggest that this **knowledge-level analysis** can be both easily formulated in concrete terms that a non-specialist in optimisation can understand, and mapped onto the optimisation algorithm in such a manner that a plan of incremental improvement and experimental programming is possible. To this end, it will be *outlined* how the forma analysis [Radcliffe 94] formalism to be discussed in Chapter 3 can be used to formalise each of the knowledge sources, and in most cases, make their effect felt upon the fitness landscape. From this, a number of additional design heuristics, based again on hillclimbing experiments will be proposed and justified to provide an experimental protocol for these knowledge sources. In addition, the design of the evaluation function and search control will be re-evaluated in the light of the KBS framework proposed here.

### 1.6.3 Empirical Evaluation

Needless to say, it is necessary to find out whether this proposed methodology will actually work in practice. The approach taken by this work is to use two case studies. To set the scene for them, Chapter 5 (Experimental Methodology and Sequencing Overview) will: state the scope of investigation of the case studies; justify the choice of case studies (both sequencing problems); describe the experimental set-up used; review the neighbourhood operators which are to be examined; and finally outline the structure of experimentation used to evaluate the proposed design heuristics.

**Chapter 6: The Flowshop Sequencing Problem**

Chapter 6 describes an investigation of a well-studied problem in operational research, the flowshop sequencing problem, and uses this as a basis for a thorough evaluation of the validity of the proposed design heuristics. In addition, it will be shown how idle-time information can be exploited using this methodology to increase optimiser performance on this problem.

**Chapter 7: Emergency Resource Redistribution in the Developing World**

The final case study is described in Chapter 7, the methodology tests the methodology on a real-world problem. A prototype system will be developed for the redistribution of relief/medical supplies in the developing world (specifically the People's Republic of China).

Apart from adding additional weight of evidence in favour of the design heuristics, it will be shown how the methodology proposed here was used to produce a system, evaluated on real data, that can deal with larger problems of this type than the current state-of-the-art.

### 1.6.4 Final Chapter and Appendices

This thesis will then be concluded in Chapter 8 which will summarise the methodology proposed here, note its contributions, relate it to other work in the literature, and suggest promising areas for future research.

Appendices are also included which include supplementary information such as the statistical methodology used to analyse the results, and the papers that have been published from the work described in this thesis.

## 1.7 Summary

This chapter has briefly introduced the techniques of interest and the subject of this thesis — the principled design of neighbourhood search optimisers. An overview of the thesis and the structure of the argument taken was also provided. The next chapter will set the scene by providing a review of the techniques of interest.

# Chapter 2

# Optimisation with Hillclimbing on Steroids — An Overview of Neighbourhood Search Optimisation Methods

This chapter will expand upon the treatment of neighbourhood search optimisers given in the introduction, reaffirm their commonalities, and suggest some useful analogies for understanding qualitatively how they work. As a result of this review, this chapter will provide an introduction for readers new to the field, as well as stating and clarifying the terminology and some theory that will be used in the remainder of this thesis.

The chapter will thus give a review of the techniques considered in this thesis to convey the design options the designer of a domain-specific optimiser has at his disposal. It should be noted that this review is not exhaustive and for a wider coverage the reader is also directed to other introductions [Reeves 93b, Osman 95, Osman & Kelly 96, Osman & Laporte 95].

## 2.1 A Common Framework

As noted in the introduction of this thesis, neighbourhood (or local) search extends hillclimbing in some fashion, usually by relaxing the acceptance criterion, in order to escape local optima.

As a result of this it is often convenient to place these techniques in a common implementational framework [Rayward-Smith 94] such as that described by the pseudo-code in Algorithm

1 below:

**Algorithm 1** AN IMPLEMENTATIONAL FRAMEWORK FOR NEIGHBOURHOOD SEARCH

**let** $P, Q, R \subset S$
$P$ = INITIALISE(); *// Generate starting solution(s)*
**while** !FINISHED($P$) **do**
  $Q$ = SELECT_SOLUTION($P$); *// Choose solution(s) to perturb*
  $R$ = CREATE_MOVES($Q$); *// Apply move operator(s)*
  $P$ = MERGE_SETS($P, Q, R$); *// Merge to obtain a new set* $P$
**end while**
**return** $P$;

Although it is usual to have only one solution in the sets $P$, $Q$, and $R$ above, **evolutionary algorithms** described later, are the exception to this rule. The above framework requires us to implement four functions specific to the problem and search algorithm: INITIALISE() to generate the starting solution(s); SELECT_SOLUTION() to select which of the solutions go on to have move operators applied to them by CREATE_MOVES(); and finally, MERGE_SETS() which implements the evaluation function and acceptance criterion, and decides which of the original and newly-generated solutions comprise the next neighbourhood search iteration.

Therefore given that the problem encoding, move operator(s), quality measure, and the above functions are specified appropriately, any of the versions of neighbourhood search described in the remainder of this chapter can be implemented. The reader should therefore keep this in mind when reading the detailed descriptions of the algorithms later on as it will assist in understanding the commonalities between these methods.

### 2.1.1 Analogy with Agenda-Based Search

In addition to the above, an analogy with exact agenda-based search methods [Hart *et al.* 68] (eg. the infamous **A**$^*$ algorithm) can be made [Jones 95, Jones & Forrest 94, Poli & Logan 96]. Mow modify the above to have two lists of solutions, both initially empty. One list is then an **open** list which denotes all of the solutions from which we can generate moves from; and the other is the **closed list** which is the list of solutions evaluated so far (so that we do not revisit points).

First, let INITIALISE() place a solution in both the open and closed lists. Now iterate through the following until the termination criterion has been reached: use SELECT_SOLUTION() to pick a solution in the open list to modify (usually the best quality solution), and remove it from the open list[1]; now use CREATE_MOVES() to generate the set of moves from these solutions that are not already in the closed list, and MERGE_SETS() to add these potential solutions to the open and closed lists.

We can therefore see that if we remove the closed list, and introduce the idea of a current solution as being the best found so far, we in effect change the agenda-based search into a hillclimber without a memory of previously visited solutions. Though this now makes the search heuristic in nature (as there is now no guarantee that the global optimum will be found), this has the advantage that we now longer have to store and manipulate lists of data structures that can be as large as the search space itself (!), with their associated infeasibly expensive computational requirements. In fact some of the techniques described later do use some form of memory of previous solutions, though unlike agenda-based search it is imperfect and often deals only with recently-visited solutions.

## 2.2 Iterated Hillclimbers and GRASP

One way out of the problem of local optima is to restart the hillclimber with a different initial solution when a local optimum has, or suspected to have, been found — this is known as **iterated** or **multi-start** hillclimbing. A common criterion for restart is when a certain user-defined number of evaluations have been made without an improvement in solution quality — the search will then resume in a different part of the search space with a different, and possibly better quality, local optimum. Such a straightforward approach has been proved to be quite effective in some cases. For example, [Johnson 90] showed that a multi-start version of the Lin-Kernighan TSP heuristic [Lin & Kernighan 73] could obtain the same solution quality as a simulated annealing procedure on a 1000-city TSP in less than one-sixth of the time.

One particular variant of this approach, **GRASP** (Greedy Randomised Adaptive Search Procedure, a trade mark of Optimization Alternatives, Austin, Texas) [Feo *et al.* 91a] incorporates a **construction phase** where the new starting point is generated using a randomised greedy algorithm which is then followed by a local search **improvement phase**. The intelligent **initialisation procedure** used in GRASP attempts to start the search in the vicinity of good solutions. More emphasis is thus placed on the initialisation procedure than the other com-

[1] In the event that the open list is empty, an unvisited solution (ie. one that is not on the closed list) is selected and placed on the open list.

ponents of neighbourhood search, to the extent that its design is highly problem-specific and is often *adaptive* in nature, making use of past experience in the search — this contrasts with the role of the improvement phase which is merely to locate a local optimum. Applications of GRASP include vehicle routing [Hjorring 95], the 2-partition problem [Laguna *et al.* 94], and single machine scheduling [Feo *et al.* 91b]. Finally [Feo *et al.* 91a] provides a review of other applications, such as flight scheduling for airlines, as well as directions for further reading.

## 2.3 Simulated Annealing and Threshold Methods

One of the most common approaches to avoiding local optima is to make some adjustment to the acceptance criteria so as to occasionally override the decision not to accept a move of worse quality if the drop in quality is not unduly high. The techniques below are designed to do this either stochastically or deterministically for stochastic/any-accept hillclimbers (though stochastic hillclimbing is most commonly used).

### 2.3.1 Simulated Annealing

**Simulated Annealing** (SA) [Kirkpatrick *et al.* 83] is based upon an analogy between optimisation and the annealing process in physics. In terms of neighbourhood search, it can be thought of as a form of stochastic/any-accept hillclimbing with a modified acceptance criterion that accepts lower quality solutions with a probability (the **Metropolis criterion**) given by the equation below:

$$p(accept) = \exp(\frac{quality(s_{curr}) - quality(s_{new})}{T_k})$$

where $T_k$ is the **temperature** at time-step $k$; this controls how likely it is for a lower quality solution to be accepted, and thus allows SA to escape local optima. The temperature varies with time according to a **cooling schedule** where, usually, the temperature is reduced as the optimisation progresses, to allow **exploration** of the search space at the start of the search, followed by **exploitation** of a promising region later on; the technique-specific choices of initial and final temperatures and the form of the cooling schedule are important so to obtain a balance between **exploration** and **exploitation** (also termed **diversification/intensification**).

This is because the behaviour of SA can be seen as standing between hillclimbing (when $T = 0$) and a random walk over the fitness landscape (when $T = \infty$).

However, there is no reason why the cooling schedule should be of any particular form — the choice is problem-dependent. That said, two common cooling monotonically decreasing schedules are: $T_{k+1} = \alpha T_k$, or $T_{k+1} = T_k/(1 + \beta T_k)$ [Lundy & Mees 86] — both of which have been found to be generally satisfactory. That said, [Boese 96] amongst others, have noted that non-monotonically decreasing cooling schedules often perform better in practice.

Finally, other work on SA has looked at applications such as sequencing [Osman & Potts 89, Ogbu & Smith 90], timetabling problems [Abramson 91], as well as the Steiner problem in graphs [Dowsland 91]. Further information on SA can be found in a variety of sources such as [Collins *et al.* 88, VanLaarhoven & Aarts 88, Aarts & Korst 89, Boese 96].

### 2.3.2 Threshold Methods

**Threshold methods** are additional extensions of stochastic/any-accept hillclimbing, that use the idea of a **threshold** which sets a level below which new solutions will not be accepted (i.e. acceptance is deterministic). For similar reasons to SA, the threshold, $L_k$, is time-varying.

- **Threshold Accepting** (TA) [Dueck & Scheuer 90] accepts a new solution if its quality is not below a set threshold relative to the current solution (ie. $quality(s_{new}) \geq quality(s_{curr}) - L_k$).This was independently discovered by [Hajek & Sasaki 89] in what they called the **threshold random search** algorithm.
- **Record-To-Record Travel** (RTRT) [Dueck 90] accepts a new solution if its quality is not below a certain threshold relative to the best solution or **record**, $s_{best}$, found during the search so far (ie. $quality(s_{new}) \geq quality(s_{best}) - L_k$).
- The **Great Deluge Algorithm** (GDA) [Dueck 90] accepts a new solution if its quality is not below an *absolute* quality threshold — the current **water-level** (ie. $quality(s_{new}) \geq L_k$).

These techniques differ in the way the threshold is used. For all of these techniques it is usual to vary $L_k$ according to the schedule $L_{k+1} = L_k + \alpha$, though there is no reason why others cannot be used. In general, the results obtained with these techniques are comparable to those obtainable with simulated annealing, though threshold accepting did do better in a comparative study involving the travelling salesman problem [Dueck & Scheuer 90] and [Johnson & McGeoch 96] compared the GDA with SA on the TSP with identical neighbourhoods and found that the GDA was significantly faster than SA in finding solutions of comparable quality. However, the first result has been questioned by [Johnson & McGeoch 96] who compared their own implementation of SA against the results obtained in the work by [Dueck & Scheuer 90] and found that the performance of the two algorithms was identical. That said, the threshold methods described here are a definite simplification of SA and are probably more efficient as, due to the deterministic nature of their acceptance criterion, they do not require as many calls to a pseudo-random number generating function. Finally, for further details and applications of the threshold methods, as well as related methods such as the **noisy method** [Charon & Hurdy 93], the reader is referred to the papers [Althofer & Koschnick 91, Sinclair 93].

## 2.4 Tabu Search

**Tabu Search** (TS) [Glover 89, Glover 90a, Glover 90b] is based on steepest-ascent or first-ascent hillclimbing, and avoids local optima in a deterministic way based on an analogy with memory. The technique has its historical roots in research on methods designed to cross boundaries of feasibility or local optimality, that were based on surrogate constraint methods [Glover 68] and cutting plane [Glover 66] approaches. It is also related to ideas presented in [Hansen 86] for the **steepest ascent/mildest descent** method.

In tabu search, the acceptance criterion of the hillclimber is altered slightly so that if no improving move can be found after the neighbourhood has been fully examined, then the move that gives the *least* drop in quality is taken. However, there is the distinct possibility that the search will return back to the previous solution. This **cycling** in an already explored area of the search space is prevented by the *explicit* use of memory that is used to guide the search away from previously visited areas of the search space to areas thought to be more promising.

### 2.4.1 The Four Dimensions of Tabu Search

The memory structures in tabu search are designed to represent four types of memory relevant to the search process. The first of these is **recency**, which is *short-term* in nature. When a move is made, or a solution visited, a record of this is made on the **tabu list**; though in practice this is usually achieved by using an **attributive memory**, where certain attributes of the move/solution are stored instead of the full solution. The tabu list thus stores solutions/moves that have been made/visited in the recent past (**tabu tenure**) of the search in what is effectively a FIFO data structure. This memory is then exploited by comparing the contents of the candidate moves with the contents of the tabu list. Any move/solution that matches to a **tabu-active** attribute in the tabu list is then prohibited from being made/revisited. This can thus be used to prevent the problem of cycling described above.

**Frequency** is a form of *long term* memory, which is often used to decide whether to move the search to a new region (diversification), or to intensify the search in what is thought to be a productive region of the search space. It is usual practice to consider frequency measures to be ratios where the numerators represent one of two measurements concerning each attribute of the solutions visited so far. For instance the **transition** measure records the number of iterations on which an attribute *changes* during the search trajectory (the set of solutions that have at one time been used as current solutions) — this is almost certainly a subset of the solutions evaluated. This is complemented by the **residence** measure which records the number of iterations that an attribute has *appeared* on the search trajectory (or some other subset of the solutions evaluated so far). One common use of frequency is to direct the search by adjusting the quality function (eg. solutions closer to a frequently visited area of the search space are penalised more).

**Quality** refers to the differentiation of the merit of solutions encountered during the search. This is often used to intensify or diversify the search in conjunction with frequency. For example, a high residence frequency coupled with a high-quality domain may indicate that an attribute is attractive (ie. associated with high-quality solutions) [Glover & Laguna 97]. Finally, **influence** is a measure of change in solution structure [Glover *et al.* 93], and is often used as part of an aspiration criterion (see below) — although it should be noted that quality can be seen as a special form of influence.

The above memory dimensions are then tied together with a **logic** component which specifies how the memory is organised and exploited to guide the search. Of course, all of these memory structures, the way that they are used, and the relative importance of possibly conflicting indicators of how the search should proceed have to be defined for the problem being solved. The reader is directed to [Glover & Laguna 97] for some guidelines on how to do this.

### 2.4.2 Aspiration Criteria

**Aspiration criteria** provide a mechanism with which to override the tabu status under certain circumstances. In this situation, two tests are made to determine tabu status: a **tabu test** (such as the recency test above) to indicate a preference towards a move being excluded, and an **aspiration test** to indicate a preference to the contrary; the results of these are then balanced, usually by use of a scorecard. The simplest of the aspiration criteria is **aspiration-by-default** which states simply that if the entire neighbourhood is tabu, then the least tabu of these moves is selected.

### 2.4.3 Candidate List Strategies

As neighbourhoods can be quite large and thus expensive to search fully, **candidate list** strategies are often used to choose subsets of the neighbourhood to search *in lieu* of the entire neighbourhood. A simple minded approach would to be randomly select a subset of moves to evaluate and then accept the best of them — this **random subset** candidate list has worked well in practice [Reeves 93a]. More sophisticated varieties of candidate list strategies have been devised [Glover & Laguna 97].

### 2.4.4 Current Research and Applications

To give a flavour of the diversity of work using tabu search, some additional example applications of TS include problems in: vehicle routing [Semet & Taillard 93], graph colouring [Hertz & deWerra 87], assignment problems [Laguna *et al.* 95], as well as those in path assignment [Oliveira & Stroud 89]. Current research in this area has looked at a variety of issues. For example, hashing functions can be used to identify tabu solutions [Hasan & Osman 95, Carlton & Barnes 95] more efficiently. Other extensions (eg. ejection chain methods [Glover 96]) have also been developed. Hybrids with other methods is also an active area of research, for example [Fox 93, Osman & Christofides 94]. Theoretical investigations of tabu search have also been undertaken [Faigle & Kern 92, Fox 93].

Finally, it should be noted that in practice, a recency-based approach with a simple neighbourhood structure, a simple random subset candidate list strategy, and a simple **fixed-length** tabu list often work well [Reeves 93b]. In these cases it if often sufficent to determine by a combination of experimentation and possibly the use of empirical rules-of-thumb. Examples of these are: the tabu tenure should be around $\sqrt{n}$, where $n$ is the length of the solution, or tabu tenures of length $\geq 7$ [Reeves 93b].

For more information on the issues raised above, alternative introductions to the area, and other further developments, the reader is directed to the book [Glover & Laguna 97] and the references [Glover 89, Glover 90a, Glover *et al.* 93, Glover 94, Hertz *et al.* 97] in addition to the general introduction given here and the references therein.

## 2.5 Evolutionary Algorithms

**Evolutionary Algorithms** (EAs) are based upon the theory of evolution by natural selection [Darwin 59] — a population of candidate solutions is maintained, and allowed to 'evolve'. Three main styles of EA exist: **Genetic Algorithms** (GAs) [Holland 75], **Evolutionary Programming** (EP) [Fogel *et al.* 66], and **Evolution Strategies** (ESs) [Rechenburg 73] — but the basic idea behind them is the same and the differences can be considered historical. In addition, new EA variations such as **Differential Evolution** [Storn & Price 97], which uses differences between solutions to produce moves, are being introduced all the time.

The diversity of EA variants is a result of each of the EA stages possessing a wide choice of alternatives. That said, all EAs have **population-based** acceptance criteria as a *set* of solutions are maintained and a new set of solutions are produced by applying the available move operator(s) and then somehow merging the two sets. Also EAs are often characterised by their heavy reliance on biological metaphors, whether or not this is actually warranted or useful [Reeves 94b]. As an example of the concept, the following describes a simple EA with **steady-state reproduction**:

1. Generate an initial population of solutions.

2. Select two **parent** solutions from the population according to their quality.

3. Apply move operator(s) to generate a new solution, $s_{new}$.

4. If $s_{new}$ is superior to the worst member of the population, $s_{worst}$, then $s_{new}$ replaces $s_{worst}$.

5. Go back to step 2 until the termination criterion is reached.

Two types of moves are commonly used: **mutation** (roughly analogous to asexual reproduction), which is equivalent to the conventional move operator; and a *binary* move operator which is roughly analogous to sexual reproduction. Known as **crossover**, or **recombination**, this operator selects two candidate solutions and (probabilistically) swaps information between them. An example is 'two-point' crossover which randomly picks two points along the strings and swaps the contents of the string between those points, to produce a child as shown in Figure 2.1.

$$\begin{array}{c|c|c} 0\ 0 & 0\ 0\ 0\ 0 & 0\ 0 \\ 1\ 1 & 1\ 1\ 1\ 1 & 1\ 1 \end{array} \xrightarrow{\text{Crossover}} \begin{array}{c} 0\ 0\ 1\ 1\ 1\ 1\ 0\ 0 \\ 1\ 1\ 0\ 0\ 0\ 0\ 1\ 1 \end{array}$$

Figure 2.1: An Example of Two-Point Crossover

The usual rationale for why crossover is a useful search operator is that the recombinative process can bring together portions of separate strings associated with high fitness to produce even fitter children. This is in much the same way that biological traits that are useful in separate parent organisms (such as good eyesight and strength in predators) can be usefully combined together in the parents' children faster than it would take mutation to perform the same task[2]. Finally, EAs can avoid the problem of the search being trapped in local optima as result of both the population-based and stochastic nature of the search, and the large moves made by the crossover operator.

Following on from this, the population-based nature of EAs also has advantages in that solutions in multiple local (or possibly global) optima can be maintained. Termed **speciation**, this

[2] It should be noted that this is not necessarily the reason why sexual reproduction evolved in nature — see [Ridley 94] for an excellent coverage of current work and views on this question.

is useful in the context of, for example, multi-objective optimisation where a number of solutions that represent different trade-offs between the objectives are desired. Some have argued, for example [Ross & Corne 95], that EAs have particular strengths over other neighbourhood search methods in this regard.

### 2.5.1 Formulation Options

The above description encompasses a very large number of possible implementations, and a large number of formulation options. A structured overview of some of the most common options is given here. It should also be noted however that **self-adaptation** methods have been investigated with the aim of producing optimisers that 'self-tune' — though their success is somewhat problem-dependent. For a full review of these methods, see [Smith & Fogarty 97, Tuson 95, Tuson & Ross 98].

#### Improvement Pressure

The bais towards finding better quality solutions, or **improvement pressure**, can be expressed in one or both of two parts of the general framework described in the early part of this chapter (Algorithm 1). The first is to use SELECT_SOLUTION() to bias the set of solutions selected for mating to those of higher quality — this is termed **selection with emphasis**. The second method is to use MERGE_SETS() to bias the new population towards containing solutions of higher quality (**termination with prejudice**). Also, the manner in which replacement is made can be varied between the two extremes of **generational** replacement, where all of the parents are replaced at each EA iteration, and **steady-state** replacement where only one parent at a time is replaced. A comprehensive review of EA selection and replacement methods can be found in [Hancock 94, Bäck *et al.* 97].

#### Move Operator Selection

An additional issue that arises here is that, in conventional EA applications, there are at least two move operators, Therefore some mechanism (as part of CREATE_MOVES() in the general framework described earlier) needs to be present to decide which operator to apply when and how often.

For example, a balance needs to be struck between crossover and mutation, and the relative perceived importance of their roles. Historically, the genetic algorithm community has been characterised by a view of crossover as the dominant operator, relegating mutation to the role of a background operator that maintains population diversity; the evolution strategy and evolutionary programming communities have in their turn been characterised by their emphasis on the mutation operator [Eiben 96]. However most practitioners take the middle ground as it is recognised that the relative importance and roles of these two move operators is dependent on the problem, its encoding and operators, and the other EA options.

#### Population Models and Speciation Methods

The **population model** of an EA is a description of how the set of solutions used by the EA is maintained, and how interactions between solutions with respect to selection and replacement are handled. The standard EA has an unstructured (or **panmitic**) population in that, fitness considerations aside, solutions are equally likely.

Other approaches are possible: the **island** model [Tanese 87] which separates the EA population into a number of sub-populations that occasionally exchange solutions; and the **cellular** model [Mühlenbein 89] which lays solutions out on a (usually) 2-dimensional grid, and allows crossover/replacement between solutions that are within a maximum number of grid squares distant. These methods also have the advantage that they make EA parallelisation more straightforward.

### 2.5.2 GA Theory

An overview of GA theory will now be given as later chapters will make use of many of the concepts described here.

The 'traditional' way to model EA dynamics has been with the use of the **schema theorem** [Holland 75]. This models the frequencies over time of **schema** (pl. **schemata**), which are matching templates (or equivalence relations) to what usually are the binary-encoded solutions in the search space (the templates are of the form 1 * * * 0 *, where * is a 'don't care' symbol) . A generalised version of the schema theorem for a generational EA with fitness-proportionate selection is given below:

$$E(N(H, t+1)) \geq N(H, t) \times \frac{f(H, t)}{\bar{f}(t)} \times (1 - \epsilon(H, t))$$

where $N(H, t)$ is the number of solutions that match the schema $H$ at time $t$; E(N(H, t+1)) is the *expected* number of matching solutions at time $t + 1$; $f(H, t)$ is the fitness of a schema $H$ at time t (calculated as the average fitness of *all* solutions in the population that match the schema); $\bar{f}(t)$ is the average fitness of the solutions in the population at time $t$; finally, $\epsilon(H, t)$ is a term describing the probability that a given schema $H$ will be removed from the population as a result of the move operators.

Therefore the above expression says that the expected number of instances of a schema in the next EA iteration is bounded from below by the product of the number of instances of that schema in the current generation, the fitness of that schema relative to the population average, and the probability that that schema will not be disrupted by the move operators. This captures well the intuitions expressed earlier about the role of recombination, and leads to the **building block hypothesis** [Goldberg 89c]:

> "Effective processing by genetic algorithms occurs when *building blocks* — relatively short, low order schemata with above average fitness values — combine to form optima or near-optima."

The reference to short, low order schemata is that schemata with many defined bits, or that have defined bits far apart are more likely to be removed by the action of move operators (i.e. $\epsilon(H, t)$ will be high).

However, though [Goldberg 89c] and others consider the above to be (quote) 'the fundamental theorem of genetic algorithms', the above suffers from some serious flaws that weaken its strength as a *quantitative* model of EA dynamics. The first of these is that the above expression is an inequality (though [Bridges & Goldberg 87] and [Nix & Vose 91] have since addressed this issue). Also other selection models apart from fitness-proportionate are difficult to model.

As the population will almost certainly contain a subset of the instances of the schema $H$, the *estimated* fitness of a schema $\hat{f}_{pop}(H, t)$ should be substituted for the above. This takes into account the implicit assumption that the fitness of schemata present in the EA population accurately samples the situation for the search space as a whole [Grefenstette 93], but this assumption and the above analysis present two immediate problems.

The first is that, after the first EA iteration, $\hat{f}_{pop}(H,t)$ will necessarily be a *biased* estimator of $f_{pop}(H,t)$. Therefore, any static analysis of the *actual* schema fitness may not be of any value in predicting EA dynamics — in fact this has led [Grefenstette 93] to propose a **dynamic building block hypothesis** to account for this. The second problem follows from this as in all but the simplest problems schemata interact. In other words, the observed fitness of a schema $H$ in a population depends upon which other schemata are also present in the solutions that contain $H$. As the schema theorem deals only with expected numbers of schema, it provides insufficient information for any predictions more than one generation ahead in all but the most trivial of problems.

In addition though, a somewhat separate issue to the above arises. Work by [Holland 75] argued that EAs in fact process schemata, and do so at a rate given as $O(N^3)$, where $N$ is the size of the population (and this is only for binary-encoded problems) — this is termed **implicit parallelism**. This suggests that a representation should be low cardinality, in other words binary, as this will maximise the number of schemata in the EA population and this lead to more information being processed (this is known as the **principle of minimal alphabets**). However, this has been disputed by [Radcliffe 92] amongst others, and it should be noted with interest that most successful EA applications used whichever encoding seems most appropriate to the problem at hand — which is usually *not* binary [Michalewicz 92]!

Finally it should be noted that other approaches to modelling EA dynamics have been explored which overcome some of the above difficulties. These include the use of Walsh functions [Goldberg 89a, Goldberg 89b], the application of Price's theorem [Altenberg 94], and the statistical mechanical formulations due to [Prügel-Bennet & Shapiro 94].

### 2.5.3 Example Applications

Applications of EAs are extremely varied, covering fields as diverse as applications to chemistry [Cartwright & Harris 93], machine learning [Goldberg 89c], and OR. Example applications in OR include: sequencing problems [Reeves 95a], vehicle routing [Thangiah 95], and timetabling [Corne *et al.* 93]. A variety of textbooks are available though Michalewicz

[Michalewicz 92] is a good starting point for those interested in how to apply EAs, whereas the recently-released 'Handbook of Evolutionary Computation' [Bäck *et al.* 97] is recommended for its in-depth coverage of the field.

## 2.6 Summary

The neighbourhood paradigm was first outlined and analogies between it and other search techniques were noted. This chapter then gave a review of each of the methods considered in this thesis and noted some representative applications.

Finally, this chapter will close with the admission that it has made little mention of *how* to apply these techniques in practice. This is for a very good reason: in short, no such effective guidelines exist. Addressing this problem of principled design is essential if these techniques are to be more widely applied and this will be the subject of the remainder of this thesis.

# Chapter 3

# Paying For Lunch in Neighbourhood Search

This chapter will start to explore the main issue of this thesis, the principled design of neighbourhood search optimisers. First of all, the need for the exploitation of knowledge in optimiser design will be covered in detail. This will then lead into a consideration of *where* the knowledge needed to design an optimiser comes from.

Central to this thesis are the ideas that (1) domain knowledge can be usefully divided in a number of well-defined categories; (2) optimisers are usefully considered as systems *embodying* the designer's knowledge which should be made explicit; and (3) effective experimental protocols exist that allow the correctness of hypotheses concerning the problem domain to be evaluated and transferred across different optimisation techniques.

To this end, it will be shown how aspects of a neighbourhood search optimiser can be mapped to the designer's knowledge and it will then be argued that design should first take place in terms of the problem domain theory. In addition, this chapter will argue that hillclimbing experiments can form an effective experimental protocol and shall propose and justify a number of design heuristics to form the basis of this.

## 3.1 The No Free Lunch Theorem

Recent theoretical work [Wolpert & Macready 95], argues that optimisers (strictly, search algorithms that don't revisit points are considered) give the same average performance (for any arbitrary performance measure) when averaged over all problems — this is known as the No-Free-Lunch (NFL) theorem. As many of the arguments made in this chapter will take advantage of the mathematical framework outlined in [Radcliffe & Surry 96], the notation used there will be described here for later use.

### 3.1.1 The Concept of a Search Space

A search problem can be characterised by a search space $\mathcal{S}$ of possible solutions, and an objective function $f$ which can be thought of as a mapping from $\mathcal{S}$ to the space of **objective function** values, $\mathcal{R}$ (i.e. $f : \mathcal{S} \to \mathcal{R}$). Two points should be made. First though $\mathcal{R}$ is usually taken from the set of real numbers, it is not required for the NFL proof. Second, the set of mappings from $\mathcal{S}$ to $\mathcal{R}$ is denoted $\mathcal{R}^{\mathcal{S}}$, so $f \in \mathcal{R}^{\mathcal{S}}$.

In addition, an **encoding**[1] of $\mathcal{S}$ is defined as the combination of the set $\mathcal{E}$ and a **surjective** function that maps $\mathcal{E}$ onto $\mathcal{S}$ (i.e. $g : \mathcal{E} \to \mathcal{S}$). More simply put, $g$ must map at least one point in $\mathcal{E}$ to each point in $\mathcal{S}$. This is more formally described as $g \in \mathcal{S}^{\mathcal{E}}_{>}$, where $\mathcal{S}^{\mathcal{E}}_{>}$ denotes the set of surjective functions from $\mathcal{E}$ to $\mathcal{S}$. The above definitions now allow the objective value of an encoding solution $e \in \mathcal{E}$ to be found by applying $f$ and $g$, i.e. by using $f(g(e))$.

These distinctions between the search and encoding spaces are important, as search algorithms do not search directly in the search space (which could be thought of as an abstraction of the space of real world solutions), but in an **encoding space** which is derived from the search space.

### 3.1.2 Sequences, Permutations, and Search Algorithms

Some additional definitions are also required, before the main proof is given. The first of these is the concept of a **sequence set**, $S(\mathcal{A})$, which for a given set $\mathcal{A}$, gives the set of all sequences of a finite length over $\mathcal{A}$:

$$S(\mathcal{A}) \stackrel{\triangle}{=} \{\langle a_1, a_2, \ldots, a_n \rangle \mid a_i \in \mathcal{A}, 1 \leq i \leq n\}$$

[1] Radcliffe used the term ‘representation’, but as later discussion will reveal that there is a strong duality between encoding and operators, the term encoding was thought more appropriate; also, this thesis will discuss representation in a wider sense than in Radcliffe’s work.

Second, the NFL proof is achieved by re-mapping (permuting) the elements of one of the spaces (especially $\mathcal{E}$). A **permutation** of a set $\mathcal{A}$ is defined as a relabelling (*invertible* mapping) of the elements of the set, i.e. $\pi : \mathcal{A} \to \mathcal{A}$. $\mathcal{P}(\mathcal{A})$ will be used to denote the set of all permutations of the objects in $\mathcal{A}$.

The concept of a **search algorithm** needs to be defined. The NFL proof concerns itself with **deterministic** algorithms, with no loss of generality as **stochastic** search algorithms can be modelled as a deterministic algorithm with the addition of a seed for a pseudo-random number generator.

A search algorithm is defined as a **generator function** that when given a sequence of points $\langle x_i \rangle$ with each $x_i \in \mathcal{E}$, and their objective function values, generates a new point $x_{n+1} \in \mathcal{E}$. This can be defined by the mapping:

$$A : S(\mathcal{E}) \times S(\mathcal{R}) \to \mathcal{E}.$$

This allows the definition[2] of the first and subsequent points in the search sequence, $\langle A_i \rangle \in s(\mathcal{E})$, associated with the generator function $A$:

$$A_1 \triangleq A(\langle\rangle, \langle\rangle) \qquad A_{n+1} \triangleq A(\langle A_i \rangle_{i=1}^{n}, \langle f(g(A_i)) \rangle_{i=1}^{n})$$

The above notion of a search algorithm is more than general enough to include all of the neighbourhood search algorithms, as well as random and stochastic search [Radcliffe & Surry 96]. Also as the NFL proof applies strictly only to algorithms that do not revisit points in the encoding space, then such an algorithm must produce a **non-repeating search sequence** in $\mathcal{E}$, which sequence of points $\langle e_1, e_2, \ldots, e_n \rangle$, where $e_i \in \mathcal{E}$, such that all the points in the sequence are unique (i.e. $\forall i, \forall j,\ e_i \not\equiv e_j$)[3].

Finally, the notion of a **performance measure**, $\mu$, for a given search algorithm will be defined as any measure that is solely dependent upon the sequence of objective values obtained under a given mapping function, $g$, of the points in $\mathcal{E}$ visited by the algorithm. Formally, $\mu$ is any function $\mu : S(\mathcal{R}) \to \mathcal{M}$, where $\mathcal{M}$ is any set used to measure the performance of $S(\mathcal{R})$.

[2] **NOTE**: as in the original paper, the notation will be abused somewhat by identifying a search algorithm with its defining function. In other words the mapping $A$ defines a search algorithm that is also denoted by $A$.

[3] This is equivalent to the definition of a **valid search sequence** in $S$ in the original paper; however the property is more simply defined with respect to $\mathcal{E}$.

Note that the search algorithms is assumed to run until they have covered the entire encoding space, $\mathcal{E}$, though performance measures can use a smaller number of the objective function values (which allows algorithms that cover a subset of $\mathcal{E}$ to be considered).

### 3.1.3 A Simple View of the NFL Proof

The NFL proof given in [Radcliffe & Surry 96] is one of the more straightforward to outline in 'plain-English' terms. This proof relies the notion that all search algorithms, in effect visit a series of points in an encoding, $\mathcal{E}$, of the search space, $\mathcal{S}$, where the relationship between the two is given by what Radcliffe terms a **growth function** $g$. Also it is assumed that a performance measure of an algorithm can be derived with reference to a sequence of points visited so far.

The proof proceeds as follows. Given that such an algorithm was to be non-repeating with respect to $\mathcal{E}$, then such an algorithm has to produce a permutation of the elements of $\mathcal{E}$. Therefore it must be possible to transform a given algorithm, $A$, into another, $B$, by simply permuting the mapping, $g$, of $\mathcal{E}$ to $\mathcal{S}$, as this would result in the elements in $\mathcal{S}$ being visited in the same order.

It is therefore possible to transform two given non-repeating algorithms so that they are equivalent with respect to the sequences of points visited in $\mathcal{S}$. Now given a set $\mathcal{G}$ of growth functions it can be seen that $A$ can generate all of the search sequences that $B$ can, and therefore their performance must be equal.

From this, a number of corollaries show that the above result is equivalent to saying that all algorithms give the same overall performance when averaged over all of the possible problems: The No Free Lunch Theorem. This has since been extended [Surry 98] to the case where $g$ is restricted to polynomial permutations of $\mathcal{E}$.

### 3.1.4 Alternative Formulations/Proofs

In should be noted that similar proofs for algorithm applicability have been proposed in the machine learning literature [Watanabe 69, Mitchell 80]. That said, other recent NFL proofs for optimisation have also been proposed [English 96, English 98, Culberson 96]. The notion of a search algorithm remains the same as above (all of the papers use effectively the same formalisation of a search algorithm), as well as the assumption of a non-repeating search algorithm. Therefore all of the NFL proofs are, in a strong sense, equivalent.

## 3.2 Implications and Limitations of NFL

This section will outline the implications and limitations of the NFL theorems and then introduce more fully the question we need to answer — *how* to use domain knowledge to construct an optimiser.

### 3.2.1 Repeating vs Non-Repeating Algorithms

The first point to make is that the NFL theorem only strictly applies to algorithms that do not revisit points. The authors of the original paper consider this to be a technicality, citing the reason that a non-repeating algorithm represents a 'compacted' algorithm template for a set of repeating algorithms, that when their revisited points are removed, give the same permutation of points in $\mathcal{E}$. However this *does* make a difference. As noted in [Radcliffe & Surry 96, Surry 98], if we assume that an algorithm is devised that visited all of the points that an neighbourhood search algorithm can, minus the revisited points then, all things being equal, it obviously would be considered to be more efficient. Therefore there will be cases where two algorithms *can* be considered to have different average performance over all problems. Also, in practice, implementing a non-repeating algorithm is often not feasible as a result of the memory requirements and the computation time required to process the large amount of data generated.

As neighbourhood search algorithms can revisit points, the theorem is therefore an approximation (albeit a useful one). Fortunately this approximation can be considered to be sufficiently close to the situation in practice as for most of the search few points are revisited (it would mostly occur once the algorithm has located a local optima), and the amount of revisiting would be expected to be roughly the same across the class of algorithms that we are considering (given their underlying similarities).

### 3.2.2 The Case for Domain Knowledge

The implications of this theorem can easily be misunderstood. For example, there can still exist a technique-of-choice for a particular *class* of problem — the problem is deciding *what* it is. An interpretation of the theorem that is more useful, is to relate the effectiveness of the optimiser to the amount of knowledge of the problem provided. For instance, if we have no knowledge of the problem, then any choice of optimiser will be as good as any other, *a priori*. However, if we obtain information about the problem, we can then use this to select (or design) a more effective optimiser. In short, domain knowledge is *necessary* for effective optimisation.

In some respects this view is not new. In the mainstream AI community the dominant paradigm (symbolic functionalism) is knowledge-intensive by its very nature [Stefik 95], and in the OR community the importance of problem-specific neighbourhood operators has been long recognised [Reeves 93b]. However, the impact of the NFL theorem has made itself most acutely felt in the Evolutionary Computation community, as succinctly put by [Culberson 96]:

> "Still, these [NFL] theorems have generated much controversy throughout the Evolutionary Algorithms (EA) community in part because not everyone seems willing to accept the full implications."

In the Evolutionary Computation community, much of the original research arises from a desire to study adaptive systems (such as Darwinian evolution) and not optimisation. When it was later found that Evolutionary Algorithms could be used for optimisation [DeJong 75], it became rather common to over-sell their abilities, partly due to a 'nature knows best' attitude amongst members of the community and a misinterpretation of the schema theorem (which will be discussed later). Unfortunately, this is a view that has persisted until recently. For instance the overview of future directions of EA research in [Schwefel 97] spends most of its time discussing the transfer of biological metaphors to EAs (see quote):

> "Let us look first at some deficiencies of current EAs compared to organic evolution. The benefit will be that this kind of approach may lead directly to further improvements in evolutionary computation. either by increasing the efficiency and/or range of applicability of EAs."

Of course it should be noted that dissenting voices were raised long before the NFL theorems — the main advocates for a more aggressive stance towards the exploitation of domain knowledge came from those involved in using evolutionary algorithms in practice [Davis 89, Michalewicz 92]. Also [Reeves 94b] has argues that EAs can be looked at from a number of non-biological perspectives, and [Radcliffe 92] was one of the first to cite theoretical objections. It is from their tradition and work that some of the arguments made in this work will be drawn.

### 3.2.3 What Now?

The discussion above, although identifying the need for the inclusion of domain knowledge if a search algorithm is to be effective, remains silent about how to design such an optimiser. Also, from an engineering viewpoint, we not only have the requirement that the optimiser satisfactorily performs the task for which it is to be used, but also that the designer can justify *how* that behaviour was produced — if only because people tend to be reluctant to adopt technologies when they do not understand how they work. This brings us back to our need for a methodology that can examine a problem for useful domain knowledge, and to map it onto the optimisation algorithm in a structured and useful way, because if such a methodology was designed and used to produce a working system, then its behaviour could be justified.

In addition, as noted earlier in the introduction, the process of constructing an effective optimiser is also itself a search problem; though this time one of the various options that make up one of the techniques considered here. Could we just somehow search the space of search algorithms instead? This approach immediately runs into two problems: first, the NFL theorems must also apply to this meta-problem as it is itself a search problem; second, this meta-problem will necessarily be of a larger size than the original optimisation problem because if we are designing an optimiser to effectively search the original problem of search space size $S$, then the meta-problem of searching for a good optimiser must have a search space size of a least $S!$. In fact, the situation is worse in practice since the mapping between the search techniques and their parameters often introduces redundancy, also we are dealing with non-repeating algorithms which in their turn also increase the space of possible optimisers.

Referring back to the original statement of the NFL theorem, it is impossible to say whether one search algorithm is better than another in the *absence* of knowledge about the structure of the problem. However we do have some knowledge of the structure of the optimisation problem in terms of both the way that neighbourhood search optimisers work and of the problem domain. It then follows that if we are to to make any headway in devising a principled methodology for optimiser design we need to examine, from first principles, the nature of the search techniques we are interested in to identify what aspects of their design lead to effective optimisers and how to obtain these from the problem. To this end, the following sections of this work will closely reexamine the fundamentals of neighbourhood search algorithms and argue that an approach to designing neighbourhood search algorithms can be devised which satisfies the criteria specified so far.

## 3.3 Where Does The Knowledge Lie?

The issue of where the knowledge required to construct an effective optimiser can be obtained from will now be addressed in detail. The formalism used in the NFL theorem suggests that domain knowledge can be placed into an optimiser in two ways. The first is to permute the solution-quality mapping (the growth functions $f$ and $g$) so that a given optimisation algorithm (which works in the encoding space $\mathcal{E}$) can find high quality solutions sooner. The second is to fix the solution-quality mapping and to devise an algorithm (the generator function $A$ above) that visits the high quality solutions sooner. Therefore there is a duality between the solution-quality mapping and the search algorithm. This section will take a somewhat different decomposition of an optimisation algorithm, relate it to the above and note how proposed decomposition relates to the designer's knowledge.

### 3.3.1 Fitness Landscape = Encoding + Operators

A common way to consider the operation of neighbourhood search is to think of a fitness landscape (see 3.1), which can be pictured as the union of three constituents forming a graph in the fashion described in, for instance, [Jones 95] and based on the biological concept due to [Wright 32] in the early 1930's: each of the nodes corresponds to a solution in the search space and has its fitness value attached, and the edges connect the nodes according to whether nodes can be reached by one application of the neighbourhood operator.

The concept of a landscape has received a significant number of attention over the year, especially when one notes that (for example) the concept of a 'cost surface' has been common in the context of methods such as simulated annealing [Boese 96]. In recent times, landscapes have been the focus of a number of studies in the EC community. Formal definitions can be found in [Weinburger 90a, Weinburger 90b] and [Manderick *et al.* 91] for example, and has drawn upon work by Schuster, Stadler, Fontana, and others in the theoretical chemistry literature (eg. [Standler & Happel 92]). Other works on landscapes will also be covered later in this chapter.

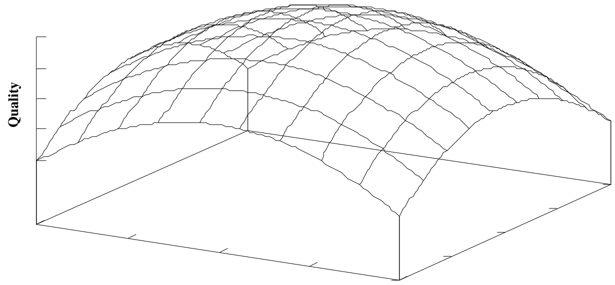


Figure 3.1: A Simple Example of a Fitness Landscape

A fitness landscape can be formally defined as the following tuple $\mathcal{L} = (\mathcal{E}, o, f, g)$ where $\mathcal{E}$, $f$, and $g$ have been defined previously, and $o$ is a given neighbourhood operator. In this formalism, the nodes are all of the points $e \in \mathcal{E}$, and the vertices connected between nodes according to $N_o(e)$ (the neighbourhood of a solution $e$ with respect to an operator $o$). Note that the resulting graph is strictly speaking **directed**. However, it will be assumed from now on, for simplicity, that the graph is undirected, as many move operators in neighbourhood search are in fact **reversible** (ie. $\forall x, y : x \in N_o(y) \Leftrightarrow y \in N_o(x)$). The work in [Jones 95] then used this formalism to then define fitness plateaus, local optima, saddle points, and global optima. The relationship between the components of the fitness landscape is shown more clearly in Figure 3.2 below which shows how the neighbourhood operators structure the encoding space.

Finally, it should be noted that the concept of a landscape used here is somewhat simplified compared to that of [Jones 95], for instance that work added transition probabilities to the vertices of the graph (thus in effect including some traversal rule information in the landscape). The choice of formalism used here was for reasons of simplicity and readability — though the formalism used in [Jones 95] is arguably more general, it is more complex than is required to make the points to be raised in this thesis.

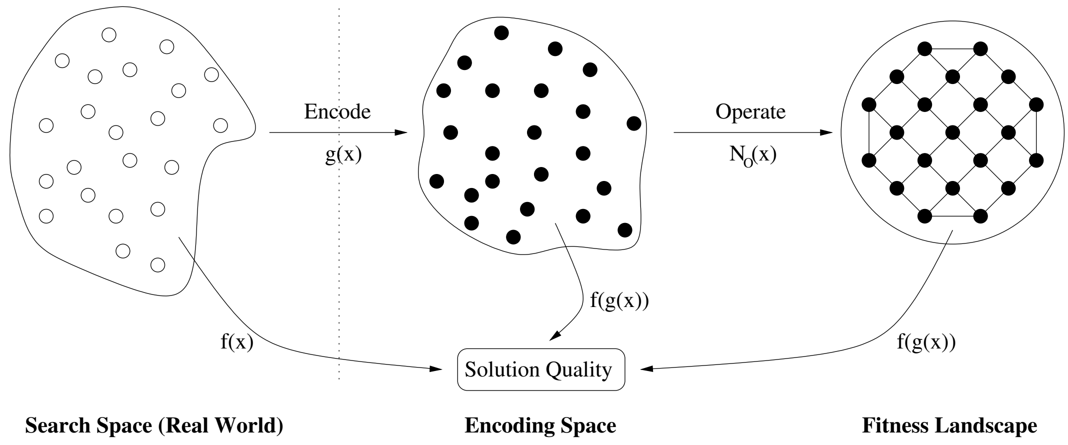


Figure 3.2: The Fitness Landscape in Terms of $\mathcal{E}$, $o$, $f$, and $g$

### 3.3.2 Algorithm = Fitness Landscape + Traversal Rules

It should be clear from the above that we can thus visualise the search process as a traversal of point(s) on this landscape with respect to well-defined **traversal rules**. Therefore we have decomposed the NFL idea of an algorithm (a function $A$ which visits a sequence of points in the encoding space) into two parts: the neighbourhood operator (which forms the landscape), and the traversal rules (which we shall denote $t$) — in other words, $A = (o, t)$. It was also noted earlier, that there was a duality between the encoding and the algorithm $A$. It follows from this that there must also be an equivalent duality between the fitness landscape and the traversal rules. This is because now we have, in effect decomposed the former duality of optimiser behaviour into 3 components (the encoding, operator, and traversal rule), and then moved the role of the neighbourhood operator from the algorithm to the landscape. Therefore if there was not a duality between the landscape and traversal rules, then we would have a contradiction with the earlier duality with respect to $\mathcal{E}$ and $A$.

Moreover, turning our attention back to Figure 3.1, it should be straightforward to see that by either permuting the assignments of fitnesses to nodes, or by changing the operators, we change the nature of the fitness landscape and thus the difficulty of the search with respect to a given set of traversal rules. In other words, we also have a duality between $\mathcal{E}$ and $o$ given a fixed $t$, and therefore have established that any one of the three components of an optimiser outlined here can be used to produce a given optimiser behaviour even when the other two are fixed.

## 3.4 Landscape vs Optimiser Design

The first question that needs to be addressed in proposing a methodology for optimiser design is which component of the optimiser should be considered first by the designer — the landscape or the traversal rules? This issue is discussed below, and a design heuristic will then be proposed.

### 3.4.1 Difficulty of Traversal Rule Selection/Design

Of course the discussion above would be academic if it was just as straightforward to design a search algorithm with respect to the traversal rules, as it is in respect to the neighbourhood structure and encoding. This is not the case for three reasons: the first is that, as will be discussed later in Section 3.7, it is difficult to characterise landscapes; the second is that it still remains very difficult to predict the effectiveness (relative or absolute) of an algorithm on a given problem/fitness landscape even when it is fully characterised [Ross *et al.* 96].

Finally, the above is further compounded by the fact that the choice of traversal rule for a given landscape can depend on the performance measure ($\mu$) used. For instance, consider the example in Figure 3.3 and for the purposes of this discussion take the performance metric as being the quality of the best solution found after $N$ evaluations[4]. The landscape is characterised as a large basin with smooth sides and a central region with multiple local optima.

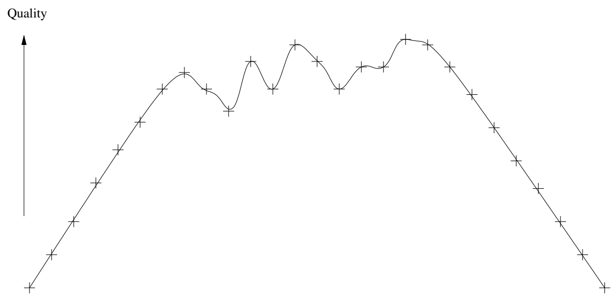


Figure 3.3: A Landscape Where the Choice of Algorithm Varies with Available Evaluations

Given that most of the starting points are on the sides of the basin, the search is most likely to start there. Now if $N$ is sufficiently low to produce search mostly in the smooth region, then

[4] Recall the performance metric refers to the algorithm and thus will invariably make some reference to computational resources an the expected quality of the solutions evaluated under these constraints (whereas solution quality is dependent only the the solution being considered).

a simple hillclimber would be a good choice, as any mechanism for escaping local optima is unlikely to be useful (and if present could slow down the search). However as $N$ is increased, the likelihood of meeting a local optimum increases and the need for such an optima-escape mechanism becomes greater (and therefore the choice of traversal rule has changed as a result of a change in the performance metric).

The problem of the design of traversal rules has been summarised by [Mitchell 96] (for EAs) in the quote below:

> "...it seems unlikely that any general principles about parameter settings can be formulated *a priori*, in view of the variety of problem types, encodings, and performance criteria that are possible in different applications".

In short, a method to characterise landscapes, and an effective working theory for exploiting this information is still a long way off.

### 3.4.2 Duality Breakdown and Non-Repeating Algorithms

Unfortunately, in practice, it is not possible for the designer to have total freedom in manipulating the traversal rules to produce an effective optimiser.

First of all, we have already decided upon algorithms based on a certain class of traversal rule: neighbourhood search, all of which are based upon the idea of hillclimbing, and this duality (between landscape and traversal rules) breaks down for as we are considering a subset of the space of traversal rules — therefore we are constrained as to what changes can be made to the traversal rules to improve search before we can no longer justify calling an optimiser neighbourhood search.

In other words, all of the traversal rules (neighbourhood search techniques) available rely upon the search space being, to a lesser or greater degree, correlated in the sense that moves to nearby solutions lead to small changes in the solution quality in such a way that directs the search to higher quality solutions (this will be expanded upon later). If this is not the case, neighbourhood search algorithms will fail, no matter how they are extended.

In addition, [Radcliffe 94] argues that even if the above argument did not hold, the duality described above between landscape and traversal rules would break down for the case of non-repeating search algorithms (which are those that are used in practice). This is because the number of landscapes available will be unchanged upon going to non-repeating algorithms, though the space of possible algorithms (in the NFL sense) will become much larger (possibly infinite). Therefore, there will be more possible search sequences in $S$, than there are landscapes to match — thus the duality breaks down.

### 3.4.3 The Origins of the Designer's Knowledge

Even if it were possible, in principle, to make the necessary changes to the traversal rules and be able to do this effectively, there would still be a *pragmatic* reason for not considering the two components as being equivalent. This relates to what aspects of the designers domain knowledge are used when constructing/selecting the fitness landscape and traversal rules. In short, the designer can draw upon either his **problem domain theory** or **search dynamics theory** in the construction of an optimiser. These can be directly mapped onto the landscape and traversal rules respectively, as will be now shown.

The construction of a fitness landscape is via operators and solution encoding, which are usually chosen to reflect the problem domain in that the encodings directly correspond to relevant features of the problem description, and the operators to transformations to a candidate solution that a human expert would make — this is what [Michalewicz 92], amongst others, call **natural representations**. The main exception to this approach is the argument from some in the EA community that low cardinality encodings should be used — this argument will be discussed (and rejected) later. Therefore it should be apparent that the design of the fitness landscape comes from the designer's theory of the problem domain.

However, the design of a set of traversal rules requires a different approach. First of all, as the effectiveness of any given set of rules depends on the fitness landscape, a fitness landscape needs to be assumed. The fitness landscape then needs to be analysed and a suitable set of traversal rules designed. Note that none of this makes any reference to the problem domain whatsoever — instead the design here is in terms of the designer's knowledge of the landscape and his theory of the search dynamics of the traversal rules.

From this, it can be argued that designing the optimiser in terms the of the fitness landscape is preferable when tacking real-world optimisation problems. Put simply, one of the criteria for a useful methodology outlined in the introduction was that the methodology should be accessible to non-specialists in optimisation (the end-users). In addition, it is well noted that end-users are loathe to adopt systems without some understanding of how they perform their task — therefore designing an optimiser in terms of the problem domain allows the system's behaviour to be more easily described in terms the end-users can understand.

Finally, given that some experimentation will be required, then designing the optimiser as far as possible in terms of the problem domain theory will mean that experimentation will increase our understanding of that theory (which is useful to both the designer and domain expert/end-user), rather than the search dynamics (which is of most interest to optimisation experts).

### 3.4.4 The First Design Heuristic

The above three considerations therefore lead to the first of the design heuristics to be proposed in this work:

> **Design Heuristic 1:** *The design options concerning the fitness landscape should be examined before the design options concerning the traversal rules.*

It should be noted that the picture of a landscape presented so far does not currently cover all of the methods of including domain knowledge into an optimiser (eg. removing unwanted solutions, indirect encodings, etc). However, the later parts of this thesis will show how they can be reconciled and integrated with the picture presented here — which will suffice for present.

The above discussion does however leave a few unanswered issues. The first is in specifying exactly *how* the landscape corresponds to the domain theory. The second concerns the process of experimentation. If we take the constructed landscape as an *hypothesis* about some aspect of the problem, then its correctness can in principle be evaluated by accessing its ability to produce an optimiser. The question is therefore how to go about this so to make the experiments as effective as possible?

Therefore, more ground needs to be covered before the other design heuristics can be introduced. The following sections will address the above points by first showing how the connection between the fitness landscape and the problem domain theory can formally be made. Then the idea of an algorithm behaviour space will be formalised, to show that it is in fact sensible to think of an optimiser as a system that embodies a number of hypotheses about the problem domain and search dynamics theories which can in principle be experimentally evaluated.

Finally, methods for evaluating the effectiveness of fitness landscapes will be reviewed and discussed. This will motivative and set the scene for the introduction of design heuristics that deal with the issue of formulating an effective series of experiments to design an optimiser.

## 3.5 Equivalence Relations: A Link to Reality?

One aspect of the above discussion that is lacking for our purposes is that the relationship between the landscape and the problem domain is not clear. This arises because of the fact that *any* arbitrary set of symbols and transformations can be used to produce the landscape, so long as the connectivity of the solutions in the search space is preserved. Unfortunately, such an arbitrary representation would leave us without any reference to the problem domain, and thus how the behaviour of the optimiser is produced. Therefore we need to identify, make explicit, and formalise the aspects of the solutions that are thought important in solving the problem, if we are to make any progress in systematically relating our knowledge of the problem domain to the design of the optimisation algorithm.

Fortunately a method for the formalisation of such aspects does exist: **forma analysis** due to [Radcliffe 91a]. Devised originally as a generalisation of the notion of schema in evolutionary algorithms[5], this takes the approach of supposing that each solution is described by a set of relevant features (eg. eye colour if we are considering the set of all people) that are thought to relate strongly to solution quality. Forma analysis then formally maps each feature to an **equivalence relation** $\psi$ which has a set of **equivalence classes (formae)** denoted by $\xi \in \Xi_{\psi}$[6] which would correspond to the set of possible eye colours.

**N.B.** The notation used here differs slightly to that used in [Radcliffe 94]. For the purposes of

[5] As noted in [Radcliffe 94], this work is related to the schema generalisation work by Vose [Vose & Liepins 91].

[6] Formae can also be used to denote the subset of the solutions in the space space that match a given equivalence relation/class combination, ie.:

$$\xi(\psi, j) \triangleq \{x \in S \mid \psi(x) = j \}$$

this discussion, $\psi$ refers to a **basis** equivalence relation, which is a member of the minimum set, $\Psi$, of equivalence relations required to uniquely describe any solution in the search space $S$. Of course other equivalence relations can be produced by combining these basis relations — these combinations of the basis equivalence relations will be denoted as $\psi^{'}$, their set as $\Psi^{'}$, and their formae as $\xi^{'} \in \Xi_{\psi'}$.

### 3.5.1 Formalisation

To formalise the above further, for each equivalence relation $\psi$ we can now define a function/predicate $\psi(x, y)$ that is true (or returns 1 in an arithmetic expression), if for the equivalence relation $\psi$ both solutions $x, y \in S$ have the same equivalence class; otherwise the relationship is false (or zero is returned). In other words, given the function $\psi(x)$ which returns the equivalence class of the equivalence relation $\psi$ for the solution $x \in S$, we can write:

$$\psi(x, y) \triangleq \psi(x) = \psi(y)$$

or if an arithmetic expression is desired:

$$\psi(x, y) = \begin{cases} 1 & : \psi(x) = \psi(y) \\ 0 & : otherwise \end{cases}$$

Therefore given the above we can fully specify any solution in the search space in terms of the vector of the *basis* features/equivalence relations in $\Psi$. So the set of all solutions can be described by the following:

$$\Xi_\Psi \triangleq \prod_{\forall \psi \in \Psi} \Xi_\psi$$

and the set of all basis formae by[7]:

$$\Xi(\Psi) \triangleq \bigcup_{\forall \psi \in \Psi} \Xi_\psi$$

We can also denote $c(\psi)$ the **cardinality**[8] of a given equivalence relation $\psi$ as is the number of equivalence classes it contains.

[7] If the combinations of the basis formae are also desired, replace $\Psi$ with $\Psi'$

[8] Radcliffe terms this **precision** [Radcliffe 94]. Cardinality is used here to make explicit the link between this and the concept of the same name used in the evolutionary computing literature — exactly why will become clearer later.

### 3.5.2 Derivation of Solution Encoding

From the above, any particular solution, $x \in S$, can be described in terms of a **representation function**, $\rho$. This is defined in terms of a set of *partial* functions for each of the equivalence relations in $\Psi$:

$$\rho_\psi : \mathcal{S} \to \Xi_\psi \text{ where } \rho_\psi(x) \stackrel{\triangle}{=} [x]_\psi$$

and $[x]_\psi$ is the equivalence class of $x$ under $\psi$. The representation function for the string as a whole, $\rho_\Psi : \mathcal{S} \to \Xi_\Psi$, is thus given by the combination of the partial representation functions, $\rho_{\psi_i}$, for all $\psi_i \in \Psi$:

$$\rho_\Psi(x) \stackrel{\triangle}{=} (\rho_{\psi_1}(x), \rho_{\psi_2}(x), \dots, \rho_{\psi_n}(x)) \equiv ([x]_{\psi_1}, [x]_{\psi_2}, \dots, [x]_{\psi_n}).$$

**N.B.** However, it may be the case that there are certain constraints on what equivalence classes can be used for a given equivalence relation with respect to the equivalence classes adopted for other equivalence relations — in which case the search space $S$ will use a subset of the above set of combinations.

The work in [Radcliffe 94] then notes that the above formalism suggests an encoding for the encoding space $\mathcal{E}$ as an image (direct encoding) of the induced equivalence classes in $\rho_\Psi(x)$. Also, he notes that provided that $\rho_\Psi$ is injective, then the growth function $g$ is simply the inverse of $\rho_\Psi$. Therefore it would appear that we have succeeded in our task of making the connection between the problem domain and the fitness landscape explicit and formalisable.

Finally, to find a suitable set of equivalence relations $\Psi$, the criteria of **coverage** must be satisfied (this ensures that $\rho_\Psi$ is injective). That is, the basis set of equivalence relations, $\Psi$, used to describe the search space, $S$, is said to cover it if and only if the following condition applies:

$$\forall x \in S,\ \forall y \in S \setminus \{x\},\ \exists \psi \in \Psi : \neg\psi(x, y)$$

### 3.5.3 Derivation of the Neighbourhood Operators

In addition, with the above formalisation, it is now possible to derive suitable neighbourhood operators. [Radcliffe 94] does this by producing *specifications* for the distance metric and neighbourhood operator. The distance metric, $d(x, y, \Psi)$, can now be defined as the number of features/equivalence relations in $\Psi$ that are *not* equivalent for two solutions $x, y \in S$ that are of interest:

$$d(x, y, \Psi) = \sum_{\forall \psi \in \Psi} 1 - \psi(x, y)$$

The neighbourhood of a solution (and therefore the behaviour of the neighbourhood operator) is specified in terms of a **minimal mutation** which define the set of solutions that differ *minimally* with respect to the basis set of equivalence relations, $\psi \in \Psi$. More formally for the operation $N : S \times \mathcal{K}_N \to S$, it must be the case that the set of solutions in the neighbourhood, $N(x, \Psi)$, of $x \in S$ is given by:

$$N(x, \Psi) \stackrel{\triangle}{=} \{y \in S \mid \neg\exists z \in S \setminus \{x, y\} : d(x, z, \Psi) < d(x, y, \Psi)\}$$

Given that we can change a solution by only one equivalence relation (as is often the case), the minimal mutation defined above for the neighbourhood returns the set of solutions in $S$ for which the distance is one, and the distance metric is the minimum number of such moves that are required to get from solution $x$ to solution $y$. This has an exact correspondence with the terms used for the fitness landscape, as was originally desired.

### 3.5.4 An Illustrative Example

The above will become more clear with a concrete example. Consider the max-ones problem — a common problem in the evolutionary computation literature where the aim is to maximise the ones in a string of binary digits of length $l$. As the value of the bits clearly correlate with fitness, a set of features can easily be set up to describe each solution for this problem by defining $l$ basis features, $\Psi = \{\psi_1, \ldots, \psi_l\}$, one for each of the bits in the problem where each feature can have an equivalence class (forma) from the set $\Xi_{\psi_i} = \{\xi_{(0,i)}, \xi_{(1,i)}\}$ — the encoding is then just the image of this. The distance metric for this feature set is then, from the definition above, the number of bits that are different — the hamming distance. In addition, the neighbourhood is obtained by taking one of the $l$ bits and changing its values. Both of these are the standard distance metrics and operators, and define exactly the fitness landscape, though with the additional benefit of making explicit and formalising in a general way the connection between the problem domain and the landscape.

Finally, it should be noted that the above will be described in more detail with more concrete examples to demonstrate that suitable feature sets can be found in practice, and extended to deal with recombination operators later. The coverage above will however suffice for the present.

## 3.6 Formalising the Search Through Algorithm Behaviour Space

This section will formalise the notion of the search of the space of algorithm behaviours given in the introduction. This formalisation should bring the following benefits.

- As noted above, there is a duality between encodings and operators; it would therefore be sensible to make the role of the fitness landscape on algorithm behaviour explicit.
- A given search technique, as defined by a set of transition rules, in fact covers a wide range of possible search algorithms (as defined by the NFL proofs), and therefore a large number of algorithm behaviours (in terms of the sequences of points in $\mathcal{S}$ visited).
- The set of search algorithms covered by such a search technique is usually a subset of the space of all search algorithms.
- The formalisation will assist in some of the arguments to be made later, as well as highlighting some related points made here.

First, a formal definition of a given **behaviour space**, $\mathcal{B}$, with respect to $\mathcal{S}$ is given which for the purposes of this discussion will be restricted to the case of non-repeating algorithms (though this will not affect the later arguments):

$$\mathcal{B}(\mathcal{E}, o, f, g, t) \equiv \mathcal{B}(\mathcal{L}, t) \stackrel{\triangle}{=} \{p_t(S_i|\mathcal{L}) \mid S_i \in S(\mathcal{S})\}$$

where $\mathcal{E}$, $\mathcal{L}$, $o$, $f$, $g$, and $t$ have been defined previously. $\mathcal{B}$ is defined as the set of probabilities, $p_t(S_i|\mathcal{L})$, (ie. the probability distribution) of the search technique in question visiting the search points in the sequence $S_i$ with a given $\mathcal{L}$ and $t$. Therefore probability theory considerations dictate that $\sum_{\forall i} p_t(S_i|\mathcal{L}) = 1$. The repeating counterparts of the above can in principle be modelled as well by employing similar extensions as those outlined earlier for NFL search algorithms.

The above captures the idea that the expected behaviour of an optimiser is an *ensemble* of its possible behaviours (since we do not know which of the possible search sequences will be taken). From this we can define a **performance measure** of an optimiser $\mu(\mathcal{L}, t)$, derived from the performance measure for a search sequence in $\mathcal{S}$, ie. $\mu(S)$. This measure is thus simply the expected performance over the probability distribution given by $\mathcal{B}$:

$$\mu(\mathcal{L}, t) \triangleq \sum_{\forall S_i \in S(\mathcal{S})} p_t(S_i|\mathcal{L}) \times \mu(S_i).$$

Suitable functions for obtaining $\mathcal{B}(\mathcal{L}, t)$ can be readily defined for all of the neighbourhood search techniques we are interested in by defining $p_t(S_i|\mathcal{L})$ appropriately, which in turn can be calculated from the product, $p_t(s_i|\mathcal{L})$, of the conditional probabilities for a point each of the search points, $s_i \in \mathcal{S}$ in the search sequence $S_i$, as shown below.

$$p_t(S_i|\mathcal{L}) = \prod_{\forall j} p_t(s_j|\langle s_0, \ldots, s_{j-1}\rangle, \mathcal{L})$$

where $p_t(s_j|\langle s_0, \ldots, s_{j-1}\rangle, \mathcal{L})$ is the probability that the partial search sequence will next visit point $s_j$ given the current search sequence, $t$, and $\mathcal{L}$. For our purposes it is more convenient to express this in terms of the weight (ie. relative likelihood) of a point being selected given a certain search history, given by the function $\omega(\langle s_i|s_{i-1}, \ldots, s_0\rangle, s_c, \mathcal{L})$, where $s_c$ is the point from which the search is currently being conducted from. Therefore $\omega_t(S_i|\mathcal{L})$ for the entire search sequence can be calculated in a similar fashion to $p_t(S_i|\mathcal{L})$ above. From this, $p_t(S_i|\mathcal{L})$ can be derived by a normalisation as shown below:

$$p_t(S_i|\mathcal{L}) = \frac{\omega_t(S_i|\mathcal{L})}{\sum_{\forall S_j \in S(\mathcal{S})} \omega_t(S_j|\mathcal{L})}$$

Of course, the above would only be useful if it were possible to produce a general expression for the techniques we are interested in. To illustrate that this is possible, take a (non-repeating) random walk as a first example, where the relative likelihood of visiting the first point is defined[9]:

$$\omega_{rw}(\langle s_0\rangle, s_c, \mathcal{L}) = \{\ 1\ :\ \ (s_0 \in \mathcal{S}) \wedge (s_c := s_0)$$

which simply states that the search can start anywhere in the search space, with assumed equal relative likelihood. From this we can define $\omega(\langle s_i|s_{i-1}, \ldots, s_0\rangle, s_c, \mathcal{L})$ for each of the $i$-th

[9] As the solution currently being visited/evaluated by the optimiser, $s_i$, will not necessarily be the solution that the next stage of search will be conducted from (denoted $s_c$), it is necessary to state explicitly when $s_c$ is changed — this is denoted by the expression $(s_c := s_i)$.

points, $s_i$, visited in the search space as follows:

$$\omega_{rw}(\langle s_i|s_{i-1},\ldots,s_0\rangle, s_c, \mathcal{L}) = \begin{cases} 1 & : \quad s_i \in N(s_c,\mathcal{L}) \;\wedge\; s_i \notin \langle s_0,\ldots,s_{i-1}\rangle \wedge (s_c := s_i) \\ 1 & : \quad \forall s_j \in N(s_c,\mathcal{L}):\; s_j \in \langle s_0,\ldots,s_{i-1}\rangle \\ & \quad\;\; \wedge\, s_i \notin \langle s_0,\ldots,s_{i-1}\rangle \wedge (s_c := s_i) \\ 0 & : \quad \text{otherwise} \end{cases}$$

where $N(s_{i-1}, \mathcal{L})$ denotes the elements in $\mathcal{S}$ that can be reached by applying the neighbourhood operators. The above expression states that the only possible additions to a given subsequence are those that are in its neighbourhood, unless the entire neighbourhood has been previously visited — in which case any unvisited point is acceptable (this is to ensure that all points in $\mathcal{S}$ are visited). The above equations can now be extended to a non-repeating version of any-ascent hillclimbing by simply changing $\omega_{rw}(\langle s_i|s_{i-1},\ldots,s_0\rangle, s_c, \mathcal{L})$ so as to disallow non-improving moves as follows:

$$\omega_{hc}(\langle s_i|s_{i-1},\ldots,s_0\rangle, s_c, \mathcal{L}) = \begin{cases} 1 & : \quad s_i \in N(s_c,\mathcal{L}) \;\wedge\; s_i \notin \langle s_0,\ldots,s_{i-1}\rangle \\ & \quad\;\; \wedge\, f(s_i) \geq f(s_{i-1}) \wedge (s_c := s_i) \\ 1 & : \quad s_i \in N(s_c,\mathcal{L}) \;\wedge\; s_i \notin \langle s_0,\ldots,s_{i-1}\rangle \\ & \quad\;\; \wedge\, f(s_i) < f(s_{i-1}) \\ 1 & : \quad \forall s_j \in N(s_c,\mathcal{L}):\; (s_j \in \langle s_0,\ldots,s_{i-1}\rangle \\ & \quad\;\; \vee\, f(s_j) < f(s_{i-1})) \\ & \quad\;\; \wedge\, s_i \notin \langle s_0,\ldots,s_{i-1}\rangle \wedge (s_c := s_i) \\ 0 & : \quad \text{otherwise} \end{cases}$$

Equations for other neighbourhood search techniques can be formulated in a similar fashion. For example, the equations for the threshold methods and first-ascent hillclimbing can be derived by a change to the fitness criteria in the above expression.

### 3.6.1 Optimiser Design as Transformations on $\mathcal{B}(\mathcal{L}, t)$

Now that the idea of a behaviour space of an optimiser has been formalised, it provides more of an handle upon the concept, presented in the introduction, of a search through the space of algorithm behaviours. The aim of this search is, in neighbourhood search terms, to find a $\mathcal{B}(\mathcal{L}, t)$ such that $\mu(\mathcal{L}, t)$ is maximised (most likely over a number of problem instances).

We can now see that changes to $\mathcal{L}$ and $t$ result in changes to $\mathcal{B}$. For instance, changing the traversal rules from those for hillclimbing, to (say) threshold accepting would expand the number of search sequences, $S_i$ in $\mathcal{B}(\mathcal{L}, t)$ that had non-zero probabilities. These additional search sequences would correspond to those that we 'blocked' by the presence of local optima. Now, if local optima were a problem for the landscape in that they prevented the search from moving to high quality points in the search space, we would then expect the new sequences in $\mathcal{B}(\mathcal{L}, t)$ to contribute to an increase in $\mu(\mathcal{L}, t)$

Similarly, a change of landscape would effect a change in $\mathcal{B}(\mathcal{L}, t)$. Now if the new landscape was more tractable to local search (this issue will be discussed later), then the new behaviour space, in combination with local search, would be expected to have a higher $\mu(\mathcal{L}, t)$ than before.

In summary, the process of optimiser design can be seen as a iterated hypothesis-experiment procedure. The designer uses the available knowledge to form hypotheses about which $\mathcal{L}$ and $t$ would lead to more efficient search. The validity of these hypotheses (and thus the designer's knowledge) are then tested by observing the effect the changes in $\mathcal{B}(\mathcal{L}, t)$ have on $\mu(\mathcal{L}, t)$. Unfortunately this formalism only takes us so far. It should be clear from the above that it would be unwieldy to calculate $\mathcal{B}(\mathcal{L}, t)$, and therefore $\mu(\mathcal{L}, t)$ directly.

For most of the remainder of this chapter, it will be described how $\mu(\mathcal{L}, t)$ can, for different landscapes, be estimated empirically and used to best effect.

### 3.6.2 Postscript: A Note on Hybrid Methods

As noted elsewhere [Rayward-Smith 94], **hybrid methods** have been touted as an answer to the current problems faced by practitioners of neighbourhood search, the premise being that some combination of, say simulated annealing, tabu search, EAs, and possibly some domain-specific heuristic is the *key* to practitioners being able to solve difficult real-world problems — thus by implication advocating that research effort should be directed towards developing more 'powerful' hybrid methods. However, it has been argued already that the designer should concentrate his design efforts towards aspects of the optimiser that correspond to the problem domain (i.e. the fitness landscape) rather than the search control — therefore placing emphasis on hybrid methods seems to contradict this philosophy.

For instance, an additive combination of two different search techniques will result in a repertoire of algorithm behaviours that is at least as large as the larger of the two sets (as the set of possible hybrids contains the techniques on their own). Therefore though it is a truism that the desired behaviour is more likely to be found in some expanded set of algorithm behaviours, the problem remains that this behaviour has to be found! Given that the task of selecting a suitable optimiser for a given landscape is already very hard, the task of selecting suitable values for the (more numerous) parameters for a hybrid would seem to be much harder. In short, the idea that hybrid methods, in themselves, can solve the problem of optimiser design is a red herring as it ignores the central issues of principled design and the exploitation of the designer's knowledge.

## 3.7 What Makes a Landscape Tractable?

Now that the connection between the features of the landscape and the problem domain, and their effect on optimiser behaviour (and therefore performance) has been made formal, it should be clear that the constructed landscape in many respects represents an hypothesis about the structure of the problem that has been used to assist the search. Therefore it is reasonable to take the position that *the effectiveness of a given optimiser is a direct function of the correctness of the hypotheses concerning the problem domain it embodies*.

In the face of a new problem it may well be possible to form a number of such hypotheses and thus it would be desirable to compare and test them in order to throw light on the correctness of these hypotheses and thus the structure and theory of the problem domain.

Given the above, some method by which the effectiveness of landscapes can be measured is required. This section will review and critique the current suite of methods in the literature that have been used to do this. It will be shown that although these methods correctly highlight components of what makes a tractable landscape, they do not capture the whole picture. The remainder of this chapter will turn its attention towards showing how direct measurements using hillclimbing experiments can be used as an effective experimental protocol to this end.

### 3.7.1 Problem Size and Density of States

The first and most obvious indicator of search difficulty is the number of points in the search space. Intuitively, the more points there are in the search space, then the more search has to be performed. This suffers from two problems. First of all, search difficulty for local search depends upon the structure of the fitness landscapes, and it therefore follows that a small landscape that has many local optima may well be harder to search than a larger landscape that has not. This is illustrated by Figure 3.4 below.

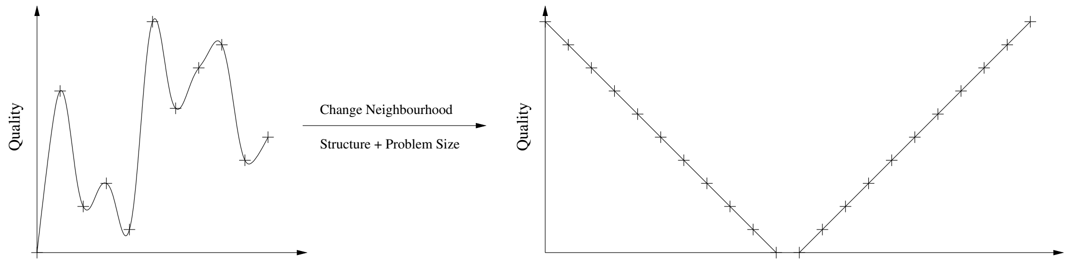


Figure 3.4: Proof That Size Does Not (Always) Matter

In addition, the *proportion* of points in the search space that are of the required quality is also important. This property of the search space can be measured graphically and quantitatively by the **density of states**; a concept taken from statistical thermodynamics. In effect, a plot is made of the number of search points/solutions of a given solution quality against their quality. Recent work [Rose *et al.* 96] has suggested that such a measure can be used as a measure of problem difficultly for a EA (and by implication, other neighbourhood search methods). This does make some intuitive sense as if the search space contains a high concentration of good quality solutions, then there are more to be found.

Unfortunately, in reality, this is only a reliable measure of search difficulty for *random* search. This is because this search difficulty metric, like the size of the search space, does not take into account the structure of the fitness landscape. An illustrative counterexample is given by Figure 3.5. Here a fixed search space undergoes a transformation to give a different neighbourhood structure, with the new neighbourhood structure clearly being less tractable for neighbourhood search than the original due to the presence of local optima.

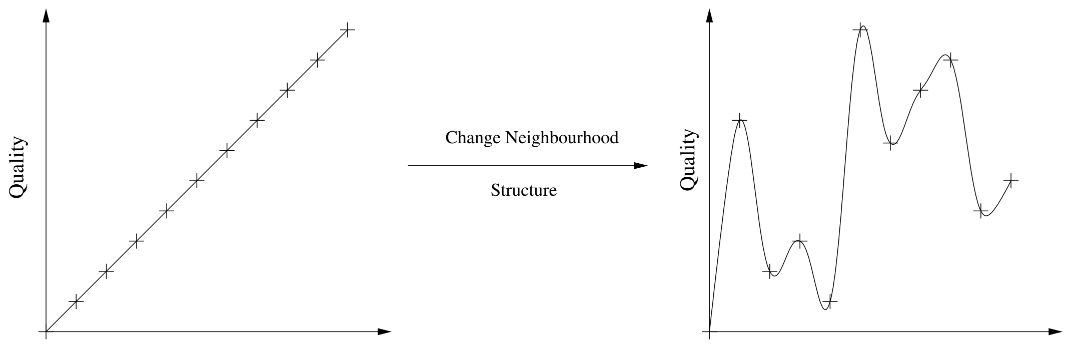


Figure 3.5: A Counter-Example to the Density of States

A related point arises from the desire to show statistically that transformations on the search space that remove poor quality or even invalid solutions from the search space do not remove the global or near global optima — an application of the the density of states metric was given in [Kappler *et al.* 96]. Unfortunately, as noted above, because this metric takes no account of the neighbourhood structure, the transformation may lead to a poorly correlated landscape and therefore the exercise may make the landscape *less* tractable. In addition, ensuring that the global optimum has not been accidently discarded, though desirable, may not be necessary. This is because we are after the best quality solution in the time available and therefore if a transformation is made that increases the optimiser's ability to do this *on average*, then the loss of the (small) chance of finding the global optimum is possibly justified. In fact, to produce this statistical check of whether the global optimum was removed [Kappler *et al.* 96] used the **Boltzmann ensemble method** [Rose *et al.* 96] to obtain the density of states distribution, this method, at a first glance, bears a striking similarity to a very slow simulated annealing procedure which begs the question of whether this is an effective use of CPU resources.

### 3.7.2 Neighbourhood Size and Path Length

Discussion of neighbourhood suitability in the OR literature [Papadimitriou & Steiglitz 82, Anderson 96], tends to concentrate on the size or **strength** of a neighbourhood. Simply put, large neighbourhoods, all things being equal, have fewer local optima. For example, given a defined basis seat of features (in the case of the TSP edges) a solution that is in a 2-edge local optimum may not necessarily be in a 3-edge local optimum, but all solutions in a 3-edge local optimum will be, by definition, in a 2-edge local optimum.

Given that the avoidance of local optima is the main reason for the extensions of hillclimbing described in this thesis, one would expect the search to be easier in this respect for large neighbourhoods. Apart from the obvious (and fatal) objection that the above metric of landscape suitability does not take into account the structure of the fitness landscape[10], the above argument also oversimplifies the situation somewhat.

Figure 3.6 shows the transformation of one landscape with a single optimum to another landscape, also with a single optimum, but also with a larger neighbourhood (four as opposed to two). It should be clear that the distance from any point to the optimum is less for the land-

[10] If the choice of basis features is poor then there will be little correlation in any landscape derived from it, even if you increase the neighbourhood size by changing more features at once.

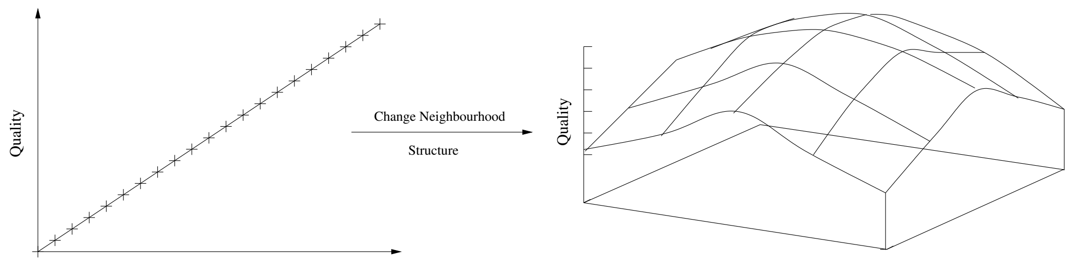


Figure 3.6: How Neighbourhood Size Changes Path Length

scape with the larger neighbourhood, and therefore we would expect a hillclimber to find the local optima faster on the landscape with the larger neighbourhood.

Though this alone would seem to further support the argument above, these gains come at a price. Exploring large neighbourhoods requires more evaluations before the acceptance criteria is met than small neighbourhoods, and this cost increases going from any-ascent hillclimbing, to first-ascent hillclimbing, and finally to steepest-ascent hillclimbing. Therefore a balance has to be struck between these competing factors, and no analytical work as been performed, as yet, to address this[11]. Therefore though the strength of a neighbourhood is a relevant factor it cannot, alone, be a metric for landscape difficulty.

### 3.7.3 Cardinality Arguments

In the EA community, the view remains in some quarters that the cardinality of the representation's alphabet is an important determiner of its suitability of a representation. This arises from considerations from the schema theorem [Holland 75] and studies of how many schema (i.e. formae) are processed by an EA. It was shown by [Holland 75], with extensions by [Bertoni & Dorigo 93], that schema are processed at a rate given as $O(N^3)$, where $N$ is the population size — this is termed **implicit parallelism**. This suggests that a representation should maximise the number of schemata (formae) in the EA population and lead to more information being processed (known as the **principle of minimal alphabets**). This argument thus suggests that the best representation be low cardinality, in other words binary. This view can however be objected to on a number of grounds, which cast great doubt upon its validity. These will be discussed in turn.

[11] At least not to my knowledge — discussion with mathematicians has indicated that solving even the random-walk case may well involve devising some new results in graph theory and Markov chain processes.

### The Case for *High* Cardinality Alphabets

Another way of looking at implicit parallelism is that a solution provides partial information about the fitness of all the subsets of the search space defined by the schema (formae) to which it belongs — the more sets available, the more are implicitly processed, and therefore more information is processed.

From this, the first of these objections was given in [Antonisse 89] who argues that in fact the don't care '$\#$' symbol in a schema (whose usual form is a template such as $1\#2\#\#0\#$) should instead be viewed as indicating all subsets of symbols at a particular position. Therefore a schema would indicate the *power set* of the encoded solutions that matched it, and as a result *high* cardinality alphabets would be preferable as they generate larger power sets!

### Forma Analysis Arguments

Work by [Radcliffe 91b, Vose & Liepins 91] argued that both arguments failed to note that the schema theorem applies to *any* arbitrary subset of the search space. Therefore if the notion of implicit parallelism is correct, the number of such subsets thus processed by the optimiser is *independent* of the representation used and is always $2^{|\mathcal{E}|-1}$ (where $|\mathcal{E}|$ is the size of the encoding space). Therefore, no representation leads to more or less information processing than another — thus resolving the contradiction between the principle of minimum alphabets and Antonisse's arguments above (they are both wrong!).

In addition, even if a particular cardinality representation were to be appropriate for the problem at hand, the issue still remains [Radcliffe 92] that there are $|S|!$ possible mappings, $g$, between the encoding space and the search space. Needless to say, the vast majority of these will lead to uncorrelated fitness landscapes and thus will be difficult to search. This therefore suggests that attention should instead concentrate on finding a mapping and neighbourhood operator that leads to a correlated landscape.

### Independence of Landscapes from Encodings

The final and possibly most damning objection to the minimal alphabets argument comes from the fact that, as noted earlier, landscapes are independent of any particular encoding scheme.

That is, for any encoding/operator pair that generates a neighbourhood structure, the same neighbourhood structure can be obtained for any other encoding, if given a suitable operator. Therefore any general claims for a particular cardinality of encoding seem, in this light, simply ridiculous.

Finally, the argument for the equivalence of encodings has been proved more rigorously in [Fogel & Ghozeil 97], which showed that no intrinsic advantage can be obtained by any cardinality of solution encoding, or for that matter any particular type of unary or binary neighbourhood operator. The latter point also extends work by [Culberson 95] which argued that EA crossover and mutation operators are, in fact, equivalent in a strong sense.

**Practical Experience**

Theoretical considerations aside, the empirical evidence for the effectiveness of non-binary alphabets is overwhelming. In the EA community itself, [Davis 91] noted early on that domain-specific operators are often useful, and the book [Michalewicz 92] devotes itself to the case for domain specific operators and describes a number of case studies that show how EAs with domain-specific operators can be used to solve difficult problems.

A particularly telling example of the above is in the application of EAs to function optimisation. According to the above cardinality arguments, some form of binary or grey coding would be appropriate and much of early work in the GA community was spent on investigating which of the two was more appropriate [Bäck 97], and some recent work has looked at this further [Whitley & Rana 98]. The ES and EP communities in parallel, decided upon encoding real numbers directly and using Gaussian mutations [Fogel *et al.* 66].

However, a number of studies (for example [Ingber & Rosen 92, Radcliffe 91b]) have shown that EAs and other neighbourhood search algorithms with real encodings have outperformed their binary coded counterparts. This has also been backed up by theoretical arguments for the use of real encodings such as [Salomon 96]. Finally, it should be noted that, rather depressingly, the use of binary codings for function optimisation problems still occurs, eg. [Donha *et al.* 97], which indicates that the low cardinality arguments still hold sway in some quarters.

### 3.7.4 Number of Optima

Given that our intuition of how neighbourhood search optimisers work, it would seem obvious to include the number of optima in the landscape in any list of problem difficulty metrics. This has, in fact been proposed by [Whitley & Rana 97, Whitley *et al.* 98, Whitley & Rana 98]. Though finding the *exact* number of optima would require evaluating *all* of the solutions on the landscape, this quantity can be estimated statistically by procedures such as capture/recapture. However, it is rather straightforward to devise landscapes that have no local optima, but would cause difficulties for neighbourhood search. The first landscape (on the left) in Figure 3.7 is an example of a 'needle on a haystack' landscape. As can be readily seen, this landscape would cause problems for neighbourhood search as there is no information available to guide the search to the optimum, thus reducing it to a random walk.

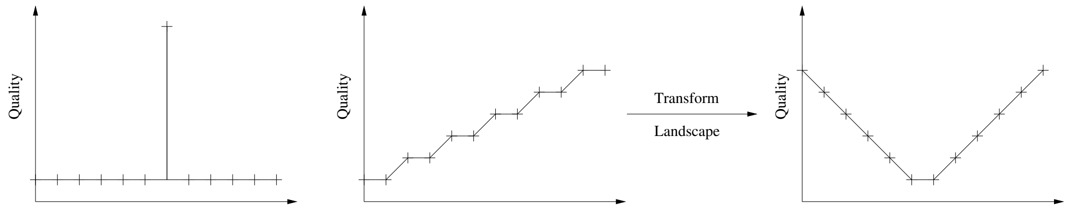


Figure 3.7: Where the Number of Optima Misleads

Furthermore a second objection, also shown in Figure 3.7 is possible. Here a landscape with one optimum causes some impediment to neighbourhood search due to the presence of plateaus. Transforming this landscape to one with two optima as shown above removes these plateaus and therefore allows the hillclimber to find one of the (global) optima more quickly.

Finally, it should be noted that some neighbourhood search algorithms (such as optima linking) exploit the relationship between local optima — in other words, it is assumed that local optima of high quality are close by other optima of high quality. Therefore the relative quality of the optima found in the search can be used to guide to search to higher quality optima. Though this 'big valley' structure has been found for a number of problems [Boese 96, Reeves 98], it is entirely possible that the optima will not possess this relationship. Therefore, the *arrangement* of local optima, and not just their abundance, is also of importance.

### 3.7.5 Correlation Analysis

None of the above metrics attempt to capture in any way the notion that the fitness landscape should be correlated. **Correlation analysis** is a blanket term for techniques that attempt to quantify the property. The techniques used in the literature are described below with comments and criticisms, where appropriate.

#### Fitness-Distance Correlation

The most notable of these techniques is **Fitness-Distance Correlation** (FDC) due to work in [Jones 95, Jones & Forrest 95], which requires that the global optimum be known already (which makes its practical use rather academic, though a suitably high quality solution could be used as a proxy). FDC takes a random sample of $N$ search points, $\{s_1, s_2, \ldots, s_n\}$, with their fitnesses (quality values), $\mu(s_i)$, and distances from the/a known global optimum, $d(s_i)$. The FDC coefficient, $\rho_{FDC}$, is thus defined as:

$$\rho_{FDC} = \frac{\sigma_{\mu,d}}{\sigma_\mu \sigma_d}$$

where

$$\sigma_{FD} = \frac{1}{N}\sum_{i=1}^{N}(\mu(s_i) - \bar{\mu})(d(s_i) - \bar{d})$$

and where $\sigma_\mu$, $\sigma_d$, $\bar{\mu}$, and $\bar{d}$ are the standard deviations and means of the fitnesses and distances from the optimum respectively. $\sigma_{\mu,d}$ is the *covariance* of $\mu$ and $d$ given above. From this [Jones 95, Jones & Forrest 95] defined three classes of problem difficulty for an EA. These were: easy ($\rho_{FDC} \leq -0.15$), difficult ($-0.15 < \rho_{FDC} < 0.15$), and misleading ($\rho_{FDC} \geq 0.15$). These were compared against a difficulty ranking of a number of binary coded EA benchmark problem, and a good correspondence was found between that and the difficulty predicted by FDC.

However, this metric suffers from a number of problems — a couple of counterexamples have been found empirically which casts doubt on its ability to predict when an EA would find a given landscape difficult [Altenburg 97]. For the purposes of this discussion, it is quite straightforward to construct a simple example where FDC misleads. Figure 3.8 below shows a plot of fitness against distance from optimum for the landscapes shown in Figure 3.6 earlier.

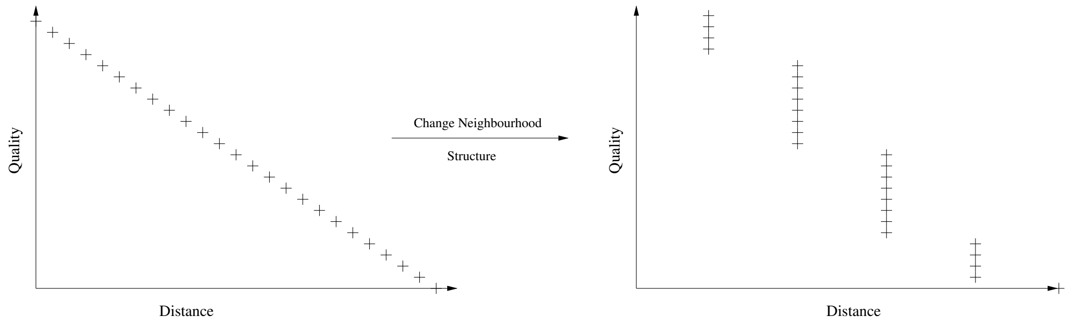


Figure 3.8: Where Fitness Distance Correlation Misleads

Thus, as the neighbourhood size is increased, solutions that were of different quality and distance from the optimum, now have the same distance. Therefore, as FDC measures the relationship between fitness and distance from the optimum, the perfect relationship $\rho_{FDC} = -1$ in the first case becomes reduced and $\rho_{FDC}$ becomes closer to zero. Unfortunately, as discussed earlier in 3.7.2, the second landscape is in fact the easier one for a hillclimber to search, and therefore FDC has been shown to mislead.

Finally, another illustrative counterexample to FDC is based on the argument that has been given by [Ross 98]. Consider the 'needle in a haystack' problem shown in Figure 3.7. Changing the quality value of the optimal solution, also changes the value of $\rho_{FDC}$. This is nonsensical as the problem is equally difficult for local search, no matter the quality of the optimal solution. That said, this problem of *scaling* could be fixed if *ranked* fitnesses where used instead.

### The Fitness Correlation Coefficient

The second of these methods equates a correlated landscape with the requirement that the fitnesses of the parents are correlated with those of their children. In other words, high quality solutions give rise to high quality solutions, and low quality solutions give rise to other low quality solutions when the neighbourhood operator is applied. This is measured quantitatively [Manderick *et al.* 91] by the **fitness correlation coefficient**, $\rho_{OP}$, which is given by:

$$\rho_{OP} = \frac{\sigma_{\mu_p,\mu_c}}{\sigma_{\mu_p}\sigma_{\mu_c}}$$

where for a $g$-ary operator, $OP$, $N$ sets, $i = 1, \ldots, N$, of $g$ parents, $\{p_{i1}, p_{i2}, \ldots, p_{Ng}\}$, are generated with their fitnesses, $\{\mu(p_{i1}), \mu(p_{i2}), \ldots, \mu(p_{Ng})\}$. These are then used to generate, using $OP$, one offspring each, $c_i$ with fitness $\mu(c_i)$. The expression above then calculates the correlation between parent and child solutions by calculating their covariance and standard deviations ($\sigma_{\mu_p,\mu_c}$, $\sigma_{\mu_p}$, and $\sigma_{\mu_c}$ respectively).

Therefore, if the underlying assumption of this metric is correct, then a high value of $\rho_{OP}$ should indicate that the operator, $OP$, and the landscape it induces is correlated and thus suitable for neighbourhood search. This assumption is supported by empirical evidence for the TSP and job shop scheduling problems [Manderick *et al.* 91].

However this metric, like FDC, takes no account of the effect of increasing the neighbourhood size. Therefore the example shown (in Figures 3.6 and 3.8), produces a similar problem for this metric as it does for FDC. In short, increasing the neighbourhood size *decreases* the correlation $\rho_{OP}$, implying greater landscape difficulty, when in fact Section 3.7.2 showed that this made the landscape more tractable. Finally, [Altenburg 95] notes that $\rho_{OP}$ can give misleading results in situations where the average offspring fitness does not increase with fitness, but the variance does. In this case, it was argued that an EA would be effective even if $\rho_{OP} = 0$.

**Autocorrelation Distance**

Another approach, [Weinburger 90a], is to analyse the quality of sequence of solutions visited by a random walk, denoted $\{s_0, s_1, \ldots, s_n\}$, and indexed by $h$, over the landscape where the pairs $s_{h-1}, s_h$ and $s_h, s_{h+1}$ are neighbouring search points for all $h = 1, \ldots, N-1$. From this we can define the **autocorrelation function**, $\rho(h)$, given below:

$$\rho(h) = \frac{1}{\sigma_\mu^2} R(h)$$

For all values of $h$ the value of the **auto-covariance function**, $R(h)$, can be estimated by the equations below:

$$R(h) = \frac{1}{N} \sum_{i=0}^{N-h} (\mu(s_i) - \bar{\mu})(\mu(s_{i+h}) - \bar{\mu})$$

where $0 \geq h > N$ and:

$$\bar{\mu} = \frac{1}{N+1} \sum_{i=0}^{N} \mu(s_i)$$

Therefore the autocorrelation function $\rho(h)$ measures how correlated points that are distance $h$ from each other are. It can also be shown for many problems, [Standler & Happel 92, Standler & Schnabl 92] that $\rho(h)$ is a exponentially decreasing function (ie. of the form $\rho(h) = e^{-ah}$). From this the **correlation length** $\tau$ can be defined as the distance, $h$, that $\rho(h) = 1/2$. Therefore large values of $\tau$ indicate smooth landscapes as this means that relatively distant points on the random walk are still correlated, and small values of $\tau$ indicate rugged, and uncorrelated landscapes that will be hard for neighbourhood search — empirical work supports this hypothesis [Manderick *et al.* 91]

Unfortunately, this suffers from a similar problem to the other correlation analysis techniques so far. In short, changing the neighbourhood size, as in Figures 3.6 and 3.8, increases the ruggedness of the random walk sequence, given that we would expect random walk distance to correlate with landscape distance, at least initially, as the tendency will be for the random walk to drift away from the starting position. Therefore for the purposes of this discussion, we can use random walk distance as a proxy for landscape distance (and vice versa).

Once this link has been made, then we can see that this metric also examines the correlation between the distance and fitness of solutions (although possibly in a more implicit manner). Therefore the same problems as described earlier will arguably arise and thus increasing the neighbourhood size in the example in Figure 3.6 will lead to a decrease in $\tau$, though this landscape is the easier to search.

Finally, it should be noted that this work has been extended to explore other methods of analysing the random walk statistics. For example, [Hordijk 96] extends the above to use a time-series analysis approach (the **Box-Jenkins** analysis [Box & Jenkins 71]). Also, [Vassilev 97] uses an information theoretic approach to derive the **information content** and **information stability**, analogous to $\rho(h)$ and $\tau$ respectively of the random walk sequence (though the information stability is defined with respect to the quality differences between solutions, rather than steps on the random walk). However, as the problem described above lies with the random walk sampling and how a change in neighbourhood size changes the fitness correlation between points, rather than the way it is analysed, these methods also suffer from the above drawback.

### 3.7.6 Analysis of Epistatis (Forma) Variance

The final metric of landscape performance arises from an analysis of the representation from which the landscape was derived. **Epistatis variance** was first suggested by [Davidor 90], and then later the **forma variance** metric was proposed by [Radcliffe & Surry 94a]. These metrics arise from a consideration of the building block hypothesis in EAs, and the realisation that landscape correlation is related to the amount of interaction between the elements of the representation (ie. genes, variables, or more specifically the features manipulated). In EAs this interaction is called as **epistatis**, and it is known that high epistatis implies an uncorrelated (even random) landscape with many local optima, whereas zero epistatis denotes a fully correlated search space [Kauffman 89].

The measurement of epistatis has also been cited as a way of identifying whether a landscape is particularly suitable for EAs, and more specifically recombination. In short, no epistatis means that hillclimbers will be more effective, high epistatis rules out neighbourhood search altogether, and therefore EAs are most suited to landscapes between these two extremes — however current arguments [Kauffman 89] for this remain qualitative.

In any case, the rationale given above for the use of epistatis to compare landscapes with respect to suitability to neighbourhood search seems reasonable. The approach in [Davidor 90] was, in effect, to decompose the variance in solution quality to the effects due to the linear contributions of the representation components (main effects), and the contributions due to their non-linear interactions (interaction effects). A high amount of interaction effect variance thus is equivalent to a high level of epistatis. The work in [Radcliffe & Surry 94a] approaches this differently by plotting the within-forma variance. However, this is broadly equivalent to the above. In both cases, the metrics devised were able to successfully predict representational difficulty for EAs.

Work in [Reeves & Wright 95a] has since compared the above approaches to statistical experimental design and **analysis of variance** (ANOVA). From this a more principled and straightforward metric was devised. It is well known in ANOVA that the summed square deviations of the main and interaction effects can be combined to give the total summed square deviation, or in other words:

$$SS_{Total} = SS_{Main} + SS_{Interaction}$$

From this [Reeves & Wright 95a, Reeves & Wright 95b] measures the amount of epistatis by the following ratio:

$$\eta = \frac{SS_{Interaction}}{SS_{Total}}$$

Experimental work in [Reeves & Wright 95b] was shown to indicate problem difficulty for an EA. However, [Reeves & Wright 97] note two objections to the general applicability of this metric. The first is that not all epistatis is harmful. This is because the interaction effects can either *reinforce* or *oppose* the main effects with respect to the direction that the quality information leads the search — these were called, in [Reeves & Wright 97], **benign** and **malign** interactions respectively. The second, and final, objection to the above comes from a general problem with **indistinguishable contrasts** or **alias sets**. In short, it is impossible with a sample to determine with certainty whether an apparent effect in the above analysis is really a result of the interactions to which is is supposedly related. Therefore, any sample contains insufficient information to enable a certain decision of the nature of the epistatis in the problem, thus forcing any analysis of the type above to be treated with caution.

### 3.7.7 Closing Remarks

In summary, it would appear from the arguments above that none of the above measures are entirely satisfactory with respect to comparing the effectiveness of landscapes with respect to algorithm performance. This arises because the metrics above consider either path length or landscape ruggedness/correlation, but not both. Unfortunately, no theory currently exists by which these factors can be usefully combined in a predictive manner. One possible reason for this, apart from its difficultly, may lie in the origin of all of the ruggedness metrics, the EA community, who have solely looked at landscapes arising from binary representations. Of course, if such a decision to fix the cardinality is taken, then the objections above do not arise — though such a choice, from the discussion above is arbitrary at best.

In addition, all of the techniques above require quite an extensive sampling of the search space, often with the assumption that the underlying landscape is isomorphic which it need not be. Such measures are thus expensive in terms of CPU time and this begs the question whether it would be less trouble just to run the algorithms we are interested in and measure and compare their performance directly! (Though they may still be useful for examining what aspect of a given landscape makes it (un)successful).

Finally, it would also be desirable to measure landscape effectiveness in such a way that it is independent of the optimiser being used. This would seem reasonable in light of both the separation made above between optimiser and landscape, and the commonality between the techniques we are considering.

## 3.8 Design Heuristics and Hillclimbing Experiments

The proceeding sections have demonstrated how the fitness landscape can be formally attached to the problem domain, and that the current suite of metrics are unequal to the task of measuring landscape suitability for neighbourhood search. From this, what is needed is a direct, accurate, way of estimating the *relative* performance of a number of *landscapes* from the relative values of their optimiser performance, $\mu(\mathcal{L}, t)$, such that the relative landscape performance is transferable across local search optimisers (ie. a change in $t$).

The transferability is an important, though as yet unexplored issue, as we would like to construct an optimiser in terms of the problem domain. Therefore consider the case where two different landscapes (hypotheses about the problem), $\mathcal{L}_1$ and $\mathcal{L}_2$ were compared with each other using, say, simulated annealing and threshold accepting. Now, if it was the case that for SA $\mu(\mathcal{L}_1, t_{sa}) \geq \mu(\mathcal{L}_2, t_{sa})$, but for TA $\mu(\mathcal{L}_2, t_{ta}) \geq \mu(\mathcal{L}_1, t_{ta})$ then this would mean that in effect a landscape for each optimiser needs to be constructed separately, *and* that the idea of a correct hypothesis about the nature of the problem would be different for each algorithm (which would make it difficult to construct an optimiser in terms of the problem domain theory).

This section will address this issue by arguing from first principles that the unfortunate situation above would not be expected to occur in practice. Therefore, hillclimbing experiments will be shown to be able to produce a direct and transferable test of landscape effectiveness — the design heuristic described here.

This will set the stage for later sections which will show how this test can be used to derive suitable operators for an EA. This will lead to the proposal of two additional design heuristics.

### 3.8.1 Design Heuristic: Relative Landscape Performance

The first of the design heuristics described here is aimed at addressing the first of the above problems. Its aim is to circumvent the problems of the landscape performance metrics discussed in Section 3.7 by directly sampling the landscape performance, $\hat{\mu}(\mathcal{L}, t)$ (recall that performance is defined by the user to suit the situation at hand). Therefore most of the discussion will focus on arguing that hillclimbing experiments can provide a suitable robust method of measuring $\hat{\mu}(\mathcal{L}, t)$ that is a good estimator of *relative* landscape performance across a range of neighbourhood search optimisation techniques. The design heuristic in question is given below.

> **Design Heuristic 2:** *In practice, it can be assumed that the relative performance of all fitness landscapes is invariant for all optimisers based on a given type of hillclimbing.*

More formally, for all traversal rules, $t$, in a **local search class** $\mathcal{T}$, and landscapes, $\mathcal{L}_A$ and $\mathcal{L}_B$, then we can assume that the following holds:

$$(\exists t \in \mathcal{T} : \mu(\mathcal{L}_A, t) \geq \mu(\mathcal{L}_B, t)) \Rightarrow (\forall t : \mu(\mathcal{L}_A, t) \geq \mu(\mathcal{L}_B, t))$$

The term 'local search class' demands some clarification. For the purposes of the design heuristics described here, local search algorithms are classified according to their acceptance criteria (see Chapter 1 for an explanation). Therefore the classes are any-ascent (stochastic), first-ascent and steepest ascent. The rationale for the above distinction, arises from the discussion about the effect of neighbourhood size earlier in Section 3.7.2. It was noted there that the cost of exploring a neighbourhood before a move is accepted increases as the acceptance criterion is moved from any-ascent to steepest-ascent. Therefore steepest-based optimisers would favour smaller neighbourhoods than any-ascent hillclimbers, and this effect could be strong enough to change the landscape ordering between optimisers based on a different local search class (this effect is investigated later in this thesis).

### 3.8.2 Justification

This design heuristic now needs to be justified. The first reason for adopting this heuristic is that the behavioural repertoire of hillclimbers is a subset of those (behaviours) of the techniques derived from it. In other words, these techniques cannot do worse than hillclimbing (as they can revert to being hillclimbers). As these extensions cannot be *too* severe (because otherwise they could not be called neighbourhood search optimisers), the implication is that the differences in the landscapes being considered will have a much larger impact on optimiser performance than the traversal rules.

This argument arises from one of the justifications for the first design heuristic — that is the ability of the set of possible landscapes to affect algorithm behaviour is far greater than that of the (restricted) set of local search traversal rules. Of course, a possible objection to the proposed design heuristic would then be to claim that the designer will not be considering the entire space of landscapes, just the landscapes that correspond to 'sensible' hypotheses about the problem domain. In this case, it could be argued that these landscapes would be sufficiently similar to allow the effect of the traversal rules to affect their relative performance. In reply, it should be noted with some interest that most of the large gains made in the design of optimisers for particular problems in the literature arise from a change in the neighbourhood structure rather than the specific optimisation technique — which speaks against this counterargument[12].

### Behaviour Space Argument

A more formal variant of this argument arises from an examination of *how* these hillclimbing extensions transform $\mathcal{B}(\mathcal{L},t)$. Let us modify the definition of $\omega(\langle s_i|s_{i-1},\ldots,s_0\rangle, s_c, \mathcal{L})$ given in Section 3.6 to that for a non-repeating threshold accepting traversal rule (where a non-repeating any-ascent hillclimber is the case where the threshold, $L$, is set to zero).

$$\omega_{ta}(\langle s_i|s_{i-1},\ldots,s_0\rangle, s_c, \mathcal{L}) = \begin{cases} 1 & : \quad s_i \in N(s_c,\mathcal{L}) \ \wedge \ s_i \notin \langle s_0,\ldots,s_{i-1}\rangle \\ & \quad \wedge\ (f(s_i) \geq f(s_{i-1}) - L) \ \wedge (s_c := s_i) \\ 1 & : \quad s_i \in N(s_c,\mathcal{L}) \ \wedge \ s_i \notin \langle s_0,\ldots,s_{i-1}\rangle \\ & \quad \wedge\ (f(s_i) < f(s_{i-1}) - L) \\ 1 & : \quad \forall s_j \in N(s_c,\mathcal{L}),\ (s_j \in \langle s_0,\ldots,s_{i-1}\rangle \\ & \quad \vee (f(s_j) < f(s_{i-1}) - L)) \\ & \quad \wedge\ s_i \notin \langle s_0,\ldots,s_{i-1}\rangle \wedge (s_c := s_i) \\ 0 & : \quad \text{otherwise} \end{cases}$$

Inspection of the above shows that as $L$ is increased, additional search sequences become possible. Furthermore it is should be clear after some thought, that these new sequences will be somewhat hillclimbing-like. More specifically, upon examination of the search sequences

[12] Of course a rigorous empirical study is required to confirm this — which is why this the main part of the experimental aspect of my PhD.

introduced by increasing $L$, much of the sequence would correspond to that given for hillclimbing, because those transitions between points would also be allowed by the hillclimbing traversal rules. The differences would arise, therefore, where hillclimbing traversal rules would not allow the move between neighbouring points because of their fitness differences, but these difference are within the bounds of $L$ — in this case these transitions would also be allowed by the random walk traversal rules.

Therefore, it should be obvious that $\mathcal{B}(\mathcal{L}, t_{hc}) \subseteq \mathcal{B}^{'}(\mathcal{L}, t_{ta}) \subseteq \mathcal{B}^{'}(\mathcal{L}, t_{rw})$[13]. In other words, as $L$ increases, the optimiser behaviour changes from purely local search to a random walk in a smooth progression. Though this implies that the set of local search traversal rules is fuzzy, it also means as the optimiser becomes less local-search like, the effect of different landscapes will become more similar (as a random walk is, relatively speaking, very insensitive to the fitness landscape). Therefore, at worst, we would expect differences in the relative order to disappear rather than be reversed (which is another reason why the $\geq$ relation was used in the formal definition of this design heuristic[14]). Also, in the case where the modifications to the traversal rules involve restarts (such as GRASP), the local search behaviour can be viewed as becoming more like enumeration/random search — which is *undoubtedly* insensitive to the underlying landscape!

### Local Optima, *etc* Cause Problems for *all* Optimisers

The second, more intuitive, reason for adopting this heuristic, is in many respects an extension of the above. In simple terms, problems that cause problems for hillclimbers, also cause problems for their derived techniques. To illustrate this, three problems common to *all* local search optimisers are shown in Figure 3.9 below.

Each case will be dealt with separately. In the case of local optima consider two similar landscapes, one of which has fewer local optima present. In either case, overcoming the local optima in the landscape will, from the above arguments, involve either a random walk behaviour, or a restart, which both slow down the search compared to the 'ideal' version of landscape where the search algorithm would be accepting improving moves all of the time. Therefore

[13] Where $\mathcal{B}^{'}(\mathcal{L}, t) = \{\ S_i \in S(\mathcal{S}) \mid p(S_i) > 0\}$, ie. the search sequences that are in the optimiser's behaviour space.

[14] The other is that as we will be, in practice, sampling $\mu(\mathcal{L}, t)$ and dealing with stochastic algorithms, it becomes possible that differences in $\mu(\mathcal{L}, t)$ may become hidden in the statistical noise.

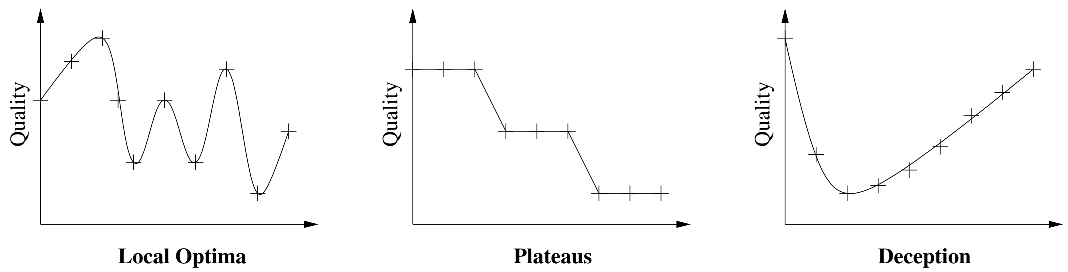


Figure 3.9: Problems Common to *all* Local Search Optimisers

no matter the degree of local search character of the optimiser, we would expect optimiser performance to be better for the landscape with fewer local optima.

Again, in the case of plateaus, any form of local search algorithm would be forced to adopt a random walk behaviour. In which case, any form of local search will perform worse on landscapes that possess plateaus than landscapes where these plateaus have been replaced by some local correlation (as noted above, most variants are somewhere between local search and a random walk). Finally, in the case of deception, *any* mechanism that exploits local correlations in the fitness landscape (ie. anything with a degree of local search character) will be misled away from the global optimum.

In summary, the common forms of traversal rule modification made to local search, although being able to reduce the impact of deviations from the ideal, such as local optima, do not overcome them completely. In short, though different local search traversal rules may be better suited to different deviations from the ideal landscape than others, they are all still affected by them. This point is especially valid as the above arguments have not yet considered implementational constraints such as the additional computational effort of these traversal rule modifications.

### 3.8.3 Implications

Given that the validity of this design heuristic is upheld, then it has useful practical implications. Most importantly, this design heuristic would allow the users to devise and experiment with hillclimbers first of all to ascertain which of the available candidate landscapes is most suited to neighbourhood search optimisers.

Moreover, in conjunction with forma analysis we now have the basis for an **experimental**

**programming** approach to the design of optimisers. Hypotheses concerning the features of the problem domain thought relevant to search can then be formalised and used to implement a working system, therefore allowing the hypotheses about the problem domain to be tested by examining the system's behaviour on the problem. This would appear to have distinct advantages given that focus of this work is upon designing optimisers in terms of the problem domain.

Finally, the approach outlined has the potential to reduce the amount of experimentation required — because this approach prescribes that the difficult issues concerning search control are left until the available hypotheses about the problem domain are fully studied. Additional investigations of options concerning the choice of traversal rules can be conducted *if required*. Given that the aim of optimiser design is to devise an algorithm that can find 'good enough' solutions in the time available, this approach should therefore minimise the amount of such experimentation.

## 3.9 Possible Shortcomings

The above design heuristic is currently a conjecture — so why propose it? The reason given earlier was if the ordering of landscape performance was to often vary between local search optimisers, then the ability to experimentally evaluate the correctness of hypotheses about the problem domain would be compromised. The section will highlight some possible shortcomings and how later parts of this thesis will try to address these concerns[15].

#### Academic Problems are too Easy

The other objection is that 'academic' combinatorial optimisation problems, such as the TSP are typically problems with linear constraints and a linear objective function. In fact could one expect that any 'reasonable' neighbourhood operator would produce a correlated landscape?

In response, at worst all this would mean is that in practice the hypotheses that a designer will draw up are not that far from the truth and thus the performance differences between the landscapes they induce may not be that large. This in itself comes as little surprise in light of the ease that many practitioners have in producing initial working systems!

[15] With thanks to Andy Harrison for his useful input here.

In any case, the possibility cannot be dismissed that problems with constraint/objective function non-linearities could produce such counter-examples. This work addresses this issue of the possible effect of (non)linearity as one of the case studies considered later is a real-world problem with non-linear constraints and objective function — thus providing a test of whether these types of problems cause difficulties to the proposed design heuristic.

### 3.9.1 A Counter-Example

Given that we are dealing with a *heuristic*, there will of course be counter-examples. Though the duality between landscape and traversal rules has been broken, there is still some scope for the choice of traversal rules to affect the comparative landscape performance (but hopefully not much — which is what the above justifications are in effect arguing). However, given that the assumptions behind this design heuristic have been made explicit above, even its failure should provide useful direction for further experimentation (and therefore the design process). Furthermore, it can be argued that the counterexamples would not present too much of a problem, if we keep in mind what we are trying to achieve. These points will be illustrated by an example. Figure 3.10 shows two landscapes. The first is characterised by a tooth-saw pattern with many local optima, and the second by two optima (one local, one global).

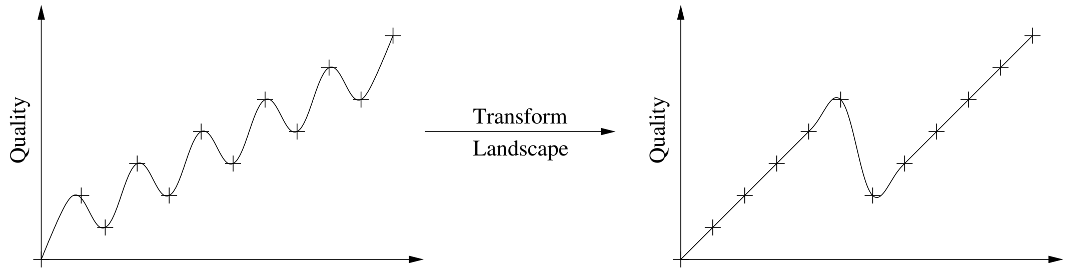


Figure 3.10: An Interesting Counterexample to the Second Design Heuristic

Now consider the action of a basic any-ascent hillclimber (without restart) on the two landscapes — it should be apparent that the hillclimber will find, on average, a higher quality solution for the second landscape. However, the situation could in principle be reversed when simulated annealing is used. This is because SA can 'hop' over the smaller local optima in the first landscape (thus improving its performance), though in the case of the second landscape, the ability to take backwards moves will slow down the optimiser's ability to locate local optima (though the ability to escape local optima may not be enough to get over the higher 'walls' between the basins of attraction).

In this case, the design heuristic appears to be in trouble (of course the factors could balance out so this is not a problem). That said, the use of another form of search control, such as iterated hillclimbing, would have the opposite effect — that is to further favour the second landscape. This is because restarting the search means that finding an optimal (or near optimal) solution is reduced to the chance event of the search starting in that optima's basin of attraction — it should be clear on inspection of Figure 3.10 that the chance of the search starting in the 'right' basin of attraction is greater for the second landscape.

It is therefore pertinent to question in this case whether it would be better to accept the results of the design heuristic (which gives us a landscape upon which hillclimbing does relatively well), and then, if necessary, select some extension of the hillclimber to address any deviations from the ideal case that may arise.

### 3.9.2 Design Heuristic = Design Conjecture?

A significant part of this chapter has tried to justify why one would not *expect* to encounter violations of this design heuristic *in practice*. The arguments used above, have been largely intuitive and, for example, the use of the one-dimensional line search arguments may not necessarily hold at higher landscape dimensionalities. That said, given the intuitive appeal of this approach, some solid theoretical arguments as to *why* and *when* this (commonly used) analogy breaks down and affects the above justification would be required before it is rejected out of hand.

In any case, the above approach is largely due to the current limitations of the theory of how neighbourhood search optimisers traverse landscapes — hopefully this work may suggest some later theoretical work to supplement the arguments made here. Therefore, we must resort to an empirical approach to evaluating the above design heuristic and those proposed later, and the experimental studies later in this thesis are thus an important test of the above arguments. This will be the aim of the later chapters of this thesis.

## 3.10 Design Heuristic: Do Hillclimbing Dynamics Approximate EA Dynamics?

So far the discussion has (implicitly) concentrated upon optimisers that maintain one search point from which the current search is conducted. However, evolutionary algorithms maintain a set of such points during the search, which means in some respects that they can be considered as a local search class in their own right. The question of whether the design of these population-based techniques can also be informed by hillclimbing experiments is addressed by the design heuristic below.

> **Design Heuristic 3:** *Hillclimbing dynamics approximate evolutionary algorithm dynamics sufficiently well*[16] *for evolutionary algorithms to be considered any-ascent (stochastic) hillclimbers for the purposes of relative landscape performance.*

In more formal terms, this design heuristic can be stated as:

$$(\exists t_i \in \mathcal{T}_{shc} : \mu(\mathcal{L}_A, t_i) \geq \mu(\mathcal{L}_B, t_i)) \Rightarrow (\forall t_j \in \mathcal{T}_{ea} : \mu(\mathcal{L}_A, t_j) \geq \mu(\mathcal{L}_B, t_j))$$

The implications of the above should be obvious. Given that we have performed experiments using an any-ascent hillclimber (or an optimiser derived from it), then if the use of an evolutionary algorithm was, for whatever reason deemed appropriate then the results of the earlier landscape comparisons would be transferable — therefore removing the need to repeat those experiments and making the choice of mutation operator effectively automatic. Moreover, we can see that the second design heuristic described in Section 3.8.1 also applies to EAs by applying modens ponens to the above design heuristic, assuming that the precedent of the above implication is true (which corresponds to the second design heuristic for the case of an any-ascent hillclimber).

### 3.10.1 Justification

The assumption made by the above design heuristic is that hillclimbing dynamics approximate evolutionary algorithm dynamics sufficiently closely for any differences in algorithm

[16] Of course the algorithm dynamics will not be *exactly* the same as they are not the same algorithm.

behaviour to be subsumed by the differences in optimiser behaviour due to the fitness landscapes being considered.. This is directly analogous to the argument made in Section 3.8.1. Therefore we need to establish that the search dynamics of these two neighbourhood search classes are closely related.

First of all, any-ascent hillclimbers are a special case of an evolutionary algorithm. More specifically, they can be considered to be a mutation-only (naive) EA with a population containing one solution. Therefore, given the arguments presented earlier (Section 3.8.2) in support of the previous design heuristic, showing how evolutionary algorithm dynamics relate to any-ascent hillclimbing dynamics as the population increases, in the sense of whether they rely upon similar landscape properties being present, is necessary. If it can be shown that they are sufficiently similar then the above design heuristic is justified.

Fortunately, this can be seen by inspection of a model of EA dynamics from the EA/population genetics literature — **Price's covariance theorem** [Price 70]. This was proposed as a model of EA dynamics by [Altenburg 95], and has found use in other EA investigations, for example [Langdon 98]. Consider the dynamical system defined by a canonical evolutionary (genetic) algorithm:

$$p_{t+1}(s_x) = \sum_{s_y, s_z \in \mathcal{S}} p_t(s_x|s_y, s_z) \frac{\mu(s_y)\mu(s_z)}{\sum_{s_i \in \mathcal{S}} \mu(s_i) p_t(s_i)} p_t(s_y) p_t(s_z)$$

where $p_t(s_i)$ is the probability that $s_i$ present in the EA population at the $t-$th EA iteration, and $p_t(s_x|s_y, s_z)$ is the conditional probability of $s_x$ being produced by an EA operation.

**N.B.**: We are currently considering a naive EA (ie. no recombination), therefore this expression is more general than is currently needed as it contains the notion of a binary neighbourhood operator (this case will be considered shortly in the context of the next design heuristic). However, a naive EA can be seen to correspond to the case where $p_t(s_x|s_y, s_z)$ is independent of $s_z$, and the terms referring to $s_z$ are removed. As a result, the conclusions that will now be made are unaffected.

Price's theorem can then be used to predict the expected values of **measurement functions**, such as the population mean fitness $\bar{\mu}$ in the next EA iteration (generation):

$$\bar{\mu}_{t+1} = \bar{\phi} + Cov\left[\phi_{s_y s_z}, \frac{\mu(s_y)\mu(s_z)}{\sum_{s_i \in \mathcal{S}} \mu(s_i) p_t(s_i)}\right]$$

where $\bar{\phi}$ is the contribution due to selection only, and the second term describes the covariance between the parent and child solutions. The expressions for $\bar{\phi}$ and $\phi_{s_y s_z}$ are given below:

$$\phi_{s_y s_z} = \sum_{s_x \in \mathcal{S}} \mu(s_x) p_t(s_x | s_y, s_z) \quad \text{and} \quad \bar{\phi} = \sum_{s_y, s_z \in \mathcal{S}} \phi_{s_y s_z} p_t(s_y) p_t(s_z)$$

In short, Price's theorem shows that the covariance between parent and child fitness is what drives the evolution of the population [Altenburg 95]. More specifically, a corollary of this [Altenburg 95] is that the expected *increase* in the population mean fitness (indirectly) depends upon the above covariance being high. As this is defined by the properties of the fitness landscape, this implies that suitable landscapes of EAs would possess a high parent-child fitness covariance.

Now recall that in the earlier discussions (Section 3.7.5) one of the aspects of a 'good' landscape for local search is that the landscape is correlated in the sense that good quality solutions are close to other good quality solutions, therefore accepting improving moves will guide the search towards higher quality solutions. This notion was emphasised by the fitness correlation coefficient which measured parent-child covariance directly.

Given this commonality, one could assume that the structure of a landscape that is suitable for an any-ascent hillclimber will also be suitable for a naive EA (and the above design heuristic holds). However, as noted in earlier discussions, other factors also influence landscape suitability and so for the moment we must assume that these do not act to contradict this design heuristic. Thus, an empirical study is needed again, and the two case study chapters later in this thesis will provide this.

## 3.11 Forma Analysis Revisited: The Derivation of Recombination Operators

What has not been explicitly discussed so far is that EAs possess an additional *binary* neighbourhood operator — crossover (or recombination). We have now obtained a formalisation of the problem features that induce unary neighbourhood operators (and therefore landscapes), and an experimental protocol for assessing the correctness of the hypotheses leading to the selection of these features. However, it would be useful if this could also be used to inform the design of recombination operators. This can be split into two separate questions. First, can the forma analysis formalism be used to specify suitable recombination operators? Second, is the choice of features suggested by the hillclimbing experiments appropriate for both unary and binary neighbourhood operators.

The answer to the first question can be answered in the affirmative — it is indeed possible to formally specify EA recombination operators. This arises from a consideration of the desirable properties of recombination operators, and their formalisation performed by [Radcliffe 94]. These properties are described below:

- **Purity**. This requirement is simply that if the two parent solutions are identical, then the child solution is identical to its parents as well.
- **Respect**. Features common to both parents should be present in the child solution. If this is achieved then the recombination operator is **respectful**. [Radcliffe 94] refers to the set of solutions that satisfy this condition as the **similarity set** of the parent solutions.
- **Transmission**. For **full transmission**, every feature in the child solution must also be present in at least one of the parents. The set of solutions that satisfy this requirement is termed the **dynastic potential** [Radcliffe 94]. Note that transmission includes respect (and therefore purity) as it is a stronger condition. If transmission can only be achieved some of the time, then the recombination operator is said to be **weakly transmitting**.
- **Assortment**. All combinations of formae that are present in the parents should appear in at least one of the child solutions generated by the recombination operator, if the combinations are compatible. If this can be performed by one application of the recombination operator then it **properly assorts**, if multiple applications of the recombination operator is required, then the recombination operator **weakly assorts**.

All of the above arise from considerations on what the role of an effective recombination operator is. Purity is important as otherwise the child solution will not be between the parents with respect to the metric space induced by the equivalence relations. Similarly, respect and transmission also arise from the need, discussed earlier in Section 3.12.1, for the child solution to be between the parents, either by sharing common features (respect) or by consisting entirely of parental material (transmission). Finally, assortment arises from an analogy with the principle of meaningful building blocks [Goldberg 89c] described earlier.

### 3.11.1 Representation Independent Operators

The task is now to define recombination operators of the form $X : \mathcal{S} \times \mathcal{S} \times \mathcal{K}_X \rightarrow \mathcal{S}$[17] that satisfy the design criteria above, given a basis set of equivalence relations $\Psi$. Radcliffe achieves this by defining 3 basic *representation independent* operators that are described and discussed below. It should be noted that the description of these operators is purely *declarative* and therefore they make no commitment about *how* they are to be implemented with respect to both the actual data-structures used[18], or the procedural aspects of the implementation.

**Random Respectful Recombination ($\mathbf{R}^3$)**

This operator ensures that all features that are common to both parent solutions are in the child solution. Thus the operator randomly selects a child solution, $z$, out of the similarity set of the parent solutions $x$ and $y$. The similarity set, $x \odot y$ can be defined as the most specific forma $\xi' \in \Xi_{\Psi'}$ that contains both $x$ and $y$:

$$x \odot y \triangleq \bigcap \{\xi' \in \Xi_{\Psi'} \mid x, y \in \xi'\}$$

Now, as the set of solutions produced by $R^3$ just has to confine itself to the solutions in $x \odot y$, a formal specification of $R^3$, in terms of equivalence relations, is given by:

$$R^3(x, y, \Psi) \triangleq \{z \in S \mid \forall \psi \in \Psi : \psi(x) = \psi(y) \Rightarrow \psi(z) = \psi(x) = \psi(y)\}$$

or, more concisely, in terms of the basis formae:

$$R^3(x, y, \Psi) \triangleq \{z \in S \mid \forall \xi \in \Xi_\Psi : x \in \xi \land y \in \xi \Rightarrow z \in \xi\}$$

where the actual child solution, $z$ is chosen out of the set above at uniform random.

**Random Transmitting Recombination ($\mathbf{RTR}$)**

This operator selects a child solution, $z$, out of the dynastic potential of the parent solutions $x$ and $y$. Thus a formal specification of $RTR$, in terms of equivalence relations, is given by

[17] Where $X$ denotes the recombination operator, and $\mathcal{K}_X$ the control parameter of the operator.

[18] This is why the operators are defined in terms of $\mathcal{S}$ and not $\mathcal{E}$ The basis equivalence relations are defined with respect to properties of the solutions in the *real world* and therefore any operator implementations that produces the same behaviour in $\mathcal{S}$ are equivalent as far as optimiser behaviour (and thus performance) is concerned. The specification of an operator in terms of $\mathcal{E}$ arises when these relations are later mapped (via $g^{-1}$) onto actual data-structures.

saying that for all basis equivalence relations, the equivalence class for the child solution, $\psi(z)$, *must* be present in either or both of the parent solutions. A formal specification of $RTR$, in terms of equivalence relations, is given by[19]:

$$RTR(x, y, \Psi) \triangleq \{z \in S \mid \forall \psi \in \Psi : \psi(x) = \psi(z) \ \oplus \ \psi(y) = \psi(z)\}$$

or, more concisely, in terms of the basis formae:

$$RTR(x, y, \Psi) \triangleq \{z \in S \mid \forall \xi \in \Xi_{\Psi} : x, z \in \xi \oplus y, z \in \xi\}$$

where the actual child solution, $z$ is again chosen out of the set above at uniform random.

#### Random Assorting Recombination ($\mathbf{RAR}_\omega$)

In the case of $RAR_\omega$, the specification is far more naturally and concisely given in terms of the combinations of the set of basis formae $\xi^{'} \in \Xi_{\Psi'}$. In these terms, $RAR_\omega$ is an operator that returns the set of child solutions, such that for all *combinable* pairs of formae present where the first parent has one formae and the second parent the other, there is at least one child solution, $z$ in the set of child solutions that contains both formae together. Therefore $RAR_\omega$ is best defined in terms of the assortment condition below:

$$\forall \xi_1^{'}, \xi_2^{'} \in \Xi_{\Psi'}, \forall x \in \xi_1^{'}, \forall y \in \xi_2^{'} : RAR_\omega(x, y, \Psi) \cap \xi_1^{'} \cap \xi_2^{'} \neq \emptyset$$

where $RAR_\omega(x, y, \Psi)$ denotes the set of solutions generated by the application of this operator for a given $x$, $y$, $\Psi$. The parameter, $\omega$, is an implementational detail, as in some representations (see below) it is not possible to have strict assortment whilst also having respect and/or full transmission. In this case, the selection of child solutions that also happen to respect their parents is biased in their favour according to the value of $\omega$.

### 3.11.2 Orthogonality and Separability

As noted above, depending on the basis set of equivalence relations used, it is not always possible to have proper assortment whilst also having respect and/or full transmission. Examples include problems involving permutation of $n$ elements, eg. an ordering of jobs to a machine, $\mathcal{P}_n$ which have to be optimised[Radcliffe 94]. Given that this is the type of problem that will

[19] Note that $\oplus$ denotes ‘exclusive-or’.

be tackled in the later case studies, some further definitions are needed for later use (Chapter 5).

A set of formae $\Xi_\Psi$ induced by the basis set $\Psi$ is **separable** if and only if respect and proper assortment are compatible, and **g-separable** if and only if full transmission and assortment are compatible [Radcliffe 94] (it should be obvious that g-separability implies separability).

One important class of g-separable basis sets are those that are **fully orthogonal**[20]. To define this, **orthogonality to order** $k$ is first defined for a basis set as: given any $k$ basis formae, induced by $k$ different basis equivalence relations ($\mathcal{O} \subset \Psi$), then the condition for orthogonality to order $k$ is satisfied if their intersection is non-empty.

$$\forall \mathcal{O} \subset \Psi(|\mathcal{O}| \leq k)\ \forall \xi \in \Xi_{\mathcal{O}} : \bigcap_{i=1}^{|\mathcal{O}|} \xi_i \neq \emptyset$$

A fully orthogonal basis set is defined as one that is orthogonal to order $|\Psi|$.

### 3.11.3 Comparison with 'Standard' EA Operators

By far the most common representations used in the EA literature are for problems formulated as optimisation over binary and $N$-ary decision variables. Taking each of the decision variables to form a basis set of equivalence relations, both of these representations are fully orthogonal. Therefore for $N$-ary strings, $RTR$ and $RAR$ are both equivalent to uniform crossover [Radcliffe 91a], $R^3$ is equivalent to a crossover operator that preserves common variable values that are in the parents, but assigns random values to the other variables, and a minimal mutation is the selection of one of the variables, following by assigning that variable a value randomly from the set of allowed values. In the binary case, all of $R^3$, $RTR$ and $RAR$ are equivalent to uniform crossover [Radcliffe 91a], and as noted in Section 3.5.4, the minimal mutation is standard bit-flip mutation.

Other operators are of course possible, though they can often be seen as variants of the standard forms above. For instance, there may be case to select a subset of any of the above operators, or to bias the selection of the child solution towards solutions that are thought to be of higher quality. In addition, other variants can be derived from relaxations of the above criteria. That said, such variations are describable in the declarative and representation-independent manner

[20] Radcliffe in fact uses the term 'orthogonal', this term is made more specific here to avoid confusion

above.

One such example of this is the representation-independent analogue of the N-point crossover operator (Generalised N-point Crossover — GNX) devised by [Radcliffe & Surry 94b]. As will be described later, interactions between formae often have structure which can be exploited, which for some problems have a large impact upon performance. GNX may be thought of as a version of RTR[21] where a subset is selected that satisfies the following condition. Given that we have imposed some spatial ordering upon the basis set of equivalence relations[22], we can then count the number of *changes* in the parental origin of the material. For example if the equivalence classes $\psi_{1,2,\ldots,j}(x) = \psi_{1,2,\ldots,j}(z)$ and $\psi_{j+1,j+2,\ldots,n}(y) = \psi_{j+1,j+2,\ldots,n}(z)$ then there has been *one* change in the parental source of the child's equivalence classes while going from $\psi_1$ through to $\psi_n$. We can therefore set an upper limit $N$ on the number of such changes allowed before a solution is excluded from the set of solutions produced by GNX. For orthogonal representations such as the binary and $N$-ary representations, this is exactly equivalent to the standard N-point crossover operator.

## 3.12 Design Heuristic: Recombination Issues

Given that the discussion so far has been able to use hillclimbing experiments to make transferable decisions concerning unary neighbourhood operators, it would also be desirable if the same approach could be used for binary neighbourhood operators. This consideration leads to the proposal of the final design heuristic covered here.

> **Design Heuristic 4:** *EA recombination operators should operate in the same metric space as the mutation operators.*

In forma analysis terms, Section 3.11 shows that this is equivalent to saying that the recombination operators should manipulate the same features (which in turn define the distance metric of a landscape — Section 3.5.3). This design heuristic will now be justified and then its implications discussed.

[21] In [Radcliffe & Surry 94b] the requirement for full transmission in the GNX operator is, in fact, relaxed slightly.

[22] For example, by numbering them $\psi_1, \psi_2, \ldots, \psi_n$ corresponding to the positions of the features on a string in the case of the max ones problem.

### 3.12.1 Justification

Various metaphors have been used to motivate the use of the recombination operator. From a biological perspective, the role of recombination is to bring together features associated with high quality solutions in the hope that they can be combined to produce even higher quality children (this is the idea behind the EA building block hypothesis [Goldberg 89c]) — this is also the view taken by the epistatis (forma) variance work described in Section 3.7.6. That said, for the earlier discussions, mutation landscapes/metric spaces that are amenable to local search, are derived from features that are strongly related to fitness — exactly what is needed for effective recombination.

Alternatively, a metaphor that is more suited for our purposes is given by work in [Reeves 94a] where it is argued that the recombination operator can be seen in neighbourhood search terms as a binary operator that searches the landscape *between* solutions (rather than around them as for the unary operators). In short, given a binary neighbourhood operator that takes two parent solutions $s_y, s_z \in \mathcal{S}$ and produces a child solution $s_x \in \mathcal{S}$, then recombination operators, given a landscape $\mathcal{L}$ can be considered to be those that satisfy the following condition:

$$\forall s_x, s_y, s_z \in \mathcal{S} : d(s_y, s_z, \mathcal{L}) \geq d(s_x, s_y, \mathcal{L}), d(s_x, s_z, \mathcal{L}) \geq 0$$

N.B.: Work in [Jones 95] argues that each neighbourhood operator, including recombination, should be viewed as possessing its own landscape. However this brings with it a number of complications, one of which is that the recombination landscape is dependent upon the current state of the EA population. Therefore though the view advocated in [Jones 95] is correct, the equivalent view of examining recombination operators with respect to the mutation landscape will be used here.

Moreover, from the discussion above (Section 3.10.1) concerning the EA dynamics work on Price's theorem and its implications for the properties of suitable landscapes, it should be clear that a suitable crossover operator should possess a high parent-child covariance. Therefore, if a relationship can be shown between the mutation landscape and the parent-child covariance, then this would support the above design heuristic. The relationship between crossover performance and the mutation landscape is best illustrated by considering the effectiveness of recombination as viewed on correlated and uncorrelated mutation landscapes between the same two points in the search space (Figure 3.11).

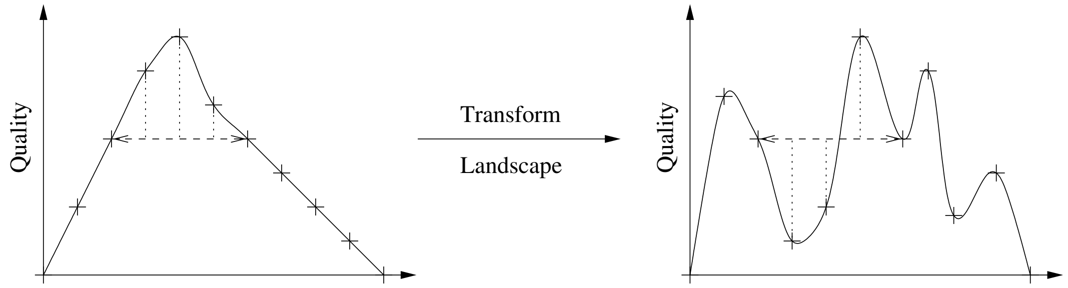


Figure 3.11: How Crossover Performance Relates to the Mutation Landscape

The landscape on the left has a high parent-child fitness covariance and is tractable to local search. In addition, it can be seen that recombination is effective also, especially in the important sense that high quality solutions are more likely to produce other high quality solutions. A transformation of the landscape is then made to produce a landscape that is less tractable to neighbourhood search (there are local optima, and the parent-child fitness covariance is reduced). In this case, as can be seen, an uncorrelated mutation landscape is unsuitable for any recombination operator based upon it — the quality of the parent solutions is a poor indicator of the quality of the child solution produced. This example therefore demonstrates that there is a commonality between mutation and crossover landscapes and supports the above design heuristic (which will be shown empirically later in this thesis).

### 3.12.2 Implications

This point has important and useful implications for EA design, because the user can use hillclimbing or a naive EA to ascertain whether his beliefs in what features of the problem to use are correct, then transfer them with confidence to the crossover operator. Therefore it now appears perverse that some of the literature (eg. [Cartwright & Tuson 94, Reeves 95a]) ignores this by using crossover operators that do not correspond to the mutation operators. That said, a possible reason may be that recombination is almost always depicted as swapping portions of strings — whether or not this was reflected in the actual choice of mutation operator.

## 3.13 Summary

This chapter has shown how a systematic problem-centred approach to the design of neighbourhood search optimisers is both desirable and necessary. It advocates a formalisation of the fitness landscape that attaches it to the concepts in the problem domain theory. Moreover, an experimental programming approach was proposed that, in conjunction with the above formalisation, allows hillclimbing experiments to be used to make design decisions that are transferable across optimisers.

Examination of the methodological criteria presented in 1.5.1 shows that most of the criteria have been addressed. The methodology proposed here emphasises both the common features of local search optimisers and frames the optimiser design mostly in terms of the problem domain which the end-user can more readily understand. Furthermore, it also outlines a theoretically justified and systematic procedure for the design of these optimisers.

However, how the above can help to organise the literature, assist in the teaching of these techniques, and to suggest future lines of research is still unclear. That said, this chapter is not the end of the journey (merely the beginning). Also, apart from addressing the remaining methodological criteria, there are in fact other ways in which knowledge of the nature of the problem domain can be exploited by a neighbourhood search optimiser. These issues will be addressed in the next chapter.

# Chapter 4

# No Optimisation Without Representation

This chapter will first describe the knowledge based systems paradigm and highlight how it can be seen as a natural extension of the arguments made in Chapter 3. The origins and roles that domain knowledge plays in neighbourhood search will then be described and, from this, the various sources of knowledge to the optimiser will be described. It will then be outlined how each knowledge source can be mapped onto the optimisation algorithm, a formalisation suggested, and then additional design heuristics, analogous to those proposed in the previous chapter, will be proposed and justified. Issues concerning the design of objective functions and search control mechanisms will also be discussed in the context of this KBS framework.

## 4.1 A Change in Viewpoint: A KBS Approach

This chapter explores the possibility of taking an approach from classical/symbolic AI. This entails a shift of viewpoint, from those so far dominant in the EC and OR communities to thinking in terms of a classical AI **symbol system** — a machine, such as a computer, that operates on symbolic structures (more on what a symbol is in Section 4.1.2). This notion forms the basis of much of AI research, as shown by one of its main hypotheses [Newell & Simon 76]:

> **The Physical Symbol System Hypothesis**. A physical symbol system has the necessary and sufficient means for general intelligent action.

Of course, this work (and its potential audience) are not especially interested in building intelligent machines, but rather in solving a given optimisation problem! However, the symbolic AI approach has lead to success in more pragmatic endeavours, namely in the area of Knowledge Based Systems (KBSs). This section will outline what a KBS is, what its components are, and will outline why it may be appropriate to view neighbourhood search optimisers as KBSs.

### 4.1.1 What is a KBS?

Early work in AI concentrated on the development of computer systems that *replicated* expert knowledge and performance (**expert systems**). However, it turned out that a useful system need not do this, if only because humans and computers have different strengths (eg. computers can calculate and do repetative tasks better than humans). A KBS relaxes the need for the system to perform in the same way as a human expert, allowing it to make use of its domain knowledge in whichever way the KBS designer sees fit. A succinct working definition of a KBS has been provided by [Stefik 95]:

> "... a computer system that represents and uses knowledge to carry out a task."

Designers of such systems (**knowledge engineers**) take a strongly 'knowledge-is-power' approach to their design. This is exemplified by the **knowledge principle** [Lenat & Feigenbaum 91]:

> **The Knowledge Principle**. To achieve a high level of problem-solving competence, a symbol system [ie. a KBS] must use a great deal of domain-specific, task-specific, and case-specific knowledge.

This has much in common with the discussion of the implications of the NFL theorems, which show that domain knowledge is strictly necessary for the design of an effective optimiser. This provides some positive indication that a KBS approach is in the spirit of what we are trying to achieve. The question therefore becomes one of whether we can usefully map neighbourhood search optimisers to a KBS framework.

It is common to relate a given KBS architecture to a certain human problem-solving approach. For example, case-based reasoning [Kolodner 93] is based upon human problem-solving by referring to and adapting past examples. From this viewpoint, neighbourhood search optimisers can, in their turn, be seen as computational models of **search by trial-and-error**. This analogy intuitively makes sense — given a starting solution, modifications are made to it and accepted in they lead to an improvement, and this process is then repeated. Therefore, a KBS approach would not only capture the spirit and aims of the work in the previous chapter, but it also potentially allows for the reuse of a large body of research into such systems. A good example of this is in the process of obtaining the knowledge required to build the KBS (knowledge acquisition).

### 4.1.2 The Knowledge Level

Before the form of the knowledge base for neighbourhood search optimisers can be specified, the meanings of 'knowledge', 'symbol system' and 'representation' need to made clear. This is of particular importance in the context of this work because, despite the realisation that domain knowledge is required, we must be clear about how this relates to neighbourhood search, KBSs, and the aims of this work. The AI literature fortunately addressed this issue some time ago. Work by [Newell 82] considers computer systems as consisting of **levels of description** such as:

- The **Knowledge Level**: a description of the knowledge contained in the system.
- The **Symbol Level**: a description of the data structures used by the system, and their manipulations.
- The **Program Level**: the implementation of an effective procedure (algorithm) that carries out the manipulations specified for the system.
- The **Hardware Level**: the physical realisation of the system.

Newell's contribution was to note that in addition to the symbol and lower levels, there was in fact a higher level of description — the **knowledge level** which corresponded to the knowledge used by the system [Newell 82].

> **The Knowledge Level Hypothesis**. There exists a distinct computer systems level, lying immediately above the symbol level, which is characterised by knowl-

edge as the medium and the principle of rationality as the law of behaviour.

Some points need clarification at this stage. The first is that the above can be seen as viewing knowledge as being a **transferable medium**, that is it implies that the knowledge required to build a KBS is sitting ready for use in the domain expert's head, and that all that needs to be done is to transfer this knowledge to the system. However, [Stefik 95] rejects this view as being incomplete as it ignores the processes of knowledge creation and use. Instead he suggests that knowledge is the **codified experience** of an agent (in our case the domain expert and designer), and can be revised and augmented by experimentation. This is the view adopted here. The second point concerns the **principle of rationality**. This refers to the assumption that a problem-solving behaviour can be explained in terms of the systems body of knowledge (ie. knowledge base) [Motta 97] — in other words there is a causal relation between the agent's (ie. the KBS's) knowledge and its behaviour.

Therefore we can see that a symbol system is a symbol-level implementation (aka. **representation**) of a computer system defined at the knowledge level. In other words, a representation is the mapping between the knowledge and symbol levels of a KBS. This discussion leads us to the main advantage of the separation of levels: if optimiser design is made at the symbol level, there clearly is more than one way that a given useful item of domain knowledge can be implemented. Therefore, if the principle of rationality is to hold then the relationship between the knowledge and symbol level must be clear — as this confers the advantage that the system's behaviour can be justified in terms of its body of knowledge. It therefore follows that in the definition of a KBS in Section 4.1.1 earlier, it is the representation condition that is the most important.

In summary, the discussion so far has described what a KBS is and has shown that a KBS approach shares many commonalities with the approach outlined in previous chapter, plus some additional advantages. In short, there is a definite case for considering the design of neighbourhood search optimisers as an exercise in knowledge acquisition and representation[1]. However there is still a need to specify and formalise what constitutes a knowledge base for a neighbourhood search KBS, and to formulate design heuristics for the other sources of domain knowledge apart from the solution encoding and operators. These issues will be addressed in

[1] Hence the title of this chapter!

the following sections which will show that a knowledge base can be constructed for neighbourhood search optimisers that satisfies the criteria set out above, and that the work in Chapter 3 in fact forms a very important part of this.

## 4.2 Classifying Knowledge Roles

So far, this thesis has only dealt with one facet, although a very important one, of how domain knowledge can be provided to an optimiser. What is required is a framework for providing domain knowledge to an optimiser. However for such a framework to be devised, a classification of the roles that knowledge plays in the optimiser is also required. Such a classification can be set up in a rather traditional KBS division of labour:

- **Problem Solving Knowledge** — this knowledge assists the search by providing knowledge and problem-solving strategies to guide the search.
- **Problem Specification (Goal) Knowledge** — specifying what desirable solution(s) are (ie. the evaluation function).
- **Search Control Knowledge** — given a search space, how do we go about searching it, our knowledge of search is represented here.

Of course such a division of **knowledge roles** also needs to fit naturally with the neighbourhood search paradigm, and in particular with the work in the previous chapter which has examined this issue from first principles. To this end, neighbourhood search optimisers were split into two components: the first was the **fitness landscape** [Jones 95] which was the combination of the solution encoding, neighbourhood operators, and objective function; the second was the set of **traversal rules** which search the landscape. Now the divisions made above and in 3 correspond closely to each other. The fitness landscape can be seen to be a representation at the knowledge level of the problem-solving and goal knowledge. The problem-solving knowledge being represented by the solution encoding and operators, and the goal knowledge being represented as the objective function. Similarly, the traversal rules can be seen to be a symbol level representation of the search control knowledge. In addition, it should be apparent from the discussion in 3.3 that as the fitness landscape and traversal rules related to the problem domain and search dynamics theories respectively, then the knowledge roles above must be similarly related. This is summarised in Figure 4.1 below:

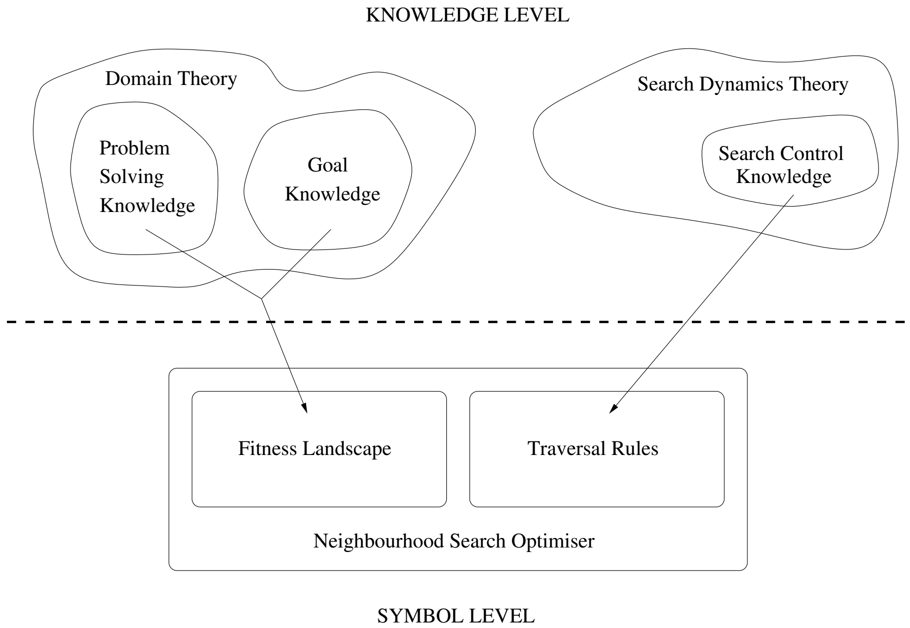


Figure 4.1: The Relationship Between Theories, Knowledge Roles, Levels, and Neighbourhood Search

The natural correspondences highlighted above, suggest that the classification of knowledge roles used here is justified, and provides a reason for focusing our attention, at least early on, to the acquisition and representation of problem-solving and goal knowledge. This can be seen by generalising the first design heuristic proposed in 3.4.4. This design heuristic stated that the options concerning the fitness landscape should be considered before those options pertaining to the traversal rules. Therefore, making the analogy with the knowledge roles, we can propose a modified version of this design heuristic:

> **Design Heuristic 1a:** *The knowledge roles should be investigated in the following order: goal knowledge, problem solving knowledge, and the search control knowledge*.

This is in a sense more specific that the earlier design heuristic as it provides additional ordering information for the design process. The rationale for this is that it is important, first of all, to be able to specify what a good solution is, as this defines the problem to be solved. Obviously, it would simply be foolish to think of a way of solving a problem *before* it has been specified. The issues pertaining to goal knowledge and problem specification will be discussed later.

### 4.2.1 Design by Knowledge Sources

However, as noted above, there are other ways that domain knowledge can be incorporated into a neighbourhood search optimiser. For example, it is common to start the search with a heuristically generated initial solution of comparatively good quality in the hope that this will reduce the amount of search required. The key to the approach outlined here is to classify the domain knowledge by its role, and then to consider each item of domain knowledge as a **knowledge source** that allows the designer to inform the optimiser on how to solve the problem at hand. This is illustrated by Figure 4.2 below.

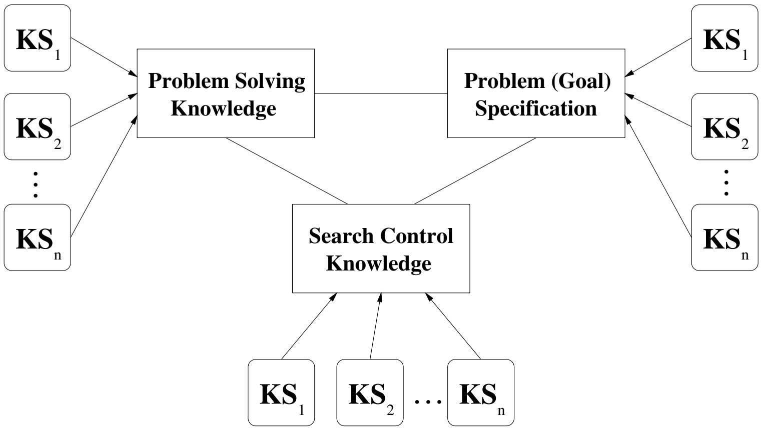


Figure 4.2: Knowledge Roles and Sources in Neighbourhood Search

This thesis proposes that these can be both expressed easily in concrete terms that a non-specialist in optimisation can understand, and mapped (represented) onto the optimisation algorithm in such a manner that a plan of incremental improvement and experimentation is possible. This arises from the view that, in practice, users examine a problem and expresses their hypothesis about the problem domain, which they can then incorporate into the algorithm to test whether they are correct.

One advantage of this incremental experimental programming approach (and of neighbourhood search algorithms) is that large amounts of explicit domain knowledge do not have to be represented to the optimiser before it can be used (unlike other KBS architectures such as a rule-based system), just that search will be more effective if it is used (which facilitates rapid prototyping). However to achieve this, the issue of *what* constitutes these knowledge sources, and therefore the knowledge base and its representation needs to be addressed (we have merely identified its structure at present). Furthermore, additional design heuristics, in the style of those in Chapter 3, need to be proposed to allow experimentation with the other knowledge sources. It is these issues that this chapter will now address.

### 4.2.2 Desirable Properties of a Representation

Given that the importance of the role of representation has been highlighted, it is necessary to define what makes a good representation before we go any further. One pragmatic answer to this is given below:

> ”My only comment is to remark that the quality of a representation depends on how well it fulfils the purposes for which it is intended, and to underline the need to specify exactly what these purposes are, and how the representation is to be used in achieving them” — Christopher Longuet-Higgins in [Desain & Honing 92].

However, aside from the (rather obvious) statement that a good representation should lead to an effective system, the above answer does not directly specify the properties that lead to a representation's ability to specify its purposes and method of achieving them. To this end the following three properties of a representation, adapted from [Stefik 95], will be adopted — that is a desirable representation should be:

- **Direct**. The condition states that for every knowledge-level object, or relationship between objects of interest, in the problem domain, there is an object or connection between objects at the symbol level, and a one-to-one correspondence between these knowledge and symbol level entities. (This is also termed **analogical**, or **vivid** [Levesque 86]).
- **Explicit**[2]. This property relates to how closely the representation is interpretable to what it represents. It has the following three conditions:

[2] A representation is termed **implicit** if it is not explicit.

  - **Modularity**. The representation is self-contained in the sense that there is a defined and bounded set of symbols that comprise it. Also, these symbols are separate from the programs that interpret and use them, and are accessed by a well-defined interface.
  - **Semantics**. The representation must have a well-defined semantics — in other words the meaning of the symbols and their relationships should be formal and clearly interpretable.
  - **Causal Connection**. There must be a causal connection between the representation and the system's behaviour such that changing the representation will cause the system's behaviour to change in a way that is consistent with its changed semantics.

- **Declarative**. This states that a representation should specify *what* the knowledge is rather than *how* it is used — this ensures that the representation is explicit. For example, a declarative representation would state that a certain condition needs to be met, whereas a procedural representation of the same condition would specify how this condition is checked in terms of an algorithmic procedure. This procedural description can obscure the purpose of the condition check, thus making the knowledge contained in the representation harder to interpret and the representation less explicit. Also, a declarative description lends itself to a number of equivalent procedural interpretations, and there is no *a priori* reason why a particular procedural interpretation must be chosen until the system is actually implemented.

It should be clear that the above is necessary for the principle of rationality to hold. This is because if any of the above conditions were not met, then it becomes more difficult (or even impossible) to relate the system's knowledge to its behaviour as the link between to two cannot be clearly made. These issues will be kept in mind later in this chapter, when the knowledge base of neighbourhood search methods, and how its body of knowledge is represented, is described.

## 4.3 Formulating Problem-Solving Knowledge

What exactly *is* meant by problem-solving knowledge? One working definition is that it is all of the knowledge that can be directly related to the problem domain, that is not involved with specifying the quality of a solution. However, an alternative definition due to [Motta 97] that is equally applicable is:

> ”Problem-solving knowledge is knowledge that is brought to bear during a problem solving process, when a system is faced with uncertainty in choosing among a number of alternatives.”

The above definitions place emphasis upon different aspects of what problem solving knowledge is. The first concentrates on where the knowledge comes from (ie. the problem domain theory), whereas the second examines its role. In short, the optimiser traversal rule has to make a number of choices which are, due to the heuristic nature of the algorithms we are considering, uncertain in the sense that there is no certainty that the choice made is the best one available (or even any good). Now these choices, and thus the system’s behaviour, can be influenced by the knowledge represented by the system via the landscape. For example, the fitness landscape specifies the set of solutions available in the neighbourhood, and which of these are likely to be accepted — which in turn also influences how effectively the traversal rules explore the fitness landscape, and thus the search space. In short, problem knowledge can be defined for the purposes of this study as:

> **Definition**: *Problem-solving knowledge is knowledge from the problem domain theory, that is not used in specifying the quality of a given solution, but instead is used to guide the traversal rules’ decisions (eg. by structuring the fitness landscape) so that they produce an effective search behaviour.*

In the context of problem solving knowledge, a number of relevant knowledge sources were identified by an informal examination of the current methods by which practitioner incorporate domain knowledge into neighbourhood search optimisers (and so may not be exhaustive). These sources are listed below in the form of questions the designer needs to answer:

1. **Features**. Which problem features correlate with solution quality?

2. **Linkage**. What is the structure and strength of interaction between certain problem features?

3. **Search Space Reduction**. Which solutions can be excluded from the search?

4. **Initialisation**. In which areas of the search space do good solutions lie?

5. **Move Selection**. Given a set of possible moves, which is most likely to result in a better quality solution?

Some reflection should suggest to the reader that the above sources can be framed as questions that a domain expert would be able to understand and answer without necessarily having any familiarity with optimisation or KBS jargon. This is intentional, since this work aims to exploit the domain expert's knowledge and to construct a system in terms that he can understand.

Each of these knowledge sources and how they can be mapped onto the optimisation algorithm will be now described in more detail. It is important to note that of these roles, the first is the most important to get right; in fact, as will be described later, all of the others require a correlated landscape in order to work effectively. In addition, the first two knowledge sources will be considered together because, as will soon be made clear, they are both realised in the construction of the fitness landscape.

The importance of the design heuristics in this methodology should be highlighted in the context of the above statement. In practice mere identification of the knowledge sources, though useful, is not enough as it does not take into account *how* these sources may interact with each other. In Chapter 3, the way that the fitness landscape and traversal rules were likely to interact with each other was exploited to usefully order the experiments (ie. landscapes were examined before traversal rules).

Finally, a similar process will therefore be used here for each of the knowledge sources to suggest design heuristics that order the sequence in which these sources should be considered, and allow the use of hillclimbing experiments in such a way that different hypotheses pertaining to a particular knowledge source can be compared so that they are transferable across optimisers. This is important because, as noted previously (Chapter 3), it would otherwise become difficult to design an optimiser in terms of the problem domain theory if the different hypotheses about the problem domain had different relative performance with different neighbourhood search optimisers.

## 4.4 Features and the Fitness Landscape

The first two knowledge sources relate to the design of the fitness landscape, which has been already discussed at length. From this, it is already known that the encoding and operators define, in conjunction with the evaluation function, the shape of the fitness landscape. If the shape of this landscape makes search tractable then the search will be efficient. There is, however, more that can be said about this component of neighbourhood search.

It was also discussed earlier that the problem features thought relevant to the search can be modelled by equivalence relations, and that the underlying theory (forma analysis) provides specifications for suitable neighbourhood operators. The reason behind the adoption of this modelling formalism was so as to provide some formal link between the problem domain theory and the neighbourhood structure. In other words, from a knowledge-level viewpoint, the equivalence relations provide a *formal* semantics of the neighbourhood structure, and of what features of the problem could usefully be used to index each solution in the search space uniquely. These equivalence relations can then be represented at the symbol level by a derived solution encoding and operators, which in combination manipulate these equivalence relations, with forma analysis providing a formal method by which to do this via the operator definitions described in 3.5.

However, the definition of unary operators currently used in the literature can contain more information than the above. For example, in the Travelling Salesman Problem (TSP) it is common in the literature to use neighbourhood operators that directly manipulate edges. As shown in [Surry 98], the minimal mutation for edge feature is the classic 2-opt operator. However, it is common to use other edge-based unary operators instead such as 3-opt or even $k$-opt, and these have been found to improve local search optimiser performance (presumably for reasons similar to those noted for the effect of neighbourhood size in 3.7).

There are two ways of integrating these operators into the forma analysis framework. The first is to define a different basis set of features for each of these operators which yields the operator when the minimal mutation heuristic is used. Unfortunately, this requires the formalisation task to be repeated for all of the possible $k$-opt operators, and may obscure the fact that these operators are closely related. That said, a more convenient way of defining these operators would be to retain the same basis set of features, but instead define additional operator specifications that allow for large changes between the parent and child solutions. Therefore, let the **generalised $k$-opt** operator be defined as follows:

$$N_{k-opt}(x, \Psi, k) \triangleq \{y \in S \mid d(x, y, \Psi) \leq k\}.$$

It can be seen that the minimal mutation operator is a special case of this operator when $k = d_{min}$ (where $d_{min}$ is the smallest number of changes that can lead to another valid solution). The forma analysis work in [Radcliffe 94, Surry 98] takes a similar approach to extending the reach of the unary minimal mutation neighbourhood operator. Though they instead define the **binomial minimal mutation** (BMM) operator, which is the result of applying a minimal mutation successively to the parent string a number of times given the **binomial distribution** $B(d_{max}, p_m)$[3] (where $d_{max}$ is the maximum distance, or the number of equivalence relations in the basis set, and $p_m$ is the probability that a given equivalence relation will be changed). Now this corresponds to the generalised $k$-opt operator by the expression below, if the assumption[4] is made that no equivalence relation is changed twice:

$$N_{bmm}(x, \Psi) \triangleq N_{k-opt}(x, \Psi, k) \quad \text{where} \quad p(k) = B(d_{max}, p_m)$$

where $p(k)$ above denotes the probability that a move of size $k$ will be made.

### 4.4.1 An Example: Real Numbers

Both of the above approaches provide definitions of operators that correspond closely to those used in the literature. For instance, the forma analysis of the real-numbered optimisation domain performed in [Surry & Radcliffe 96b, Surry & Radcliffe 96a, Surry 98], formalises the real numbers as a set of **Dedekind cuts**. In other words, the range of each real-numbered parameter is discretised into $N$ equally sized domains with boundaries $b_i$. For example the range $[0, 1)$ can be divided into 10 sections with 9 boundaries: 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, and 0.9.

[3] Where $B(d_{max}, p_m) = \frac{(d_{max}-d_{min})!}{(d_{max}-k)!(k-d_{min})!}\, p_m^{k-d_{min}}(1-p_m)^{d_{max}-k}$.

[4] This assumption is valid as in practice the chance of a given equivalence relation being changed twice is small for a large $d_{max}$ and small $p_m$ ($p_m$ is usually of the order of $1/d_{max}$).

This then allows the definition of equivalence relations as two solutions which are equivalent under an equivalence relation $\psi_{b_i} \in \Psi$ if they *both* have parameter values above $b_i$, or below $b_i$ — thus the features can take values $\Xi_{\psi_{b_i}} = \{\xi^0_{b_i}, \xi^1_{b_i}\}$, denoting each of the two cases. Naturally, there are constraints between the equivalence classes that can comprise a solution. For example if $\Xi_{\psi_{0.2}}$ had the value $\xi^0_{0.2}$ (which we will take to be that the described parameter is *less than* 0.2), and $\Xi_{\psi_{0.4}}$ had the value $\xi^1_{0.2}$ then this is clearly non-sensical — we would be saying that the parameter, $x$, is bounded by the condition $0.4 < x < 0.2$ and this leads to a contradiction given that we know that $0.2 < 0.4$. The above explanation should suffice for our current purposes, though the reader is directed to the original papers for a full explanation.

Getting to the point, the full forma analysis shows that BMM in this case, as the quantisation gets progressively finer, corresponds to the Gaussian mutation commonly used in many parts of the evolutionary algorithms literature and simulated annealing. It should also be clear that, by the analogy between BMM and $k$-opt above, that $k$-opt corresponds to change in real-number parameter values of size $\pm k$ times the size of the quantised domains, over a uniform distribution — this, at fine quantisations, corresponds to the 'creep' mutation operator commonly-used in the EA literature. Therefore, the forma analysis comes up with the desired result.

### 4.4.2 Landscape Families

It appears that the description of the role of features described so far is somewhat misleading as the above discussion implies that a set of basis features can be used to define a number of landscapes. The question is what, if any, effect does this have upon the design heuristics proposed there?

The way to answer this question is to view a basis set of features as not defining a single landscape, but instead defining a *family* of landscapes. The best way of visualising this is that the semantics provided by forma analysis define a (suggested) encoding, and more importantly a **metric space** that provides a **topological distance** between points in the search space (and therefore imposes a structure on the search space) — work in [Ronald 98] defines the conditions required for a distance or metric function, $d(x, y, \Psi)$, such as that proposed by the forma analysis discussion in 3.5 to be valid[5]. The distance metric for a basis set of equivalence rela-

[5] These conditions can be formulated as 4 axioms. For all $x, y, z \in \mathcal{S}$, the axioms are:

1. $d(x, y, \Psi) \geq 0$ and $d(x, x, \Psi) = 0$.

tions, $d(x, y, \Psi)$, satisfies these criteria. Now the use of a given $k$-opt operator can allow the definition of a **derived** metric space $d_k^{'}(x, y, \Psi)$ where the two distance metrics are related by the expression:

$$d_k^{'}(x, y, \Psi) = \left\lceil \frac{d(x, y, \Psi)}{k} \right\rceil$$

and therefore $d_k^{'}(x, y, \Psi)$ also satisfies the above criteria for distance metrics. These derived landscapes therefore also embody the metric space information defined by the features, it is just that as $k$ increases the distance differences get progressively ignored. As a result the effects of larger neighbourhoods noted in 3.7 arise: local optima disappear, and the minimum number of moves required to get from one point to another decreases. In other words, the neighbourhood operator contains information about both the structure of the landscape, and also about the appropriate size of move made. That is a neighbourhood operator in fact is a representation of two knowledge sources: one from the problem domain theory which *induces* the metric space and solution encoding; and the other from the search dynamics theory which represents a search strategy component regarding the size of the moves taken.

Moreover, as the discussion in 3.7 concerning the effect of neighbourhood size notes that this factor has an influence on the tractability of the landscape, there is a need to be able to somehow separate the examination of the two knowledge sources detailed above. To this end, the next design heuristic listed here addresses this issue.

> **Design Heuristic 5**[6]: *For two landscapes induced by $\Psi_A$ and $\Psi_B$ and operators with neighbourhoods of similar size, $N$, the relative performance of the two $\mathcal{L}_{N,\Psi_A}$ and $\mathcal{L}_{N,\Psi_B}$ can in practice be assumed invariant over all N for all sets of traversal rules, $t$ in a hillclimbing class $\mathcal{T}$.*

or expressed more formally as the assumption that:

$$\forall N : \; (\exists t \in \mathcal{T} : \; \mu(\mathcal{L}_{(N,\Psi_A)}, t) \geq \mu(\mathcal{L}_{(N,\Psi_B)}, t)) \Rightarrow (\forall t : \; \mu(\mathcal{L}_{(N,\Psi_A)}, t) \geq \mu(\mathcal{L}_{(N,\Psi_B)}, t))$$

2. $d(x, y, \Psi) = d(y, x, \Psi)$ (symmetric).
3. $d(x, z, \Psi) \leq d(y, z, \Psi) + d(y, x, \Psi)$ (triangle inequality).
4. $x \neq y \Rightarrow d(x, y, \Psi) > 0$.

[6] Recall that design heuristics 2, 3, and 4 were proposed in Chapter 3.

where the term **neighbourhood size** refers to the number of solutions in the neighbourhood, and not the value of $k$ (which will now be termed **neighbourhood order** to prevent confusion).

**Justification**

This design heuristic has an *informal* justification. From the discussion in 3.7, the number of local optima is dependent upon 2 factors: the amount of structure/correlation in the metric space defined by the chosen features (measured by metrics such as autocorrelation distance) and the effect of the neighbourhood size. Now assume that $\mu(\mathcal{L}_{N,\Psi_A}, t) \geq \mu(\mathcal{L}_{N,\Psi_B}, t)$, then the effect of increasing $N$ will be to reduce the number of local optima, and possibly other undesirable features such as plateaus. Assuming all things to be equal, we would expect the above effects of increasing $N$ to be similar for both metric spaces. Now also assuming, which seems likely, that the landscape $\mathcal{L}_{N,\Psi_A}$ will have fewer undesirable features than $\mathcal{L}_{N,\Psi_B}$, then the value of $N$ should have no effect on the *relative* number of these undesirable features in the two landscapes, and so their relative performance should remain unaffected.

### 4.4.3 A Caveat

A corollary of the above argument leads to the following caveat to the second design heuristic:

> **Caveat to Design Heuristic 2**: *Given the interaction between the choice of features/metric space and neighbourhood size, it is advisable when comparing two metric spaces to ensure that the neighbourhood operators used are of similar size.*

An illustrative example of this caveat appears later in this thesis. The example presented there arises, in abstract terms, in the case where a given optimiser is given two metric spaces defined by $\Psi_A$ and $\Psi_B$ that are quite similar (ie. they often give similar answers for pairwise comparisons of points in the search space). Now assume that the metric space defined by $\Psi_A$ was (slightly) more correlated than that defined by $\Psi_B$ (ie. for similar neighbourhood sizes $\mu(\mathcal{L}_{\Psi_A}) > \mu(\mathcal{L}_{\Psi_B})$), and that the size of a minimal mutation neighbourhood for $\Psi_A$ was greatly different from that for $\Psi_B$. In this case, the effect of the different neighbourhood sizes could oppose and override the above relationship between $\Psi_A$ and $\Psi_B$. However, if a $k$-opt operator for $\Psi_A$ was used instead so that the sizes of $k{-}\mathrm{opt}_{\Psi_A}$ and $\mathrm{BMM}_{\Psi_B}$ were comparable, then the landscape induced from $k{-}\mathrm{opt}_{\Psi_A}$ would perform better.

In summary, if the above caveat is not noted then the use of hillclimbing experiments to compare hypotheses about relevant problem features/metric spaces may mislead (as comparing landscapes is not *quite* the same as it tests a combination of the choice of metric space and a component of the search control knowledge). However, it should be noted that this problem is easily fixed — the experimenter should ensure that the neighbourhood sizes used are comparable. That said, this caveat should not detract from the other advantages of the experimental programming approach noted in Chapter 3, though it may be instructive to note that the landscape metrics such as FDC described there may also find a useful role in diagnosing whether this problem arises.

## 4.5 Linkage

There is an additional knowledge source that has not been considered so far which finds its representation in the neighbourhood operator. **Linkage** arises from the structure of the epistatic interaction between features of the candidate solution. For example, consider the optimisation of landscapes produced by spin-glass models, which are based upon aspects of solid state physics. In the case discussed here, consider a 1-dimensional linear array of 'spins' (bits) each described by a feature/equivalence relation $\psi_i$, where $i$ is the index of the array element. Due to electro-magnetic interactions, a lower energy state is attained when two given spins oppose each other. Furthermore, the strength of this interaction diminishes as the two spins are placed further apart in the array. From this, the energy of the system (which is to be minimised) is given by the expression below:

$$E = \sum_{i=1}^{N} \sum_{d=1}^{D} \alpha_d \times (\delta(\psi_i, \psi_{i+d}) + \delta(\psi_i, \psi_{i-d}))$$

where $N$ is the number of 'spins' in the array, $D$ is the maximum distance of spin interaction considered, $\delta(\psi_i, \psi_{i+d})$ returns 1 if the two spins are same direction, and zero otherwise, and $\alpha_d$ is the strength of the interaction at distance $d$ where this is the difference between the two array indices $i$ and $j$, ie. $d = |i - j|$ (also note that $\alpha_d > \alpha_{d+1}$). Therefore, as the fitness contribution of a given feature, $\psi_i$ interacts most strongly with features that are indexed close by (according to $d$), it would make sense in an operator that changes multiple features to bias the operator towards changing strongly interacting features at the same time. In other words, changing $\psi_i$ and $\psi_{i+1}$ together should be preferable to changing $\psi_i$ and $\psi_{i+k}$ together (where

$k > 1$). For more information on spin glass and NK landscape models the reader is directed to [Bäck *et al.* 97].

In other words, it is often possible to formulate hypotheses about the relative strength of the epistatic interaction between problem features. At the knowledge level this would be manifested as a statement such as the one in the example above about the relative locality of strongly interacting elements, possibly formalised by a function, $link(\Psi)$, which could be used to express which sets of features should be treated as single elements. At the symbol level, this knowledge would be represented and exploited in a combination of the choice of unary neighbourhood and crossover operators, and often also in the indexing scheme used for the features.

### 4.5.1 Formalisation

How the linkage knowledge source can be formalised will now be outlined. For instance in the above example, it is straightforward to see that as interactions are localised, then if multiple features are to be changed then they should be in a contiguous block. This can be used to define a variant of the generalised $k$-opt operator that only makes these changes:

$$N_{k-adj}(x, \Psi) \stackrel{\triangle}{=} \{y \in S \mid d(x, y, \Psi) \leq k \wedge block(x, y, k)\}$$

where $block(x, y, k)$ is defined by:

$$block(x, y, k) \stackrel{\triangle}{=} \exists i, \forall j \ (i < j < i + k) : \ \psi_i(x, y) = \psi_j(x, y)$$

or if circularity is required (ie. the operator 'wraps' the block over the end of the solution):

$$block(x, y, k) \stackrel{\triangle}{=} \exists i, \forall j \ (i < j < i + k - 1) \vee (j < i + k - n) : \ \psi_i(x, y) = \psi_j(x, y)$$

Therefore, it can been seen that linkage is represented as a **linkage specialisation** of the neighbourhood operator. In other words, the knowledge level description, $link(\Psi)$, of the linkage between features is used to define a mapping, $L_{link(\Psi)} : N_{k-opt}(x, \Psi) \rightarrow N'_{k-opt}(x, \Psi)$ which reduces the neighbourhood of $N_{k-opt}(x, \Psi)$ to include only those moves that are considered to respect the linkage between features in the manner described earlier. Linkage specialisation can also be transferred to recombination operators. For example, the GNX operator template for N-point crossover due to [Radcliffe & Surry 94b] can be used, in conjunction with the feature basis set used here, to derive a standard 2-point crossover operator that swaps a contiguous block of spins between 2 parent solutions. In this case, $link(\Psi)$ has been used to specialise the RTR operator to produce the G2X operator template, which is then instantiated to the binary 2-point crossover operator.

### 4.5.2 Design Heuristics for Linkage

Of course, as hypotheses about linkage cannot be formulated until a basis set of features have been decided upon, this leads to a further extension of the first design heuristic.

> **Design Heuristic 1b**: *Options concerning linkage should be investigated after the basis set of features/metric space has been decided upon.*

Furthermore, it would be reasonable and in keeping with the discussion so far to assume that hillclimbing experiments can be used to test hypotheses concerning linkage. This consideration leads to the proposal of the next design heuristic:

> **Design Heuristic 6**: *For a given landscape induced by $\Psi$, the relative effect on performance of various linkage specialisations induced by $link_i(\Psi)$, can in practice be assumed invariant over all optimisers of a given hillclimbing class. Furthermore, previous design heuristics concerning the transferability of these results to evolutionary algorithm mutation and crossover operators also apply.*

which can be expressed more formally as:

$$(\exists t \in \mathcal{T} : \mu(\mathcal{L}_{link_A(\Psi)}, t) \geq \mu(\mathcal{L}_{link_B(\Psi)}, t)) \Rightarrow (\forall t : \mu(\mathcal{L}_{link_A(\Psi)}, t) \geq \mu(\mathcal{L}_{link_B(\Psi)}, t))$$

where $\mathcal{L}_{link_i(\Psi)}$ is the landscape induced by applying the mapping $L_{link_i(\Psi)}$ to the operator that defines the unspecialised $\mathcal{L}$.

#### Justification

The validity of this design heuristic can be justified by noting that the above is simply a specialisation of the second design heuristic described in 3.8.1, as the linkage specialisations each correspond to a different landscape. Therefore if the second design heuristic is valid, so must this one be, as the earlier design heuristic applies regardless of which component of the neighbourhood operator is being varied. In addition, it is also possible by virtue of the above arguments, and of the third and fourth design heuristics described in 3.10 and 3.12, to transfer the results of hillclimbing experiments to the EA mutation and recombination operators.

However, implicit in the above argument is that a certain basis set of features (and neighbourhood order) have already been selected so as to fix the other components of the neighbourhood operator for the above experiment.

### 4.5.3 Examples of Linkage Exploitation

It should be noted that such knowledge concerning linkage has been exploited in previous work. A clear example is in problems which a matrix is to be optimised, such as the source approportionment problem [Cartwright & Harris 93], and the use of a voxel-based representation in shape optimisation [Baron *et al.* 97a, Baron *et al.* 97b, Baron 97]. In these situations, it becomes apparent that elements in the matrix that are spatially close together interact more strongly than those that are spatially more distant; however, this locality would be lost if the matrix was transformed into a linear string, and thus the linkage of interacting elements would be reduced. In both cases it was found that the use of a ‘matrix-aware’ recombination operator such as UNBLOX [Cartwright & Harris 93] increased GA performance dramatically (ie. a speed increase of a factor of 2) with respect to an encoding and crossover operators (n-point crossover) that treated the solution as a linear string of elements. The action of the UNBLOX operator is illustrated by Figure 4.3, as can be seen, it randomly selects a rectangular section of the matrix and swaps the information there between the two parent solutions to produce a child — in this sense it is a generalisation of the 2-point crossover operator in 2-dimensions.

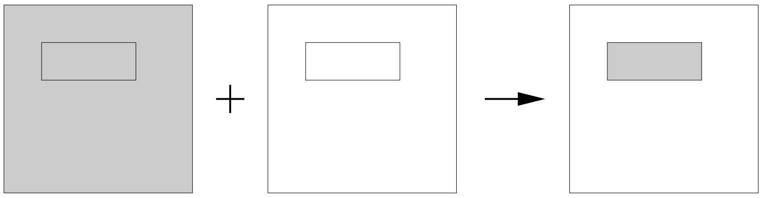

Figure 4.3: The UNBLOX Crossover Operator

The linkage structure of the problem was also exploited in the mutation operators used – in fact it was found that it was impossible to produce an evolutionary algorithm that could solve even a simple 2D beam optimisation without these specialised operators. One such operator was the smoothing mutation operator that selected a rectangular portion of the array and set all of the bits in the selected area to the most common value in that area, as shown in Figure 4.4.

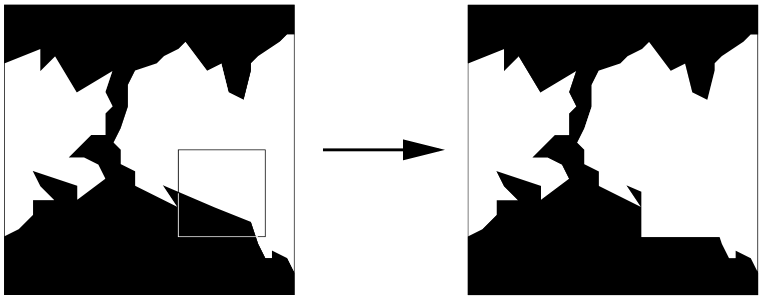

Figure 4.4: The Smoothing Mutation Operator

Finally, additional $k \times k$ operators, which can be thought of as a 2D version of the $k$-opt operator that operates on a square portion of the matrix, were also investigated in the above study and found to produce significant gains in EA performance. These operators were then later used as a basis for the optimisation of an aircraft annulus with interesting results [Baron *et al.* 99].

### 4.5.4 Summing Up

In summary, the above argument shows that the neighbourhood operator is more than just a function that specifies the connectivity between solutions — in fact it is central to the principled design of neighbourhood search optimiser. The above discussion has shown that its role is threefold: to represent the designer's knowledge concerning the appropriate features of the problem domain that are used to index the solutions and induce a metric space, their interactions, and some additional search control knowledge. This decomposition of the roles of a neighbourhood is summarised in Figure 4.5 below.

Also, examination of the above criteria for a desirable representation given earlier in Section 4.2.2, shows that the approach above satisfies these conditions. Forma analysis can be used to directly index both relationships of interest between solution features (as equivalence relations), and entire solutions (as a set of equivalence classes). The notion of coverage also helps to ensure that there is a one-to-one mapping between symbols and objects, given that the chosen feature basis set is not redundant. The representation suggested by forma analysis also satisfies the modularity criterion as it defines a bounded set of symbols in a fashion that

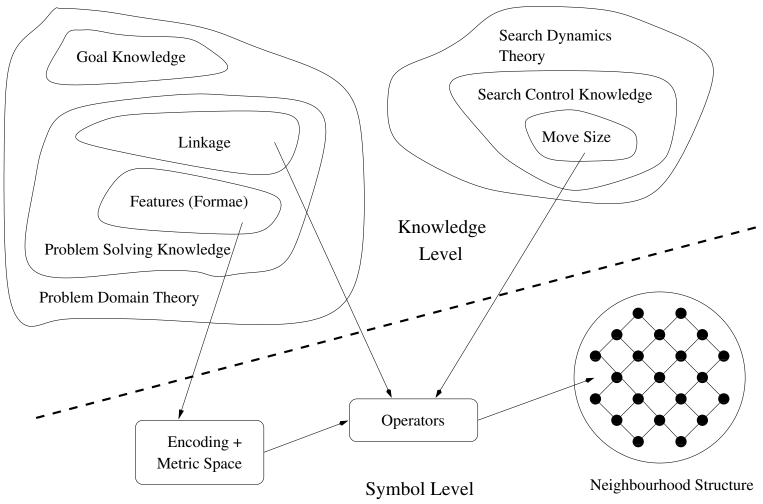


Figure 4.5: The Relationship Between The Neighbourhood Structure and the Knowledge Level

is separate from the interpretation program (in fact, no reference to such a program needs to be made). In addition, not only does forma analysis provide a formal semantics for the fitness landscape, it also does so in a declarative manner as both the basis set of features and the operator definitions make no reference to their procedural implementations. Finally, as the choice of the features makes an explicit reference to the problem domain, a change in the basis set of features to index a different set of problem features leads to a change in operator, and therefore optimiser, behaviour that is consistent with the changed semantics — therefore the condition that there is a causal connection between the representation and the system's behaviour is also satisfied.

From this, it is now possible to address the objections in the introduction of this thesis (in 1.5) to the notion of 'natural' encodings and operators. The objections centred around the fact that the definition of what makes a representation 'natural' was somewhat vague. However, it should now be clear that natural encodings/operators can be usefully equated to those that both meet the representational conditions laid out earlier, *and* represent valid hypothesis about the problem domain in that, when implemented, they result in an effective optimiser.

In addition, the design heuristics proposed in Chapter 3 have been extended to take into account the fine structure of the neighbourhood operator detailed here. In fact, given the above, it is indeed difficult to argue that there is any significant difference between the approach in Chapter 3 (and its extensions above) and a conventional KBS. In fact, the only two objections that may arise is that, so far, the range of knowledge sources is rather impoverished, and the issue of whether a KBS viewpoint allows current issues in the literature to be examined in a new, more useful, fashion. These issues will be dealt with in the remainder of this thesis.

Two additional points are worthy of mention at this stage. First, these equivalence relations are encoding independent, and therefore there is no need to encode the solution in the same way as these relations. In fact this may not be desirable, as they do not necessarily comprise an intuitive description of the solution for the user (eg. schedulers are often most comfortable with Gantt charts, and a list of features would be quite alien to them). However, it is still possible to use forma analysis to derive suitable operators and provide formal semantics.

Finally, it should be noted that there are other ways by which linkage can be represented other than by the above. For example, the linkage structure could indicate that the problem is readily decomposable into subproblems — an example of this occurs in 'divide and conquer' evolutionary algorithms (for example [Gonzales-Hernandez 95]) that encodes each subproblem separately and also the method by which they are brought together for evaluation.

## 4.6 Search Space Reduction

Attention so far has been upon the knowledge sources that find their expression as deformations in the fitness landscape. There is, however, one other knowledge source that can also find expression in this manner. This knowledge source, **search space reduction**, arises because it is also possible to form beliefs about areas of the search space, $\mathcal{S}$, which can be excluded from the search, by not representing them in the encoding space, $\mathcal{E}$. This will make the encoding space smaller as a result, and thus, hopefully, easier to search. Of the knowledge sources described here, this is most probably the least heuristic in nature, as it is often possible to exclude solutions from the search in such a fashion that optimal, or near-optimal, solutions are guaranteed not to be 'accidentally' excluded.

One clear example arises in domains that can be formulated as constraint satisfaction problems (CSPs), where it is possible with the use of arc and path consistency[7] algorithms [Tsang 93], to exploit the constraints to reduce the search space. This brings the additional advantage that

[7] This refers to the propagation of constraints to reduce the amount of search required. For example, if we know that variable $A$ must take a value $x$, then with the binary inter-variable (arc) constraint that $A \neq B$ then we can remove the value $x$ from $B$'s domain (and the search).

sometimes some of the constraint checks on feasibility are no longer needed, thus making the evaluation of solutions faster. This approach has been used in an informal manner in the secondary structure prediction of RNA molecules [Tsang 96] where this procedure has proved vital in producing a working system.

A final example comes from [Kappler *et al.* 96] which examined the problem of scheduling the exchange/replacement of fuel rods in a nuclear reactor where the rods are arranged in a 2D array. Rods in the centre of the array wear out more quickly than those at the edges and fuel efficiency is also location dependent in a non-linear fashion. Therefore the task is find a schedule of rod exchanges that maximise the life and efficiency of the available fuel rods. In this case the 8-fold axial symmetry present in the fuel array was exploited to reduce the size of the encoding space to be searched by considering only 1/8 of the array.

### 4.6.1 Formalisation

The notion of search space reduction can be formalised in the mapping, $g$ from $\mathcal{E}$ to $\mathcal{S}$. The action of search space reduction is to modify $g$ such that some items in $\mathcal{S}$ cannot be represented in $\mathcal{E}$ (and vice versa), thus effecting a reduction in the number of search points considered by the optimiser (as they will be discarded). The CSP domain will now be used in the following discussion to make the formalisation easier to understand.

The formalisation of this reduction can be performed as follows. In the review of forma analysis in 3.5, it was noted that in some domains, not all combinations of features are possible — a classic example of this are permutation problems. Therefore, implicit in the definition of a basis set of equivalence relations, is a set of constraints between their allowed values. Now given that $legal(\Psi)$ explicitly denotes the set of such constraints, then it is possible to add additional constraints to $legal(\Psi)$ to exclude the unwanted solutions in $\mathcal{S}$. Therefore the mapping between $\mathcal{S}$ and $\mathcal{E}$ should be more properly defined as: $g_{legal(\Psi)} : \mathcal{E} \rightarrow \mathcal{S}$. In other words, the relationship between the knowledge sources considered so far and the neighbourhood operators is based on the instantiation and then successive ‘specialisations’ by constraints upon a generic neighbourhood operator template (Figure 4.6).

In the case of search space reduction, $legal(\Psi)$, can be used to describe a formal knowledge-level description of the *unrepresented* solutions $\mathcal{S}$, and therefore this suffices to formalise the

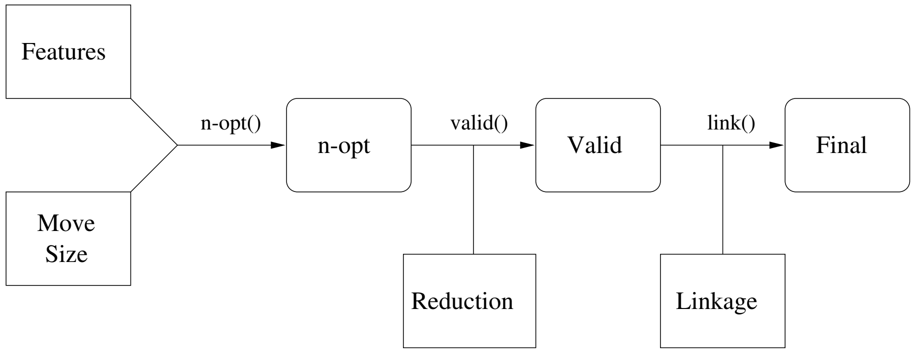


Figure 4.6: Operator Design as Successive Specialisations by Knowledge Sources

designer's hypotheses concerning this knowledge source. For example, in the case of the CSP, $legal(\Psi)$, can be augmented to enforce node and arc consistency in the permissible solutions. However, this approach should come with a warning. In the CSP example considered here, it is entirely possible to place the entire CSP into $legal(\Psi)$. Therefore though the optimiser will be able to solve the CSP in one move, the neighbourhood operators will be required to solve an NP-hard problem. As a result of this, there is a tradeoff involved between how efficiently the constraints are imposed by $legal(\Psi)$, and its effect upon the number of search points that are needed to be evaluated by the optimiser.

### 4.6.2 Design Heuristics for Search Space Reduction

The first of the design heuristics is concerned with the issue of *when* this knowledge source is considered, and is another extension of the first design heuristic in 3.4.4:

> **Design Heuristic 1c**: *Search space reduction should be considered not before the basis set of features have been decided upon.*

This is due to the pragmatics of the situation. The examples above decide to exclude solutions on the basis of the semantics of the solutions in the search space . As this is given by the basis set of features, $\Psi$, any reduction of the search space, denoted by $legal_i(\Psi)$, must wait until $\Psi$ has been defined. Furthermore, as noted above, the effectiveness of a redefined mapping is only useful if the landscape remains correlated. Therefore a basis set needs to be found that ensures this. In addition to the above heuristic, the following design heuristic can also be proposed for this knowledge source:

**Design Heuristic 7**: *For a given landscape induced by* $\Psi$*, the relative effect on performance of various search space reductions induced by* $legal_i(\Psi)$*, can in practice be assumed invariant over all optimisers of a given hillclimbing class. Furthermore, the previous design heuristic concerning the transferability of these results to evolutionary algorithms also applies.*

which can be expressed more formally as:

$$(\exists t \in \mathcal{T} : \mu(\mathcal{L}_{legal_A(\Psi)}, t) \geq \mu(\mathcal{L}_{legal_B(\Psi)}, t)) \Rightarrow (\forall t : \mu(\mathcal{L}_{legal_A(\Psi)}, t) \geq \mu(\mathcal{L}_{legal_B(\Psi)}, t))$$

where $\mathcal{L}_{legal_i(\Psi)}$ is the landscape induced by applying the mapping/constraints $legal_i(\Psi)$ to the basis set of features that defines the untransformed $\mathcal{L}$. This design heuristic can be justified in the same way as Section 4.5, in that it is a specialisation of the second design heuristic. Therefore hillclimbers can again be used to verify problem-solving knowledge sources in a transferable manner.

## 4.7 Move Selection

This knowledge source arises because, for each solution in the search space, there are a number of 'moves' to other solutions available (most of which are non-improving). Almost all neighbourhood search implementations examine these moves in a random or fixed lexicographic order. However, if a method existed that could cheaply determine which moves were improving (without evaluating them), it would then result in improving moves (and hopefully high-quality solutions) being found more quickly — assuming that the search is not led into a local optimum. This is because it would no longer be necessary to evaluate solutions arising for non-improving moves. Fortunately, it is often possible to examine a solution and to form hypotheses about which moves would be most likely to improve the solution. These hypotheses comprise the **move selection** knowledge source.

### 4.7.1 Formalisation and Implementation

Before this knowledge source can be formalised, it should be noted that this knowledge source introduces a *temporal* aspect to the neighbourhood which has not been considered so far. To capture this aspect of the neighbourhood, it should be considered as being an *ordered set* (or sequence) of the solutions (or equivalently moves) to be evaluated. So we can define *ordered* set of moves, $\langle m_1, m_2, \ldots, m_n, \rangle$, induced by move selection, $movesel(m_i, m_j, \Psi)$, as given below:

$$\forall m_i, m_j : i < j \Leftrightarrow movesel(m_i, m_j, \Psi)$$

where move selection is expressed by the (transitive) comparison predicate $movesel(m_i, m_j, \Psi)$ which returns true if the move $m_i$ is preferred to the move $m_j$. From this the neighbourhood $N(s_c, \mathcal{L})$ is the sequence of solutions generated by applying the set of available moves (with $s(m_i, s_c)$ mapping a move $m_i$ from $s_c$ to the solution in $N(s_c, \mathcal{L})$).

From an implementational viewpoint, there are two ways in which this knowledge can be used by the optimiser:

- A **candidate list** strategy might select a biased subset of the possible moves to try. This has the dual effect of both reducing the size of the neighbourhood, but also in ensuring that the most improving moves are not inadvertently left out (as these are the moves that we are most interested in). This approach has been used in tabu search [Glover *et al.* 93], and the **directed** candidate list strategy proposed later in this thesis is a general implementational framework for this approach.
- The second approach is the **directed neighbourhood** approach. Instead of taking a subset of the available moves, the *temporal* aspects of how the neighbourhood is examined in any-ascent and first-ascent hillclimbing is exploited. In this case, a move is accepted as soon as it meets the acceptance criterion, without waiting for the remainder of the neighbourhood to be considered. Therefore if the improving moves could be tried earlier, then the non-improving moves would not be considered thus saving otherwise wasted search effort. This approach has been successful for timetabling problems [Ross *et al.* 94], where exams with a high number of constraint violations were more likely to be moved (where it was termed **directed mutation**); it is also related to the work by [Zweben & Davis 92, Zweben *et al.* 92] on iterative repair scheduling, and by [Minton *et al.* 90] on the 'min-conflicts' heuristic. In addition, it has found other application in EA work, for example in [Baron *et al.* 97a, Baron *et al.* 97b] where the smoothing operator described earlier was modified to target regions of high beam stress,

and in the optimisation of array paddings in FORTRAN codes[8] [Altmann 97]. Finally, move selection has been used in a similar manner for simulated annealing (though in [Zegordi & Enkawa 95] it was termed **move desirability**).

### The Effect upon the Landscape

Given the formalism and move ordering described above, one suitable candidate list strategy can be defined by considering only the $S$ most preferred moves in the neighbourhood $N_{dir}(s_c, \mathcal{L})$ (where $S$ is the size of the candidate list), to give the revised neighbourhood $N_{dir}(s_c, \mathcal{L})$ below:

$$N_{dir}(s_c, \mathcal{L}) = \{\langle s(m_1, s_c), s(m_2, s_c), \ldots, s(m_n, s_c)\rangle \mid m_i \in M(s_c, \mathcal{L}) \wedge n = S\}$$

where $s(m_i, s_c)$ denotes the search point in $\mathcal{S}$ that is produced by applying the move $m_i$ to the current solution $s_c$. This reduction in the neighbourhood will also produce a corresponding transformation in the set of search sequences considered by the optimiser, $\mathcal{B}(\mathcal{L}, t)$, given by the behaviour space formalism in 3.6. Assuming that $movesel(m_i, m_j, \Psi)$ is valid, then the expectation will be that this transformation will effect an increase in optimiser performance — due to the retention of search sequences corresponding to the cases where a high proportion of improving moves were considered.

In the case of a deterministic directed mutation implementation, an *implicit* neighbourhood reduction occurs in the case of any-ascent or first-ascent optimisers. This is because of the ordering on the moves imposed by $movesel(m_i, m_j, \Psi)$. Consider the case where there are a number of moves that can be accepted in the neighbourhood. Given that the moves are tried in a given order, then the first move that *can* be accepted, will be, and all moves after that will not be considered. Therefore given that the **effective neighbourhood**, $N_{eff}(s_c, \mathcal{L})$, can be considered to be the set of solutions that will actually be evaluated, we get the following:

$$\begin{aligned} N_{eff}(s_c, \mathcal{L}) = \{\langle s(m_1, s_c), s(m_2, s_c), \ldots, s(m_n, s_c)\rangle \mid \quad & m_i \in M(s_c, \mathcal{L}) \wedge \\ & \neg\exists i < n : f(s(m_i, s_c)) \geq f(s_c) \wedge \\ & f(s(m_n, s_c)) \geq f(s_c)\} \end{aligned}$$

[8] So to optimise the arrangement of data arrays in a direct-mapped primary processor cache, and thus maximise the frequency that the processor accesses the cache rather than the (slower) main memory.

From the above, if $movesel(m_i, m_j, \Psi)$ represents a valid hypothesis about the relative likelihood of the moves $m_i$ and $m_j$ being improving, then the proportion of the neighbourhood that needs to be evaluated before a solution is accepted should be reduced — thus producing an effect similar to that for an effective directed candidate list strategy. Therefore in both cases, the effect of the move selection knowledge source is to induce a further (though possibly implicit) specialisation upon the fitness landscape in a similar fashion to the linkage specialisation. Moreover, even in the stochastic case where the ordering is perturbed somewhat, the same argument should apply though $N_{dir}$ will vary somewhat (however the orderings will be expected to be quite similar due to the bias imposed). Finally though only unary neighbourhood operators have been considered here, it is entirely possible to devise similar heuristics for EA recombination operators.

### 4.7.2 Design Heuristics for Move Selection

These considerations lead to the following design heuristics, the first of which is a further specialisation of the first design heuristic proposed in 3.4.4.

> **Design Heuristic 1d**: *move selection should be considered after the move operator(s) have been decided upon.*

The primary reason for this is the fact that move selection is defined for a set of available moves. Therefore the designer will only know what these are once the earlier knowledge sources regarding the nature of the landscape has been decided upon. Furthermore, this knowledge source needs to be defined for a correlated landscape to be effective, because as noted above, move selection allows the optimiser to hillclimb faster by removing 'useless' non-improving moves. Of course, if the search space was uncorrelated, then local optima would merely be found faster, and deceptive regions exploited faster to 'false' optima. As a result the gains obtained by move preference on a poorly chosen landscape would be modest. Therefore it would be sensible for the designer to first obtain an effective fitness landscape by exploiting the earlier knowledge sources.

> **Design Heuristic 8**: *For a given landscape induced by $\Psi$, the relative effect on performance of various move selection strategies induced by $movesel_i(m_1, m_2, \Psi)$,*

*can in practice be assumed invariant over all optimisers of a given hillclimbing class. Furthermore, the previous design heuristic concerning the transferability of these results to evolutionary algorithms also applies.*

which can be expressed more formally as:

$$(\exists t \in \mathcal{T} : \mu(\mathcal{L}_{movesel_A(m_j,m_k,\Psi)}, t) \geq \mu(\mathcal{L}_{movesel_B(m_j,m_k,\Psi)}, t)) \Rightarrow$$
$$(\forall t : \mu(\mathcal{L}_{movesel_A(m_j,m_k,\Psi)}, t) \geq \mu(\mathcal{L}_{movesel_B(m_j,m_k,\Psi)}, t))$$

where $\mathcal{L}_{movesel_i(m_j,m_k,\Psi)}$ is the landscape induced by applying $movesel_i(m_j, m_k, \Psi)$ to the basis set of features that defines the untransformed $\mathcal{L}$. This design heuristic can be justified in the same way as Section 4.5, in that it is a specialisation of the the second design heuristic. Therefore hillclimbers can again be used to verify this problem-solving knowledge source in a transferable manner.

## 4.8 Heuristic Initialisation

It is common, especially in the OR literature, to start the search with a heuristically generated solution — for example, in studies [Osman & Potts 89, Ogbu & Smith 90, Reeves 95a] that have attempted the flowshop sequencing problem, the NEH heuristic [Nawaz *et al.* 83] was used to find a high-quality solution with which to begin the search. This was found, in these studies, to result in significant performance gains. In addition, the GRASP methodology [Feo *et al.* 91a] places emphasis upon the construction of high-quality initial solution, to the extent that feedback from previous starting points is exploited. The fact that GRASP has been shown to be effective for a number of problems indicated that the initialisation component of a neighbourhood search optimiser can be used to improve performance.

The above are examples of knowledge being provided to the optimiser via the **heuristic initialisation** knowledge source. In the context of the framework discussed here, the initialisation procedure of neighbourhood search is effectively a representation of the user's hypothesis concerning *what* collection of features generally constitute a high-quality solution — thus, assuming a correlated landscape, an implicit assumption is make concerning *where* on the fitness landscape high-quality solutions are located.

Two issues must be kept in mind when devising an effective heuristic initialisation strategy. The first is cost, in that the computational effort required to produce an initial solution must be less than the computational effort saved by the reduction in the number of solutions the optimiser has to consider. The second is its effect on global optimisation. In other words, the initialisation strategy must start the search in a productive region of the fitness landscape. However, the concern that heuristic initialisation could mislead has been expressed most strongly in the evolutionary computation community (where it has been used least often):

> "Heuristic initialisation may be helpful but must be done carefully to avoid premature convergence, since the GA is likely to exploit the opportunity to converge to the regions of the search represented by the heuristically chosen structures." [Grefenstette 87].

That said, these concerns have been empirically rebutted in [Surry & Radcliffe 96c] where for a number of demanding real-world problems, heuristic initialisation produced substantial gains in performance. In summary, the literature therefore indicates that, in practice, it is possible to devise effective heuristic initialisation strategies.

### 4.8.1 Implementation Issues

In addition, this knowledge source can be implemented in a number of ways, and these can be broken down into two orthogonal components. The first component is concerned with how the initial solution is *obtained*. This can be achieved in a number of ways: the use of constructive/greedy heuristics, such as NEH above, is common in the OR literature; a **case-based reasoning/retrieval** (CBR) system [Kolodner 93] could be used to obtain good initial solutions from a database of solutions to previously-tackled problems; finally, **immune networks** [Bersini 91] which are based on models of mammalian immune systems, can also be used to quickly generate initial solutions.

The second component determines *how* the initial solution is made available to the optimiser. Two methods will be briefly mentioned here: the first, termed here as **direct seeding** introduces the solution into the optimiser unmodified. If a population-based optimiser is being used, the remainder of the population can be generated randomly — this was the approach used in [Reeves 95a], amongst others. The alternative, exemplified by the work in [Surry & Radcliffe 96c] is termed **biased seeding** here. Here the solution is perturbed by applying large move(s) to it before it is presented to the optimiser. In the case of a population-based optimiser, each of the initial solutions are produced by applying a different sequence of moves to the same solution.

### 4.8.2 The Nature of Heuristic Initialisation

This knowledge source considered differs from the others in that it is not expressed in the neighbourhood structure of the optimiser. As a result, before design heuristics for this knowledge source are formulated, it is first necessary to make a distinction between problem-solving knowledge sources that are expressed in the neighbourhood operator, and those that are not.

More formally, consider $\mathcal{S}_{int}$ to be the set of solutions available for initialisation. Then our initialisation strategy, denoted by $initial(\mathcal{S})$ at the knowledge level is a mapping $initial(\mathcal{S}) : \mathcal{S} \rightarrow \mathcal{S}_{int}$ that determines the solutions that are used by the initialisation strategy. Therefore we can expect $initial(\mathcal{S})$ to represent a useful hypothesis about where high quality solutions are located on the fitness landscape if the solutions in $\mathcal{S}_{int}$ are on average both closer to high-quality (or even optimal) solutions than those in $\mathcal{S}$, *and* a path to these solutions exists that is traversable by the traversal rules.

Furthermore, in the behaviour space formalism given in 3.6, it can be seen that $initial(\mathcal{S})$ at the knowledge level maps onto $\omega_{rw}(\langle s_0 \rangle, s_c, \mathcal{L})$ (the distribution of the starting solutions) of the traversal rule description. For example, in the case of direct seeding, $\omega_{rw}(\langle s_0 \rangle, s_c, \mathcal{L})$ would be redefined as follows:

$$\omega_{rw}(\langle s_0 \rangle, s_c, \mathcal{L}) = \begin{cases} 1 & : \quad s_0 \in initial(\mathcal{S}) \wedge (s_c := s_0) \\ 0 & : \quad \text{otherwise} \end{cases}$$

Therefore, the use of heuristic initialisation effects a reduction of the number of search sequences in $\mathcal{S}$ in $\mathcal{B}^{'}(\mathcal{L}, t)$. Obviously if $initial(\mathcal{S})$ is a correct hypothesis about the problem domain, then we should expect to see an increase in the expected optimiser performance, $\mu(\mathcal{L}, t)$, as the less effective search sequences (ie. those far away from high quality solutions) are discarded.

### 4.8.3 Design Heuristics for Heuristic Initialisation

One important consequence of the above discussion is that, unlike the other knowledge sources covered so far, the second design heuristic in 3.8.1 concerning the comparison of landscapes cannot be specialised to derive design heuristics for this knowledge source. Therefore, the design heuristics presented here will be justified from scratch.

> **Design Heuristic 1e**: *Problem-solving knowledge sources affecting the landscape should be considered and decided upon before considering heuristic initialisation.*

This is another specialisation of the original version of the first design heuristic. This is because heuristic initialisation requires a correlated landscape to be effective, as the hypothesis that a good initial solution is in the vicinity of better solutions which is only realisable if the landscape is correlated (as correlation implies that good solutions are closer to each other than poor solutions). This is illustrated by Figure 4.7 below, where the quality (and distance from the optimum) of the starting solution(s) is a much poorer guide to the location of high-quality solutions for the less correlated landscape.

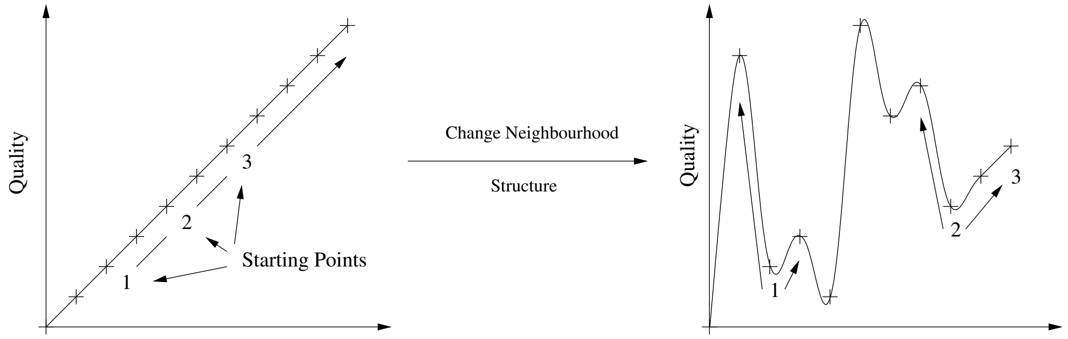


Figure 4.7: The Effect of Landscape on Initialisation Effectiveness

In other words, there is no point in comparing initialisation strategies until a tractable landscape has been found as they would not be effective. Furthermore, if the landscape was to change, then the comparisons would have to be performed again which leads to wasted experimental effort.

The next design heuristic arises from the issue, as noted in 3.8.2, of whether the relative effectiveness of different variants of a knowledge source (in this case the initialisation strategy) vary with a change of local search traversal rules. If this were the case, then the notion of a current hypothesis about a problem would also vary. Therefore it would become extremely difficult to design an optimiser in terms of the problem domain theory (which is one of the stated aims of this work). If this can be shown to be the case then the following design heuristic applies:

> **Design Heuristic 9**: *For a given landscape, $\mathcal{L}$, the relative effect on performance of various hypotheses concerning the heuristic initialisation knowledge source, $initial_i(\mathcal{S})$, can in practice be assumed invariant over all optimisers of a given hillclimbing class. Also, the third design heuristic concerning the transfer of results to the evolutionary algorithms holds for this knowledge source.*

this can be stated in more formal terms as follows:

$$(\exists t \in \mathcal{T} : \mu(\mathcal{L}, initial_A(\mathcal{S}), t) \geq \mu(\mathcal{L}, initial_B(\mathcal{S}), t)) \Rightarrow$$
$$(\forall t : \mu(\mathcal{L}, initial_A(\mathcal{S}), t) \geq \mu(\mathcal{L}, initial_B(\mathcal{S}), t))$$

The independence of initialisation from the search control options can be argued as follows. Recall from 3.8.2 that modification of the traversal rules from the hillclimbing case invariably involves either adopting a random walk or random search behaviour for part of the search sequence. Now, first consider the case of a correlated landscape in Figure 4.7 above. There the effect of any search control extensions would be to slow down the search, as the additional search sequences in $\mathcal{B}'(\mathcal{L}, t)$ will represent variants of the original search sequences that possess some random walk/search character. This will have two effects: firstly those starting points with search sequences that were previously able to locate high quality solutions will still be able to do so; secondly, the introduction of new search sequences with some random walk/search character will invariably slow down the search as they represent cases where the information provided by the fitness landscape has been ignored. That said, solutions of high quality (and therefore closer to the optimum) will still be expected to get there sooner, though the performance differences would diminish until the search degenerates into a random walk/search where the performance of the different starting points can be considered equivalent.

Now consider what happens when any of the landscape features that reduce the tractability of a landscape are introduced by transforming the landscape (as in Figure 4.7 above). Though quality is now not as reliable a guide to initialisation performance as before, the addition of further search control would not be expected to affect the relative performance of the different starting points greatly. This can be seen by splitting the starting points into two classes with respect to their search sequences depending on whether they can, with the traversal rules provided, reach high-quality solutions (defined appropriately) within a certain number of steps. Now let $\mathcal{B}'_{yes}(\mathcal{L}, t_{hc})$ and $\mathcal{B}'_{no}(\mathcal{L}, t_{hc})$ denote the behaviour space of the two sets of starting points that can and cannot access high-quality solutions with a hillclimber respectively. Progressively increasing the amount of random walk/search character by invoking (say) threshold accepting with threshold, $L$, will expand the behaviour space such that $\mathcal{B}'_{yes}(\mathcal{L}, t_{hc}) \subseteq \mathcal{B}'_{yes}(\mathcal{L}, t_{ta(L)})$ with results similar to the fully correlated landscape discussed above. On the other hand, the corresponding expansion of $\mathcal{B}'_{no}(\mathcal{L}, t_{hc})$ to $\mathcal{B}'_{no}(\mathcal{L}, t_{ta(L)})$ makes it more likely that search sequences will be introduced that will allow starting points which were previously not able to find high-quality solutions in the time available (due to local optima and other barriers being in the way) to now access these solutions. Therefore the expectation is that the expected performance of $\mathcal{B}'_{yes}(\mathcal{L}, t_{ta(L)})$ will be higher than $\mathcal{B}'_{no}(\mathcal{L}, t_{hc})$.

The question is now whether the performance of these two sets of solutions ‘cross over’ at some point. The discussion in 3.8.2 shows that the amount of random walk/search character increases smoothly, with higher and higher barriers between regions of the fitness landscape being progressively overcome. Therefore, it can be seen that these two sets would be expected to have equivalent performance at the limit where $\mathcal{B}'(\mathcal{L}, t)$ is fully random search/walk in behaviour, and as a result no such crossover in relative performance should occur. From this argument, the relative performance of initialisation strategies is expected to be preserved. Consider the two sets of solutions, $initial_A(\mathcal{S})$ and $initial_B(\mathcal{S})$, and suppose that the set $initial_A(\mathcal{S})$ contains more starting solutions that can access high-quality solutions with a hillclimber than $initial_B(\mathcal{S})$. Therefore as the random walk/search character of the optimiser is increased, the optimiser based on $initial_A(\mathcal{S})$ would be expected to become relatively less effective whereas $initial_B(\mathcal{S})$ becomes relatively more effective (from the above argument), finally reaching equivalent performance at the limit of random walk/search behaviour. Therefore it would be reasonable to assume that the relative performance of initialisation strategies would be insensitive to the choice of local search optimiser used.

## 4.9 Problem (Goal) Specification Knowledge

"Be careful what you wish for — you might just get it!" — Unknown

In neighbourhood search algorithms, the objective function plays the roles of defining what the desired solution is to be by stating the relative quality of the solutions in the search space. Naturally, it is important to get this right, otherwise the wrong problem would be optimised. This view makes claims such as 'What You Want Is What You Get' (WYWIWYG) [Koza *et al.* 96a, Koza *et al.* 96b], used by Koza in the context of Genetic Programming [Koza 92] sound over-optimistic in the extreme. For lovers of acronyms, Hancock's [Hancock 92] 'What You Test Is What You Get' (WYTIWYG) is far more accurate. Therefore, this thesis argues that the design of the objective function is itself an exercise in knowledge representation. To this end, an appropriate formalism for performing this task will first be identified and outlined and issues arising from two examples of using this approach discussed.

### 4.9.1 The Role of Utility Theory

The first questions to address are how to formulate the user's desires at the knowledge level and how to represent this knowledge to the optimiser. Fortunately, **utility theory** provides a way of doing this [Keeney & Raiffa 76]. From this perspective, all that is required is that we can provide **preference-indifference** relations, $A \succ B$ ($A$ is preferred to $B$), or $A \sim B$ (A and B are equally preferable) between any two given points in the search space. The two conditions most commonly applied are: **transitivity**[9] and **orderability**[10]. These conditions are important as the aim of the systems that we are trying to construct is that they assist in rational decision making. Therefore if the answer produced contains logical inconsistencies then it would be simply wrong (and therefore useless), that said in some situations the transitivity condition (ie. orderability) may be relaxed, for example when the comparision of two solutions has a stochastic element.

Therefore a knowledge level description of the user's desires is possible by use of a **compar-**

[9] Given any three solutions, if $s_A$ is preferred to $s_B$ and $s_B$ is preferred to $s_C$ then $s_A$ is preferred to $s_C$ (this is to ensure consistency). That is, $\forall s_A, s_B \in \mathcal{S} : (s_A \succ s_B) \wedge (s_B \succ s_C) \Rightarrow (s_A \succ s_C)$.

[10] Given any two solutions, one must be preferred over the other, or rated as equally preferable (ie. the optimiser must know what it wants). In other words, $\forall s_A, s_B \in \mathcal{S} : (s_A \succ s_B) \vee (s_B \succ s_A) \vee (s_A \sim s_B)$.

**ison function** $s_A \succsim s_B$ that returns which solution, if any, is preferred. This knowledge can then be represented in two main ways. First it is possible in principle to construct one of a number of scalar objective (or **utility**) functions from the axioms of utility which dictate the form of the function[11].

Another alternative is to use the comparison function directly. As most of the techniques make direct comparisons between pairs of solutions, this is easily implemented. That said, it should be noted that, from this viewpoint, a hillclimber, tabu search or even an EA with rank-based or tournament selection would appear to be a more natural choice than techniques such as simulated annealing that *do* rely upon a scalar objective function. In fact if this formalism is to be used then some techniques may not be appropriate if for some reason it is not possible (or convenient) to construct a scalar fitness function. This loss of choice is acceptable as it is more important to get a good model of the problem, than to be wedded to any one type of optimisation technique. This will be illustrated later.

### 4.9.2 Problems with Multiple Objectives

The first difficult case to be discussed here is a **multi-objective** optimisation problem where there is more than one objective to be optimised, and where a gain in one objective may be offset by a loss in at least one of the others. One such example in scheduling would be minimising both work in process (WIP) and makespan, as a low makespan implies heavy resource utilisation, which in turn increases WIP. This transforms the concept of an optimal solution to a set of compromise solutions known as the trade-off surface or **Pareto-optimal** front in the **objective space** (Figure 4.8). These solutions are optimal in the sense that improvement in any objective can only be achieved by degrading at least one of the other objectives.

Therefore the difficulty arises of how to represent such problems to the optimiser. These issues will be explored in the context of the two representational options described earlier.

[11] See [Keeney & Raiffa 76] for a review, or [Russell & Norvig 95] for a gentle introduction, however the basic axioms of utility are:

- $\forall s_A, s_B : f(s_A) > f(s_B) \Leftrightarrow s_A \succ s_B$.
- $\forall s_A, s_B : f(s_A) = f(s_B) \Leftrightarrow s_A \sim s_B$.

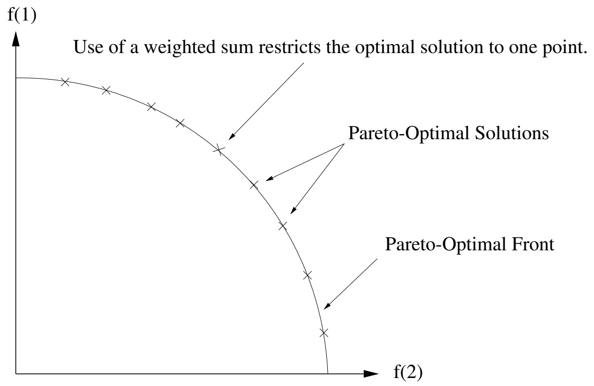


Figure 4.8: A Pareto-Optimal Front

### Multi-Attribute Utility Theory

As noted earlier, it is possible to construct a scalar fitness function to capture the users preferences. These situations can be tackled with **multi-attribute utility theory** (MAUT). This attempts to identify regularities in the preference behaviour so to select an appropriate **representation theorem** that suggests a suitable form of the objective function. More formally, let $f(s)$ be the overall objective function, and $o_i(s)$ be the functions that return the objective values. From this the representation theorem suggests an utility (objective) function of the form:

$$f(s) = F(o_1(s), o_2(s), \dots, o_n(s))$$

From this it is hoped that $F(o_1(s), o_2(s), \dots, o_n(s))$ is a simple function. Due to this desire for simplicity, the two most common forms used are **additive** and **multiplicative** utility models which correspond to a weighted sum/product of the objective values. This approach suffers from a number of objections, however. First of all, combining the objectives into a single value, by using for instance a weighted sum, would restrict the optimal solution to a single point on the front which may not be desirable if there is uncertainty about what a suitable tradeoff is. Furthermore, though the representation theorem may suggest a *form* for the objective function, it does not say anything about specifying the relative importance of the objectives according to their weightings. Therefore, even if for example, an additive model was appropriate, it may not be straightforward to decide upon suitable weightings for the objective function (which the uncertainty about a suitable tradeoff between the objectives only makes worse).

#### Preference-Based Selection and Dominance

Of course, an alternative would be to directly use the comparison function. From this, one way to tackle this problem is to admit our ignorance of the situation and to construct a **partial comparison function** based upon the idea of **dominance**, which says that one solution is better than another if and only if it has equal or better performance for all of the objectives and that at least one objective is superior [Fonseca & Fleming 95]. More formally, let $\mathcal{O}$ be the set of objectives being considered, then dominance can be defined as given below:

$$s_A \succ s_B \stackrel{\triangle}{=} (\forall o_i \in \mathcal{O} : o_i(s_A) \geq o_i(s_B)) \wedge (\exists o_j \in \mathcal{O} : o_j(s_A) > o_j(s_B))$$

The user can therefore get around the problem of not knowing the trade-off between objectives by obtaining a number of solutions scattered along the Pareto-optimal front and selecting the one desired. The above does, however, allow for additional preference information to be added — in this case if the above expression decides that there is no preference between the two solutions, then additional test(s) can be invoked to break the tie.

Finally, the approach described in [Fonseca & Fleming 95] uses the above comparison function directly, though it should be noted that other, more implicit, methods have been proposed in the EA literature such as the VEGA (**vector evaluated genetic algorithm**) due to [Schaffer 84, Schaffer 85].

### 4.9.3 Constrained Optimisation Problems

The above shows one case where utility theory considerations can be used to shed light upon how to represent user preferences. The next example considered here looks at **constrained optimisation problems** where the aim is to optimise a given objective, $o(s)$, whilst satisfying constraints upon feasibility (denoted by $\mathit{feasible}(s)$). Again, the utility theory formalism allows us to shed light upon the number of approaches that have been proposed to deal with this class of optimisation problem (some of which will now be considered in turn).

### Penalising Infeasible Solutions

The simplest method is to assign a very low fixed objective value (such as zero) to the infeasible solutions as shown below.

$$f(s) = \begin{cases} 0 & : \neg feasible(s) \\ o(s) & : \text{otherwise} \quad (\forall s : o(s) > 0) \end{cases}$$

The above has the disadvantage that the infeasible region(s) of the landscape will become plateaus and therefore no information is made available to the optimiser as to where the feasible region(s) are. One possible way around this problem is to return a (small) value for infeasible solution which is higher the closer the solution is to a feasible region — however it may not always be possible to devise a method by which to do this. Also, this addition may well be little different to using a penalty function approach, where a (large) penalty is added to the objective function according to how many of the constraints $c_i \in \mathcal{C}$ are broken (in which case $c_i = 1$, and is zero otherwise) as shown below.

$$f(s) = o(s) - \sum_{\forall c_i \in \mathcal{C}} \alpha_i \times c_i$$

This approach in not without its problems. First of all, the above additive model may not reflect the actual structure of the user's preferences. One example of this is in problems where some constraints may be relaxed so long as others are satisfied (**soft** as opposed to **hard** constraints). Even given that the interactions between hard and soft constraints can in principle be represented by this approach, finding a suitable set of weights for the different types of constraints that may be present in the problem may be non-trivial, especially as they will also interact with the values returned by $o(s)$. Finally, both of the above approaches carry the very real danger that the feasible regions of the fitness landscape may be 'cut off' by virtue by being separated by infeasible regions of poor quality, which have the effect of confining the search to the first feasible region found by the optimiser.

### Preference-Based Selection

As some of the above issues are also common to multi-objective optimisation, it should not be surprising to find that constrained optimisation problems can also be cast in this manner. The

COMOGA approach in [Surry *et al.* 95, Surry 98] considers the degree of feasibility of the solution to be another objective in a multi-objective optimisation problem. This approach was then applied to the optimisation of gas pipeline networks. One possible variant to the above is to use a **hierarchical comparison function** (which was hinted at earlier). Here solutions are first compared on the basis of their degree of (in)feasibility, and if there is a tie (for instance if both solutions are feasible), then solutions will be compared on the basis of their values of the objective, $o(s)$. This can also be used in the multi-objective case by looking at the most important objectives first, then, if there is a tie, the next most important objectives (and so on...).

### Search Space Reduction and Repair Operators

Another approach is to ensure that the infeasible solutions have been excluded from the search. The classical example of this is the GENOCOP EA system [Michalewicz 92] for the function optimisation of problems with linear constraints (initially expressed as either equalities or inequalities) and a non-linear objective function. The first stage of GENOCOP is to reformulate the problem to convert the equality constraints into their corresponding inequalities. Such problems have the property that the feasible region is convex and therefore assuming that all but one of the variables to be optimised is fixed, then there will exist a feasible range $\langle \mathit{left}(k), \mathit{right}(k) \rangle$ for that free variable $k$. Therefore, [Michalewicz 92] devised a number of mutation operators that modified only one variable at a time to a value within its current feasible range. Therefore only feasible solutions were generated by mutation. As regards recombination operators, the position is more straightforward. For instance, GENOCOP again exploits the fact that the feasible region is convex. All solutions intermediate between two points in the feasible region will themselves be feasible — this was used to produce the arithmetical crossover operator.

Formally, this above approach equates to moving the problem of infeasible solutions from the objective function to the search space reduction knowledge source. This circumvents many of the difficulties noted above, *and* makes the search space smaller as a result. In addition, any barriers between feasible solutions will be removed as a result (Figure 4.9).

To this end, a specification of the infeasible solutions will have to be represented in the set of constraints upon the allowed solutions such that $feasible(s)$ is incorporated into $legal(\Psi)$. Of

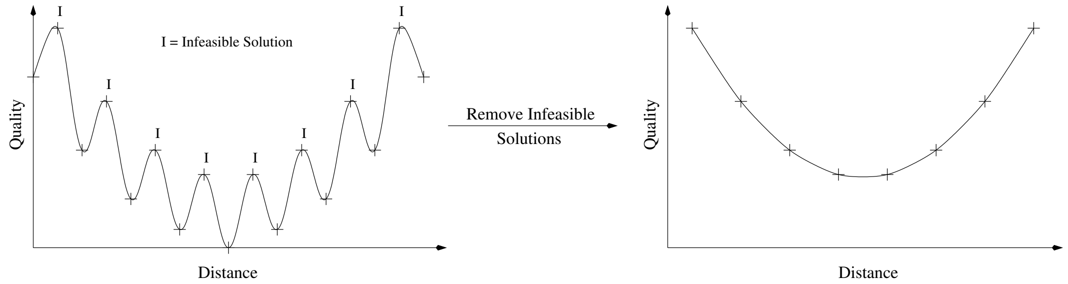


Figure 4.9: The Effect of Removing Infeasible Solutions from the Landscape

course operators need to be defined so that the above exclusions are possible. Though forma analysis can, in principle, be used to derive suitable operators, an equivalent though less formal approach is to use standard operators and then, if the generated solution is infeasible, to apply a **repair** procedure (which is often heuristic in nature).

Of course the caveat noted in Section 4.4.2 remains — the costs of ensuring feasibility (ie. in more computationally demanding neighbourhood operators) must be outweighed by the gains possible by excluding the infeasible solutions. Therefore the use of *heuristic* repair methods in some cases can be justified as providing a reasonable compromise in the face of the above trade-off. In addition, it is not always possible to preserve properties such as convexity, for example it is possible to produce a landscape with unconnected 'islands' of solutions that are unconnected with each other.

### 4.9.4 Summing Up

In summary, the above examples illustrate that specifying the designer's knowledge of the user's desires is an exercise in knowledge representation, and that such knowledge can be represented to the optimiser in a number of ways which map quite naturally to the view described here. In fact the importance of this knowledge role is such that design heuristic 1a earlier stated that is should be considered first of all.

The above said, each of the approaches to the examples described above does result in a different landscape — therefore the overall effect of the issues raised above can be compared and evaluated by the use of hillclimbing experiments (due to the design heuristics described earlier). Finally, excellent alternative overviews of the above methods can be found in [Michalewicz 92].

## 4.10 Search Control Knowledge

The final knowledge source, search control knowledge, comprises the remaining optimiser knowledge which is drawn from the search dynamics theory. Therefore, by default, all of the technique-specific additions that control and parameterise the various hillclimbing extensions (techniques) reviewed in Chapter 2 fall under this class.

Quite correctly, emphasis has been placed here on designing optimisers in terms of the problem domain theory, as the mapping between optimisation technique parameters and the landscape being searched is usually *implicit* and poorly characterised, and the underlying search dynamics theory is very incomplete (which was also the main reason for the first design heuristic in Chapter 3 and its later specialisations). That said, search control knowledge has a role to play. This is because problem solving knowledge is almost always heuristic in nature, and therefore may occasionally be misleading to a hillclimber (in that the resulting landscape may contain, for instance, local optima). Therefore, some mechanism to allow the optimiser to occasionally override/ignore the choice suggested by the problem solving knowledge is often advantageous. This can be achieved in a number of ways, for instance randomisation in the case of simulated annealing, or explicit rules based on the search history in tabu search.

Of course in order to devise such mechanisms, some knowledge of how the landscape produced by the problem solving knowledge deviates from the 'ideal' case is required (as noted in Chapter 3 this is equivalent to finding out how accurate the hypotheses from the problem domain theory used to build the landscape are). Therefore, such mechanisms can be constructed in terms of the search dynamics theory by characterising the deviations from the ideal and modifying the traversal rules to deal with them.

Given that the emphasis here is upon exploiting problem domain theory knowledge, the discussion here will be comparatively brief and argue that the results of the problem solving and goal specification knowledge acquisition process can be used to inform the process of designing traversal rules.

### 4.10.1 The Impact of Problem Solving and Goal Knowledge on Search Control

It should be noted that apart from its impact upon the landscape, the problem solving and goal knowledge can be used to assist in the design of suitable traversal rules. First of all, the earlier discussion of constrained optimisation problems in Section 4.9.3 suggests that landscapes based on the penalty function approach will be characterised with feasible regions separated by high fitness barriers of infeasible solutions. In this case, some form of restart search control may be useful here, as a mechanism, such as simulated annealing, that allows the optimiser to escape from local peaks will be ineffective as a temperature high enough to allow the optimiser to move between feasible regions will also be too high to exploit the correlations between feasible solutions. However, if search space reduction is used to remove the infeasible solutions, then the barriers between feasible solutions will be removed and the local optima would be expected to be (relatively) small and local which suggests an optimiser such as simulated annealing or threshold accepting.

Furthermore, many search control mechanisms are (or can be) defined in terms of the problem solving knowledge. A good example of this is in tabu search where many of the search control mechanisms (eg. recency, frequency) are defined in terms of solution attributes — it should be clear that these attributes correspond to the basis set of features, $\Psi$, used to define the landscape. This point will be illustrated below with a more detailed example from the evolutionary algorithms literature.

### 4.10.2 A Detailed Example: Sexual Selection

Traditionally in EAs the second parent for recombination is either selected on the basis of fitness, or randomly. In natural (biological) systems, this does not occur as organisms often select on the basis of similarity to maintain an optimal level of inbreeding/out-breeding — a form a sexual selection described in [Miller 94]. This can be cast into optimisation terms by considering the multi-modal landscape (Figure 4.10) below.

Consider a solution (marked ‘S’ in Figure 4.10) that has been selected for recombination with one of the five solutions so marked in Figure 4.10. It should be apparent that of the alternative solutions, the nearest is the best choice as recombination in that case will have the highest expected gain in solution quality from parent to child — in the other cases there is a significant

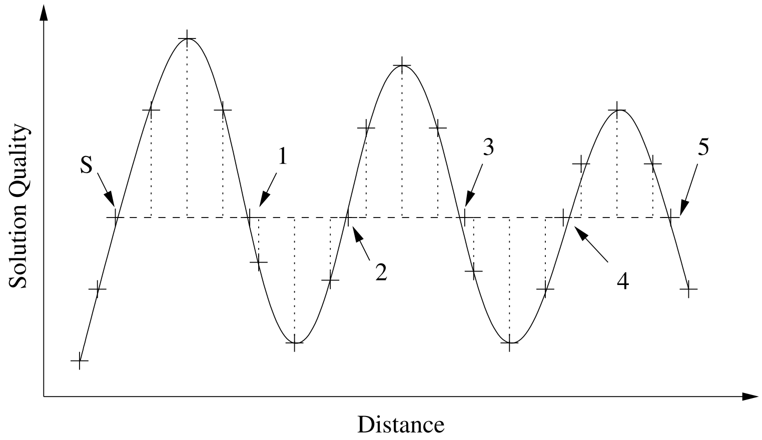


Figure 4.10: A Multi-modal Landscape to Illustrate Sexual Selection

probability that recombination would generate a solution in one of the low-quality ‘barriers’ between the basins of attraction.

In other words, in multi-modal landscapes such as those in Figure 4.10 solutions that are far apart may well be in different basins of attraction, and thus the quality of the parent solutions solutions is often a poor guide to the quality of points in the landscape in-between them. However, in this situation, it is evident that recombination between solutions that are very similar will also greatly diminish recombination’s ability to perform a more exploratory search between solutions, as the moves produced begin to resemble mutation.

Therefore **sexual selection** has been introduced to deal with the above. One aim of this method is to maintain a productive balance between inbreeding and out-breeding (ie. mating between very close and very distant solutions is discouraged), so as to select the most useful parents [Miller 94]. Work by [Ratford 96, Ratford *et al.* 97] investigating the utility of sexual selection in an optimisation context over a range of test problems, found that it produced significant improvements in performance, as well as enhancing the EA’s ability to obtain multiple solutions.

In order to implement sexual selection, an effective distance metric is required — which obviously has to be the same as that used for the landscape/recombination operator, otherwise it would not be a useful guide for relating the distance between solutions to their effectiveness with respect to the landscape. As forma analysis defines this distance metric in terms of the number of feature with different values for their equivalence classes, then once the beliefs about the correct landscape have been found, the sexual selection distance metric follows automatically. In other words, an important aspect of the design of this search control mechanism can be derived from the problem-solving knowledge.

## 4.11 The Proposed Design Procedure

From the design heuristics proposed so far, a suitable ordering of steps for a systematic procedure for the design of neighbourhood search optimisers can now be outlined.

1. First of all, find a suitable domain expert and find out what is required of the system and outline a model of the problem being solved. This is all rather standard requirements analysis/specification advice, but is worth repeating.

2. Devise a suitable evaluation/comparison function and validate it (design heuristic 1a).

3. Now investigate the problem solving knowledge sources (design heuristic 1a) as follows.

   (a) Devise hypotheses about which features of the problem correlate with solution quality and use forma analysis to derive suitable operators. Adopt the most suitable hypothesis suggested by hillclimbing experiments, as well as the most effective hillclimbing type (noting the later experiments will need any-ascent/stochastic hillclimbing if they are to be transferrable to EAs). Remember to take note of the caveat to design heuristic 2 and design heuristic 5 noted in Section 4.4.2 earlier.

   (b) Devise and formalise hypotheses concerning linkage and search space reduction, and use these to modify the neighbourhood operator appropriately. Then use hillclimbing experiments to find the most suitable hypotheses and adopt them.

   (c) After this, now that the move operators have been decided upon, devise and evaluate hypotheses concerning move preference and compare using hillclimbing experiments.

   (d) Consider the hypotheses concerning heuristic initialisation and use hillclimbing experiments to evaluate them.

4. Now consider the various search control options (design heuristic 1). The design decisions made so far stand by virtue of design heuristics 2, 5, 6, 7, 8, and 9. Also note that

design heuristic 3 allows transfer of the problem solving knowledge to the EA recombination operator (with the aid of forma analysis), when any-ascent/stochastic hillclimbing experiments have been used.

5. Finally, validate and then deploy the system. Maintain/modify as required.

The linear nature of the above suggests some similarity with the **waterfall model** of software development [Royce 70] and thus implies that this is the approach to be used. However, it should be stressed that this is not the case, and it should be noted that the waterfall model is now considered somewhat outdated. What is important to note here is the design heuristics *suggest* what aspects of the system should be considered before others, and describe *whether* changes that are made in one part of the optimiser affect earlier design decisions. For instance, changing the set of basis features would affect *all* of the design decisions made regarding the problem solving and search control knowledge, but the validity of decisions regarding the goal knowledge would remain unaffected. Therefore, given that these guidelines are observed, any development model that is thought appropriate to the circumstances can be used. For example, the **rapid prototyping** [Vonk 87] or **incremental development** [Glib 98] models could be argued to be well-suited to the design of neighbourhood search methods as all of their knowledge sources do not need to be specified in order to work (as defaults can be used). Finally, the reader is best directed to a standard software engineering textbook such as [vanVilet 93] for a general coverage of these issues.

## 4.12 A Discussion of Related Work

No work exists in a vacuum and the work here and in Chapter 3 is no exception as it draws upon a number of previously expressed ideas, and is related to a few others. Therefore the success of this work also partly depends upon how it combines, extends, and improves upon the previous works and addresses their shortcomings. To this end, this section will now discuss further the contribution of this work, and also how it differs from and extends the previous work it draws upon.

### 4.12.1 Natural Operators

Much inspiration for this work has come from the work of [Michalewicz 92] which advocates the case for the use of 'natural' encodings and operators in evolutionary algorithms that are appropriate to the problem at hand to produce what he call **evolution programs**. The discussion in [Michalewicz 92] also credits earlier work such as [Davis 89] for noting the need for non-binary, domain-specific formulations of evolutionary algorithms. This view is mirrored by the OR community (eg. [Reeves 93b]) which has long noted that the choice of neighbourhood operator is important and should reflect the nature of the problem.

Unfortunately, the earlier discussion in Chapter 3 notes that despite the fact that this approach succeeds in many cases, it does suffer from a number of methodological drawbacks because the concept of an appropriate operator can be considered tautological, and the issue of what makes a good encoding/operator is not addressed. Also, this work does not address the need for an experimental protocol that can be used as the basis of a systematic and principled design procedure (which was one of the criteria outlined in Chapter 3). Finally, the focus of optimiser design in [Michalewicz 92] was at the symbol level, rather than the knowledge level, as the term 'natural' in [Michalewicz 92] refers to data structures that are used by the evolutionary algorithm. In fairness, this is true of virtually all current work in neighbourhood search, though some (for example [Maini *et al.* 94]) decide to call their operators 'knowledge-based' even though a knowledge-level analysis is absent.

In the light of these objections, this work makes a number of useful contributions. First of all, the issue of what makes a good operator/landscape was discussed at some length in Chapter 3, and the fact that the fitness landscape can be formally attached to hypotheses concerning the problem domain which correspond to a number of knowledge sources addresses the question of what makes a good operator, as well as providing additional information on their design. Furthermore, the concept of a natural operator was further clarified in the context of the representational requirements outlined earlier in 4.2.2. Finally, the experimental protocols suggested by the design heuristics proposed here and in Chapter 3 provide the required experimental protocol, and allow the structured evaluation of the neighbourhood operator components outlined here.

### 4.12.2 Radcliffe and Surry's Forma Analysis

This work draws heavily upon the forma analysis formalism due to [Radcliffe 94, Surry 98] to form the basis of the formalisation of the semantics of the fitness landscape, and some of the other knowledge sources. However, the use of forma analysis in this work differs from the original work both in intent, and in the results produced.

The original intention of forma analysis was to generalise the EA schema theorem [Holland 75] to arbitrary representations. From this, its role was extended to allow the definition of **representation independent** operators [Radcliffe 94] with well-defined properties, such as respect and assortment for recombination operators, that were felt to be desirable. This development has recently [Surry & Radcliffe 96b, Surry 98] been extended to allow for the definition of **representation independent evolutionary algorithms** which can then be instantiated to produce different concrete optimisers depending upon the features chosen.

This thesis first of all addresses the fact that the original work has solely been applied to evolutionary algorithms, and has shown that its scope is in fact far more general. More importantly, with the addition of the design heuristics proposed here, forma analysis becomes a powerful tool in the design of neighbourhood search optimisers by addressing the lack of a justified experimental protocol in the above works. For example, hillclimbing experiments can now be used to decided upon what features the recombination operators should use, with forma analysis then enabling the derivation of a family of suitable operators.

In addition, the work here has extended forma analysis in a number of ways. The role of forma analysis in knowledge representation and providing landscape semantics was not as well characterised in the previous work as it has been here – though the previous work did make the more implicit connection between the choice of features and hypotheses concerning those features thought to be correlated to fitness [Radcliffe & Surry 94a, Surry 98]. This was presumably because of the different intent of this work. Furthermore, the useful distinction between knowledge and symbol level descriptions of the optimiser was not made, and the decomposition of the other knowledge sources (such as move direction) was not addressed in the previous work.

Finally, the work presented here shows that the neighbourhood structure is in fact induced not only by the choice of basis features, but also by considerations such as linkage, search space reduction, and the appropriate size of move to be made. Therefore this work has proposed extensions to the forma analysis formalism to account for this. These extensions support the incorporation of the other knowledge sources, and it has been shown how this formalism can also impact upon the search control knowledge. Examples include the generalised $k-$opt operator, the various specialisations induced by the other knowledge sources (such as linkage), and the discussion of the suitable distance metric for sexual selection.

### 4.12.3 Heuristic Search Frameworks

A number of frameworks have been proposed for heuristic search, and more specifically neighbourhood search optimisers. One such framework was outlined earlier in Chapter 2 and was taken from the work in [Rayward-Smith 94]. The work there aimed to highlight the similarities between the various neighbourhood search techniques and therefore was not especially detailed. More recent attempts to produce formal detailed general frameworks for heuristic search have been proposed in the literature, and notable attempts are due to [Pirlot 93, Rayward-Smith 94, Eiben *et al.* 95, Harrison 99]. These aim to place both iterative and constructive search into a general search procedure from which the 'standard' search techniques can be instantiated. Though the work presented here concurs with these attempts to build general search frameworks — as they allow the similarities and differences between search techniques to be usefully highlighted, made explicit, and therefore allow their implementations be to better structured — there are a number of significant differences.

First of all, this work concentrates upon one class of iterative search technique (neighbourhood search) though there is no reason why the analysis here could not also be carried out for other types of search. Second, all of the frameworks consider the neighbourhood operator to define a transition between search points. Though this in itself is not a mistake, the arguments presented in this thesis suggest that a finer level of granularity may be appropriate and useful in the context of operator design. Also, the most important difference between these frameworks applies to the level of description that they address. In short, these frameworks correspond to *symbol* level descriptions of these optimisers, as opposed to the knowledge level description proposed here. Therefore the work presented here should be viewed as being *complementary* to the work on producing search frameworks. Finally, due to the *implementational* nature and intent of these frameworks, the important issue of structuring experimentation in the design process is not addressed. Therefore the design heuristics proposed here and in Chapter 3 constitute an additional contribution.

### 4.12.4 Problem Solving Methods

The final point of contact between this work and those of others is with the KBS literature itself. Recently, **problem solving methods** (PSMs) [Marcus 98, Benjamins 93] have been the focus of some interest in the KBS community. Problem solving methods have been characterised [Fensel *et al.* 97] as "[describing] in a domain-independent way which reasoning steps and which types of knowledge are needed to perform a task". In other words, the reasoning steps (ie. search mechanism) are abstracted and then (re)instantiated to produce a suitable concrete problem solver in a (hopefully) structured manner. This obviously has many similarities, at least in intent, with the work presented here. Therefore its should come as no surprise that a PSM analogous to neighbourhood search, **propose and revise** [Marcus *et al.* 98, Zdrahal & Motta 95] has been formulated as a reasoning pattern based on human search by trail and error. In addition, work in [Zdrahal & Motta 95, Motta 97] has argued that such problem solving methods can be placed in a structured taxonomy and, more specifically, that the propose and revise PSM is in fact a specialisation of the **generate and test** (ie, enumeration) PSM — a point which neatly corresponds with the heuristic search frameworks described above.

There are, however, significant differences between the work presented here and that in the PSM literature. For example, this work approaches the KBS formation of neighbourhood search optimisers in a 'bottom-up' manner. That is, instead of looking at and decomposing a particular reasoning pattern, a simple hillclimber was first considered and the possible ways of extending or better informing its behaviour were described. More importantly, the description of the knowledge sources proposed here is in fact much more detailed than that given by the propose and revise PSM. The differences will be summarised here. First of all, the propose and revise PSM currently considers only the initialisation procedure, the move operator, and the move selection procedure (analogous to move preference in this work) as knowledge sources. The knowledge-level analysis here differs in a number of ways. To begin with, the knowledge sources for the propose and revise PSM are defined as **procedural attachments** whereas the knowledge sources in this work are formulated and represented in a more formal and declarative manner which, as noted in Section 4.2.2, is preferable. Moreover, the work presented here argues that the neighbourhood operator is in fact composed of a number of knowledge sources, such as linkage, which are not considered by the propose and revise PSM. As a result, the knowledge level description of a neighbourhood search operator can be usefully made at a finer level of decomposition than that suggested by the propose and revise PSM. In summary, one notable contribution of this work is that it in fact extends and *improves* upon the PSM formulation of neighbourhood search optimisers!

Also, in common with the other related works described here, the PSM work does not consider experimental methodology. That is, though the knowledge-level analysis of the propose and revise PSM has identified knowledge sources, it does not consider *how* they interact, and from this in *what* order they should be investigated. In this work, the proposed design heuristics perform this important and neglected task.

Finally, it should be noted that the PSM work has been constructed purely from the viewpoint of AI and therefore is somewhat separate from the work being carried out in the OR and EC communities. As the work here and in Chapter 3 argues a KBS view from first principles, one contribution of this work is that it, in effect, transfers the work on neighbourhood search from those communities into the context of the KBS work described above. One such example of this is that knowledge representation issues pertaining to evolutionary algorithms were explicitly considered in this work.

## 4.13 Summary

This chapter has extended the discussion in Chapter 3, to advocate a KBS approach to the design of neighbourhood search optimisers, combined with systematic and justified experimental protocols such as those suggested in the previous chapter. To this end, the KBS approach, and the suitable properties for a representation, was reviewed and it was argued that the approach described in Chapter 3 mapped naturally to a KBS framework.

From this, the roles that knowledge plays in neighbourhood search were outlined and the first design heuristic was extended to take account of this. The problem-solving knowledge sources were then considered, discussed with reference to the current literature and suggestions for their formalisation provided. Concurrently further design heuristics were proposed and justified so that hillclimbing experiments could be used to empirically evaluate items of domain knowledge corresponding to these new knowledge sources in an effective and transferable manner.

Finally, the goal and search control knowledge roles were then considered and issues pertaining to them were discussed so to make the case that they too could be considered as exercises in knowledge representation. The remainder of this thesis will now concentrate on evaluating whether the approach described so far in fact constitutes a workable and effective methodology for the principled design of neighbourhood search optimisers.

# Chapter 5

# Experimental Methodology and Sequencing Overview

This chapter aims to set the scene for the later case studies. First of all, it will state which aspects of the work discussed in the earlier chapters are to be empirically investigated, and explain why they were chosen. The configuration of the optimisers used for the case studies will then be described.

Also, both of the case studies optimise sequencing problems. Therefore, a brief review of this domain will be given in the light of the forma analysis formalism described previously, and a recently proposed taxonomy of sequencing operators. From this, the operators that are to be used in the case studies will be described and the reasons for their selection given.

Finally, an outline of the statistical analysis used to ascertain the validity of the experimental results obtained will be given, and the reader is referred to the appendices for a full discussion.

**NOTE**: this chapter assumes that the reader is familiar with the material presented in Chapters 3 and 4, especially with regard to the design heuristics proposed there. To assist the reader, some indication of where this material can be located will be given in this chapter.

## 5.1 What is to be Investigated?

To arrive at the viewpoint in earlier chapters, a large number of issues were raised — too many, in fact, to be fully investigated in this work. Therefore some rationalisation of what is to be investigated is required. The aim is to test sufficiently representative aspects of the points raised so far. The selection will be on the the basis that if the experiments are successful, it would strongly support the acceptance of the entire package.

The hypotheses raised by the discussion so far that are most amenable to an experimental evaluation are the design heuristics. Also, as they focus upon the experimental protocols for problem solving knowledge, they comprise the most important contribution of the viewpoint proposed in this work. Therefore much of the validity of this view of neighbourhood search optimiser design rests upon the validity of these design heuristics. To this end, *which* of the design heuristics will be experimentally evaluated in this thesis are listed and then justified below[1]:

- **Landscape Independence** — relative landscape performance can be assumed independent of the choice of search control (design heuristic 2).
- **The Caveat to Design Heuristic 2** — the need for the effect of neighbourhood size to be taken into account.
- **Transferability to EAs** — relative performance of knowledge sources can be assumed the same for any-ascent hillclimbers and EAs (design heuristics 3 and 4).
- **Move Selection** — relative performance of move selection strategies can be assumed independent of the choice of search control (design heuristic 8) .
- **Heuristic Initialisation** — relative performance of move selection strategies can be assumed independent of the choice of search control (design heuristic 9).

The exclusion of the first design heuristic[2] (and most of its specialisations) are a result of their dependence upon the other later design heuristics. In other words, the first design heuristic is a pragmatic statement about the ordering of experiments. Now if the later design heuristics were to be incorrect in that changes to the choice of optimiser were to change the relative performance of hypotheses concerning the problem domain, then much of the utility of the first design heuristic would be lost. Therefore the validity of the first design heuristic is in

[1] The reader is referred back to the relevant sections of Chapters 3 and 4 for a full description of these design heuristics.

[2] As discussed in detail in 3.4.4, this states that options concerning landscape design should be considered before those concerning search control.

a large part dependent upon the validity of the later heuristics, and so they should be the heuristics tested.

These later design heuristics, as noted above, concern themselves with the transfer of problem solving knowledge sources between optimisers. Though, of these, the second and third design heuristics[3] concern themselves with the transfer of *landscapes* which themselves, as noted in Chapter 4, comprise a number of knowledge sources. As a result of this, the later design heuristics proposed in Chapter 4 for these component knowledge sources are specialisations of the second and third design heuristics and therefore are valid if the more general second and third design heuristics are.

Moreover, testing the fifth design heuristic concerning the effect of neighbourhood size is also necessary in this context — this can be done by seeing whether its corollary, the caveat to design heuristic 2, occurs in practice[4]. Furthermore, the fourth design heuristic which informs the design of EA recombination operators needs to be tested[5].

Finally, the design heuristics concerning move selection and heuristic initialisation do need to be tested separately[6]. In the case of the move selection design heuristic, though it is based upon the second design heuristic, its effect on the fitness landscape is rather implicit, which means that there are additional factors present that may affect its validity. However, the need for a separate test of the heuristic initialisation design heuristic is even greater as its justification is quite separate from the second design heuristic for landscape comparisons.

### 5.1.1 The Case Studies

In this work, these hypotheses will be evaluated in the context of two case studies which not only provide a realistic test of the above points, but also allows for the investigation of these problems in their own right. The first of the case studies, a standard OR problem, will be used to evaluate *all* of the above. The second case study is a real world problem and therefore is

[3] Informally stated, the relative performance of a set of landscapes can be assumed to be independent of the choice of search control in practice and is transferable to EAs (see 3.8.1 and 3.10 for details).

[4] This is described fully in 4.4.2, and notes that comparisons between landscapes to assess the suitability of problem features/formae should be conducted with operators with similar neighbourhood sizes.

[5] This states (in 3.12) that the problem features/formae/metric space used in the unary operation should also be used in the recombination operator.

[6] These are described fully in 4.7 and 4.8 earlier in this thesis.

more directed towards testing the ease of applicability and utility of the methodology proposed here. Therefore, as will become clear during the case study, it was only felt necessary to conduct a comparison of the neighbourhood operators (design heuristic 2). A more detailed description and rationale for the choice of case studies can be found in their respective chapters later.

**Are Two Case Studies Enough?**

One immediate question is whether two case studies are sufficient to support the arguments presented here. In response, it should first be said that each of these case studies required a sizeable amount of work and effort to perform and analyse thoroughly, and therefore two was the limit of what could be achieved within the limitations of a PhD.

Second, two MSc and undergraduate projects supervised by the author [Nakata 97, Ramos 97] provide supporting evidence of the proposed design heuristics in the context of two other problems. Other supervised projects have also made use of the ideas presented here, though in a more piecemeal and informal fashion. Therefore, there is more supporting experimental evidence available than it first appears. These additional studies will be covered in more detail in the conclusion (Chapter 8) of this thesis.

In any case, given the breadth of the material and ideas proposed earlier, it would be frankly impossible to exhaustively evaluate them all over the set of problem types that practitioners are likely to face. Therefore, the reader should see these case studies as providing an *initial* evaluation of whether the view argued here has validity, and whether follow-up studies to further evaluate and build upon the work presented in this thesis are worthwhile. The conclusion will outline the directions on which this work can be taken forward.

## 5.2 Optimiser Configuration

In order to test the design heuristics concerning the transferability of hypotheses across optimisers, a representative range of optimisers needs to be selected. To this end, this study considers a number of any-ascent/stochastic, first-ascent and steepest-ascent optimisers[7]. Due to the open-ended choice of parameters and extensions, the infeasibility of tuning complex

[7] The reader is referred back to Chapter 2 for a detailed description of these techniques.

implementations of them all, and the fact that excessive search control is against the philosophy of this method, simple 'textbook' or 'off-the-shelf' implementations of the more common neighbourhood search optimisers were used in both of the later case studies. The optimisers chosen, and their configurations will now be described in turn.

### 5.2.1 Any-Ascent/Stochastic Hillclimbing Based Optimisers

The stochastic (ie. any-ascent with a random ordering of moves) hillclimbing techniques considered in this study are described below.

- **Hillclimbing**. A simple stochastic hillclimber (SHC) without restart was used as the baseline for the investigations carried out here.

- **Simulated Annealing** (SA) was used with a simple cooling schedule of the following form: $T_{k+1} = \alpha \times T_k$, where $k$ is the number of annealing steps, and $\alpha$ was given by:

$$\alpha = \exp\left(\frac{ln(T_0/T_f)}{max\_evals/change\_freq}\right)$$

  where $T_0$ and $T_f$ are the starting and finishing temperatures; with the two parameters $max\_evals$ and $change\_freq$ denoting the number of evaluations the optimiser is to be run for, and the frequency of temperature changes (ie. the number of evaluations in an annealing step) respectively. The choice of cooling schedule was based on the set-up used in [Ross & Corne 95] for evolutionary timetabling, and one of the standard schedules described in [Lundy & Mees 86].

  Only the initial temperature, $T_0$, was tuned extensively. The other parameters such as the final temperature, $T_f$, were fixed for both case studies at 0.05, with temperature changes every 100 evaluations ($change\_freq$). These values were chosen on the basis of formative experiments. A range of values (based on an informal examination of previous SA implementations on permutation problems) were experimented with for each of the tunable parameters to evaluate their effect. Of these, it was found that the initial temperature had the most marked effect, and that the values stated for the other parameters were generally robust.

- **Threshold Accepting** (TA) was used with a linearly decreasing threshold schedule of the form $L_{k+1} = L_k - \alpha$, where $\alpha$ was set by the formula:

$$\alpha = \left( \frac{L_0 - L_f}{max\_evals - change\_freq} \right) \times change\_freq$$

  where $L_0$ and $L_f$ are the starting and finishing thresholds; with the other parameters being equivalent to those used by SA. Only the initial threshold, $L_0$, was tuned extensively. As for SA, $L_f$ was fixed for both case studies (at 0), with $change\_freq$ set to 100 evaluations. These values were chosen on the basis of formative experiments, like those performed for SA.

- **Record-to-Record Travel** (RTRT) was used with a fixed constant value of $L$ which was tuned for the problem at hand.

All of the above optimisers were implemented in ANSI C.

### 5.2.2 The Evolutionary Algorithm

For this study, a Davis-style, GENITOR [Whitley 89] steady-state (with kill-worst replacement), EA with an unstructured (panmitic) population model was implemented. The other EA design options used were as follows:

- EA Population Size: 100.
- Crossover: $p(Crossover) = [0.0, 1.0)$.
- Mutation: $p(Mutation) = 1.0 - p(Crossover)$.
- Selection: Rank-Based with bias = 1.5.
- Mate Selection: Random.

The choice of EA and the above values were chosen as being generally applicable and robust, based on results for sequencing problems in the EA literature (such as [Mott 90, Reeves 95a, Tuson 95]). The crossover/mutation probability (the probability that crossover is used as the neighbourhood operator) was the sole tunable parameter. Code for this technique was implemented in ANSI C.

### 5.2.3 Steepest/First-ascent Hillclimbing Based Optimisers

In addition to the above, the second design heuristic also contains the condition that it is only applicable to optimisers based upon the same type of hillclimbing. Therefore it is also useful to see whether it also applies to first and steepest-ascent hillclimbing based optimisers. To this end, the following optimisers were selected.

- **Hillclimbing**. First-ascent (FAHC) and steepest-ascent (SAHC) hillclimbers were implemented without restart,
- **Tabu Search**. Simple recency-based implementations of both (FATS) and steepest-ascent (SATS) tabu search with a static fixed length FIFO tabu list were used. No 'advanced' TS extensions such as additional aspiration criteria, candidate list strategies, or dynamic tabu tenures were implemented.

The form of the attributes for the tabu list is neighbourhood-operator dependent and therefore the attributes used will be described later in this chapter on a operator-by-operator basis.

### 5.2.4 Tuning and Performance Metrics

As implied by the discussion above, owing to the open-ended nature of the optimiser tuning process, it was felt that an exhaustive search of all of the tunable optimiser parameters would be unnecessary and undesirable. Instead, as a comparison of *landscape* and not *optimisers* will be taking place, the emphasis was moved to perform a relatively 'quick and dirty' tuning regime that is most of all equally *fair* to all of the optimisation techniques being considered in that a comparable amount of resources were used in tuning each one (which therefore hopefully excludes any implicit bias from that source). Of course, since each problem instance could produce a landscape with different characteristics, tuning will take place on an instance-by-instance basis (at least initially).

As is evident from the description of the SA implementation used, it was decided to use informal formative experiments to set what were felt to be 'sensible' values for all but one of tunable parameters (the one felt to have the most impact on optimiser performance) which was then exhaustively investigated.

In addition, to test the design heuristics a suitable performance metric needs to be selected. As the aim of these optimisers is, as stated earlier in Chapter 2, to find a 'good enough' or 'good as possible' answer in the time available, the metric used is the solution quality obtained after a set number of solution evaluations (which is determined by the context of the problem being solved)[8]. This was felt to be the most realistic metric used in practice where optimisation is time-limited. This is also an aspect that some comparative studies of optimisers tend to ignore — the performance metric used is often the best solution quality obtained for an arbitrary long time, which ignores the solution quality/time tradeoff which is central to the utility of heuristic methods (whereas this work makes an explicit assumption where this trade-off lies). As the proposed design heuristics make no assumptions about the optimiser performance metric used (consistency is all that is required), this choice will not affect their experimental evaluation.

The reader should note that detailed descriptions of the actual tuning regimes and performance metrics used will be given for each of the case studies later. Also in all cases, though comparisons will be made between the 'tuned' optimisers, results of the tuning experiments can be found in the appendices.

## 5.3 Sequencing Operator Overview

Both of the case studies focus upon sequencing problems that is, loosely speaking, problems that involve an ordering (permutation) of items. This field has been well studied with many neighbourhood operators being proposed — too many in fact to give a full review here. The derivation of suitable operators using forma analysis is really only necessary when no operators that manipulate the desired features are available. Therefore, the case studies will use the above formalism to select neighbourhood operators commonly used in the literature which are suitable for the investigation of the applicability of the proposed design heuristics.

These sequencing operators will now be described and reviewed below. The review will be in the context of the forma analysis formalism described earlier and a proposed taxonomy of sequencing operators. Additional to the operators defined for a permutation encoding two additional operators will be used for the purposes of comparison and to help evaluate the usefulness of this taxonomy.

[8] As the implementations of the optimisers above also returned the number of evaluations made before the best solution was found, this information is also included in the results in the appendices but not used in the analysis.

### 5.3.1 The Natural Permutation Encoding

Recent work [Mattfeld *et al.* 96] has proposed a taxonomy of sequencing recombination operators based upon the building blocks (formae) that they manipulate and the types of sequencing problem each is suited to. He proposes the following types of relevant building blocks.

- **Position/Absolute Order** is the absolute position of a item (eg. item 5 is at position 4); this was proposed to be suitable for assignment-type problems.
- **Precedence/Relative Order** is whether one task is performed before another (eg. item 5 appears before item 4) and was thought useful for 'scheduling' problems.
- **Edges/Adjacency** is whether two items are next to each other (eg. item 5 is next to item 4); operators based on these features were thought to be suitable for routing problems.

The following sections will formalise the above features, assuming a permutation of $N$ elements in the set $N = \{1, \ldots, n\}$. Basis set of equivalence relations will be described for each of the above, and then suitable operators from the literature will be described in terms of this formalism and the representation-independent operators reviewed earlier.

#### Position-Based Operators

The formalisation of position/absolute order, as described above, is performed as follows. Let $\Psi_{pos}$ be the set of basis **position equivalence relations** such that $\Psi_{pos} = \{\psi_{pos(i)} \mid i \in N\}$. In each case $i$ refers to the $i$-th position on the permutation and the equivalence classes are the set of $N$ positions, ie. $\forall i \in N : \Xi_{\psi_{pos(i)}} = \{\xi_1, \ldots, \xi_n\}$. In addition, the constraint needs to be added that a valid permutation exists if and only if none of the elements occupy the same position:

$$\forall i, j \ (i \neq j) : \ \psi_{pos(i)} \neq \psi_{pos(j)}$$

From this the distance metric can be defined simply as the number of elements in the permutation that are in different positions.

This formalisation allows us to specify a number of unary neighbourhood operators. Of course, the above constraint prevents the use of a 1-opt neighbourhood as this would involve placing two elements in the same position, but a 2-opt minimal mutation can be defined as the set of solutions generated by swapping all pairs of elements — this is the familiar **permutation swap** operator which will be used in the later case studies. Its operation is shown by the diagram (Figure 5.1) below, where the elements that are bounded by a rectangle are the two jobs selected to be moved by the operator.

$$1\ 3\ \boxed{6}\ 5\ 4\ \boxed{2}\ 7\ 8 \xrightarrow{\text{Swap}} 1\ 3\ 2\ 5\ 4\ 6\ 7\ 8$$

Figure 5.1: The Permutation-Swap Operator

This can be extended into the $k$-opt case where $k$ elements are reassigned positions. Applying the linkage specialisation[9] that the $k$ selected elements must occupy adjacent positions of the permutation, then produces the common **scramble-sublist** operator [Davis 89]. Both operators are given in Figure 5.2 below.

Attention can now be turned to the EA recombination operators. Now, [Radcliffe 94] proves that though position formae are separable, they are not $g$-separable — though both respect and proper assortment are simultaneously possible, strict transmission and proper assortment are not. Therefore if proper assortment is felt important, we can obtain this and maintain respect by using the **Position RRR** operator which copies across the positions that have common elements and randomly fills in the remaining elements (Figure 5.3).

This is, in fact, equivalent to the **C1 crossover** operator proposed in [Reeves 95a] — though

[9] Where the neighbourhood is reduced by allowing changes only to feature combinations with high mutual fitness interactions — see 4.5.

$$1\ \boxed{3}\ 6\ 5\ \boxed{4}\ 2\ \boxed{7}\ 8 \xrightarrow{\text{k-opt}} 1\ 4\ 6\ 5\ 7\ 2\ 3\ 8$$

$$1\ 3\ 6\ \boxed{5\ 4\ 2\ 7}\ 8 \xrightarrow{\text{Scramble}} 1\ 3\ 6\ 2\ 7\ 5\ 4\ 8$$

Figure 5.2: The Position $k$-opt and Scramble Sublist Operators

$$
\begin{array}{l}
\boxed{3\ 6}\ 5\ 4\ 2\ 7\ 8\ \boxed{1} \\
\boxed{3\ 6}\ 2\ 5\ 4\ 8\ 7\ \boxed{1}
\end{array}
\xrightarrow{\text{Remove}} 3\ 6 * * * * * 1 \xrightarrow{\text{Replace}} 3\ 6\ 7\ 4\ 8\ 2\ 5\ 1
$$

Figure 5.3: The Position RRR/C1 Crossover Operator

it was not found to be particularly successful as due to its disruptive nature (ie. low chance of forma transmission), therefore it is not commonly used (though it will be used here for comparative purposes).

Not surprisingly, less disruptive recombination operators are invariably used. Therefore the ‘Modified PMX’ operator [Mott 90], found to be far less disruptive of strings than C1/RRR, will also be used. A two-point crossover is performed upon the two strings selected. A repair procedure then analyses one string for duplicates: when one is found it is replaced by the first duplicate in the second string. This process is repeated until both strings are legal (Figure 5.4).

$$
\begin{array}{l}
3 \mid 6\ 5 \mid 4\ 2\ 7\ 8\ 1 \\
3 \mid 6\ 2 \mid 5\ 4\ 8\ 7\ 1
\end{array}
\xrightarrow{\text{Crossover}}
\begin{array}{l}
3\ 6\ 2\ 4\ \boxed{2}\ 7\ 8\ 1 \\
3\ 6\ 5\ \boxed{5}\ 4\ 8\ 7\ 1
\end{array}
\xrightarrow{\text{Repair}}
\begin{array}{l}
3\ 6\ 2\ 4\ 5\ 7\ 8\ 1 \\
3\ 6\ 5\ 2\ 4\ 8\ 7\ 1
\end{array}
$$

Figure 5.4: The Modified-PMX Crossover Operator

It should be finally noted that this operator, though *highly* transmitting is not *strictly* transmitting and is therefore not equivalent to Position G2X (though it can be considered a relaxation of it). This can be shown by counter-example in Figure 5.4, where it should be clear that elements 2 and 5 have not been transmitted.

**Precedence-Based Operators**

Precedences in permutations can be modelled in a similar fashion. First, let $\Psi_{prec}$ be the set of basis **precedence equivalence relations** for a permutation of $N$ elements, $\Psi_{prec} = \{\psi_{prec(i,j)} \mid i, j \in N \wedge i \neq j\}$ — in all cases $i$ and $j$ refer to the elements in the permutation. Therefore the equivalence classes are simply a true/false answer to whether the element $i$ proceeds the element $j$ in the permutation (ie. $\forall i, j (i \neq j) : \Xi_{\psi_{prec(i,j)}} = \{\xi_0^{i<j}, \xi_1^{i<j}\}$).

However the above basis set can, in principle, describe $2^{N^2-N}$ solutions which is more than the $N!$ possible permutations. This is because there are again constraints upon which combinations of equivalence classes conform to valid solutions (actual permutations). The first of the constraints notes that if $i$ is before $j$ then the reverse relationship cannot possibly be true:

$$\forall i, j \in N \ (i \neq j) : \psi_{prec(i,j)} \Leftrightarrow \neg\psi_{prec(j,i)}.$$

In addition, the constraint needs to be added that a valid permutation exists if and only if the relationship between the precedences is consistent (in that the transitivity condition is preserved) in the manner shown below:

$$\forall i, j, k \in N \ (i \neq j \neq j) : \ (\psi_{prec(i,j)} \wedge \psi_{prec(j,k)} \Rightarrow \psi_{prec(i,k)}).$$

Applying these constraints to the $N^2 - N$ basis relations gives us the $N!$ possible permutations desired. Also, the distance metric can be specified as the number of differing precedence relations between two solutions.

The issue of what a minimal mutation for position formae is now arises. First of all, precedence one-change is clearly impossible by virtue of the first of the above constraints. However a 2-opt operator is possible, which corresponds to the **swap-adjacent** operator which exchanges 2 elements of the permutation that occupy adjacent positions in the solution (Figure 5.5). The neighbourhood size of this operator is $N$ moves which is much smaller than the swap-neighbourhood ($N^2 - 1$). Therefore, if design heuristic 5 is to be respected, then a 'precedence-aware' operator with a similar neighbourhood size needs to be found.

$$1\ 3\ 6\ 5\ \boxed{4\ 2}\ 7\ 8 \xrightarrow{\text{Swap-adjacent}} 1\ 3\ 6\ 5\ 2\ 4\ 7\ 8$$

$$1\ \boxed{3\ 6}\ 5\ \boxed{4\ 2}\ 7\ 8 \xrightarrow{\text{k-opt}} 1\ 6\ 3\ 5\ 2\ 4\ 7\ 8$$

Figure 5.5: The Precedence 2-opt and $k$-opt Operators

One option would be to use a $k$-opt operator such as shown in Figure 5.5. However, a more commonly used operator, however, is the **permutation-shift** operator (Figure 5.6) which selects and removes an element from the permutation and re-inserts it elsewhere in the sequence.

Also, as will be noted in a later case study, this is an operator that appears to be quite natural (in that it would be an operation that a human would be likely to use) for a number of scheduling problems in that it captures the concept of a 'block of jobs' in the schedule.

$$1\ 3\ \boxed{6}\ 5\ 4\ \boxed{2}\ 7\ 8 \xrightarrow{\text{Shift}} 1\ 3\ 5\ 4\ 2\ 6\ 7\ 8$$

Figure 5.6: The Permutation-Shift Operator

This operator can in fact be viewed as a linkage specialisation of the $k$-opt operator where the $k$ precedences modified are those between the element removed and the elements between it and the second selected element it is inserted before/after (both elements are boxed in Figure 5.6). This operator has a neighbourhood size of $N^2 - N$ (ie. $O(N^2)$) which is comparable in size to the permutation swap operator. Therefore both this, and the swap-adjacent, operator will be used in the later case studies to test the caveat to the second design heuristic.

Interestingly, it would also appear that the position and precedence metric spaces are closely related. This can be seen from inspection of the shift operator. As the number of precedences changed by the shift operator increases, so does the number of positions in a smooth progression. This relationship is further illustrated in [Yamada & Reeves 97] by plots of the precedence distance against the position distance for a large number of solutions; this plot showed a high degree of correlation between the two metric spaces.

Finally a suitable recombination operator needs to be selected. Fortunately an operator has recently been devised that strictly transmits precedence formae: **Precedence Preservative Crossover** (PPX) [Mattfeld *et al.* 96]. This is the recombination operator that will be used in this study.

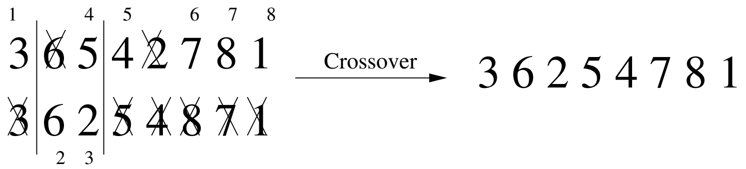


Figure 5.7: The 2-point PPX/Precedence G2X Crossover Operator

Figure 5.7 above illustrates the working of this operator. Two positions, one in each parent, are selected for crossover. It is also assumed in this example ' that the process also starts on the uppermost, 'current' solution in Figure 5.7. Now working from the left hand side of the current parent solution, elements are placed from the current parent solution into the child (building up the solution from left to right) and simultaneously removed from each of the two parent solutions. When a crossover position is reached, the current parent solution is changed. This process continues until the child solution is constructed. For ease of interpretation, the elements in the parent solutions are numbered in the order that they are taken to construct the child solution.

**Edge-Based Operators**

Let $\Psi_{edge}$ be the set of basis **edge equivalence relations** for a permutation of $N$ elements, ie. $\Psi_{edge} = \{\psi_{edge(i,j)} \mid i, j \in N \wedge i \neq j\}$. As **undirected** edges will be considered here, the equivalence classes are simply a true/false answer to whether the element $i$ is adjacent to $j$ in the permutation (ie. $\Xi_{\psi_{edge(i,j)}} = \{\xi_0^{i-j}, \xi_1^{i-j}\}$), such that $\forall i, j \in N(i \neq j) : \psi_{edge(i,j)} = \psi_{edge(j,i)}$. Again, the above gives rise to more combinations of equivalence classes (in fact $2^{N^2}$) than there are solutions to represent. This is overcome by adding the constraint that each vertex of the graph corresponding to the permutation can only participate in two edges and still be a valid permutation[10]:

$$\forall i, j \in N \ (i \neq j) : \ (\psi_{edge(i,j)} \Rightarrow !\exists k, l \in N \ (i \neq j \neq k \neq l) : \psi_{edge(i,k)} \wedge \psi_{edge(i,l)}).$$

From the above, the distance metric between any two solutions in the search space can be taken to be the number of common edges that they possess. Furthermore, as expected, the minimal move for this feature is the familiar **edge 2-opt** operator which removes two edges from the tour and then re-completes the tour by placing two new edges in. For a permutation encoding, this corresponds to the **permutation reverse** operator that selects a contiguous section, which can wrap over the ends, of the permutation and reverses the order of the elements within it as shown in Figure 5.8 below.

The above operator has a neighbourhood size equivalent to that of the swap operator. The reverse operator does, however, operate in a very different metric space to that of the shift and

[10] Note that $!\exists s \in S$ is interpreted as 'there exists *only one* object, $s$, in the set $S$'.

$$1\,\boxed{3\;6\;5\;4\;2}\,7\;8 \xrightarrow{\text{Reverse}} 1\;2\;4\;5\;6\;3\;7\;8$$

Figure 5.8: The Permutation-Reverse/Edge 2-opt Operator

swap operators. For the purposes of this discussion, let $L$ be the number of elements affected by the move operator — in the example in Figure 5.8, $L = 5$. Now irrespective of the value of $L$, the number of edges changed by the reverse operator is always two. However, the number of element positions changed is $L$, and the number of precedences changed is $L(L-1)$. Therefore a large proportion of solutions that could be considered to be close by in an edge metric space will be far apart in the position and precedence metric spaces, and vice-versa. In other words, the edge and position/precedence metric spaces are poorly correlated.

Selection of a suitable recombination operator now becomes an issue. Unfortunately, the nature of the formalism above makes this difficult. This is because comparison of two solutions using $\psi_{edge(i,j)}$ can flag equivalence in two ways: when elements $i$ and $j$ are adjacent in both solutions, and when both elements are not. This gives rise to two corresponding sets of formae which are termed **positive** ($\xi_1^{i-j}$) and **negative** ($\xi_0^{i-j}$) formae respectively. Work in [Hofmann 93] notes that it is the latter case that causes the problem. Some desirable operator properties, such as strict assortment, require that it be known whether a combination of formae corresponds to an empty set of solutions. In the case of negative formae, this is equivalent to the Hamiltonian Cycle Problem which is NP-complete [Garey & Johnson 79]. In addition, [Radcliffe 94] shows that edge formae are non-separable and therefore respect/transmission cannot be achieved without sacrificing assortment.

This study adopts a more informal 'edge-aware' recombination operator that was found to be effective for the TSP. This operator is the **enhanced edge recombination** operator devised by [Starkweather *et al.* 91] which has a high (98.8%) rate of edge transmission as illustrated by Figure 5.9 below.

The operator first constructs an edge table which contains, for each element, the elements in the two solutions it is adjacent to. The edge table corresponding to Figure 5.9 is given by Table 5.1 below. In addition, entries in the edge table that involve an edge common to

$$\begin{array}{l} 3\,6\,5\,4\,2\,7\,8\,1 \\ 3\,6\,2\,5\,4\,8\,7\,1 \end{array} \xrightarrow{\text{Crossover}} 3\,6\,2\,5\,4\,8\,7\,1$$

Figure 5.9: The Enhanced Edge Recombination Operator

both parent solutions are flagged (by a '+' in Table 5.1). This helps ensure respect and is the main modification between this operator and the original **edge recombination** operator [Whitley *et al.* 89]

| Element | Edge List | Element | Edge List |
|---|---|---|---|
| 1(F) | +3, 7, 8 | 5 | 2, +$4^4$, 6 |
| 2 | 4, $5^3$, 6, 7 | 6 | $2^2$, +3, 5 |
| 3(S) | +1, +$6^1$ | 7 | +$1^7$, 2, 8 |
| 4 | 2, +5, $8^5$ | 8 | 1, 4, +$7^6$ |

Table 5.1: An 'Edge Table' for Enhanced Edge Recombination

Given such an edge table, the procedure that this operator adopts is best described by the pseudo-code given in Algorithm 2 below. This procedure is further illustrated by Table 5.1 above which shows the order that entries in the edge table are considered so to produce the result described by Figure 5.9 earlier.

Finally, for a more in-depth and detailed discussion of the forma analysis work in this domain, the reader is referred to [Hofmann 93].

### 5.3.2 The Ordinal Encoding

The next neighbourhood structure considered here is the **ordinal** encoding originally due to [Grefenstette *et al.* 85], and detailed in [Michalewicz 92]. A string of N variables, numbered $i$ from left to right, with values in the range 1 to $N-1$ is used to encode the permutations in the form of a 'pick-list' (see Figure 5.10). The string is then decoded by proceeding from the start of the string and taking (and removing) the $j$-th element from the (ordered) permutation $\{1, 2, \ldots, N\}$, where the value of $j$ is given by the value of the string at that point — the process is then repeated until a permutation is produced. This is described by Algorithm 3 and Figure 5.10 below.

To formalise this encoding, let $\Psi_{ord}$ be the set of basis **ordinal equivalence relations** for a

**Algorithm 2** THE EDGE RECOMBINATION OPERATOR

Set up the Edge Table;
Let $P = \emptyset$;
*// Where P is the permutation to be constructed*
Let $E = \{1, \dots, N\}$;
*// Where E is the (unordered) set of elements in the permutation*
**repeat**
  Let $e \in E$ be randomly selected;
  **if** there is a common-edge element $c \notin P$, in the element's edge table entry $entry(e)$ **then**
    $P$ = APPEND$(P, c)$;
    $E$ = REMOVE$(E, c)$;
    Let $e = c$;
  **else if** there is *any* element $d \notin P$, in the element's edge table entry $entry(e)$ **then**
    $P$ = APPEND$(P, d)$;
    $E$ = REMOVE$(E, d)$;
    Let $e = d$;
  **else if** $entry(e) \subseteq P$ **then**
    Randomly select an available element $e \in E$ such that $e \notin P$;
    $P$ = APPEND$(P, e)$;
    $E$ = REMOVE$(E, e)$;
  **end if**
**until** $E = \emptyset$;
Return the permutation $P$ thus produced;

**Algorithm 3** TRANSFORMATION FROM AN ORDINAL ENCODING TO A PERMUTATION

Let $O$ be a list containing the ordinal representation of the solution;
Let $P = \emptyset$;
*// Where P is the permutation to be constructed*
Let $E = \{1, \dots, N\}$;
*// Where E is the numerically ordered set of elements in the permutation*
**repeat**
  $j$ = ITEM$(E,$ FIRST$(O))$;
  *// Take the $n$-th item from $E$ determined by $O$*
  $P$ = APPEND$(P, j)$;
  $E$ = REMOVE$(E, e)$;
  $O$ = REMOVE$(O,$ FIRST$(O))$;
**until** $E = \emptyset$;
Return the permutation $P$ thus produced;

permutation of $N$ elements, where $i$ refers to one of the $n$ positions in the encoding, ie. $\Psi_{ord} = \{\psi_{ord(i)} \mid i = 1, \ldots, n\}$. The equivalence classes are the set $\Xi_{\psi_{ord(i)}} = \{\xi_1, \ldots, \xi_{n-i+1}\}$ which correspond to each of the $j$-th remaining items in the permutation.

As the above describes a fully orthogonal $n$-ary representation, the distance metric in this case corresponds simply to the Hamming distance between the solutions. Also, the minimal mutation in this case involves taking one of the equivalence relations and changing its equivalence class as shown by Figure 5.10 below.

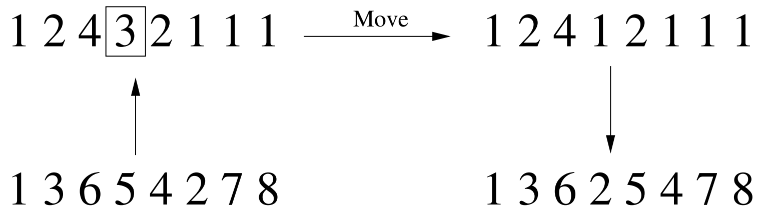


Figure 5.10: The Ordinal Neighbourhood Operator

The observant reader will note that the above example corresponds to a shift move where element 2 is removed and inserted before element 5. In fact, all of the minimal mutations have this effect. This, however, does not mean that the two neighbourhoods are equivalent, as they are not of the same size. As noted in [Ramos 97], the number of solutions in this encoding is $N(N-1)/2$ which is half the size of the shift neighbourhood — so where does the other half of the shift neighbourhood lie in the ordinal metric space? This question can be answered by reference to Figure 5.11 below which shows that the other half of the shift neighbourhood is in fact quite distant in the ordinal metric space — the distance increasing as the number of precedences changed by the shift operator increases.

Therefore, not only does the ordinal representation not correspond to any direct feature of the permutation — which means that it is unlikely be be used as a hypothesis about the structure of the problem — it also induces a metric space that is poorly correlated to any of the permutation features described above. However the main stated advantage of this encoding is that, unlike those described earlier, it is fully orthogonal. That is, standard EA operators can be used without additional constraints. Therefore for this study a standard uniform crossover operator was adopted.

Finally, for the purposes of this study, in common with earlier studies on $n$-ary representations

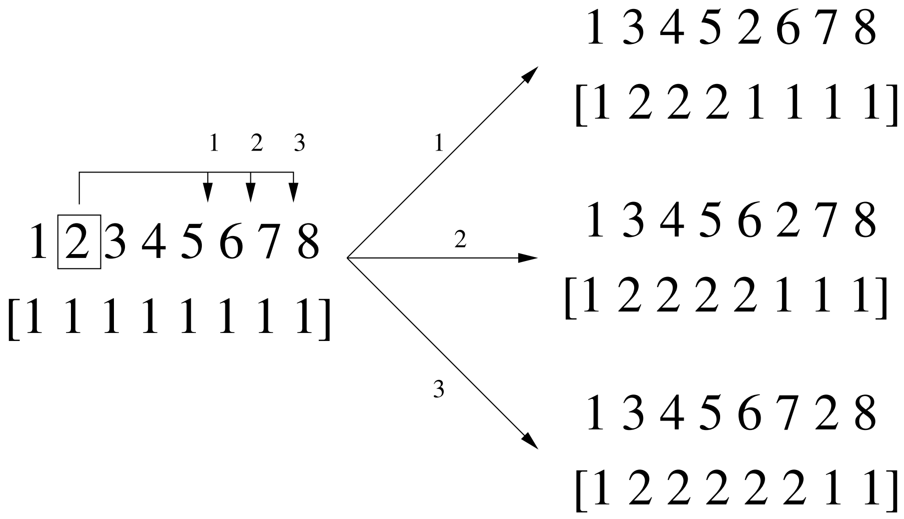


Figure 5.11: The Missing Half of the Shift Neighbourhood

such as described in [Glover *et al.* 93], first and steepest-ascent optimisers will only consider one (random) change to each of the variables/equivalence relations per iteration[11], rather than the entire set of variable/value (equivalence class/relation) changes — this also produces a smaller neighbourhood which is more easily explored.

### 5.3.3 The Random Keys Encoding

The last encoding scheme, **random keys** [Bean 94], considered here is another fully orthogonal representation of a permutation. In this case, the permutation is represented by a vector of numbers (either real or integer), as illustrated by Figure 5.12. Each variable denotes one of the elements of the permutation, and its value can be thought of as a 'priority' in that the elements are placed in the permutation in the order of decreasing priority (ie the element with the highest priority is the first element in the permutation), where tie-breaks are given in favour in the element with the lowest numerical value.

However, the form of the representation needs to be justified, especially as there is a choice of neighbourhood operators (eg. Gaussian, creep) for a real/integer encoding (which involves some notion of locality), as well as the landscape corresponding to an $n$-ary representation. The answer to this question arises from the fact that *all* the landscapes that correspond to

[11] This refers to one test of the acceptance criterion, which in the case of SAHC-based would be one complete evaluation of the neighbourhood.

real/integer/$n$-ary neighbourhoods will be characterised by plateaus. This is because, for a change in the *actual* permutation to take place, one of the numbers in the string has to become higher or lower than another. Any change smaller than any of those amounts will lead to no change in the phenotype at all and therefore the landscape will possess plateaus.

Therefore to reduce this problem, it was felt desirable to remove the locality information implicit in a real-coded representation (based on dedekind cut equivalence relations), and adopt an $n$-ary representation. This is because locality considerations would lead to an operator that favoured small changes, and by the arguments above this would mean that (unproductive) moves that correspond to no change in the underlying permutation would be more likely.

Therefore the basis set of **random keys equivalence relations**, $\Psi_{key}$, for a permutation of $N$ elements is given by $\Psi_{key} = \{\psi_{key(i)} \mid i = 1, \ldots, n\}$. The equivalence classes are the $\Xi_{\psi_{ord(i)}} = \{\xi_{min\_value}, \cdots, \xi_{max\_value}\}$ which correspond to the available priority values ($min\_value$ and $max\_value$ are set arbitrarily). As this is an example of an $n$-ary representation, the distance metric is simply the Hamming distance, and the move operator chooses one equivalence relation and changes its equivalence class as shown by Figure 5.12 below[12].

$$\begin{array}{ccc} [0.1\ 0.3\ 0.6\ \boxed{0.5}\ 0.4\ 0.2\ 0.7\ 0.8] & \xrightarrow{\text{Move}} & [0.1\ 0.3\ 0.6\ 0.75\ 0.4\ 0.2\ 0.7\ 0.8] \\ \uparrow & & \downarrow \\ 8\ 7\ 3\ 4\ 5\ 2\ 6\ 1 & \xrightarrow{\text{[Shift]}} & 8\ 4\ 7\ 3\ 5\ 2\ 6\ 1 \end{array}$$

Figure 5.12: The Random Keys Neighbourhood Operator

This operator can be considered, in a sense, equivalent to that for the shift neighbourhood. This is because increasing/decreasing the priority of one of the above variables causes it to be moved backwards/forwards in the permutation. However the shift operator cannot be considered completely equivalent as the random keys neighbourhood introduces 'null' moves where, as noted above, a move in the random keys landscape results in no change in the actual permutation represented. Of course it would be difficult indeed to argue that it would lead to more effective optimisation!

Finally, for the purposes of this study, in common with the ordinal representation described

[12] As is the case for conventional implementations of random keys, the $n$-ary values are replaced by real numbers in the range $[0.0, 1.0)$. This however makes no real difference to the operation of this representation

above, standard $n$-ary uniform crossover was used. It should be noted that, in this case, since the features (element, priority) manipulated by the above recombination operator correspond closely with the (position, element) features manipulated by operators such as Modified PMX (as priority is closely related to position) then the metric space explored by the recombination operator is also highly correlated with the position metric space. This (apparent) contradiction arises as a result of the decoding procedure used by this representation.

Also, as before, first and steepest-ascent optimisers will only consider one (random) change to each of the variables per iteration, rather than the entire set of variable/value changes.

### Possible Disadvantages of this Representation

Apart from the presence of plateaus in the fitness landscape as its possibly detrimental effect on search, there are additional concerns regarding this representation. The first would be that this could be abstracting the problem too much, in much the same way as earlier EA work advocated using binary strings to represent real numbers. Apart from the fact that this flies in the face of the work detailed here, why should abstracting permutations to reals/$n$-ary representations be any different from abstracting problems in the space of real numbers to a binary encoding (especially, as there are no spurious schema-processing arguments for using these types of representation)?

The next possible concern arises from the relationship between the changes in phenotypic and genotypic space, and that these depend on the relative values of the alleles. If they are of comparable value then small changes will be required, if they are not, correspondingly large changes will be required. This does not suggest that a highly correlated search space would be produced by this approach.

The representation is also highly redundant, and prone to the 'competing conventions' problem described earlier in Chapter 4. For example, consider two random keys representation, with values in the range $[0.0, 1.0)$, of the same permutation as shown by Figure 5.13.

Phenotypically speaking, these strings are identical (they represent identical permutations). However crossover between these two strings will produce children that are phenotypically very different from their parents. This makes it rather unlikely, especially early on in the EA run, that crossover will be able to usefully combine building blocks.

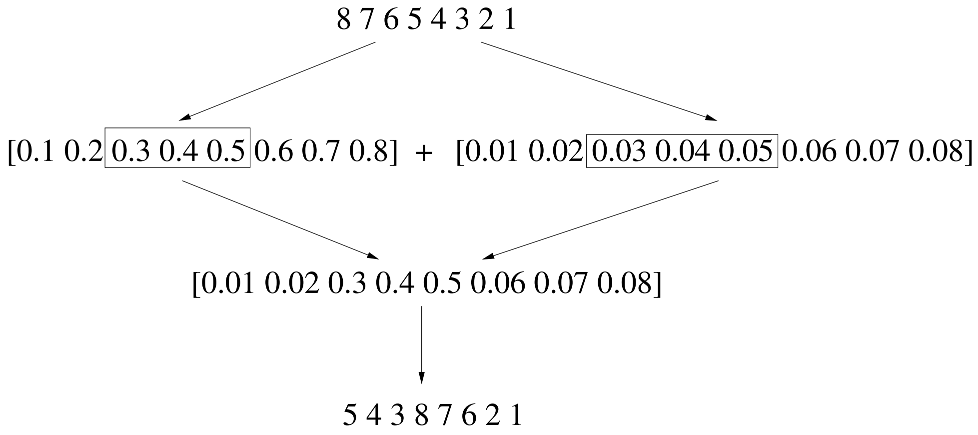


Figure 5.13: The Competing Conventions Problem for Random Keys

### 5.3.4 Tabu List Attributes

The final issue that needs to be addressed here is to devise suitable tabu structures for the tabu search implementation. These are different for each of the neighbourhood operators considered above and will therefore be dealt with in turn.

First, as the operators for the permutation encoding (i.e. permutation swap, shift, and reverse) can be specified in terms of which two positions on the permutation delimit the operator's effect, the tabu list for these operators simply contains the 2-tuple $(i, j)$ where $i$ and $j$ are the two positions on the solution. In addition, the $swap(i, j)$ and $reverse(i, j)$ operators are symmetric and so in this case, the presence of the 2-tuple $(i, j)$ will also be taken by the optimiser to imply the presence of the 2-tuple $(j, i)$.

However, the $shift(i, j)$ operator is not symmetric. For instance, position $i$ (which is element number 6 at position 3 in Figure 5.6 above) corresponds to the job that is displaced to make for the block of jobs that is to be shifted, the extent of which is given by position $j$. Exchanging the values of $i$ and $j$ in a shift operation therefore leads to a reversal of the roles of the two positions, and of the direction that the block is shifted.

The ordinal and random keys representations are $n$-ary and therefore were dealt with in a different manner. Like the neighbourhoods, the tabu list structure that was adopted in this case was based on that used for a simple tabu search implementation of the graph colouring problem described in [Hertz & deWerra 87], where all changes to that variable are then tabu. Here, the tabu list simply records the variable (basis equivalence relation) that has been changed, rather than the 2-tuple as was the case above. Thus in the case of the random keys representation, each item in the tabu list results in the exclusion of a larger number of possible moves than the definition for the shift neighbourhood above. However, due to the fact that the size of the random keys neighbourhood examined is similarly restricted, the *relative* effect is much the same.

Finally, as the swap adjacent neighbourhood swaps adjacent positions on the solution, it should be clear that $\forall i : swap(i, i+1) \Leftrightarrow swap\_adj(i)$ and so only the leftmost position that is selected to be swapped needs to be recorded by the tabu list.

### 5.3.5 Closing Remarks

This section has reviewed and discussed the operators that will be used in the later case studies. However, one aspect of the above discussion that is worth noting is that the encoding used need not necessarily be that suggested by the basis equivalence relations. For example, in the case of precedences, the suggested encoding is in fact a binary string, rather than a permutation. However, so long as the move operators are equivalent this is of little import, especially if it is noted that the permutation encoding probably makes most sense to a human user.

## 5.4 An Overview of the Statistical Methods Used

As the neighbourhood search algorithms under consideration are stochastic in nature, some form of statistical analysis is necessary. Unfortunately, despite this fact, the standard of statistical methodology in many studies can only be described as poor.

So how would one go about testing an hypothesis of the form described above? The key to this is to realise that what we need to do to falsify such a hypothesis is to find out whether the performance ranking of one of the optimisers in a given class is different from another optimiser in the same class.

Fortunately, this task is made easier because the $\geq$ relationship described above is transitive, and therefore pairwise comparisons are sufficient. This leads us to a method for falsifying the above hypothesis. If we can find cases where for one optimiser of a given class knowledge source $A$ *significantly* (in a statistical sense) outperforms $B$, but for another optimiser

$B$ significantly outperforms $A$, then these would count against the validity of these design heuristics.

Full details of the analysis used to do this is given in Appendix A. In summary, however, to test the hypotheses that are of interest in these studies, we need to see whether there is a case where a contradiction occurs in the relative ordering of knowledge source performance. This is done using three statistical tests for pairwise comparisions (Student's $t-$Test, Scheffé, and LSD).

Finally, the results of the statistical analysis are given in full in the appendices of this report[13].

## 5.5 Summary

This chapter sets the scene for the following case studies. First of all, it was outlined which of the issues raised in the earlier chapters are to be empirically investigated. The configuration of the optimisers used in later case studies were then outlined and justified. A review of the sequencing operators to be used was then undertaken (and their choice justified) in the light of the forma analysis formalism, the KBS framework described earlier, and a recently proposed taxonomy of sequencing operators. Finally, the important issue of the statistical analysis of the results was briefly discussed, and an analysis of appropriate statistical methods and their suitability in the appendices referred to.

Looking ahead, the following case studies will perform an evaluation of the proposed design heuristics by comparing the relative performance of a number of different hypotheses concerning a given knowledge source (which induce different landscapes) over a number of optimisers and problem instances. Statistical methods will then be used to ascertain whether there are cases where the predictions of the design heuristics are mistaken.

It is also hoped that, as a fortunate consequence of these case studies, useful results will be obtained in both solving these problems effectively and a better understanding of why (and why not) certain operators, initialisation methods, etc. work well.

[13] The statistical analysis was performed using Microsoft Excel for Windows 95 (Version 7).

# Chapter 6

# Case Study One: The Flowshop Sequencing Problem

The first of the studies considered in this thesis is the **flowshop sequencing problem**, which is a well-studied problem from the operational research literature. Therefore, to set the scene for the experimental investigations, this chapter will begin with an overview of this problem, its extensions, and the previous approaches used in the literature.

The design heuristics proposed so far, and selected for further investigation in Chapter 5, will then be investigated in turn. First, the second design heuristic will be evaluated by a comparison of the performance obtained using the sequencing operators outlined in Chapter 5 over a range of SHC, FAHC, and SAHC-based optimisers. Also, the caveat to this heuristic will be investigated and illustrated in the context of the above.

The next issue of import is the transferability of these results to an evolutionary algorithm. To this end, the applicability of these design heuristics (3 and 4 — see 3.10 and 3.12) will be evaluated by examining whether the results obtained for the SHC-based optimisers considered above successfully inform the choice of EA mutation and recombination operators.

The design heuristic (design heuristic 9 — see 4.8) concerning the heuristic initialisation knowledge source will then be evaluated. Three initialisation methods from the literature will be used as a basis to compare their relative performance across the above range of SHC, FAHC, and SAHC-based optimisers as well as an EA.

After this, the design heuristic (design heuristic 8 — see 4.7) concerning the move selection knowledge source will be investigated separately. This investigation will first be performed for the SHC-based optimisers and the EA in the context of a 'directed mutation' mechanism and a move selection heuristic based on 'idle time' (explained later). Attention will then be turned to the FAHC and SAHC-based optimisers where the above idle-time heuristic will be exploited in the context of a candidate list strategy.

Finally, the results obtained for the above investigations will then be discussed further in the conclusion of this chapter.

## 6.1 An Introduction to the Flowshop

**Flowshops** are quite common in the chemical industry [Ku *et al.* 87], where the continuous (but efficient) large-scale production of products required in only small amounts is undesirable; but sporadic batch processing is inefficient. To overcome this, a production line approach is taken where chemical precursors are fed through a string of reactors joined in a serial fashion (Figure 6.1).

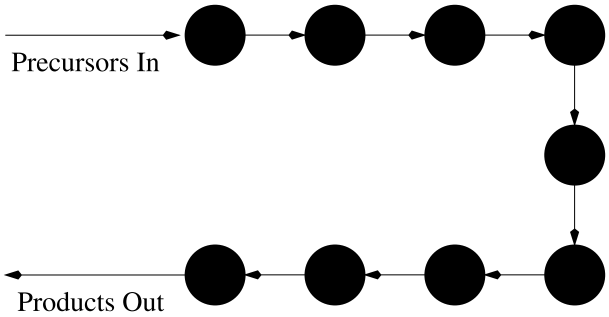


Figure 6.1: A Serial Flowshop

Flowshops operate in a *multi-product* fashion — different products can be synthesised by feeding the required precursors into the reactor and adding reagents as appropriate.

In all multi-product flowshops, each product (or **job**) has its own characteristic processing time in each reactor (or **machine**). Therefore, the efficiency of the line is dependent upon the order in which the products are produced — poor sequences can lead to temporary blockages in the flowline as completed stages cannot proceed, because a stage ahead in the flowline is still being processed (Figure 6.2). Obviously, this has a detrimental effect on throughput, and the sequence of jobs fed into the flowline is of importance; as these bottlenecks can lead to missed order deadlines, and the introduction of high avoidable costs.

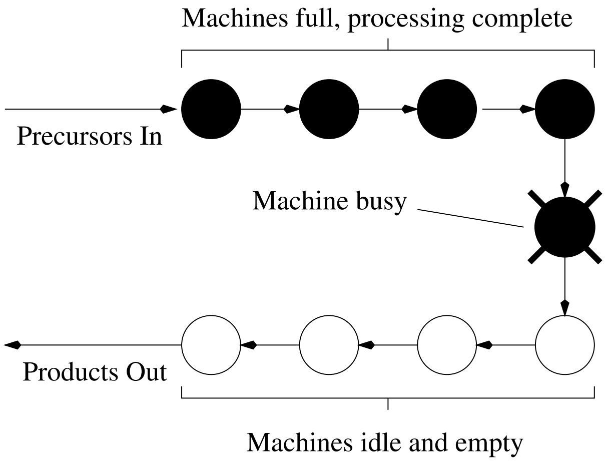


Figure 6.2: Blocking of a Flowshop at a Single Machine

To give some idea of the commercial importance of obtaining good schedules, consider the example [Mott 90] where a chemical plant may have 30 flowlines, each costing about 20 million pounds. If a 3.3% improvement in efficiency can be obtained over traditional methods, this would effectively leave an extra flowline. This saving is considerable, especially when you consider the additional potential benefits of improved customer service, energy savings and reduced manpower costs.

## 6.2 The $n/m/P/C_{max}$ Problem

The most common formulation of the flowshop sequencing problem in the research literature is to find a sequence of jobs for the flowshop to process, so as to minimise the **makespan** — the time taken for the last of the jobs to be completed. This task, termed the $n/m/P/C_{max}$ problem [Kan 76], is difficult due to the fact that it is NP-hard [Garey & Johnson 79] (the number of possible sequences is $n!$).

Of course variants of this the flowshop sequencing problem exist. Examples include job lateness [Allahverdi & Aldowaisan 98], sequence-dependany set up times [Corwin & Esogbue 74, Szwarc & Gupta 87], and stochastic scheduling [Cunningham & Dutta 72, Reeves 92].

### 6.2.1 Formalisation

This problem is formalised as follows: $n$ jobs have to be processed (in the same order) on $m$ machines; the aim is to find a job permutation $\{J_1, J_2, ..., J_n\}$ so as to minimise $C_{max}$, the makespan. Figure 6.3 illustrates this by showing a **Gantt chart** for a $n = 5, m = 4$ flowshop.

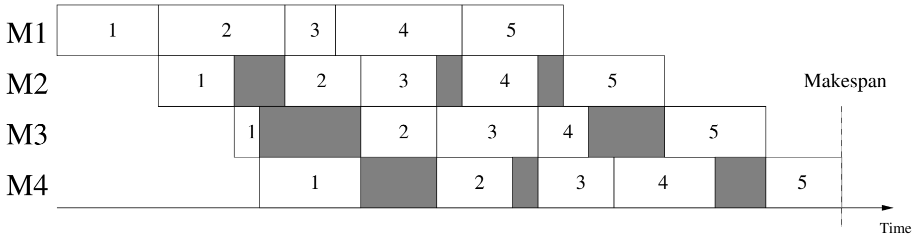


Figure 6.3: An Example Gantt Chart for a 5 × 4 Flowshop

Mathematically, $C_{max}$ is defined as follows: given processing times $p(i, j)$ for job $i$ on machine $j$ and the job permutation above, we can find the completion times by the use of the following equations:

$$C(J_1, 1) = p(J_1, 1)$$
$$C(J_i, 1) = C(J_{i-1}, 1) + p(J_i, 1) \; for \; i = 2, ..., n$$
$$C(J_1, j) = C(J_1, j-1) + p(J_1, j) \; for \; j = 2, ..., m$$
$$C(J_i, j) = max\{C(J_{i-1}, j), C(J_i, j-1)\} + p(J_i, j) \; for \; i = 2, ..., n; \; j = 2, ..., m$$
$$C_{max} = C(J_n, m).$$

The above describes the formalisation of the $n/m/P/C_{max}$ problem that will be used here.

### 6.2.2 Benchmarks

Standard benchmarks exist for this problem. Taillard [Taillard 93] produced a set of test problems which, using a very lengthy Tabu Search procedure, were still inferior to their lower bounds, with problem sizes ranging from 20 jobs and 5 machines, to 500 jobs and 20 machines. There are 10 instances of each problem — all processing times were generated randomly from a $U(1, 100)$ distribution.

## 6.3 Previous Approaches

To set the scene for the later investigations, an overview will now be given of the variety of approaches previously investigated in the literature. A number of reviews are available and the reader is directed to them [Graves 81, Ku *et al.* 87, Rodammer & White 88, Dudek *et al.* 92].

### 6.3.1 Johnson's Rule and it's Extensions

The seminal work in solving the $n/m/P/C_{max}$ problem was the discovery of a procedure, now known as **Johnson's Rule**, that can provably solve a 2-machine problem to optimality [Johnson 54] in polynomial time. Although this is of limited practical usefulness as few systems of interest are this simple; it formed the basis of a large part of the later work. Approximate methods for the $m$ machine problem include work by [Campbell *et al.* 70], the **Rapid Access Extended Search** (RAES) heuristic [Dannenbring 77], and the **Modified Rapid Access** heuristic (MRA) [Karimi & Ku 88a].

### 6.3.2 Constructive Heuristics

Constructive heuristics that 'build up' a solution sequentially from the first job have been proposed. The 'classic' construction heuristic for this problem is the **Nawaz-Emscore-Ham** (NEH) heuristic [Nawaz *et al.* 83]. As this heuristic will be used in the later investigation it will be described fully there. Other constructive methods have been devised. For example, [Palmer 65] and used geometric arguments, and more informed heuristics by [Rajendran & Chaudhuri 91, Sarin & Lefoka 93, Moccellin 95] have used idle times in constructive algorithms.

Finally, construction methods for the JSSP can be adapted. One common approach is the **Giffler and Thompson** algorithm [Giffler & Thompson 60] in conjunction with dispatch rules. In fact, this is one of the approaches used later and thus it will be described there.

### 6.3.3 Neighbourhood Search Techniques

As neighbourhood search is the focus of this work, it is fitting that there has been much work applying these methods to this problem. The utility of simulated annealing has been investigated in a number of studies, for example [Osman & Potts 89, Ogbu & Smith 90]. Evolutionary approaches have also been explored. Cleveland and Smith [Cleveland & Smith 89] used an EA to sequence a computer board assembly and test facility which was modelled as a flowshop; with a wide range of additional work by, for example, [Mott 90, Reeves 95a]. A great deal of work has also been performed using tabu search. The first such work was by [Taillard 90, Adenso-Diaz 92] which showed that this approach could outperform NEH. Improvements to this approach were then investigated, mainly using candidate list strategies; for example [Reeves 93a, Reeves 95b].

One of the currently most effective implementations is by [Nowicki & Smutnicki 96] with their tabu search formulation which exploits knowledge of the properties of moves that affect the critical path in the candidate list [Szwarc 79], as well as efficient methods of makespan evaluation, and tabu search strategies such as aspiration to produce an optimiser that can quickly solve medium and large-size instances of the Taillard benchmark problems[1].

### 6.3.4 Other Approaches

In addition to the above, a number of other approaches are applicable For example **dynamic programming** has been used [Corwin & Esogbue 74]; also a number of **branch and bound** procedures have also been devised specifically for this problem, for instance the work by [Ignall & Schrage 65, Lomnicki 65, McMahn & Burton 67, Ashour 70] (also in the above context, **elimination approaches** have also been investigated [Szwarc 71]).

Other approaches to this problem have included formulating the problem as a 0-1 **Mixed Integer Linear Program** (MILP) [Karimi & Ku 88b]. This formulation can then be solved using a commercial MILP solver such as LINDO, though this was found to scael up very poorly.

Finally, KBS and **constraint-based** scheduling methods can, and have, been employed to solve job/flow-shop problems. Examples include the TOSCA system [Beck 92, Beck 93], the CSP approach by [Cheng & Smith 95], and the graph search approach by [Sen *et al.* 96].

[1] However, it should be noted here that the aim of this work is not to compete against this algorithm but rather to evaluate the proposed design heuristic. Also, the performance criterion used here would not allow easy comparison.

## 6.4 Representational Comparisons

The first of the experimental comparisons to the undertaken is to confirm the second design heuristic. Therefore a range of optimisers and neighbourhood operators, described in Chapter 5 will be compared with respect to a common performance metric (described below). The results will be examined for any interesting trends and, more importantly, to see whether the design heuristic holds in that there are no cases where two optimisers give conflicting pairwise landscape comparison results.

### 6.4.1 Evaluation Methodology

The performance and behaviour of each of the meta-heuristics will be dealt with in turn, and then finally followed with a comparison. In all cases, experiments were performed for the first of the Talliard test set's instances of the following problems[2]: 20x5, 20x10, 20x20, 50x5, 50x10, 50x20, 100x5, 100x10, 100x20, 200x10, and 200x20.

The measure of performance used was the quality of solution (where high quality equates with low makespan) obtained after a set number, $N$, of evaluations. A sample of fifty runs was taken in each case. As a very large number of runs were made, the detailed results can be found via in Appendix C of this thesis, and summaries are given in the discussions here. In addition, the methodology described in Chapter 5 was performed and their results are available in full in Appendix C. Therefore, where performance differences are reported as being significant in the main text, this refers to the results of the statistical tests.

The value of $N$ used in these experiments was solely dependent upon $n$ (the number of jobs) and was set to 5000, 6750, 8000, and 9250 evaluations respectively. The value of $N$ at $n = 20$ was set on the basis of formative experiments that measured the time taken for no further improvements to be made for SHC. The value of $N$ for the larger problems was then scaled up by a roughly $ln(n)$ relationship, justified from empirical results from [Osman & Potts 89] for the number of evaluations required for good convergence for SA.

[2] Where the notation used is of the form 'number of jobs' x 'number of machines'.

| Problem | Swap | Shift | Swap-Adjacent |
|---|---|---|---|
| 20x5 | 1295.14 (4.73) | 1292.44 (7.77) | 1356.32 (49.16) |
| 20x10 | 1623.18 (14.23) | 1605.88 (11.00) | 1804.46 (62.32) |
| 20x20 | 2355.74 (18.04) | 2337.42 (17.18) | 2574.96 (62.12) |
| 50x5 | 2737.66 (8.24) | 2734.46 (6.76) | 2809.68 (66.73) |
| 50x10 | 3141.72 (24.76) | 3121.10 (24.73) | 3393.40 (64.93) |
| 50x20 | 4049.34 (31.64) | 4021.62 (22.52) | 4460.42 (84.30) |
| 100x5 | 5510.62 (12.91) | 5506.56 (14.14) | 5748.96 (130.14) |
| 100x10 | 5927.38 (37.54) | 5884.80 (30.06) | 6463.94 (76.80) |
| 100x20 | 6622.44 (44.06) | 6581.78 (38.13) | 7194.34 (111.92) |
| 200x10 | 11029.90 (35.02) | 11020.20 (31.61) | 11615.28 (153.94) |
| 200x20 | 11826.10 (49.64) | 11797.80 (50.06) | 12667.98 (119.44) |

Table 6.1: Summary of Stochastic Hillclimbing Results (Part 1)

| Problem | Reverse | Ordinal | Random Keys |
|---|---|---|---|
| 20x5 | 1296.78 (2.91) | 1307.16 (17.55) | 1305.44 (14.92) |
| 20x10 | 1677.78 (48.02) | 1696.16 (43.21) | 1686.26 (37.25) |
| 20x20 | 2401.52 (28.89) | 2399.50 (31.67) | 2400.32 (28.89) |
| 50x5 | 2748.32 (8.70) | 2748.00 (14.90) | 2736.74 (8.96) |
| 50x10 | 3207.64 (44.47) | 3281.32 (39.09) | 3128.84 (29.30) |
| 50x20 | 4223.90 (57.63) | 4253.40 (58.17) | 4030.92 (28.43) |
| 100x5 | 5537.14 (19.72) | 5523.80 (23.14) | 5506.88 (15.66) |
| 100x10 | 6041.30 (51.91) | 6140.66 (53.84) | 5912.10 (38.88) |
| 100x20 | 6893.24 (78.43) | 6981.76 (75.53) | 6602.34 (42.29) |
| 200x10 | 11159.82 (56.48) | 11257.36 (79.35) | 11022.60 (42.13) |
| 200x20 | 12291.92 (83.84) | 12468.80 (84.39) | 11794.40 (50.80) |

Table 6.2: Summary of Stochastic Hillclimbing Results (Part 2)

### 6.4.2 Stochastic Hillclimbing Results

Experiments were first performed to examine the performance of stochastic hillclimbing. The results obtained for SHC are summarised in Tables 6.1 and 6.2. For all of the results presented here, the standard deviation is given in parentheses.

Comparing representations for stochastic hillclimbing indicates that, for all problem instances, the shift-neighbourhood generally gave the highest quality solutions in the time available (for the first 9 of the 11 instances), with the random keys approach winning out for the largest problem instance. Further examination of the performance of the other neighbourhoods indicates, for all of the problem instances considered, that the random keys and swap neighbourhoods were the next best choices — giving results only slightly worse than that obtained by the shift neighbourhood (shift came out significantly better than swap in 8 cases, and beat random keys in 5 cases). This is not especially surprising as 5.3.3 notes that the random keys neighbourhood can be thought of as being equivalent to a shift neighbourhood with the addition of null moves, which would suggest that the performance of the random keys would be slightly worse that that of a shift neighbourhood. The performance of the swap neighbourhood can be accounted for as a result of the fact, noted in Chapter 5, that the position and precedence neighbourhoods are well-correlated and therefore we would expect similar performance.

Examination of the remaining three neighbourhoods also reveals some interesting trends. The reverse neighbourhood is consistantly worse than the shift neighbourhorhood (10 out of 11 instances), especially as the problem size increases. This is a consequence of the reverse operator being relatively more disruptive of precedence-formae (the type that the shift neighbourhood manipulates) as problem size increases, as discussed in more detail in 5.8.

The ordinal neighbourhood performed worse still, being outperformed by the reverse neighbourhood over *all* instances. As noted in the analysis given in 5.10, the ordinal neighbourhood can be viewed as half of a shift neighbourhood with the other half of the shift neighbourhood being distant distant in the ordinal metric space, which would induce a very different metric space to that of the shift neighbourhood which accounts for the differences in performance. Finally, the swap-adjacent neighbourhood performs worst of all (outperformed by ordinal for 11 out of 11 instances), although the metric space induced is similar to that for the shift neighbourhood — this is largely an effect, as noted in Chapter 3, of the smaller neighbourhood increasing both the number of moves required to make a walk from one point on the landscape to another as well as the number of local optima.

### 6.4.3 Simulated Annealing Results

Experiments were also performed to examine the performance of simulated annealing for the selected neighbourhoods. The effect of the temperature schedule was also examined by varying the initial temperature in the range 1-30 temperature units. The final temperature was, as noted earlier in 5. set to 0.05 temperature units, with temperature readjustments being carried out every 100 evaluations. The results obtained are shown in Tables 6.3 and 6.4, with the best choice of initial temperature given in square brackets.

As can be seen, the shift neighbourhood gives the best overall results cases (beating swap in 9 cases, and random keys in 1 case). Further examination of the results obtained shows that the relative performance of the neighbourhoods is, as the hypothesis being tested here suggests, the same as for stochastic hillclimbing.

In addition, examination of the initial temperatures found to be most effective show that for the

| Problem | Swap | Shift | Swap-Adjacent |
|---|---|---|---|
| 20x5 | 1295.36 (5.02) [13] | 1292.22 (8.04) [20] | 1311.16 (25.05) [29] |
| 20x10 | 1618.00 (10.78) [17] | 1606.84 (11.65) [9] | 1724.82 (47.64) [26] |
| 20x20 | 2353.42 (21.10) [18] | 2338.02 (18.70) [17] | 2451.40 (43.10) [25] |
| 50x5 | 2737.56 (9.63) [10] | 2735.44 (8.40) [13] | 2784.54 (47.13) [25] |
| 50x10 | 3128.78 (20.01) [11] | 3111.38 (21.60) [11] | 3339.94 (65.41) [23] |
| 50x20 | 4033.74 (23.82) [15] | 4006.40 (23.79) [19] | 4357.66 (74.43) [30] |
| 100x5 | 5509.80 (13.96) [3] | 5501.66 (11.31) [2] | 5731.70 (97.41) [23] |
| 100x10 | 5920.78 (36.93) [6] | 5886.08 (30.38) [10] | 6320.46 (97.63) [17] |
| 100x20 | 6599.52 (39.60) [22] | 6553.24 (35.45) [10] | 7120.64 (98.63) [24] |
| 200x10 | 11028.40 (33.55) [17] | 11014.60 (31.49) [1] | 11605.20 (157.67) [15] |
| 200x20 | 11794.20 (52.22) [14] | 11735.90 (47.00) [13] | 12621.50 (120.11) [26] |

Table 6.3: Summary of Simulated Annealing Results (Part 1)

| Problem | Reverse | Ordinal | Random Keys |
|---|---|---|---|
| 20x5 | 1295.74 (4.11) [10] | 1295.86 (5.37) [22] | 1293.96 (6.97) [3] |
| 20x10 | 1641.98 (20.08) [25] | 1669.68 (27.00) [30] | 1607.98 (12.00) [18] |
| 20x20 | 2382.52 (30.98) [12] | 2407.58 (30.16) [29] | 2341.90 (20.41) [14] |
| 50x5 | 2745.14 (11.38) [24] | 2744.08 (11.32) [12] | 2735.96 (9.32) [5] |
| 50x10 | 3178.82 (30.84) [27] | 3240.94 (33.03) [29] | 3117.46 (21.78) [9] |
| 50x20 | 4153.02 (53.07) [27] | 4204.72 (46.66) [27] | 4012.84 (22.88) [16] |
| 100x5 | 5529.00 (19.85) [11] | 5515.44 (14.72) [15] | 5504.32 (13.87) [5] |
| 100x10 | 5908.42 (34.25) [23] | 6106.40 (45.91) [28] | 5903.20 (35.56) [13] |
| 100x20 | 6819.48 (72.95) [24] | 6913.62 (59.77) [25] | 6562.80 (36.48) [14] |
| 200x10 | 11144.16 (50.12) [8] | 11178.18 (66.17) [22] | 11018.34 (38.28) [4] |
| 200x20 | 12217.48 (79.62) [30] | 12392.20 (87.09) [27] | 11753.20 (53.52) [13] |

Table 6.4: Summary of Simulated Annealing Results (Part 2)

more effective landscapes (ie. shift, swap, and random keys), the best initial temperatures were generally lower than for the other landscapes. This is a direct result of the higher tractability of these landscapes, which implies that is likely that fewer local optima occur in them — in turn this then leads to less of a need of a mechanism (ie. temperature) to escape local optima.

### 6.4.4 Threshold Accepting Results

The effect upon TA performance of the initial threshold was examined by varying it in the range 0-10 fitness units with, as noted in Chapter 5, the threshold being linearly decreased every 100 evaluations to 0, as the search progresses. The results obtained in each case are given in Tables 6.5 and 6.6, with the best choice of threshold found again given in square brackets.

As for SHC and SA, the results for the shift neighbourhood were consistently superior (e.g. shift was significantly better than swap in 5 cases), and though no differences were significant for random keys, the makespans reported were slightly higher. In addition, the trends reported for simulated annealing are also apparent here, with respect to relative landscape performance,

| Problem | Swap | Shift | Swap-Adjacent |
|---|---|---|---|
| 20x5 | 1295.32 (5.12) [14] | 1292.02 (8.07) [2] | 1307.86 (23.57) [13] |
| 20x10 | 1614.48 (13.48) [8] | 1604.88 (12.78) [12] | 1722.02 (55.11) [15] |
| 20x20 | 2346.46 (20.02) [13] | 2335.08 (22.51) [7] | 2469.84 (55.93) [15] |
| 50x5 | 2738.74 (9.29) [7] | 2734.04 (7.83) [2] | 2778.98 (40.07) [12] |
| 50x10 | 3128.22 (28.89) [9] | 3113.44 (19.07) [8] | 3307.64 (59.79) [15] |
| 50x20 | 4029.00 (22.16) [8] | 4004.72 (21.50) [11] | 4319.46 (77.17) [15] |
| 100x5 | 5508.02 (14.10) [3] | 5505.04 (14.24) [2] | 5727.02 (117.39) [8] |
| 100x10 | 5918.30 (39.55) [7] | 5881.68 (33.06) [7] | 6305.30 (94.37) [10] |
| 100x20 | 6589.60 (32.33) [9] | 6549.38 (32.21) [10] | 7096.64 (94.55) [15] |
| 200x10 | 11025.60 (37.82) [2] | 11017.40 (33.09) [9] | 11595.74 (157.42) [6] |
| 200x20 | 11782.00 (47.41) [7] | 11711.10 (43.71) [11] | 12619.20 (108.61) [8] |

Table 6.5: Summary of Threshold Accepting Results (Part 1)

| Problem | Reverse | Ordinal | Random Keys |
|---|---|---|---|
| 20x5 | 1295.92 (3.67) [15] | 1296.48 (2.39) [10] | 1293.42 (7.11) [3] |
| 20x10 | 1637.34 (16.68) [14] | 1672.16 (22.35) [12] | 1607.06 (10.96) [6] |
| 20x20 | 2377.46 (29.37) [14] | 2405.64 (29.30) [15] | 2338.44 (21.53) [12] |
| 50x5 | 2746.10 (11.62) [6] | 2742.44 (9.05) [11] | 2735.86 (9.56) [1] |
| 50x10 | 3176.78 (21.78) [14] | 3253.90 (33.90) [12] | 3121.12 (22.12) [10] |
| 50x20 | 4140.98 (57.51) [15] | 4198.56 (43.47) [13] | 4015.78 (22.96) [12] |
| 100x5 | 5532.12 (17.46) [9] | 5518.52 (18.82) [11] | 5504.20 (14.56) [1] |
| 100x10 | 6009.42 (44.68) [7] | 6111.08 (62.12) [14] | 5899.92 (35.54) [9] |
| 100x20 | 6818.32 (56.34) [15] | 6919.44 (55.12) [14] | 6559.60 (29.57) [10] |
| 200x10 | 11148.12 (56.71) [14] | 11187.16 (60.79) [15] | 11023.04 (38.83) [3] |
| 200x20 | 12228.38 (86.45) [15] | 12400.82 (87.65) [15] | 11730.54 (50.33) [11] |

Table 6.6: Summary of Threshold Accepting Results (Part 2)

most effective initial threshold, and an increase in performance compared to stochastic hillclimbing.

### 6.4.5 Results for Record-To-Record Travel

Experiments were performed to examine the performance of RTRT for all choices of neighbourhood. The effect of the threshold was also examined by varying it in the range 0-10 fitness units. The results obtained in each case are given in Tables 6.7 and 6.8, with the choice of threshold adopted given in square brackets.

As for SHC, SA, and TA, the results for the shift neighbourhood were consistently superior (shift outperformed swap in 6 cases), though random keys makespans were higher, the differences were not significant. The trends reported for stochastic hillclimbing, simulated annealing, and threshold accepting are also reflected in the results presented here, with respect to all of the following: relative landscape performance, most effective initial threshold. Finally, as expected, there was an increase in performance compared to stochastic hillclimbing.

| Problem | Swap | Shift | Swap-Adjacent |
|---|---|---|---|
| 20x5 | 1295.54 (4.85) [2] | 1292.52 (7.78) [1] | 1324.46 (37.76) [10] |
| 20x10 | 1618.74 (13.35) [7] | 1604.98 (12.60) [8] | 1767.16 (56.32) [10] |
| 20x20 | 2348.56 (25.31) [9] | 2338.64 (19.20) [9] | 2524.08 (64.18) [10] |
| 50x5 | 2738.32 (9.14) [9] | 2734.04 (7.07) [2] | 2790.76 (54.36) [5] |
| 50x10 | 3133.30 (20.59) [6] | 3118.22 (20.54) [3] | 3346.04 (60.72) [9] |
| 50x20 | 4041.76 (19.60) [8] | 4012.88 (18.88) [9] | 4383.76 (80.04) [10] |
| 100x5 | 5510.64 (13.40) [6] | 5503.90 (12.82) [1] | 5732.46 (119.95) [7] |
| 100x10 | 5917.56 (31.68) [5] | 5885.38 (25.57) [3] | 6327.90 (97.30) [9] |
| 100x20 | 6604.36 (34.85) [5] | 6564.28 (25.58) [4] | 7142.92 (110.37) [9] |
| 200x10 | 11030.30 (37.99) [8] | 11015.10 (39.19) [6] | 11605.70 (149.02) [3] |
| 200x20 | 11811.80 (42.64) [6] | 11752.80 (55.88) [5] | 12645.98 (123.86) [3] |

Table 6.7: Summary of Record-To-Record Travel Results (Part 1)

| Problem | Reverse | Ordinal | Random Keys |
|---|---|---|---|
| 20x5 | 1296.00 (3.89) [8] | 1297.10 (3.01) [10] | 1293.88 (6.60) [6] |
| 20x10 | 1649.54 (25.52) [10] | 1679.42 (28.17) [10] | 1605.80 (10.69) [9] |
| 20x20 | 2392.88 (34.61) [10] | 2421.22 (36.01) [6] | 2344.08 (25.72) [8] |
| 50x5 | 2745.46 (8.33) [2] | 2744.06 (9.42) [6] | 2735.16 (8.46) [1] |
| 50x10 | 3186.10 (30.04) [7] | 3266.00 (36.14) [10] | 3122.36 (23.58) [4] |
| 50x20 | 4192.98 (56.87) [10] | 4215.10 (51.60) [10] | 4022.00 (22.53) [8] |
| 100x5 | 5531.50 (19.95) [8] | 5517.34 (17.32) [1] | 5504.00 (11.48) [1] |
| 100x10 | 6023.04 (46.21) [3] | 6119.74 (51.81) [9] | 5906.12 (40.56) [3] |
| 100x20 | 6858.16 (78.81) [9] | 6937.84 (57.04) [7] | 6575.92 (37.75) [2] |
| 200x10 | 11147.92 (55.46) [6] | 11208.50 (78.90) [5] | 11020.52 (35.49) [2] |
| 200x20 | 12269.26 (94.32) [9] | 12418.40 (75.31) [9] | 11764.76 (52.86) [7] |

Table 6.8: Summary of Record-To-Record Travel Results (Part 2)

### 6.4.6 First and Steepest-Ascent Hillclimbing Results

Attention will now be turned to the relative landscape performance for optimisers based on the first and steepest-ascent hillclimbing classes. This will help clarify the reasons why the second design heuristic makes a distinction between hillclimbing classes. To this end any observed differences between the relative landscape performances found here and those for SHC-based optimisers will be highlighted and discussed.

The experiments performed on these optimisers are identical to those performed on the SHC-based optimisers considered previously. Again, the results here are presented on an optimiser-by-optimiser basis. For an alternative presentation of these results, the reader is again directed to the statistical analysis given in Appendix C.

#### First-Ascent Hillclimbing

The results of the experiments for FAHC are given in Tables 6.9 and 6.10. These results show a different pattern regarding relative landscape performance than stochastic hillclimbing. The random keys neighbourhood was not only the better overall performer of those presented here

| Problem | Swap | Shift | Swap-Adjacent |
|---|---|---|---|
| 20x5 | 1296.44 (3.96) | 1296.32 (3.36) | 1379.00 (46.93) |
| 20x10 | 1627.82 (13.03) | 1616.58 (10.70) | 1824.30 (59.59) |
| 20x20 | 2363.00 (15.42) | 2343.70 (22.67) | 2583.24 (59.63) |
| 50x5 | 2745.70 (11.71) | 2740.64 (10.06) | 2943.90 (92.88) |
| 50x10 | 3352.96 (40.90) | 3163.36 (25.51) | 3502.72 (90.51) |
| 50x20 | 4065.66 (29.83) | 4068.56 (34.94) | 4507.84 (93.14) |
| 100x5 | 5530.88 (21.82) | 5539.72 (32.44) | 5914.18 (128.35) |
| 100x10 | 6038.72 (61.47) | 6026.16 (47.28) | 6579.16 (80.03) |
| 100x20 | 6735.62 (56.59) | 6806.38 (60.89) | 7351.18 (124.78) |
| 200x10 | 11164.10 (62.66) | 11295.90 (77.61) | 11956.22 (183.43) |
| 200x20 | 12120.30 (69.41) | 12326.00 (89.20) | 13021.76 (159.90) |

Table 6.9: Summary of First-Ascent Hillclimbing Results (Part 1)

| Problem | Reverse | Ordinal | Random Keys |
|---|---|---|---|
| 20x5 | 1301.68 (12.44) | 1310.88 (20.77) | 1307.44 (16.47) |
| 20x10 | 1707.38 (39.70) | 1700.92 (36.22) | 1711.82 (32.59) |
| 20x20 | 2449.68 (37.64) | 2452.54 (34.72) | 2443.88 (36.62) |
| 50x5 | 2761.46 (18.10) | 2755.00 (16.09) | 2748.30 (12.64) |
| 50x10 | 3256.62 (55.38) | 3296.30 (43.13) | 3141.26 (30.54) |
| 50x20 | 4259.64 (60.49) | 4268.16 (59.82) | 4057.72 (34.03) |
| 100x5 | 5533.45 (24.65) | 5534.70 (22.63) | 5521.84 (22.68) |
| 100x10 | 6124.73 (46.02) | 6173.98 (53.66) | 5961.06 (50.69) |
| 100x20 | 6985.27 (54.63) | 6977.50 (71.49) | 6641.38 (46.24) |
| 200x10 | 11343.37 (62.84) | 11276.98 (87.45) | 11068.82 (54.89) |
| 200x20 | 12514.82 (61.26) | 12482.20 (98.64) | 11875.26 (61.58) |

Table 6.10: Summary of First-Ascent Hillclimbing Results (Part 2)

on the larger problems (it came out significantly better than shift on the 6 largest problem instances), being outperformed by shift neighbourhoods only on the 3 smallest problems. This is a result of the smaller neighbourhood size that was defined for the random keys neighbourhood and that the neighbourhood size increases at $O(n)$ rather than at $O(n^2)$ which is the order of increase for the swap and shift neighbourhoods. Though of course a candidate list strategy could also be used to remove this difference — more on this in Section 6.8 later — this represents an example of the caveat to the section design heuristic taking effect (see Chapter 4).

As was the case with SHC, the reverse neighbourhood performs consistently worse than the swap or shift neighbourhoods especially for the larger problems (significantly worse than shift for all but the smallest problem instance). The ordinal neighbourhood now appears to exhibit a different behaviour: giving comparable (ie. no significant difference found) performance to reverse for the smaller problem instances, but significantly better performance for the two largest problem instances. This is an effect of the larger and more rapidly increasing neighbourhood size of the reverse operator. The swap-adjacent operator again performed worst of all presumably due to a combination of increased path length and number of local optima.

Finally it should be noted that for all of the neighbourhoods considered, the performance of first-ascent hillclimbing was consistently worse that for stochastic hillclimbing (taking the shift neighbourhood as an example, FAHC was significantly worse than SHC for all but the 3 smallest problem instances), and therefore any relative improvements in performance between neighbourhoods occurring from going for stochastic to first-ascent hillclimbing is, in a practical sense, somewhat academic.

**Steepest-Ascent Hillclimbing**

Moving onto the results obtained for steepest ascent hillclimbing (Tables 6.11 and 6.12), it becomes apparent that the trends observed on moving from stochastic to first-ascent hillclimbing due to the differences in neighbourhood size have been further reinforced. This is best illustrated by the random keys neighbourhood which now significantly outperforms the shift and swap neighbourhoods for all but the 2 smallest problem instances due to its smaller neighbourhood size. Also, the ordinal neighbourhood now consistently outperforms (i.e significant for all but the 20x5 instance) the reverse neighbourhood — a complete reversal of the ordering obtained for stochastic hillclimbing — illustrating the dramatic effects that neighbourhood size can induce when changing hillclimbing class from stochastic to steepest ascent hillclimbing. The swap-adjacent neighbourhood again performs worst of all, being significantly worse than the ordinal neighborhood in all cases.

Another trend that appeared in FAHC but which is more pronounced for SAHC is the preference towards a swap neighbourhood over a shift neighbourhood for the larger problem instances. A possible reason for this may lie in the different size of the neighbourhoods. For a swap neighbourhood, the moves $move(i, j)$ and $move(j, i)$ are equivalent; however for a shift neighbourhood this is not the case. Therefore swap neighbourhoods are roughly half the size of shift neighbourhoods — which can make a difference when the whole neighbourhood (or a large part of it) has to be searched before a move is accepted, even if the shift operator is otherwise better suited to the problem.

As a final note, comparing the performance of the three hillclimbers representation by representation shows a progression of reduced performance upon going from stochastic to first-ascent and then to steepest ascent hillclimbing. This can be accounted for by considering the additional evaluations made by first and steepest-ascent hillclimbers before they accept a move to be a process of finding the steepest way to climb. Irrespective of whether this would lead to

| Problem | Swap | Shift | Swap-Adjacent |
|---|---|---|---|
| 20x5 | 1298.28 (6.82) | 1299.20 (8.52) | 1380.56 (45.43) |
| 20x10 | 1645.10 (23.45) | 1630.40 (17.16) | 1820.10 (55.26) |
| 20x20 | 2377.32 (22.22) | 2381.42 (28.65) | 2572.38 (63.05) |
| 50x5 | 2775.08 (23.42) | 2936.68 (103.90) | 2948.02 (99.83) |
| 50x10 | 3352.96 (40.90) | 3573.30 (95.47) | 3506.84 (92.70) |
| 50x20 | 4341.72 (54.18) | 4576.70 (99.95) | 4514.60 (99.35) |
| 100x5 | 5858.14 (118.98) | 6036.56 (147.63) | 5927.16 (138.08) |
| 100x10 | 6650.20 (108.27) | 6837.74 (127.17) | 6348.80 (75.01) |
| 100x20 | 7540.60 (139.51) | 7715.66 (164.52) | 7382.98 (127.25) |
| 200x10 | 12134.90 (197.88) | 12195.60 (199.38) | 11995.26 (176.01) |
| 200x20 | 13407.00 (172.98) | 13465.90 (172.81) | 13131.32 (146.72) |

Table 6.11: Summary of Steepest-Ascent Hillclimbing Results (Part 1)

| Problem | Reverse | Ordinal | Random Keys |
|---|---|---|---|
| 20x5 | 1299.86 (6.82) | 1295.98 (6.35) | 1295.04 (5.71) |
| 20x10 | 1655.72 (13.23) | 1623.16 (13.50) | 1621.56 (19.82) |
| 20x20 | 2380.38 (26.75) | 2360.90 (28.14) | 2362.50 (28.99) |
| 50x5 | 2800.00 (34.11) | 2757.52 (18.47) | 2746.10 (11.76) |
| 50x10 | 3399.12 (42.67) | 3300.46 (44.79) | 3154.20 (25.76) |
| 50x20 | 4398.10 (63.61) | 4276.66 (57.18) | 4063.74 (32.75) |
| 100x5 | 5784.78 (64.81) | 5544.56 (23.29) | 5528.40 (24.24) |
| 100x10 | 6693.40 (60.68) | 6183.94 (61.78) | 5981.26 (48.75) |
| 100x20 | 7611.70 (92.38) | 6995.94 (68.26) | 6697.72 (40.40) |
| 200x10 | 12209.27 (73.92) | 11321.32 (87.50) | 11141.64 (57.63) |
| 200x20 | 12624.27 (83.24) | 12544.66 (109.70) | 12179.48 (76.13) |

Table 6.12: Summary of Steepest-Ascent Hillclimbing Results (Part 2)

a shorter path to the optimum, if the number of evaluations required to obtain this information is greater than the potential gains, then performance will suffer in the manner shown here.

### 6.4.7 Tabu Search Results

The hillclimbing experiments were then repeated for tabu search to see if the addition of search control made any difference to the trends noted above. As noted in Chapter 5, simple recency-based tabu search implementations were used. The effect of the length of the tabu list was also examined by varying it in the range 0-10. The best results obtained in each case are given in Tables 6.13, 6.14, 6.15, and 6.16 with the best choice of tabu tenure found given in square brackets.

Both forms of tabu search did not perform as well as their stochastic counterparts (only in the smallest problem instance was their performance not signifcantly worse). The performance, behaviour, and the trends in the relative performance of the neighbourhood operators of both forms of tabu search are similar to those obtained for the form of hillclimbing that they were based upon.

| Problem | Swap | Shift | Swap-Adjacent |
|---|---|---|---|
| 20x5 | 1297.02 (7.80) [1] | 1293.82 (7.10) [2] | 1332.80 (39.52) [10] |
| 20x10 | 1626.48 (18.33) [10] | 1613.52 (13.11) [4] | 1686.52 (43.25) [10] |
| 20x20 | 2354.56 (24.61) [5] | 2341.00 (21.71) [7] | 2426.48 (43.29) [10] |
| 50x5 | 2753.20 (15.95) [1] | 2742.22 (11.93) [5] | 2924.82 (94.70) [10] |
| 50x10 | 3160.30 (26.06) [8] | 3161.88 (25.81) [3] | 3461.72 (74.45) [9] |
| 50x20 | 4066.10 (29.44) [1] | 4060.88 (34.16) [5] | 4462.58 (96.67) [9] |
| 100x5 | 5531.66 (22.18) [1] | 5538.18 (32.35) [10] | 5912.96 (113.15) [10] |
| 100x10 | 6038.54 (61.84) [1] | 6023.28 (45.29) [3] | 6585.34 (121.10) [8] |
| 100x20 | 6726.02 (58.99) [9] | 6800.46 (58.82) [2] | 7343.48 (123.57) [8] |
| 200x10 | 11160.90 (61.96) [5] | 11295.90 (77.61) [ALL] | 12000.08 (176.63) [5] |
| 200x20 | 12120.30 (69.41) [1] | 12328.60 (88.14) [ALL] | 12967.86 (203.22) [7] |

Table 6.13: Summary of First-Ascent Tabu Search Results (Part 1)

| Problem | Reverse | Ordinal | Random Keys |
|---|---|---|---|
| 20x5 | 1299.34 (11.71) [9] | 1296.48 (2.12) [8] | 1293.50 (8.82) [1] |
| 20x10 | 1672.48 (37.28) [5] | 1668.72 (23.62) [7] | 1612.20 (9.86) [4] |
| 20x20 | 2402.94 (34.80) [3] | 2394.40 (25.19) [8] | 2340.26 (21.85) [2] |
| 50x5 | 2760.38 (17.54) [6] | 2752.28 (17.01) [9] | 2744.34 (10.59) [10] |
| 50x10 | 3256.40 (57.24) [3] | 3276.60 (35.45) [8] | 3133.92 (32.88) [10] |
| 50x20 | 4256.24 (54.50) [5] | 4238.78 (51.41) [10] | 4036.76 (34.75) [4] |
| 100x5 | 5532.35 (37.59) [4] | 5529.18 (17.97) [6] | 5520.34 (19.65) [6] |
| 100x10 | 6123.37 (42.25) [3] | 6152.50 (53.35) [1] | 5935.22 (44.60) [3] |
| 100x20 | 6983.17 (53.65) [9] | 6974.02 (68.45) [10] | 6618.86 (50.44) [9] |
| 200x10 | 11341.10 (73.55) [6] | 11253.18 (84.24) [3] | 11060.58 (53.26) [4] |
| 200x20 | 12512.35 (76.12) [3] | 12482.96 (82.57) [8] | 11864.96 (57.33) [5] |

Table 6.14: Summary of First-Ascent Tabu Search Results (Part 2)

| Problem | Swap | Shift | Swap-Adjacent |
|---|---|---|---|
| 20x5 | 1300.22 (11.81) [1] | 1298.64 (7.07) [9] | 1331.92 (42.99) [9] |
| 20x10 | 1639.68 (21.62) [10] | 1626.52 (15.79) [6] | 1696.02 (46.46) [10] |
| 20x20 | 2370.96 (26.69) [7] | 2377.56 (30.04) [4] | 2430.06 (45.13) [9] |
| 50x5 | 2775.08 (23.42) [1] | 2936.68 (103.90) [1] | 2929.64 (96.29) [8] |
| 50x10 | 3352.34 (40.36) [10] | 3573.30 (95.47) [1] | 3469.80 (86.31) [10] |
| 50x20 | 4341.12 (53.98) [9] | 4576.64 (99.65) [10] | 4471.80 (97.36) [10] |
| 100x5 | 5858.14 (118.98) [1] | 6036.56 (147.63) [1] | 5930.92 (116.69) [9] |
| 100x10 | 6650.20 (108.27) [1] | 6837.74 (127.17) [1] | 6601.66 (131.19) [9] |
| 100x20 | 7540.60 (139.51) [ALL] | 7715.66 (164.52) [ALL] | 7379.36 (132.18) [10] |
| 200x10 | 12134.90 (197.88) [ALL] | 12195.60 (199.38) [ALL] | 12044.54 (177.53) [10] |
| 200x20 | 13407.00 (172.98) [ALL] | 13465.90 (172.81) [ALL] | 13069.26 (195.87) [7] |

Table 6.15: Summary of Steepest-Ascent Tabu Search Results (Part 1)

| Problem | Reverse | Ordinal | Random Keys |
|---|---|---|---|
| 20x5 | 1304.26 (13.93) [10] | 1296.80 (1.40) [10] | 1293.94 (6.34) [7] |
| 20x10 | 1677.26 (33.11) [7] | 1667.76 (22.14) [6] | 1610.36 (11.65) [1] |
| 20x20 | 2406.80 (29.83) [8] | 2396.92 (25.26) [9] | 2346.36 (24.95) [2] |
| 50x5 | 2799.04 (34.12) [5] | 2751.86 (14.47) [10] | 2743.22 (10.16) [5] |
| 50x10 | 3398.76 (41.99) [6] | 3277.84 (38.23) [10] | 3148.78 (33.76) [9] |
| 50x20 | 4398.10 (63.61) [1] | 4252.72 (56.69) [10] | 4043.56 (22.15) [7] |
| 100x5 | 5784.78 (64.81) [1] | 5535.26 (16.15) [8] | 5522.00 (20.83) [10] |
| 100x10 | 6491.62 (72.58) [3] | 6162.96 (60.28) [10] | 5973.78 (52.86) [7] |
| 100x20 | 7611.70 (92.38) [ALL] | 6986.96 (64.58) [9] | 6679.64 (44.95) [9] |
| 200x10 | 11209.27 (73.92) [ALL] | 11303.54 (72.45) [9] | 11132.50 (53.46) [8] |
| 200x20 | 12624.27 (83.24) [ALL] | 12521.78 (81.91) [10] | 12155.70 (75.23) [8] |

Table 6.16: Summary of Steepest-Ascent Tabu Search Results (Part 2)

In addition, there were no trends observed regarding the effect of the neighbourhood on the length of tabu tenure, with the exception of the insensitivity of steepest-ascent tabu search with a reverse neighbourhood when $n$ is large. This is due to the fact that the large number of evaluations required to explore an entire neighbourhood (just under 20000 when $n = 200$) is either more that there are available, or if not high enough to ensure that a relatively small area of the search space will be explored — this was also observed with the shift and swap neighbourhoods. It would appear that, with this exception aside, that it is not so straightforward to qualitatively relate the relative tractability/effectiveness of the landscapes to tabu tenure as it was for the other optimisers considered here.

### 6.4.8 Summary and Additional Remarks

The main aim of the experiments in this section of the case study was to evaluate the second design heuristic proposed in Chapter 3. Its main prediction was that hillclimbing experiments could be used to compare the suitability of various candidate neighbourhoods and the results transferred to other neighbourhood search optimisers based upon that hillclimber.

From the results obtained here, the arguments concerning the relative effectiveness of the landscapes appear to be vindicated — the relative performance of the landscapes for each of the neighbourhood search techniques, reflected that obtained in the experiments on the hillclimber upon which they were based. In addition, it was found that it was often possible to qualitatively relate some of the tuning parameters of the stochastic-hillclimbing based optimisers to the tractability/effectiveness of the landscape.

Furthermore trends in relative landscape performance could also be qualitatively explained by the differences between their metric spaces and their neighbourhood sizes. From this, the poor performance of the swap-adjacent neighbourhood in the context of the good performance of the precedence aware shift neighbourhood as attributed as an example of the caveat to the second design heuristic described in Chapter 4 arising in practice. That said, further experiments involving the comparsion of operators with different basis formae over a range of neighbourhood sizes would be required to confirm this.

The above does leave us with the question of why should a shift-neighbourhood create a tractable landscape? First of all, the sequencing taxonomy due to [Mattfeld *et al.* 96] states that precedences are the relevant feature for ‘scheduling’ problems (the FSSP falls into this category). This is presumably because the rationale behind this is that the graph representation of the JSSP (of which the FSSP is a special case) involves selecting a set of directed arcs (ie. precedences) that minimises the makespan.

An intuitive explanation goes as follows. If example Gantt charts are examined, it soon becomes apparent that there are blocks of jobs that are ‘well-meshed’ together (ie. have little are no idle time between then). It would appear important that these blocks should be disrupted as little as possible by the neighbourhood operator, or better still, manipulated explicitly. This is precisely what the shift-operator does. In fact, an examination of example Gantt charts shows that the way the jobs in a block mesh together is largely preserved.

In conclusion, experiments using hillclimbing can be usefully used to predict relative operator performance in other neighbourhood search optimisers based on the same hillclimbing class.

## 6.5 Transfer To Evolutionary Algorithms

Now that the second design heuristic has been shown to hold for this case study, attention can now turn towards seeing whether these results can be transferred to evolutionary algorithms. In Chapter 3 earlier, these points are addressed by the third and fourth design heuristics which state that the relative performance of landscapes is unchanged upon moving from SHC-based optimisers to an EA, and that the recombination operator should work in the same metric space as the most effective operator for stochastic hillclimbing.

The results of experiments identical to those carried out to evaluate the second design heuristic will now be detailed and discussed to evaluate the above. As before, full results and statistical analysis are available in Appendix C. The form of the EA used in these experiments was that described in Chapter 5 earlier, where the crossover probability was explored in increments of 0.1.

### 6.5.1 Results for Permutation Operators

The results presented here are for the permutation operators: PPX (precedence), Enhanced Edge Crossover (edges), Modified PMX (positions) and the Position RRR operator, along

| Problem | Shift | Swap | Reverse | Swap-Adjacent |
|---|---|---|---|---|
| 20x5 | 1261.36 (91.18) [0.2] | 1264.46 (91.35) [0.6] | 1275.46 (87.96) [0.5] | 1278.24 (86.28) [0.6] |
| 20x10 | 1577.88 (77.45) [0.2] | 1585.98 (82.64) [0.4] | 1613.74 (85.01) [0.6] | 1616.00 (81.21) [0.8] |
| 20x20 | 2267.94 (70.30) [0.2] | 2276.54 (73.91) [0.3] | 2315.12 (77.92) [0.7] | 2320.08 (73.67) [0.5] |
| 50x5 | 2804.44 (129.12) [0.1] | 2809.12 (129.57) [0.1] | 2832.54 (129.04) [0.4] | 2839.40 (129.21) [0.7] |
| 50x10 | 3155.68 (100.90) [0.2] | 3166.76 (103.22) [0.2] | 3236.44 (107.14) [0.4] | 3244.22 (102.48) [0.9] |
| 50x20 | 4020.76 (92.10) [0.5] | 4028.76 (79.94) [0.5] | 4129.10 (90.97) [0.7] | 4129.98 (95.00) [0.9] |
| 100x5 | 5390.86 (190.86) [0.3] | 5397.36 (190.49) [0.2] | 5444.92 (189.72) [0.7] | 5443.30 (185.61) [0.9] |
| 100x10 | 5843.34 (159.87) [0.5] | 5840.40 (144.06) [0.1] | 5964.42 (145.66) [0.7] | 5995.56 (140.30) [0.9] |
| 100x20 | 6824.06 (97.10) [0.4] | 6834.10 (94.11) [0.6] | 7028.06 (93.14) [0.7] | 7020.18 (98.96) [0.8] |
| 200x10 | 11067.86 (207.81) [0.1] | 11068.22 (204.37) [0.3] | 11253.66 (216.28) [0.5] | 11299.62 (205.52) [0.9] |
| 200x20 | 12263.32 (111.37) [0.4] | 12268.86 (120.53) [0.5] | 12574.20 (133.16) [0.6] | 12630.12 (140.39) [0.9] |

Table 6.17: Summary of Results For PPX (Precedence) Crossover

| Problem | Shift | Swap | Reverse | Swap-Adjacent |
|---|---|---|---|---|
| 20x5 | 1261.98 (92.20) [0.2] | 1267.58 (88.14) [0.0] | 1281.30 (90.53) [0.2] | 1277.10 (86.63) [0.8] |
| 20x10 | 1582.68 (79.05) [0.0] | 1596.76 (80.37) [0.1] | 1630.12 (80.10) [0.0] | 1619.00 (87.49) [0.5] |
| 20x20 | 2275.64 (71.26) [0.0] | 2294.12 (73.20) [0.0] | 2340.90 (70.16) [0.3] | 2321.60 (75.92) [0.5] |
| 50x5 | 2806.80 (130.46) [0.0] | 2811.86 (128.40) [0.2] | 2842.48 (127.31) [0.1] | 2850.14 (120.14) [0.8] |
| 50x10 | 3168.40 (102.21) [0.0] | 3183.62 (104.87) [0.0] | 3270.22 (99.12) [0.0] | 3306.46 (99.88) [0.6] |
| 50x20 | 4036.16 (86.16) [0.0] | 4058.32 (91.71) [0.0] | 4189.00 (92.23) [0.1] | 4212.82 (82.57) [0.3] |
| 100x5 | 5392.28 (189.96) [0.0] | 5397.94 (186.80) [0.2] | 5459.74 (184.13) [0.0] | 5480.46 (170.33) [0.9] |
| 100x10 | 5847.76 (137.19) [0.0] | 5861.74 (141.75) [0.0] | 6017.28 (154.00) [0.0] | 6095.54 (130.78) [1.0] |
| 100x20 | 6850.26 (92.04) [0.0] | 6886.46 (101.33) [0.1] | 7105.80 (104.00) [0.0] | 7169.64 (105.85) [0.2] |
| 200x10 | 11074.98 (211.15) [0.0] | 11091.38 (207.07) [0.0] | 11310.42 (184.14) [0.0] | 11485.58 (165.42) [0.4] |
| 200x20 | 12305.66 (123.45) [0.0] | 12309.84 (127.40) [0.0] | 12694.28 (126.02) [0.1] | 12861.08 (150.54) [0.5] |

Table 6.18: Summary of Results For Edge Crossover

with the permutation mutation (unary neighbourhood) operators selected in Chapter 5 and used earlier. The mean makespan obtained is shown in Tables 6.17, 6.18, 6.19 and 6.20. The standard deviation is given in brackets, with the crossover probability that was found to be most effective in square brackets.

Some care had to be taken when examining the differences between the means as the size of the standard deviations lead to a sizeable proportion of the pairwise comparisons to not be statistically significant. If in doubt, the reader is therefore directed to the statistical analysis in Appendix C.

### Trends in the Mutation Operator

Examining the effect of the mutation operator revealed some rather interesting trends. The experiments with stochastic hillclimbing, which were detailed in Section 6.4.2 earlier, showed that the shift, swap and random keys neighbourhoods performed best of all, followed by the reverse, ordinal and swap-adjacent neighbourhoods — which can be related to how far the metric space of these operators is different to that for a shift operator, or in the case of the swap-adjacent neighbourhood to the number of moves required to make a walk from one point on the landscape to another. To simplify the discussion, when concrete significance results between mutation operators are being reported, it is when PPX crossover is being used (unless it is made clear otherwise).

These previous results tie in well with the results obtained here: shift mutation gave lower average makespans than swap mutation (with the earlier caveat about standard deviations — none of the differences were statistically significant); and this was followed by further increases in makespan upon moving on to the reverse and then swap-adjacent neighbourhoods, though it should be noted that a greater (and significant) difference in makespan between these two operators was observed in the hillclimbing experiments that was not reflected in the results shown here (again the high standard deviations meant the no significance can be reported). This was found to be because, when these mutation operators where used, the best performance was at high crossover probabilities where the mutation operator was not used often enough for any difference to be observed. However, examination of the plots of makespan against crossover (PPX) probability (Figure 6.4) show, for low crossover probabilities, that the reverse mutation (significantly) outperforms swap-adjacent mutation.

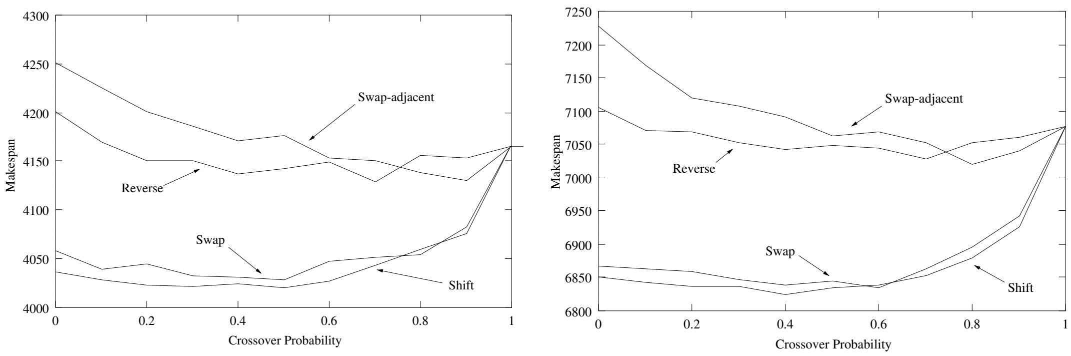


Figure 6.4: Plots of the Effect of the Mutation Operator (50x20 and 100x20 Problems)

Therefore, it is clear that the third design heuristic holds for this case study, and that the results of hillclimbing experiments can be used to inform the choice of the EA mutation operator.

### Trends in the Crossover/Recombination Operator

Attention can now turn to the crossover operators where it was immediately apparent that the relative performance of the neighbourhood operators in the hillclimbing experiments exhibits itself in the crossover operators. The precedence-aware PPX operator generally gave the lowest

| Problem | Shift | Swap | Reverse | Swap-Adjacent |
|---|---|---|---|---|
| 20x5 | 1262.48 (91.35) [0.4] | 1266.18 (90.59) [0.1] | 1274.58 (87.84) [0.6] | 1277.76 (87.21) [0.8] |
| 20x10 | 1576.00 (78.94) [0.3] | 1590.32 (80.20)[0.3] | 1618.08 (78.30) [0.5] | 1616.72 (82.81) [0.7] |
| 20x20 | 2284.18 (66.64) [0.3] | 2271.08 (69.95) [0.3] | 2319.34 (76.72) [0.7] | 2317.82 (78.50) [0.9] |
| 50x5 | 2804.54 (129.20) [0.2] | 2814.04 (129.00) [0.3] | 2831.64 (123.30) [0.5] | 2833.90 (129.97) [0.9] |
| 50x10 | 3163.40 (102.61) [0.1] | 3174.40 (101.20) [0.3] | 3229.44 (109.60) [0.9] | 3227.82 (103.46) [0.9] |
| 50x20 | 4023.10 (86.92) [0.2] | 4032.48 (87.06) [0.4] | 4113.50 (86.52) [0.5] | 4106.76 (99.43) [0.9] |
| 100x5 | 5388.80 (188.05) [0.2] | 5398.28 (181.66) [0.2] | 5442.08 (188.36) [0.7] | 5449.82 (192.22) [0.5] |
| 100x10 | 5831.74 (144.03) [0.3] | 5844.22 (148.34) [0.3] | 5951.90 (149.39) [0.9] | 5943.98 (162.44) [0.9] |
| 100x20 | 6828.68 (101.67) [0.3] | 6828.34 (81.23) [0.2] | 6993.18 (104.00) [0.8] | 6956.42 (106.51) [0.9] |
| 200x10 | 11066.80 (207.47) [0.5] | 11070.30 (213.08) [0.5] | 11220.50 (213.07) [0.8] | 11226.54 (196.69) [0.9] |
| 200x20 | 12258.50 (120.57) [0.5] | 12251.10 (128.84) [0.3] | 12517.88 (161.15) [0.8] | 12512.82 (141.11) [0.9] |

Table 6.19: Summary of Results For Modified PMX (Position) Crossover

| Problem | Shift | Swap | Reverse | Swap-Adjacent |
|---|---|---|---|---|
| 20x5 | 1262.66 (91.95) [0.0] | 1266.94 (86.85) [0.0] | 1272.58 (88.20) [0.1] | 1281.10 (83.28) [0.7] |
| 20x10 | 1583.36 (80.23) [0.0] | 1595.16 (78.53) [0.1] | 1605.10 (79.57) [0.0] | 1619.20 (81.13) [0.5] |
| 20x20 | 2276.06 (70.98) [0.0] | 2294.14 (69.61) [0.0] | 2317.54 (75.08) [0.0] | 2321.54 (72.60) [0.4] |
| 50x5 | 2807.16 (130.49) [0.1] | 2814.32 (128.08) [0.0] | 2830.82 (125.26) [0.1] | 2842.72 (131.07) [0.6] |
| 50x10 | 3168.92 (101.31) [0.0] | 3189.98 (102.35) [0.1] | 3231.60 (100.95) [0.0] | 3249.72 (105.63) [0.5] |
| 50x20 | 4040.70 (88.20) [0.0] | 4059.12 (86.83) [0.0] | 4153.98 (81.63) [0.0] | 4142.22 (83.43) [0.6] |
| 100x5 | 5391.22 (186.17) [0.1] | 5398.90 (182.91) [0.1] | 5432.18 (185.06) [0.0] | 5437.36 (184.24) [0.8] |
| 100x10 | 5846.82 (148.10) [0.0] | 5862.94 (152.12) [0.0] | 5981.32 (138.56) [0.1] | 5977.46 (152.18) [0.4] |
| 100x20 | 6845.26 (88.90) [0.0] | 6870.24 (94.12) [0.0] | 7049.06 (101.12) [0.0] | 7025.00 (104.80) [0.6] |
| 200x10 | 11083.62 (203.57) [0.0] | 11084.66 (206.23) [0.0] | 11251.96 (189.34) [0.0] | 11221.32 (195.06) [0.7] |
| 200x20 | 12310.78 (118.89) [0.0] | 12299.54 (110.85) [0.0] | 12595.98 (116.99) [0.0] | 12544.50 (142.51) [0.5] |

Table 6.20: Summary of Results For RRR (Position) Crossover

average makespan followed closely by the Modified PMX crossover operator (though again the differences in performance were not found to be significant). The edge crossover operator appeared to perform relatively poorly. That said, significant differences from shift were only found when reverse/swap-adjacent mutation were used, for the 3 largest problem instances.

Why the pattern of the above results arises is related to why the swap neighbourhood does almost as well as the shift neighbourhood in the hillclimbing experiments. That is, the metric space induced by the position formae is correlated with that induced by precedence formae. Otherwise placed in forma processing terms, position based operators of high transmission also *implicitly* process (ie. assort/transmit) precedence formae to a large extent — this idea of implicit forma processing was first introduced by [Jelasity & Dombi 96]. In fact, the need for high transmission for implicit forma processing is clear when the performance of another position-based crossover operator (RRR) is considered — its performance was generally as poor as that for the Edge crossover whose induced metric space does not correlate well with for precedence formae (the pattern of significance was very similar).

In addition, the above pattern is also reflected in the crossover probabilities that were found to be most effective. For the 'effective' crossover operators (ie. PPX and Modified PMX), the crossover probabilities for the shift and swap mutation operators lay in the region of 0.1-0.6, while for the less effective reverse and swap-adjacent mutation operators crossover was used more often (in the range 0.5-0.9). This contrasts with the RRR and Edge crossover operators that, with shift, swap, and reverse mutation operators have very low crossover probabilities (0.0-0.1), whereas with swap-adjacent mutation the appropriate crossover probability rises to the range 0.4-0.8. These trends can be accounted for by realising that the evolutionary algorithm will give its best performance when the operator that induces the most tractable metric space is used most often. So it would be expected in the cases where a relatively 'good' crossover operators but a relatively 'poor' mutation operator was used, that the most appropriate crossover probability would be higher than if a good mutation operator was used.

To illustrate the above points, let us now examine some examples of how the different crossover operators perform relative to each over with respect to their crossover probabilities. To this end, plots of the mean makespan against crossover probability, whilst using shift mutation, were made for the 50x20 and 100x20 problems — these plots are shown in Figure 6.5.

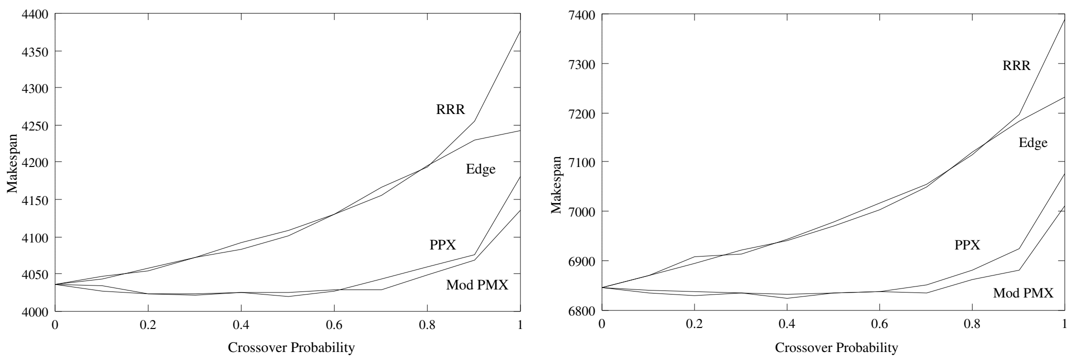


Figure 6.5: Plots of the Effect of the Crossover Operator (50x20 and 100x20 Problems)

As can be seen, the PPX and Modified PMX operators consistently gave the best performance which corresponds well to the results above. The RRR and Edge crossovers gave comparable performance at low probabilities, but this was because then the mutation operator (which was the same for all) was doing all or most of the search. These results are largely consistent with the suggestions made by the fourth design heuristic.

### 6.5.2 Results for Ordinal and Random Key Representations

The experiments above were then repeated for the ordinal and random keys representations, with the operators described in Chapter 5 — the results of these experiments are given in Table 6.21. Comparison of the results obtained with the permutation operators shows that (with the caveats described earlier) the performance of the evolutionary algorithm with the random keys representation is generally comparable with that for the PPX & Shift EA for problems of size equal or less than 100 jobs, but the results for the 100x20, 200x10, and 200x20 problems exhibit significant improvement in favour of the random keys representation.

As noted in 5.3.3, a random keys representation can be seen to be related to both the shift neighbourhood, and recombination in the position metric space. This would support the idea that the random keys and permutation precedence operators would have similar performance, with some differences due to the redundancy of the random keys representation. The hillclimbing experiments in Section 6.4.2 earlier showed that this redundancy degraded performance. However it would appear that, in this case, the possible problems with crossover in a redundant representation (ie. the competing conventions problem) do not manifest themselves — or possibly may even enhance performance by allowing the evolutionary algorithm to escape local optima or avoid premature convergence.

Support for the above may be taken from the literature on evolving neural networks where the completing conventions problem originally arose. In this context, EAs with high selection pressures which quickly force the solutions in the EA population into the same 'convention' have been found to be successful [Walker 95]. One implication of this is that, if EA performance was being limited by the fact that population diversity is too low, then the 'unproductive' crossovers may have the benefit of maintaining a suitable level of diversity. Also, the fact that this effect only becomes noticeable for large problems ties in with observations made by [McGeoch 86] that in empirical studies of algorithms it is best to use large problem instances as differences in performance are often more noticeable. A definitive test of this, though, would be to see if the above trends remain if alternative EA selection schemes/population models were used. Also, from a design point of view, if diversity was a performance bottleneck then it would be better to deal with it by some explicit mechanism rather than the implicit side-effect that the random keys representation offers.

| Problem | Ordinal | Random Keys |
|---|---|---|
| 20x5 | 1280.32 (85.55) [0.0] | 1262.18 (91.57) [0.3] |
| 20x10 | 1623.80 (85.56) [0.5] | 1582.56 (81.05) [0.6] |
| 20x20 | 2333.44 (76.90) [0.3] | 2270.96 (72.56) [0.6] |
| 50x5 | 2834.34 (121.50) [0.2] | 2811.10 (130.33) [0.3] |
| 50x10 | 3244.28 (100.13) [0.8] | 3161.24 (98.66) [0.5] |
| 50x20 | 4136.80 (88.88) [0.8] | 4001.28 (91.87) [0.9] |
| 100x5 | 5431.52 (156.64) [0.8] | 5394.98 (187.70) [0.4] |
| 100x10 | 5937.54 (136.00) [0.8] | 5822.82 (160.14) [0.9] |
| 100x20 | 7043.92 (97.06) [0.2] | 6710.92 (101.51) [1.0] |
| 200x10 | 11264.38 (179.96) [0.1] | 11037.76 (231.54) [0.7] |
| 200x20 | 12607.42 (133.75) [0.4] | 12127.96 (138.63) [1.0] |

Table 6.21: Summary of Results For Ordinal and Random Keys EAs

Examination of the results obtained for the ordinal representation showed that this representation was a poor choice for this problem — performance was consistently worse than all of the permutation operators (for the larger problems this was to a high degree of statistical significance with respect to the swap-adjacent operator). This was suggested in the analysis given in 5.10 where it was noted that the ordinal neighbourhood can be viewed as half of a shift neighbourhood with the other half of the shift neighbourhood being maximally distant. Therefore if, as appears to be the case, a shift neighbourhood is the unary operator of choice, then the metric space induced by the ordinal representation and its operators would be quite different and therefore less tractable for neighbourhood search techniques.

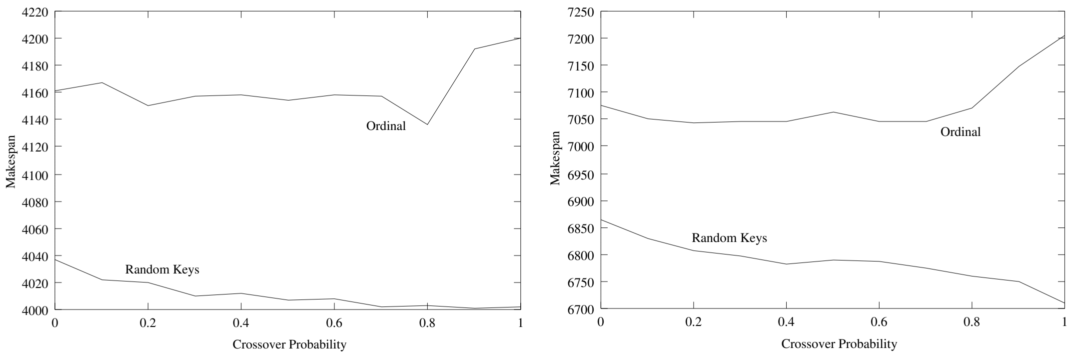


Figure 6.6: Plots of the Effect of the Crossover Operator (50x20 and 100x20 Problems)

To examine how the relative performance of these operators varies as a function of crossover operator probability, plots for the 50x20 and 100x20 problems were obtained (Figure 6.6). As can be seen, at no point does the performance of the ordinal representation approach that obtained using a random keys representation. These results were reflected in the stochastic hillclimbing experiments where random keys hillclimber performed roughly as well as a hillclimber with a shift neighbourhood, and an ordinal neighbourhood performed very poorly.

### 6.5.3 Summary and Additional Remarks

The results obtained vindicate the third design heuristic — the relative performance of the mutation operators, with the exception of the ordinal and swap-adjacent operators (which in the evolutionary algorithm gave similar performance), reflected that obtained in the hillclimbing experiments.

The validity of the fourth design heuristic was also supported by the results obtained above. The use of crossover operators that are 'precedence aware' gave the best performance. In addition, highly transmitting position-based crossover operators did well too, due to their ability to process precedences in an implicit manner, and the results of crossover operators based on the reverse and ordinal neighbourhoods were in line with the relative performance predicted by the hillclimbing experiments. The one exception (random keys) was explicable in terms of the differences in hillclimbing and evolutionary algorithm dynamics and the fact that this representation has operators that in many respects behave in a similar fashion, to the precedence aware operators (shift and PPX) This indicates that if search control ever affects the choice of neighbourhood, it would only be in cases where the neighbourhoods have very similar structures, and thus closely related hypotheses.

In conclusion, as suggested by the third and fourth design heuristics, hillclimbing experiments can be usefully used to predict relative operator performance in an evolutionary algorithm for both mutation and recombination operators.

## 6.6 Examining Heuristic Initialisation

The design heuristic concerning heuristic initialisation will now be considered and evaluated in the context of this case study. To set the scene for this study three constructive heuristics will be selected to be used as initialisation methods and described in detail.

After this a comparison will be made of the selected initialisation method's effectiveness with respect to the quality of solutions they generate. As the default initialisation method for many neighbourhood search optimisers is to start from a randomly generated solution, this method will therefore be used as a baseline with which to compare the problem-specific initialisation methods used in this study

This will set the scene for the main investigation, where the relative performance of the various initialisation methods in the context of the optimisers considered earlier. The transferability of the relative initialisation method effectiveness will be assessed in order to evaluate the validity of the heuristic initialisation design heuristic.

### 6.6.1 The SPT and LPT Priority Rules

In this study, two priority rules were used in conjunction with the Giffler and Thompson algorithm [Giffler & Thompson 60] for the makespan job shop scheduling problem (JSSP) to generate solutions for the makespan FSSP (which is a special case of the JSSP)[3]. The Giffler and Thompson algorithm is described in Algorithm 4 below.

**Algorithm 4** THE GIFFLER AND THOMPSON ALGORITHM

1: Let $t = 0$;
2: Let $Q(t) = \{1, \ldots, n \times m\}$;
   *// Where $Q(t)$ is the set of the $n \times m$ unscheduled operations at time $t$*
3: **repeat**
4: Let $c_{o^*} = min\{c_o \mid o \in Q(t)\}$;
   *// Find the earliest completion time for all $o \in Q(t)$, where $c_o$ is the earliest completion time for operation o*
5: Let $m^*$ be the machine that operation $o^*$ has to be processed on;
6: From the **conflict set** choose and schedule an operation $s \in \{o \in Q(t) \mid o$ has to be processed on machine $m^*$ and $r_o < c_o\}$; *// Where $r_o$ is the earliest start time for operation o*
7: $Q(t) = Q(t) \setminus s$;
8: Modify $c_o$ for all operations $o \in Q(t)$;
9: Set $t$ to the earliest possible machine to operation assignment;
10: **until** $Q(t) = \emptyset$;
11: Return the schedule thus produced;

The Giffler and Thompson algorithm itself only guarantees an active non-delay schedule — the quality of results obtained depends upon *how* solutions are selected from the conflict set in step 6 of the algorithm (the default is random selection). For this study, the two priority rules below were initially investigated to examine how effectively they select operations from the conflict set:

- SPT — pick the operation with the shortest processing time.

[3] Thanks to Emma Hart for running the Talliard FSSP benchmark problems through her JSSP code to provide a set of initial solutions.

- LPT — pick the operation with the longest processing time.

In the event that the priority rule being used considers more than one operation to be equally desirable, tie-breaks were resolved randomly.

For an efficient implementation, the additional constraints imposed on the FSSP with respect to the JSSP can be exploited. For both problems the optimal solution is an active non-delay schedule, however for the FSSP the optimal schedule is also a permutation schedule. Therefore the Giffler and Thompson algorithm given above was modified so that when an operation was scheduled, the other operations pertaining to the job to which the scheduled operation belongs are also scheduled — this was found to speed the algorithm up greatly [Hart & Tuson 98].

Finally, in order to allow for the time taken to construct the solution, the generation process was assumed to take the equivalent of two evaluations — examination of the length of time taken to produce solutions using the SPT and LPT priority rules showed that this was a fair approximation. This, of course, mean that the optimisers run for slightly fewer iterations - however, this does better simulate the effect of initialisation with respect to wall-clock time.

### 6.6.2 The Variant NEH Heuristic

the two priority rules described above were then compared with a heuristic specific to the FSSP. The construction heuristic used was a variant of the NEH heuristic [Nawaz *et al.* 83]. In a comparative study of a number of heuristics for the FSSP, the NEH heuristic was found to give the best results with respect to quality of solution obtained [Turner & Booth 87]. This heuristic reformulates the problem to a 2-machine problem (like RAES for example), but instead assumes that a job with a high total processing time on all machines should be given high priority.

In the variant of the NEH heuristic used here, a random labelling of the jobs was used so as to produce number of different starting points for the optimisers. Another consideration was that there was no *a priori* reason why one particular labelling scheme should be any better than another. A sequence of two jobs was then created, followed by the construction of a partial sequence of length $3, 4, \dots, n$, by successively inserting one unsequenced job into the existing partial sequence. This method is based upon the idea that the position chosen for the new job to be inserted in should be the one which minimises the resulting increase in makespan. Therefore, if at stage $k$ we have a partial sequence $\{J_1, \dots, J_k\}$ this implies $k$ calculations of the effect of inserting the unsequenced job $J_r$, between jobs $J_1$ & $J_2$, $J_2$ & $J_3$, and so on, or after $J_k$ (where the position of $J_1$ is fixed). Algorithm 5 below gives a more detailed description of the heuristic:

**Algorithm 5** THE VARIANT NAWAZ-EMSCORE-HAM (NEH) HEURISTIC

Assign a random labelling $(1, \dots, n)$ to the jobs;
Let $\sigma = \{J_3, \dots, J_n\}$; *// The list of unsequenced jobs*
Let $\pi = \{J_1, J_2\}$; *// The partial sequence being constructed*
**repeat**
    **for all** insertions of the first job in $\sigma$ $(\sigma_1)$ into $\pi$ **do**
        Let $\pi_{best} = \emptyset$;
        Evaluate the partial solution $(\pi^{'})$ thus generated;
        **if** $\pi_{best} = \emptyset \vee C_{max}(\pi^{'}) < C_{max}(\pi_{best})$ **then**
            Let $\pi_{best} = \pi^{'}$; *// Record any improvements in makespan*
        **end if**
    **end for**
    $\pi = \pi_{best}$; *// Let the best partial solution be the starting point for the next iteration*
    $\sigma = \sigma \setminus \sigma_1$;
**until** $\sigma = \emptyset$;
Return the sequence $\pi$;

In order to take into account the computational effort of constructing an initial solution, the number of evaluations required by the variant NEH heuristic, *weighted* by the length of the partial solution evaluated at each stage is subtracted from the time allowed for the optimiser to run. For example, if we called the evaluation function for a partial solution containing 10 jobs for a 20-job problem, then the cost of this evaluation would be counted as $10/20 = 0.5$ *actual* evaluations. Therefore, the variant NEH heuristic was found to require the equivalent of 132, 832, 3332, and 13332 evaluations for problems of size 20, 50, 100 and 200 jobs respectively. Again examination of the length of time taken to produce an initial solution showed that this was a fair approximation.

Unfortunately problems arose as a result of the above heuristic having a time complexity of $O(n^2)$ with respect to the number of evaluations made. It was found, upon examination of Tables 6.22, 6.23 and 6.24 later, that for the benchmark problems of size 200 the number of evaluations required to generate a solution using this heuristic was greater than the evaluations taken by stochastic hillclimbing to produce a solution of comparable quality. Therefore, for large problems it would appear at first sight that the cost of this heuristic may not be justified.

Two points should be noted with regard to the above. First, the use of Mohr's procedure [Taillard 90] for evaluating makespan does allow the NEH heuristic to be efficiently implemented with regards to wall-clock time — however Mohr's procedure does also benefit neighbourhood search optimisers in a similar fashion. Second, though earlier studies, such as [Reeves 95a], have found that the use of the NEH heuristic to produce a an initial solution for a neighbourhood search optimiser is beneficial, is not clear whether (or how) the costs involved in producing the initial solution in the first place was accounted for.

Therefore, two initialisation strategies with the variant NEH heuristic will be investigated. The first strategy (termed NEH in later experiments) takes the effort of generating the initial solution into full account by subtracting the evaluations required by NEH away from the evaluations allowed for the optimisation phase. The second strategy (termed NEH-LITE in later experiments) in effect provides the initial solution 'for free' by assigning a cost for the initial solution equivalent to that for a randomly generated solution (i.e. one evaluation).

### 6.6.3 Outline of the Comparative Study

Before the optimisation methods were examined, the quality of initial solutions generated by the initialisation methods was compared to see whether they do in fact start the search from a higher quality point in the search space than random initialisation. Therefore each of the initialisation methods was run 100 times and the mean and standard deviation of the solution quality obtained was recorded in Table 6.22. Where the results are highlighted in **bold**, this means that the result obtained was significantly better ($> 90\%$) by t-test than random initialisation. Results in *italics* indicate that the result obtained was significantly worse than random initialisation.

| Problem | Random | NEH | LPT | SPT |
|---|---|---|---|---|
| 20x5 | 1506.82 (64.44) | **1322.70 (25.35)** | *1645.00 (0.00)* | **1331.70 (2.49)** |
| 20x10 | 2024.68 (77.90) | **1711.14 (43.23)** | *2155.40 (19.94)* | 2031.08 (5.45) |
| 20x20 | 2771.16 (98.58) | **2446.48 (37.38)** | *2840.82 (30.73)* | *2825.28 (32.00)* |
| 50x5 | 3186.02 (124.29) | **2813.38 (37.99)** | *3665.74 (14.80)* | 3121.26 (11.16) |
| 50x10 | 3832.72 (130.04) | **3234.88 (40.88)** | *4181.62 (47.03)* | 3827.06 (21.04) |
| 50x20 | 4863.60 (114.04) | **4181.86 (55.44)** | *5175.30 (16.55)* | **4662.90 (45.91)** |
| 100x5 | 6159.88 (161.12) | **5557.46 (31.61)** | *7224.06 (13.89)* | 6121.06 (19.18) |
| 100x10 | 6897.78 (147.33) | **6031.56 (63.20)** | *8018.66 (17.53)* | **6763.52 (31.80)** |
| 100x20 | 7776.72 (151.56) | **6715.90 (53.57)** | *8550.88 (47.96)* | **7629.26 (42.05)** |
| 200x10 | 12305.96 (214.76) | **11056.20 (38.92)** | *14270.76 (39.51)* | **11989.38 (31.29)** |
| 200x20 | 13579.32 (186.49) | **11838.50 (62.98)** | *15477.30 (46.84)* | 13642.96 (58.77) |

Table 6.22: Initialisation Method Quality Compared

When the results were examined, it became evident that the variant NEH heuristic consistently gave the best results, with SPT occasionally (on 5 occasions) producing significant improvements with respect to random initialisation (though on one occasion it made matters worse). Perhaps unsurprisingly, LPT consistently produced the worst quality solutions and therefore it was decided that it would not be used in the following investigation.

Finally, the use of the initialisation heuristics did, in all cases, appear to result in a fall in solution quality variance. However, though the use of a F-test to see whether these differences were in fact significant was tempting, examination of the distribution of solution quality produced by the initialisation methods showed that they often deviated greatly from a normal distribution. As the F-test is known to be sensitive to deviations from normality, it was felt that the results obtained from an F-test would be misleading and therefore it was not used.

### 6.6.4 Effect on Relative Optimiser Performance

As the heuristic initialisation design heuristic is of a similar form to the design heuristics evaluated earlier, the experimental methodology used there is also applicable and therefore will be used here. Also, this decision has the advantage that the results obtained earlier will be better able to guide and inform the following experiments.

However, some changes to the experimental methodology were made which had the effect of making the amount of experimentation required tractable. First of all, only the permutation swap and shift operators (with their respective recombination operators) will be considered as these were the two best-performing landscapes in the earlier landscape comparison. Also, the results of the optimiser tuning for the earlier landscape comparison was used to select a 'canonical' setting for each optimiser that is generally applicable across the two landscapes and the benchmark problems considered here.

Finally in the tables that follow, results in **bold** and *italic* indicate results are significantly better or worse respectively (by t-test to $> 90\%$ confidence) than random initialisation. Results will now be presented for each optimiser in turn. Again, for the detailed results of the tuning experiments and the statistical analysis the reader is directed to Appendix C.

### 6.6.5 Hillclimbing Results

The results for stochastic hillclimbing are given in Tables 6.23 and 6.24. For all of the tables presented here, the mean and standard deviations of the makespan are given.

| Problem | Random | SPT | NEH | NEH-LITE |
|---|---|---|---|---|
| 20x5 | 1296.26 (3.63) | 1296.08 (4.18) | 1296.56 (3.28) | **1293.56 (6.86)** |
| 20x10 | 1633.32 (21.73) | 1631.54 (18.11) | **1623.88 (15.54)** | **1621.94 (14.01)** |
| 20x20 | 2373.16 (26.94) | *2385.22 (25.85)* | 2369.32 (31.52) | 2371.58 (31.60) |
| 50x5 | 2742.82 (9.14) | 2739.90 (9.80) | **2739.48 (10.17)** | **2736.30 (10.19)** |
| 50x10 | 3141.72 (24.76) | 3133.52 (30.12) | **3123.26 (17.72)** | **3123.52 (24.13)** |
| 50x20 | 4049.34 (31.64) | 4042.42 (35.63) | 4048.96 (32.97) | 4041.86 (23.21) |
| 100x5 | 5510.62 (12.91) | 5513.76 (13.81) | *5517.64 (17.17)* | 5508.70 (14.77) |
| 100x10 | 5927.38 (37.54) | 5935.44 (36.68) | **5904.24 (37.04)** | **5871.78 (27.41)** |
| 100x20 | 6622.44 (44.06) | **6598.84 (34.78)** | *6639.66 (32.70)* | 6576.06 (31.59) |
| 200x10 | 11029.90 (35.02) | 11028.10 (35.11) | — | **10998.88 (34.43)** |
| 200x20 | 11826.10 (49.64) | **11805.66 (43.34)** | — | **11707.10 (60.90)** |

Table 6.23: Summary of Stochastic Hillclimbing Results — Swap Neighbourhood

| Problem | Random | SPT | NEH | NEH-LITE |
|---|---|---|---|---|
| 20x5 | 1293.60 (7.26) | 1294.70 (5.87) | 1294.78 (6.02) | 1293.00 (7.61) |
| 20x10 | 1618.92 (13.13) | 1617.58 (13.44) | **1609.06 (11.31)** | **1609.62 (11.57)** |
| 20x20 | 2358.88 (33.10) | 2352.44 (30.23) | 2356.20 (25.61) | 2351.42 (26.99) |
| 50x5 | 2734.46 (6.76) | 2734.48 (7.00) | 2733.42 (8.03) | **2731.98 (6.64)** |
| 50x10 | 3121.10 (24.73) | **3108.98 (21.52)** | **3101.72 (20.47)** | **3112.26 (21.77)** |
| 50x20 | 4021.62 (22.52) | **4009.24 (19.82)** | 4020.96 (26.01) | 4019.38 (24.37) |
| 100x5 | 5506.56 (14.14) | 5506.48 (16.12) | 5509.12 (13.78) | 5502.36 (11.91) |
| 100x10 | 5884.80 (30.06) | *5897.84 (36.17)* | 5880.40 (32.55) | **5857.00 (29.94)** |
| 100x20 | 6581.78 (38.13) | **6560.18 (35.06)** | 6586.32 (30.72) | **6537.50 (31.88)** |
| 200x10 | 11020.20 (31.61) | 11012.62 (34.37) | — | **10992.62 (35.22)** |
| 200x20 | 11797.80 (50.06) | **11731.42 (63.99)** | — | **11655.46 (45.81)** |

Table 6.24: Summary of Stochastic Hillclimbing Results — Shift Neighbourhood

Examination of these results showed the following trends. First of all, irrespective of the initialisation method used, the shift neighbourhood did at least as well as the swap neighbourhood and sometimes better (e.g. for NEH-LITE shift was signifantly better in 5 cases). Second, for both neighbourhoods, the relative performance generally is, in increasing order, Random, SPT, NEH, then NEH-LITE which indicates that there is a relationship between the quality of the solution produced by the initialisation procedure and its effectiveness in starting the search in an advantageous region (this should be readily discernable from the pattern of bold results). The performance of SPT was, more often than not, around that of random initialisation (only improving on 2 and 4 occasions for swap and shift respectively).

Finally, the effectiveness of NEH-LITE over NEH was especially marked for the larger problems thus demonstrating that the cost of the initialisation procedure does need to be taken into account when evaluating the effectiveness of initialisation procedures.

### 6.6.6 Simulated Annealing Results

Simulated annealing was implemented, for all problems, with an initial temperature of 12, a final temperature of 0.05 with temperature changes every 100 evaluations according to a logarithmic cooling schedule. The results for simulated annealing are given in Tables 6.25 and 6.26.

| Problem | Random | SPT | NEH | NEH-LITE |
|---|---|---|---|---|
| 20x5 | 1296.00 (4.30) | 1296.30 (3.65) | 1296.02 (3.99) | 1295.94 (3.41) |
| 20x10 | 1619.72 (14.73) | 1621.08 (12.96) | 1622.42 (10.67) | 1618.60 (12.05) |
| 20x20 | 2357.36 (24.21) | 2357.70 (25.18) | 2351.50 (21.50) | 2355.88 (22.49) |
| 50x5 | 2741.14 (10.02) | 2739.20 (10.59) | 2740.56 (10.24) | 2739.00 (9.55) |
| 50x10 | 3135.34 (21.37) | 3131.78 (23.13) | **3126.20 (17.95)** | **3127.12 (21.82)** |
| 50x20 | 4044.52 (27.69) | 4035.76 (26.59) | 4037.24 (27.81) | **4034.18 (27.02)** |
| 100x5 | 5516.12 (13.28) | 5513.50 (15.92) | *5521.72 (14.50)* | **5509.78 (14.74)** |
| 100x10 | 5939.50 (41.34) | 5927.64 (33.35) | **5921.38 (30.82)** | **5889.30 (33.40)** |
| 100x20 | 6606.50 (31.98) | 6598.52 (35.52) | *6628.98 (39.93)* | **6576.98 (36.34)** |
| 200x10 | 11035.90 (41.03) | 11026.22 (33.66) | — | **11008.60 (33.68)** |
| 200x20 | 11797.40 (52.15) | 11789.04 (51.81) | — | **11717.32 (43.98)** |

Table 6.25: Simulated Annealing Results — Swap Neighbourhood

| Problem | Random | SPT | NEH | NEH-LITE |
|---|---|---|---|---|
| 20x5 | 1295.36 (5.19) | 1294.60 (5.78) | **1294.28 (6.11)** | **1290.58 (8.52)** |
| 20x10 | 1609.08 (10.86) | 1609.60 (12.36) | 1609.56 (12.92) | 1607.72 (11.84) |
| 20x20 | 2340.68 (22.56) | 2340.20 (24.68) | 2347.28 (23.06) | 2340.10 (22.09) |
| 50x5 | 2737.70 (7.63) | **2734.08 (7.26)** | 2737.26 (7.31) | **2734.76 (7.40)** |
| 50x10 | 3116.48 (23.13) | **3105.68 (20.41)** | 3114.64 (20.58) | 3114.28 (20.91) |
| 50x20 | 4007.50 (23.25) | 4009.92 (23.90) | 4008.72 (18.81) | 4009.46 (20.06) |
| 100x5 | 5506.14 (14.65) | 5508.16 (13.72) | *5515.42 (14.67)* | 5504.78 (13.03) |
| 100x10 | 5888.60 (34.58) | 5897.94 (35.35) | *5901.44 (35.97)* | **5866.24 (23.85)** |
| 100x20 | 6562.66 (36.39) | 6557.16 (36.05) | *6576.46 (31.13)* | **6538.86 (33.80)** |
| 200x10 | 11023.30 (38.10) | 11027.02 (38.12) | — | **11004.90 (36.55)** |
| 200x20 | 11750.30 (51.57) | **11727.22 (49.17)** | — | **11684.10 (45.29)** |

Table 6.26: Simulated Annealing Results — Shift Neighbourhood

The results obtained were similar in nature to those obtained for SHC — the trends regarding the relative performance of the neighbourhoods and initialisation procedures mirrored those obtained above. In addition, simulated annealing generally did improve solution quality compared to SHC.

### 6.6.7 Threshold Accepting Results

Threshold accepting was implemented, with an initial threshold of 9, and a final threshold of 0, with the threshold changing linearly every 100 evaluations. Again the results obtained and trends observed for threshold accepting were similar in nature to those obtained for SHC, though solution quality was improved (Tables 6.27 and 6.28).

| Problem | Random | SPT | NEH | NEH-LITE |
|---|---|---|---|---|
| 20x5 | 1296.28 (3.08) | 1296.40 (2.50) | 1295.60 (4.36) | 1295.84 (3.40) |
| 20x10 | 1620.30 (16.21) | 1619.18 (15.17) | 1619.54 (11.61) | **1612.98 (11.06)** |
| 20x20 | 2358.22 (22.21) | 2358.26 (24.87) | 2353.22 (22.97) | 2355.16 (29.30) |
| 50x5 | 2739.02 (9.35) | 2740.36 (9.28) | 2738.52 (10.19) | 2738.14 (9.77) |
| 50x10 | 3128.22 (28.89) | 3126.12 (24.03) | 3125.28 (19.36) | 3120.04 (19.47) |
| 50x20 | 4031.50 (23.21) | **4023.20 (20.85)** | 4038.00 (22.47) | 4030.88 (20.23) |
| 100x5 | 5517.66 (15.28) | 5515.30 (14.22) | 5519.30 (14.37) | **5512.52 (13.73)** |
| 100x10 | 5921.44 (31.73) | 5926.40 (36.57) | **5910.04 (31.49)** | **5876.28 (30.68)** |
| 100x20 | 6600.00 (34.02) | 6593.92 (29.78) | *6614.98 (39.34)* | **6568.34 (30.66)** |
| 200x10 | 11027.60 (34.44) | 11023.90 (32.90) | — | **11002.02 (32.11)** |
| 200x20 | 11798.70 (44.20) | **11781.80 (45.57)** | — | **11698.54 (47.41)** |

Table 6.27: Threshold Accepting Results — Swap Neighbourhood

| Problem | Random | SPT | NEH | NEH-LITE |
|---|---|---|---|---|
| 20x5 | 1295.30 (5.09) | 1296.30 (2.87) | 1294.66 (5.70) | 1294.28 (5.42) |
| 20x10 | 1604.64 (10.74) | 1607.28 (11.10) | 1602.84 (10.93) | 1604.56 (13.92) |
| 20x20 | 2341.72 (20.15) | 2335.58 (23.03) | 2342.84 (22.35) | 2339.56 (24.15) |
| 50x5 | 2735.90 (7.23) | 2736.86 (7.54) | 2734.68 (7.62) | 2734.34 (7.83) |
| 50x10 | 3115.56 (20.06) | 3113.72 (25.58) | 3115.08 (20.19) | 3111.18 (20.65) |
| 50x20 | 4007.74 (20.17) | **4000.04 (19.45)** | 4008.74 (19.72) | 4008.34 (17.38) |
| 100x5 | 5507.98 (13.64) | 5508.32 (13.77) | 5510.42 (14.24) | 5506.84 (13.04) |
| 100x10 | 5889.48 (32.69) | 5895.70 (32.53) | 5899.50 (33.35) | **5870.74 (27.81)** |
| 100x20 | 6552.14 (32.45) | 6557.54 (28.06) | *6583.80 (25.16)* | **6537.02 (27.10)** |
| 200x10 | 11017.40 (33.09) | 11014.92 (38.26) | — | **10990.18 (34.36)** |
| 200x20 | 11736.00 (46.31) | 11722.46 (42.33) | — | **11654.18 (51.05)** |

Table 6.28: Threshold Accepting Results — Shift Neighbourhood

### 6.6.8 Record-to-Record Travel Results

Record-to-record travel was implemented, for all of the test problems, with a threshold of 5 which was held constant. The results for RTRT are given in Tables 6.29 and 6.30. The results obtained were similar in nature to those obtained for SHC, though generally of higher quality and the trends observed for the other SHC-based optimisers are mirrored here.

### 6.6.9 Evolutionary Algorithm Results

Examination of the results of the earlier comparative studies lead to the adoption of a steady-state EA with kill-worst replacement, a crossover probability of 0.3, and a mutation probability of 0.7. The Modified PMX and PPX recombination operators were used as they correspond to the swap and shift operators with respect to the metric spaces they operate in. The results for the evolutionary algorithm experiments are given in Tables 6.31, 6.32, 6.33, and 6.34 below.

The changes in makespan effected by the operator for the EA were of roughly the same size and showed the same trends as those obtained for the other SHC-based optimisers considered here — which leads some support to the idea the SHC comparisons are transferable to EA

| Problem | Random | SPT | NEH | NEH-LITE |
|---|---|---|---|---|
| 20x5 | 1296.26 (3.63) | 1296.14 (3.40) | 1295.80 (4.38) | 1295.28 (4.60) |
| 20x10 | 1619.02 (14.18) | 1620.36 (19.31) | 1617.44 (14.35) | 1616.42 (15.59) |
| 20x20 | 2355.62 (22.71) | 2368.46 (23.89) | *2423.90 (29.08)* | 2362.24 (30.78) |
| 50x5 | 2739.16 (10.17) | 2738.72 (9.43) | 2737.66 (11.22) | 2739.40 (9.91) |
| 50x10 | 3138.70 (22.01) | 3133.08 (23.30) | **3126.48 (21.88)** | **3117.14 (18.54)** |
| 50x20 | 4044.50 (26.21) | **4034.46 (22.60)** | 4041.34 (21.61) | 4043.54 (24.98) |
| 100x5 | 5516.22 (15.10) | 5514.82 (17.67) | 5517.84 (15.42) | **5509.68 (14.32)** |
| 100x10 | 5917.56 (31.68) | 5927.06 (33.39) | **5901.54 (27.19)** | **5875.34 (38.53)** |
| 100x20 | 6608.70 (32.57) | **6598.50 (26.49)** | 6613.96 (35.42) | **6583.16 (34.89)** |
| 200x10 | 11033.50 (37.67) | 11040.32 (36.16) | — | **11003.50 (34.59)** |
| 200x20 | 11819.30 (37.84) | **11791.34 (49.45)** | — | **11703.32 (51.43)** |

Table 6.29: Results for Record-to-Record Travel — Swap Neighbourhood

| Problem | Random | SPT | NEH | NEH-LITE |
|---|---|---|---|---|
| 20x5 | 1294.14 (6.18) | 1294.52 (5.79) | 1294.48 (6.08) | 1293.76 (6.60) |
| 20x10 | 1609.44 (9.36) | 1608.12 (11.92) | **1604.96 (10.70)** | **1602.40 (10.90)** |
| 20x20 | 2342.06 (23.12) | 2342.04 (23.81) | *2418.94 (31.80)* | 2343.26 (26.85) |
| 50x5 | 2735.68 (8.91) | 2738.58 (9.32) | 2734.48 (8.22) | 2734.30 (9.79) |
| 50x10 | 3118.86 (22.65) | 3112.32 (22.06) | **3107.64 (18.40)** | **3108.34 (23.95)** |
| 50x20 | 4019.06 (22.79) | 4014.76 (19.38) | **4009.44 (21.47)** | 4015.98 (18.61) |
| 100x5 | 5509.24 (15.00) | **5503.98 (12.11)** | 5509.72 (13.13) | 5505.08 (12.88) |
| 100x10 | 5889.70 (33.16) | 5894.50 (34.32) | 5885.02 (29.35) | **5859.74 (26.89)** |
| 100x20 | 6567.06 (37.40) | 6559.76 (28.74) | 6565.24 (42.97) | **6530.38 (36.77)** |
| 200x10 | 11019.90 (39.49) | 11018.84 (32.92) | — | **10994.92 (36.98)** |
| 200x20 | 11752.80 (55.88) | **11726.78 (47.30)** | — | **11654.18 (50.96)** |

Table 6.30: Results for Record-to-Record Travel — Shift Neighbourhood

design. However the standard deviation of the EA results obtained are much higher than for the other SHC-based optimisers, which means that the improvements in performance cannot be said to be statistically significant.

Finally, as an aside, the much higher standard deviation again obtained compared with the other optimisers considered here could be seen to be a reason for not using an EA (or at least this type of EA) for this problem: users would prefer a greater degree of certainty of what the final solution will be, and they may chose optimisers that provide this.

### 6.6.10 First-Ascent Hillclimbing Results

The results for FAHC are given in Tables 6.35 and 6.36. Comparison with the results for SHC-based optimisers shows that the performance of FAHC is, at best, as good as that for SHC and worse for larger problems irrespective of the initialisation method used. This reduction in performance from going from SHC to FAHC is in line with the results obtained in Section 6.4.

Examination of these results showed the following trends. First of all, irrespective of the initialisation method used, the shift neighbourhood did at least as well as the swap neighbour-

| Problem | Random | SPT | NEH | NEH-LITE |
|---|---|---|---|---|
| 20x5 | 1268.76 (87.63) | 1266.28 (89.88) | 1266.54 (90.54) | 1264.50 (89.70) |
| 20x10 | 1590.32 (80.20) | 1593.66 (80.71) | 1588.30 (79.60) | 1588.56 (81.82) |
| 20x20 | 2284.18 (66.64) | 2289.06 (68.44) | 2286.94 (72.60) | 2287.68 (73.54) |
| 50x5 | 2814.04 (129.00) | 2811.90 (129.19) | 2810.22 (131.90) | 2807.84 (133.08) |
| 50x10 | 3174.40 (101.20) | 3169.70 (103.03) | 3146.68 (97.80) | 3146.74 (101.06) |
| 50x20 | 4033.84 (88.94) | 4031.06 (85.52) | **4002.14 (89.91)** | **3996.34 (92.60)** |
| 100x5 | 5401.38 (186.55) | 5402.98 (183.84) | 5398.54 (191.10) | 5389.68 (185.35) |
| 100x10 | 5844.22 (148.34) | 5845.28 (154.39) | 5817.24 (156.51) | **5793.00 (154.25)** |
| 100x20 | 6828.34 (81.23) | 6831.08 (103.31) | *6886.58 (97.48)* | **6665.84 (98.65)** |
| 200x10 | 11073.30 (208.54) | 11076.84 (208.20) | — | **10985.10 (230.08)** |
| 200x20 | 12243.10 (124.37) | 12282.46 (130.82) | — | **11892.72 (133.90)** |

Table 6.31: Evolutionary Algorithm Results — Swap and Modified PMX Operators

| Problem | Random | SPT | NEH | NEH-LITE |
|---|---|---|---|---|
| 20x5 | 1263.06 (90.78) | 1264.14 (90.11) | 1263.42 (91.07) | 1261.56 (90.47) |
| 20x10 | 1576.00 (78.94) | 1582.10 (81.21) | 1581.86 (83.87) | 1578.26 (82.98) |
| 20x20 | 2271.08 (69.95) | 2276.12 (73.11) | 2273.20 (68.70) | 2274.30 (72.09) |
| 50x5 | 2809.12 (130.13) | 2806.58 (129.08) | 2805.78 (131.39) | 2804.30 (132.88) |
| 50x10 | 3165.16 (103.64) | 3160.78 (101.29) | 3144.68 (95.85) | 3139.96 (95.35) |
| 50x20 | 4024.14 (82.78) | 4026.42 (82.54) | **3979.54 (88.93)** | **3975.46 (89.66)** |
| 100x5 | 5390.96 (185.53) | 5390.18 (190.26) | 5391.66 (190.23) | 5384.82 (191.51) |
| 100x10 | 5831.74 (144.03) | 5850.02 (154.60) | 5805.36 (155.00) | **5777.18 (158.13)** |
| 100x20 | 6836.50 (86.46) | 6832.40 (96.98) | 6859.52 (97.91) | **6649.98 (102.07)** |
| 200x10 | 11072.90 (216.02) | 11066.86 (218.27) | — | **10966.94 (227.85)** |
| 200x20 | 12267.60 (133.43) | 12258.14 (135.45) | — | **11884.24 (137.32)** |

Table 6.32: Evolutionary Algorithm Results — Shift and Modified PMX Operators

hood and sometimes better, as was the case for the SHC-based neighbourhoods. However for the larger problems the swap neighbourhood did relatively which again mirrors the trends observed earlier. As an illustration, in the case of NEH-LITE shift did better than swap in 3 cases).

Second, for both neighbourhoods, the relative performance generally is, in increasing order, Random, SPT, NEH, then NEH-LITE which indicates that there is a relationship between the quality of the solution produced by the initialisation procedure and its effectiveness in starting the search in an advantageous region. Also, compared with the results obtained for SHC, the improvements produced by various initialisation methods were more pronounced here. This is illustrated by the results for SPT which though like SHC did not show any improvement on random initialisation for the small problems, did show significant improvements for the larger problems. Finally, the effectiveness of NEH-LITE over NEH was again especially marked for the larger problems.

| Problem | Random | SPT | NEH | NEH-LITE |
|---|---|---|---|---|
| 20x5 | 1266.00 (88.95) | 1265.38 (92.24) | 1264.18 (91.37) | 1264.18 (91.37) |
| 20x10 | 1591.92 (85.28) | 1590.40 (85.51) | 1583.62 (81.73) | 1582.78 (81.98) |
| 20x20 | 2276.54 (73.91) | 2280.54 (75.92) | 2278.42 (69.39) | 2276.76 (69.16) |
| 50x5 | 2810.78 (131.24) | 2812.16 (132.07) | 2808.46 (133.12) | 2807.96 (133.46) |
| 50x10 | 3173.72 (104.94) | 3168.56 (104.65) | 3153.88 (98.22) | 3148.38 (98.99) |
| 50x20 | 4032.34 (89.43) | 4032.24 (87.10) | 4002.64 (90.64) | **3996.44 (90.01)** |
| 100x5 | 5398.54 (190.18) | 5403.02 (191.50) | 5395.46 (193.10) | 5387.78 (189.37) |
| 100x10 | 5841.48 (141.79) | 5846.20 (150.37) | 5814.38 (154.08) | 5792.90 (159.85) |
| 100x20 | 6847.10 (91.11) | 6830.12 (100.92) | *6888.36 (98.41)* | **6644.04 (100.13)** |
| 200x10 | 11068.22 (204.37) | 11084.94 (220.83) | — | **10984.18 (228.48)** |
| 200x20 | 12275.48 (124.43) | 12274.80 (128.23) | — | 11895.80 (135.05) |

Table 6.33: Evolutionary Algorithm Results — Swap and PPX Operators

| Problem | Random | SPT | NEH | NEH-LITE |
|---|---|---|---|---|
| 20x5 | 1262.12 (93.06) | 1262.22 (92.68) | 1262.90 (91.95) | 1262.80 (92.01) |
| 20x10 | 1579.58 (80.38) | 1583.58 (81.49) | 1578.92 (83.51) | 1578.22 (83.61) |
| 20x20 | 2270.02 (70.75) | 2269.24 (73.99) | 2268.26 (72.51) | 2267.92 (72.60) |
| 50x5 | 2806.96 (128.70) | 2807.78 (129.38) | 2805.12 (132.48) | 2804.48 (132.98) |
| 50x10 | 3158.54 (109.64) | 3171.04 (108.71) | 3139.06 (99.18) | 3132.50 (99.22) |
| 50x20 | 4022.26 (82.40) | 4016.46 (84.87) | **3980.70 (92.30)** | **3973.16 (92.79)** |
| 100x5 | 5390.86 (190.86) | 5399.16 (185.95) | 5394.44 (189.93) | 5384.16 (190.96) |
| 100x10 | 5848.52 (150.18) | 5841.28 (138.69) | 5807.82 (160.78) | **5778.94 (160.76)** |
| 100x20 | 6836.06 (98.81) | 6837.10 (100.87) | 6855.22 (101.49) | **6644.04 (100.13)** |
| 200x10 | 11077.52 (221.53) | 11071.66 (210.67) | — | **10963.18 (225.74)** |
| 200x20 | 12283.90 (134.45) | 12271.28 (107.97) | — | 11878.84 (138.32) |

Table 6.34: Evolutionary Algorithm Results — Shift and PPX Operators

### 6.6.11 First-Ascent Tabu Search Results

The results for FATS are given in Tables 6.37 and 6.38. The earlier comparative experiments did not come up with a definitive value for the tabu tenure for all problems, but a tenure of 7 moves was found to give acceptable performance in general and so was adopted. As expected, results obtained for FATS reflected those obtained for FAHC with respect to all of the trends noted above.

### 6.6.12 Steepest-Ascent Hillclimbing Results

The results for SAHC are given in Tables 6.39 and 6.40. Comparison with the results for SHC-based optimisers shows that the performance of SAHC is, at best, comparable to that for SHC and FAHC and more often than not worse for all but the smaller problem instances, irrespective of the initialisation method used. Again, this reduction in performance from going from SHC to FAHC to SAHC is in line with the results obtained earlier.

Examination of these results showed the following trends. First of all, irrespective of the initialisation method used, the shift neighbourhood did at least as well as the swap neighbourhood

| Problem | Random | SPT | NEH | NEH-LITE |
|---|---|---|---|---|
| 20x5 | 1300.00 (12.21) | **1296.60 (3.92)** | 1297.40 (5.69) | **1293.96 (6.94)** |
| 20x10 | 1640.56 (22.28) | 1644.46 (26.43) | **1627.36 (17.58)** | **1626.64 (17.26)** |
| 20x20 | 2383.40 (27.21) | 2379.92 (30.10) | **2375.70 (25.94)** | **2371.14 (32.60)** |
| 50x5 | 2753.20 (15.95) | **2743.70 (10.60)** | **2738.42 (12.29)** | **2741.40 (16.51)** |
| 50x10 | 3162.76 (27.62) | 3169.52 (38.75) | **3139.20 (18.78)** | **3133.82 (26.90)** |
| 50x20 | 4065.66 (29.83) | 4069.60 (34.48) | 4060.72 (35.42) | **4057.08 (28.91)** |
| 100x5 | 5531.66 (22.18) | 5539.50 (31.40) | 5527.10 (23.37) | 5515.78 (20.34) |
| 100x10 | 6038.72 (61.47) | **6005.66 (47.64)** | **5946.30 (58.75)** | **5923.50 (50.89)** |
| 100x20 | 6735.62 (56.59) | **6670.52 (43.02)** | **6635.42 (45.50)** | **6610.82 (39.25)** |
| 200x10 | 11164.10 (62.66) | **11091.00 (45.53)** | — | **11012.20 (36.28)** |
| 200x20 | 12120.30 (69.41) | **12068.02 (76.89)** | — | **11732.36 (62.43)** |

Table 6.35: First-Ascent Hillclimbing Results — Swap Neighbourhood

| Problem | Random | SPT | NEH | NEH-LITE |
|---|---|---|---|---|
| 20x5 | 1294.58 (6.34) | 1295.54 (4.72) | 1295.82 (5.63) | 1293.52 (7.41) |
| 20x10 | 1621.52 (12.05) | 1620.64 (15.83) | **1617.24 (12.81)** | **1613.20 (13.63)** |
| 20x20 | 2353.36 (27.41) | 2356.78 (23.46) | 2356.84 (31.36) | 2353.40 (35.81) |
| 50x5 | 2742.54 (11.95) | 2740.30 (7.62) | **2737.52 (12.89)** | 2738.56 (13.26) |
| 50x10 | 3163.36 (25.51) | **3144.58 (27.59)** | **3127.14 (21.43)** | **3127.40 (25.27)** |
| 50x20 | 4068.56 (34.94) | **4052.74 (23.56)** | **4049.26 (33.10)** | **4046.44 (24.77)** |
| 100x5 | 5539.72 (32.44) | 5530.80 (29.90) | **5524.82 (22.99)** | **5510.64 (15.70)** |
| 100x10 | 6038.72 (61.47) | **6011.18 (44.62)** | **5935.42 (35.22)** | **5912.74 (35.92)** |
| 100x20 | 6806.38 (60.89) | **6735.86 (45.83)** | **6630.32 (44.05)** | **6606.84 (41.65)** |
| 200x10 | 11295.90 (77.61) | **11156.24 (48.93)** | — | **11013.86 (37.74)** |
| 200x20 | 12326.00 (89.20) | **12283.04 (88.49)** | — | **11741.94 (57.11)** |

Table 6.36: First-Ascent Hillclimbing Results — Shift Neighbourhood

and sometimes better for the small problem instances (in 3 cases for NEH-LITE). However, for the remaining problems the swap neighbourhood was more effective (signficant in 2 cases for NEH-LITE) which mirrors earlier results. Second, for both neighbourhoods the relative performance generally was, in increasing order, Random, SPT, NEH, then NEH-LITE which again indicates that there is a relationship between the quality of the solution produced by the initialisation procedure and its effectiveness in starting the search in an advantageous region.

Finally as for FAHC, the improvements produced by various initialisation methods were more pronounced here compared with the results obtained for SHC. Also, the effectiveness of NEH-LITE over NEH was again especially marked for the larger problems.

### 6.6.13 Steepest-Ascent Tabu Search Results

The results for SATS are given in Tables 6.41 and 6.42. As for FATS, the earlier comparative experiments did not come up with a definitive value for the tabu tenure for all problems, but again a tenure of 7 moves was found to give acceptable performance in general. As expected, results obtained for SATS were similar in nature to those obtained for SAHC with respect to

| Problem | Random | SPT | NEH | NEH-LITE |
|---|---|---|---|---|
| 20x5 | 1298.12 (6.79) | 1297.26 (3.50) | 1296.60 (4.63) | **1294.44 (6.64)** |
| 20x10 | 1628.60 (19.03) | 1634.64 (25.42) | **1618.22 (14.84)** | **1614.32 (14.18)** |
| 20x20 | 2358.02 (24.97) | 2362.96 (25.11) | 2358.28 (25.88) | **2349.96 (25.55)** |
| 50x5 | 2753.60 (15.59) | **2743.98 (11.40)** | **2738.82 (13.96)** | **2742.14 (16.59)** |
| 50x10 | 3160.36 (26.66) | *3174.06 (37.67)* | **3140.18 (21.91)** | **3133.70 (25.96)** |
| 50x20 | 4073.28 (33.06) | 4067.88 (37.02) | 4061.88 (35.85) | **4057.42 (31.02)** |
| 100x5 | 5532.54 (21.19) | 5539.32 (31.44) | 5527.22 (23.59) | **5515.78 (20.34)** |
| 100x10 | 6039.56 (60.06) | **6004.32 (47.45)** | **5948.58 (58.84)** | **5923.12 (50.56)** |
| 100x20 | 6730.40 (65.55) | **6669.16 (40.46)** | **6632.06 (48.75)** | **6612.70 (40.66)** |
| 200x10 | 11162.90 (62.58) | **11091.06 (45.63)** | — | **11012.50 (36.77)** |
| 200x20 | 12122.70 (70.10) | **12070.80 (79.28)** | — | **11734.64 (62.79)** |

Table 6.37: First-Ascent Tabu Search Results — Swap Neighbourhood

| Problem | Random | SPT | NEH | NEH-LITE |
|---|---|---|---|---|
| 20x5 | 1295.30 (5.92) | 1294.66 (6.21) | 1296.40 (5.24) | 1293.78 (7.26) |
| 20x10 | 1613.94 (14.63) | 1617.44 (15.81) | 1611.80 (11.85) | **1608.14 (12.12)** |
| 20x20 | 2341.00 (21.71) | 2344.54 (26.92) | 2346.50 (27.31) | 2345.48 (31.61) |
| 50x5 | 2742.96 (12.16) | 2740.24 (7.62) | **2737.36 (12.85)** | 2738.84 (13.16) |
| 50x10 | 3164.14 (28.40) | **3144.74 (26.68)** | **3126.86 (21.02)** | **3128.18 (26.73)** |
| 50x20 | 4063.28 (31.75) | **4050.92 (25.07)** | **4050.14 (33.38)** | **4047.66 (27.37)** |
| 100x5 | 5537.74 (32.79) | 5530.60 (30.32) | **5523.72 (23.03)** | **5510.68 (15.66)** |
| 100x10 | 6024.54 (43.80) | 6012.20 (43.93) | **5935.54 (35.41)** | **5912.74 (35.92)** |
| 100x20 | 6802.86 (58.70) | **6736.70 (44.19)** | **6636.28 (43.49)** | **6608.04 (42.39)** |
| 200x10 | 11295.90 (77.61) | **11155.68 (48.33)** | — | **11013.88 (37.84)** |
| 200x20 | 12328.60 (88.14) | **12282.84 (88.72)** | — | **11741.86 (56.85)** |

Table 6.38: First-Ascent Tabu Search Results — Shift Neighbourhood

all of the trends noted so far.

### 6.6.14 Summary and Additional Remarks

The results obtained in this study support the validity of the heuristic initialisation design heuristic — which suggested that hillclimbing experiments could be used as an experimental protocol to compare initialisation methods in a way that is transferable between optimisers. Trends also appeared with respect to the effect of the neighbourhood and the relative performance of the hillclimbing types, reflecting those found in earlier comparison of neighbourhood operators 6.4. This is useful as it supports the assertion made earlier (design heuristic 1e — see 4.8.3) that the landscape should be selected before initialisation methods are considered. If the choice of landscape was to change with initialisation method then such a guideline would become less useful.

In addition, there appeared to be a relationship between the quality of the initial solution produced and the overall performance of the optimiser. This is not unexpected if the fitness landscape is correlated as a high(er) quality solution will be more likely to be closer to other

| Problem | Random | SPT | NEH | NEH-LITE |
|---|---|---|---|---|
| 20x5 | 1299.52 (12.60) | **1295.74 (4.06)** | 1297.70 (4.44) | **1294.20 (7.29)** |
| 20x10 | 1645.22 (25.80) | **1698.00 (22.92)** | **1629.64 (13.83)** | **1629.12 (16.18)** |
| 20x20 | 2389.76 (29.98) | 2386.58 (16.98) | **2369.04 (34.59)** | **2375.52 (36.65)** |
| 50x5 | 2775.08 (23.42) | **2744.18 (10.90)** | **2743.04 (13.14)** | **2742.28 (16.17)** |
| 50x10 | 3352.96 (40.90) | *3368.24 (30.74)* | **3157.74 (28.04)** | **3149.00 (26.84)** |
| 50x20 | 4341.72 (54.18) | **4260.78 (27.10)** | **4086.54 (37.80)** | **4076.80 (30.19)** |
| 100x5 | 5858.14 (118.98) | **5749.80 (19.51)** | **5540.64 (25.98)** | **5523.68 (20.14)** |
| 100x10 | 6650.20 (108.27) | **6541.78 (24.19)** | **5988.50 (54.20)** | **5958.38 (48.51)** |
| 100x20 | 7540.60 (139.51) | **7333.44 (37.71)** | **6675.76 (44.15)** | **6650.76 (40.02)** |
| 200x10 | 12134.90 (197.88) | **11827.40 (26.27)** | — | **11023.36 (36.00)** |
| 200x20 | 13407.00 (172.98) | **13454.80 (53.31)** | — | **11771.30 (64.80)** |

Table 6.39: Steepest-Ascent Hillclimbing Results — Swap Neighbourhood

| Problem | Random | SPT | NEH | NEH-LITE |
|---|---|---|---|---|
| 20x5 | 1299.20 (8.52) | **1296.62 (2.66)** | **1295.96 (5.47)** | **1293.66 (7.93)** |
| 20x10 | 1630.40 (17.16) | 1626.28 (10.64) | **1618.02 (13.77)** | **1615.32 (12.63)** |
| 20x20 | 2381.42 (28.65) | 2387.62 (10.58) | **2355.34 (25.74)** | **2360.30 (27.68)** |
| 50x5 | 2936.68 (103.90) | **2819.12 (20.72)** | **2743.40 (16.87)** | **2742.02 (15.53)** |
| 50x10 | 3573.30 (95.47) | 3589.44 (18.51) | **3169.38 (28.27)** | **3165.16 (26.61)** |
| 50x20 | 4576.70 (99.95) | **4418.40 (34.45)** | **4095.52 (39.31)** | **4094.08 (35.73)** |
| 100x5 | 6036.56 (147.63) | **5897.80 (22.07)** | **5541.08 (28.04)** | **5531.14 (21.59)** |
| 100x10 | 6837.74 (127.17) | **6672.32 (21.76)** | **5989.80 (60.76)** | **5988.08 (54.39)** |
| 100x20 | 7715.66 (164.52) | **7520.28 (36.22)** | **6675.26 (50.50)** | **6675.26 (50.50)** |
| 200x10 | 12195.60 (199.38) | **11891.44 (29.86)** | — | **11022.56 (37.51)** |
| 200x20 | 13465.90 (172.81) | **13518.20 (51.20)** | — | **11771.04 (65.59)** |

Table 6.40: Steepest-Ascent Hillclimbing Results — Shift Neighbourhood

high(er) quality solutions and possibly the optimum. Of course, other factors do have to be considered, most notably the cost of the initialisation procedure. This was highlighted by the relative performance of the variant NEH and NEH-LITE procedures and the fact that the variant NEH procedure was outperformed by a hillclimber for the largest of the problem instances considered here.

Finally, this study showed that the effect of using more informed initialisation procedures was more marked upon going from SHC to FAHC then to SAHC-based optimisers. This may well be due to the higher ratio of evaluations to hillclimbing acceptance steps experienced by FAHC and SAHC. This directly corresponds to a shorter possible walk across the fitness landscape in the time available. Therefore an informed initialisation procedure's ability to start the search closer to high quality solutions would have a greater impact for FAHC and SAHC-based optimisers as it would cut out many of the unnecessary and expensive steps required before solutions of a given quality are found.

| Problem | Random | SPT | NEH | NEH-LITE |
|---|---|---|---|---|
| 20x5 | 1302.42 (15.05) | **1294.72 (5.66)** | **1297.44 (4.27)** | **1293.92 (8.55)** |
| 20x10 | 1641.10 (22.67) | **1648.32 (18.56)** | **1621.60 (12.39)** | **1616.16 (13.04)** |
| 20x20 | 2370.96 (26.69) | 2375.06 (11.87) | **2356.48 (26.99)** | **2356.80 (28.90)** |
| 50x5 | 2776.60 (26.18) | 2746.40 (12.30) | 2742.92 (13.19) | 2741.86 (16.05) |
| 50x10 | 3352.62 (40.50) | *3367.36 (28.17)* | **3157.70 (28.03)** | **3149.00 (26.84)** |
| 50x20 | 4341.28 (53.78) | **4266.92 (30.36)** | **4087.36 (36.45)** | **4076.80 (30.19)** |
| 100x5 | 5858.14 (118.98) | **5751.38 (17.99)** | **5540.64 (25.98)** | **5523.72 (20.08)** |
| 100x10 | 6650.20 (108.27) | **6542.58 (21.53)** | **5989.32 (54.66)** | **5958.90 (48.53)** |
| 100x20 | 7540.60 (139.51) | **7347.70 (36.44)** | **6680.40 (53.04)** | **6650.76 (40.02)** |
| 200x10 | 12134.90 (197.88) | **11829.84 (27.34)** | — | **11023.36 (36.00)** |
| 200x20 | 13407.00 (172.98) | **13455.98 (52.42)** | — | **11771.46 (64.90)** |

Table 6.41: Steepest-Ascent Tabu Search Results — Swap Neighbourhood

| Problem | Random | SPT | NEH | NEH-LITE |
|---|---|---|---|---|
| 20x5 | 1300.14 (10.57) | **1295.84 (4.19)** | **1295.96 (6.37)** | **1293.58 (7.85)** |
| 20x10 | 1626.62 (15.85) | 1622.82 (10.84) | **1614.28 (11.82)** | **1612.34 (12.60)** |
| 20x20 | 2377.74 (29.81) | *2386.24 (10.60)* | **2348.92 (26.42)** | **2353.54 (31.03)** |
| 50x5 | 2937.24 (103.48) | **2820.28 (19.79)** | **2743.40 (16.82)** | **2742.52 (15.45)** |
| 50x10 | 3573.32 (95.48) | 3589.82 (18.72) | **3169.54 (28.01)** | **3165.60 (26.61)** |
| 50x20 | 4576.68 (100.09) | **4423.22 (33.82)** | **4095.72 (39.32)** | **4096.52 (35.08)** |
| 100x5 | 6036.56 (147.63) | **5898.22 (22.46)** | **5541.08 (28.04)** | **5531.80 (21.80)** |
| 100x10 | 6837.74 (127.17) | **6673.42 (22.30)** | **5989.80 (60.76)** | **5988.54 (54.37)** |
| 100x20 | 7715.66 (164.52) | **7523.12 (37.46)** | **6678.56 (55.82)** | **6675.34 (50.54)** |
| 200x10 | 12195.60 (199.38) | **11891.44 (29.86)** | — | **11022.56 (37.51)** |
| 200x20 | 13465.90 (172.81) | **13518.20 (51.20)** | — | **11771.04 (65.59)** |

Table 6.42: Steepest-Ascent Tabu Search Results — Shift Neighbourhood

## 6.7 Examining Move Selection

Attention will now be turned to the design heuristic for the move preference knowledge source proposed in 4.7. Again, as the form of the design heuristic is similar to those carried out previously the same experimental set-up as the heuristic initialisation knowledge source will be used, including the canonical optimiser settings. The only difference is that there are additional settings involved in the implementation of the ‘directed mutation’ mechanism for exploiting the move selection information. These will be tuned and the results of the tuning experiments and the statistical analysis are given in full in Appendix C.

The above experimentation will be set in the scene of evaluating an idle-time based move selection heuristic. Therefore, to set the scene for the comparative experiments, the idle time move selection heuristic will be first described and justified and its implementation detailed. The results of the comparative study will then be presented, as before, on an optimiser-by-optimiser basis. This investigation will then be concluded by a discussion of the implications of the results as a whole.

### 6.7.1 The Idle Time Heuristic

The heuristic for move selection that we will use for this problem, which has the objective of minimising makespan, is based upon idle times. Idle time is usually defined for a particular job and a particular machine, and it is the amount of time that a job is waiting to be processed at a machine (because the machine in question is unavailable), after it has been processed by the previous machine. This is illustrated by the Gantt chart (Figure 6.3) given earlier in this chapter — the gaps between blocks are the periods in which a machine is lying idle.

Given that the presence of idle time is detrimental as we are attempting to minimise makespan, we could base move selection on some aggregated measure of idle time associated with a given job, $I(J_i)$, such as that given below:

$$I(J_1) = \sum_{j=2}^{m} p(J_1, j-1) \ (for\ the\ purposes\ of\ this\ study\ -\ see\ below)$$
$$I(J_i) = \sum_{j=2}^{m} max\{C(J_{i-1}, j) - C(J_i, j-1), 0\}\ for\ j = 2, \ldots, m$$

with processing times $p(i, j)$ for job $i$ on machine $j$, the job permutation $\{J_1, J_2, \ldots, J_n\}$, completion time $C(J_i, j)$ and idle time $I(J_1)$.

Note that the first job in the sequence is given an idle time, equal to the sum of the processing times for that job for the machines $2, \ldots, n$. However, strictly speaking, the idle time for the first job in the sequence is not defined as the machines in front of it are empty and the job will not have to wait to be processed. Unfortunately, this would have the effect of ensuring that the first job always has a low probability of being moved. As it is unlikely that the first job in the initial solution's sequence will be the correct choice, and so should eventually be moved, it is assigned an amount of idle time.

How can this information be used to predict which moves are most likely to lead to improvements in makespan? The answer is that first, as the aim is to minimise the time it takes to process all of the jobs, having jobs waiting to be processed is not desirable; second, it is reasonable to suppose that jobs that are left waiting for a long time are in an unsuitable position in the sequence, and therefore should be moved.

It should be noted that the idea of using idle times to improve the performance of heuristic methods for the flowshop sequencing problem is not new: [Ho & Chang 91] proposed the use of a similar heuristic in an approach that is essentially a deterministic hillclimber, and [Rajendran & Chaudhuri 91] used idle times in a constructive algorithm.

### 6.7.2 Implementing The Heuristic

A variant of the successful 'directed mutation' approach [Ross *et al.* 94] for SHC-based optimisers and EAs will be used here to exploit the move selection information. The evaluation function was adapted to return the idle times associated with each job, as described above. When the neighbourhood operator was then applied, the position(s) of the move operator was decided on the basis of the idle time of the job at that position, by using one of the selection methods below:

- **Tournament Selection** is taken from the EA literature [Brindle 81, Goldberg & Deb 91]. A fixed number, the *Monte Carlo size*, of jobs are randomly selected, and the job with the highest idle time associated with it is chosen.
- **Marriage Selection** is a variant of tournament selection designed to soften its inherently high selection [Ross 96]. Possible jobs are picked at random until either one is found that is has a higher idle time than the first, or a fixed number (the *Monte Carlo* size) of possible moves have been selected, in which case the job with the highest idle time associated with it is chosen.

In this study, the effect of the move selection heuristic, and how the selection is carried out will be investigated. The tunable parameters were set by an exhaustive search of both the selection methods over the set of Monte Carlo sizes in the range 2 though 10.

### 6.7.3 Stochastic Hillclimbing Results

The results for stochastic hillclimbing are given in Tables 6.43 and 6.44. For all of the tables presented in the main body of this report, the mean and standard deviations of the makespan are given, with the selection type and pressure found to be most effective given in square brackets. The form of the information about the selection method is as follows: the capital letter represents the selection method used (M for marriage, T for tournament), and the number gives the Monte Carlo size that was found to work best. Terms such as "Heuristic & Random" refer to how the first and second jobs for the operator are selected. Where such choices would be equivalent (i.e. in the case of the swap operator, due to its symmetry), only one of them is given.

| Problem | Undirected | Heuristic & Random | Heuristic & Heuristic |
|---|---|---|---|
| 20x5 | 1296.26 (3.63) | **1291.68 (7.52)** [T9] | **1294.58 (5.68)** [M4] |
| 20x10 | 1633.32 (21.73) | 1629.32 (21.15) [M5] | *1642.66 (27.37)* [T2] |
| 20x20 | 2373.16 (26.94) | 2367.80 (29.29) [M3] | 2372.52 (23.68) [T2] |
| 50x5 | 2742.82 (9.14) | **2728.90 (4.10)** [T9] | **2732.56 (6.62)** [T4] |
| 50x10 | 3141.72 (24.76) | **3126.72 (19.98)** [M8] | 3133.68 (25.85) [M3] |
| 50x20 | 4049.34 (31.64) | **4037.62 (26.40)** [T3] | 4046.38 (25.83) [T2] |
| 100x5 | 5510.62 (12.91) | **5500.08 (11.75)** [T8] | **5495.80 (4.28)** [T8] |
| 100x10 | 5927.38 (37.54) | **5893.12 (35.40)** [T9] | **5907.42 (33.17)** [M6] |
| 100x20 | 6622.44 (44.06) | **6588.62 (35.34)** [M6] | **6605.48** (32.09) [M4] |
| 200x10 | 11029.90 (35.02) | **10995.12 (33.04)** [T9] | **10992.40 (28.64)** [M8] |
| 200x20 | 11826.10 (49.64) | **11764.14 (59.65)** [T8] | **11780.46 (54.43)** [M6] |

Table 6.43: Stochastic Hillclimbing Results — Swap Neighbourhood

| Problem | Undirected | Heuristic & Random | Random & Heuristic | Heuristic + Heuristic |
|---|---|---|---|---|
| 20x5 | 1293.60 (7.26) | 1292.58 (7.81) [M4] | **1284.00 (10.32)** [T9] | **1288.52 (9.27)** [M3] |
| 20x10 | 1618.92 (13.13) | 1615.80 (10.62) [T2] | 1615.12 (16.12) [T2] | 1617.34 (16.64) [T2] |
| 20x20 | 2358.88 (33.10) | 2361.28 (25.90) [M3] | 2352.30 (26.25) [T2] | 2364.22 (27.15) [T2] |
| 50x5 | 2734.46 (6.76) | **2731.98 (6.93)** [T8] | **2730.48 (5.15)** [M5] | **2730.92 (6.47)** [T4] |
| 50x10 | 3121.10 (24.73) | 3125.86 (19.89) [M8] | **3107.96 (21.59)** [T3] | 3117.54 (18.77) [M3] |
| 50x20 | 4021.62 (22.52) | 4019.64 (23.58) [M5] | **4011.98 (25.19)** [M3] | 4018.68 (21.74) [T2] |
| 100x5 | 5506.56 (14.14) | **5498.70 (8.53)** [M5] | **5493.24 (0.65)** [T9] | **5493.98 (1.50)** [T8] |
| 100x10 | 5884.80 (30.06) | *5895.82 (35.19)* [T2] | **5842.00 (20.49)** [T8] | **5874.08 (32.51)** [M6] |
| 100x20 | 6581.78 (38.13) | 6582.32 (36.89) [M6] | **6542.88 (36.09)** [T5] | **6556.70 (38.44)** [T2] |
| 200x10 | 11020.20 (31.61) | **11007.94 (35.52)** [M5] | **10977.52 (38.20)** [T9] | **10982.40 (23.85)** [T6] |
| 200x20 | 11797.80 (50.06) | **11770.14 (51.45)** [M3] | **11669.52 (48.67)** [T8] | **11715.88 (53.82)** [M5] |

Table 6.44: Summary of Stochastic Hillclimbing Results — Shift Neighbourhood

Where the results are highlighted in **bold**, this means that the directed result was significantly better than the undirected result. Results in *italics* indicate that the result in question performed significantly worse. The following clear differences were observed.

In general, the shift neighbourhood gave better quality solutions than the swap neighbourhood, regardless of whether the neighbourhood was directed or not (in the case of Random & Heuristic, this was significant for 7 cases). Also, the heuristic was able to produce better quality solutions, however the performance appeared to depend on a number of factors.

The first of these factors was that the heuristic had a much more beneficial effect upon the problems with a large number of jobs — this was not particularly surprising as it could be argued that the smaller problems were not sufficiently difficult for the heuristic to make any impact. Also, the selection pressures that have to be used appear to be quite high — higher than would be used for an EA population for example. This indicates that the use of 'weaker' selection methods, such as the standard implementations of fitness-proportionate and rank-based selection should be avoided.

A final observation was that the choice of how the second job to be moved should be selected depends upon the neighbourhood used. The best results for a swap operator were obtained by selecting the second job at random, whereas for a shift operator, selecting the second job in the same way as the first was found to give the best results.

### 6.7.4 Simulated Annealing Results

The results for simulated annealing are given in Tables 6.45 and 6.46. The results obtained were similar in nature to those obtained for SHC, though simulated annealing did improve solution quality somewhat.

| Problem | Undirected | Heuristic & Random | Heuristic & Heuristic |
|---|---|---|---|
| 20x5 | 1296.00 (4.30) | **1293.84 (6.54)** [T6] | 1294.98 (5.29) [T3] |
| 20x10 | 1619.72 (14.73) | 1619.52 (17.07) [T6] | *1626.76 (15.30)* [T2] |
| 20x20 | 2357.36 (24.21) | 2355.22 (22.79) [T2] | 2359.44 (21.21) [T2] |
| 50x5 | 2741.14 (10.02) | **2730.84 (6.04)** [T9] | **2735.64 (7.14)** [T4] |
| 50x10 | 3135.34 (21.37) | **3122.84 (22.93)** [T4] | 3131.44 (20.66) [T2] |
| 50x20 | 4044.52 (27.69) | **4027.76 (24.04)** [T5] | **4030.72 (25.49)** [T2] |
| 100x5 | 5516.12 (13.28) | **5500.32 (10.63)** [T9] | **5498.40 (8.56)** [T9] |
| 100x10 | 5939.50 (41.34) | **5880.32 (24.12)** [T8] | **5902.92 (24.74)** [M4] |
| 100x20 | 6606.50 (31.98) | **6582.76 (31.96)** [T4] | **6586.84 (28.20)** [T2] |
| 200x10 | 11035.90 (41.03) | **10992.02 (26.15)** [T9] | **10997.50 (22.37)** [T6] |
| 200x20 | 11797.40 (52.15) | **11755.82 (57.91)** [T5] | **11753.02 (56.75)** [M3] |

Table 6.45: Simulated Annealing Results — Swap Neighbourhood

| Problem | Undirected | Heuristic & Random | Random & Heuristic | Heuristic & Heuristic |
|---|---|---|---|---|
| 20x5 | 1295.36 (5.19) | 1294.62 (5.57) [T6] | **1280.34 (5.23)** [T9] | **1291.34 (7.85)** [M2] |
| 20x10 | 1609.08 (10.86) | 1611.32 (10.22) [M9] | 1607.86 (11.04) [M3] | 1611.12 (10.72) [T2] |
| 20x20 | 2340.68 (22.56) | 2343.24 (20.03) [T2] | 2338.96 (23.32) [T2] | 2346.84 (25.16) [T2] |
| 50x5 | 2737.70 (7.63) | **2734.34 (7.96)** [T2] | **2729.82 (5.63)** [T6] | **2731.90 (6.33)** [M5] |
| 50x10 | 3116.48 (23.13) | 3115.96 (21.62) [M3] | **3103.86 (22.42)** [M6] | 3110.80 (20.40) [M3] |
| 50x20 | 4007.50 (23.25) | 4008.46 (24.52) [T5] | 4003.22 (19.19) [M5] | 4008.78 (18.85) [T2] |
| 100x5 | 5506.14 (14.65) | **5499.00 (8.04)** [T9] | **5493.76 (0.97)** [T6] | **5494.12 (1.29)** [T4] |
| 100x10 | 5888.60 (34.58) | 5895.82 (31.93) [T2] | **5843.04 (27.54)** [M8] | **5869.80 (27.92)** [T2] |
| 100x20 | 6562.66 (36.39) | 6554.50 (27.33) [T2] | **6526.16 (37.05)** [T4] | **6536.58 (32.03)** [M9] |
| 200x10 | 11023.30 (38.10) | 11012.28 (37.97) [T9] | **10977.42 (31.08)** [T9] | **10984.04 (23.99)** [T9] |
| 200x20 | 11750.30 (51.57) | **11721.22 (51.10)** [M4] | **11644.56 (47.62)** [T9] | **11674.68 (46.54)** [M8] |

Table 6.46: Summary of Simulated Annealing Results — Shift Neighbourhood

### 6.7.5 Threshold Accepting Results

The results obtained for threshold accepting were similar in nature to those obtained for SHC, though threshold accepting did improve solution quality somewhat. This is shown by examination of Tables 6.47 and 6.48.

| Problem | Undirected | Heuristic & Random | Heuristic & Heuristic |
|---|---|---|---|
| 20x5 | 1296.28 (3.08) | **1292.96 (6.67)** [T8] | 1295.74 (3.77) |
| 20x10 | 1620.30 (16.21) | 1618.08 (11.96) [T3] | 1625.48 (16.74) [T2] |
| 20x20 | 2358.22 (22.21) | **2348.08 (20.54)** [M5] | 2359.16 (24.07) [T2] |
| 50x5 | 2739.02 (9.35) | **2731.42 (8.19)** [T9] | **2735.80 (8.09)** [T2] |
| 50x10 | 3128.22 (28.89) | 3120.84 (19.60) [M5] | 3126.46 (19.96) [T2] |
| 50x20 | 4031.50 (23.21) | 4027.04 (24.48) [T3] | 4029.38 (20.40) [T2] |
| 100x5 | 5517.66 (15.28) | **5503.96 (13.60)** [M5] | **5498.46 (8.54)** [T9] |
| 100x10 | 5921.44 (31.73) | **5881.18 (30.99)** [T9] | **5902.26 (28.12)** [M8] |
| 100x20 | 6600.00 (34.02) | **6577.48 (34.64)** [T2] | **6583.22 (33.81)** [T2] |
| 200x10 | 11027.60 (34.44) | **10990.90 (23.74)** [T8] | **11001.14 (25.87)** [T4] |
| 200x20 | 11798.70 (44.20) | **11738.06 (48.00)** [T8] | **11745.58 (43.21)** [T2] |

Table 6.47: Threshold Accepting Results — Swap Neighbourhood

| Problem | Undirected | Heuristic & Random | Random & Heuristic | Heuristic & Heuristic |
|---|---|---|---|---|
| 20x5 | 1295.30 (5.09) | 1295.40 (4.63) [M6] | **1282.10 (5.18)** [T9] | **1293.02 (6.84)** [T3] |
| 20x10 | 1604.64 (10.74) | 1606.98 (11.66) [T2] | 1604.82 (11.10) [T2] | *1609.74 (12.33)* [T2] |
| 20x20 | 2341.72 (20.15) | 2338.74 (20.17) [T2] | 2335.14 (21.11) [T2] | 2338.76 (20.10) [T2] |
| 50x5 | 2735.90 (7.23) | 2734.44 (9.17) [T5] | **2728.98 (4.56)** [T9] | **2731.68 (6.14)** [M5] |
| 50x10 | 3115.56 (20.06) | 3116.38 (21.57) [M9] | **3104.24 (20.43)** [T8] | 3110.48 (19.16) [M3] |
| 50x20 | 4007.74 (20.17) | 4003.50 (20.29) [M2] | **3998.86 (19.19)** [T4] | **4000.38 (19.69)** [T7] |
| 100x5 | 5507.98 (13.64) | **5499.20 (7.92)** [T9] | **5494.38 (2.75)** [T8] | **5494.68 (1.71)** [T6] |
| 100x10 | 5889.48 (32.69) | 5897.74 (26.39) [M8] | **5842.58 (21.70)** [T5] | **5872.64 (31.10)** [T3] |
| 100x20 | 6552.14 (32.45) | 6549.18 (34.28) [T2] | **6519.38 (27.50)** [T6] | **6530.64 (33.66)** [T2] |
| 200x10 | 11017.40 (33.09) | 11010.24 (30.76) [T8] | **10967.46 (20.10)** [T9] | **10985.76 (26.61)** [T9] |
| 200x20 | 11736.00 (46.31) | **11703.50 (43.93)** [M3] | **11631.02 (36.63)** [T8] | **11661.32 (41.49)** [M4] |

Table 6.48: Summary of Threshold Accepting Results — Shift Neighbourhood

### 6.7.6 Record-to-Record Travel Results

The results for record-to-record travel are given in Tables 6.49 and 6.50. Again, the results obtained were similar in nature to those obtained for SHC, though RTRT did improve solution quality somewhat.

### 6.7.7 Evolutionary Algorithm Results

The results for the evolutionary algorithm are given in Tables 6.51, 6.53, 6.52, and 6.54.

An examination of the results shows that the relative performance of the directed EA with respect to its operators was unchanged from the undirected EA. However, unlike the other optimisation techniques, the evolutionary algorithm did not show significant improvements

| Problem | Undirected | Heuristic & Random | Heuristic & Heuristic |
|---|---|---|---|
| 20x5 | 1296.26 (3.63) | **1293.48 (6.57)** [T5] | **1294.72 (5.46)** [M3] |
| 20x10 | 1619.02 (14.18) | 1619.18 (14.58) [M5] | *1632.74 (21.15)* [M4] |
| 20x20 | 2355.62 (22.71) | 2358.20 (24.28) [T2] | *2373.82 (24.46)* [T2] |
| 50x5 | 2737.66 (11.22) | **2733.04 (7.89)** [M9] | **2732.94 (6.74)** [T3] |
| 50x10 | 3138.70 (22.01) | **3123.00 (19.41)** [M4] | 3132.46 (22.27) [T2] |
| 50x20 | 4044.50 (26.21) | **4030.00 (26.41)** [M5] | **4032.06 (16.51)** [T2] |
| 100x5 | 5516.22 (15.10) | **5500.10 (10.93)** [T9] | **5497.38 (7.09)** [T8] |
| 100x10 | 5917.56 (31.68) | **5883.72 (31.68)** [T8] | **5906.66 (27.92)** [M9] |
| 100x20 | 6608.70 (32.57) | **6582.32 (33.18)** [M8] | **6589.02 (28.95)** [M8] |
| 200x10 | 11033.50 (37.67) | **10993.44 (25.49)** [T9] | **10999.36 (29.19)** [M6] |
| 200x20 | 11819.30 (37.84) | **11761.24 (50.38)** [M9] | **11764.22 (48.41)** [M4] |

Table 6.49: Results for Record-to-Record Travel — Swap Neighbourhood

| Problem | Undirected | Heuristic & Random | Random & Heuristic | Heuristic & Heuristic |
|---|---|---|---|---|
| 20x5 | 1294.14 (6.18) | 1294.84 (5.68) [M8] | **1281.32 (7.38)** [T8] | 1293.42 (6.14) [M9] |
| 20x10 | 1609.44 (9.36) | 1609.58 (14.52) [T2] | 1608.96 (12.39) [M4] | *1613.12 (10.58)* [T2] |
| 20x20 | 2342.06 (23.12) | 2349.18 (24.86) [M3] | 2341.70 (22.66) [T2] | 2349.62 (25.29) [T2] |
| 50x5 | 2734.48 (8.22) | **2731.46 (7.26)** [T4] | **2729.50 (5.15)** [T9] | **2730.82 (6.02)** [T4] |
| 50x10 | 3118.86 (22.65) | 3114.46 (23.30) [T2] | **3106.14 (23.40)** [T3] | 3113.06 (17.91) [T2] |
| 50x20 | 4019.06 (22.79) | **4008.44 (18.45)** [T2] | **4005.30 (19.72)** [M5] | **4010.48 (21.58)** [M4] |
| 100x5 | 5509.24 (15.00) | **5499.40 (8.58)** [T8] | **5493.82 (1.83)** [T9] | **5495.34 (4.16)** [T9] |
| 100x10 | 5889.70 (33.16) | *5901.92 (31.62)* [T2] | **5838.00 (24.97)** [T9] | **5873.42 (24.88)** [M9] |
| 100x20 | 6567.06 (37.40) | 6568.18 (34.04) [M4] | **6533.82 (27.74)** [M5] | **6540.46 (28.62)** [M3] |
| 200x10 | 11019.90 (39.49) | **11006.70 (28.39)** [T3] | **10977.66 (25.76)** [T8] | **10984.26 (23.36)** [T8] |
| 200x20 | 11752.80 (55.88) | 11741.34 (48.07) [M3] | **11647.04 (37.13)** [T9] | **11691.72 (41.39)** [M3] |

Table 6.50: Summary of Results for Record-to-Record Travel — Shift Neighbourhood

in performance when the neighbourhood was directed, except for the largest problems. Why should this be?

Examination of the results showed again that the changes in makespan effected by the directed mutation operator for the EA were of roughly the same size as obtained for the other optimisers considered here. However the standard deviation of the EA results are much higher, which means that the improvements in performance cannot be said to be statistically significant.

### 6.7.8 Summary and Additional Remarks

First, like the experiments performed earlier, the shift neighbourhood consistently outperformed the swap neighbourhood, and this was irrespective of whether the idle-time heuristic was being used. This supports the design heuristic (1d) that suggests that the landscape should be decided upon before move selection. As was the case for the heuristic initialisation knowledge source, if the choice of landscape was to be affected by the different knowledge source options then the ability of design heuristic 1d to effectively order experimentation would be reduced (see 4.7.2).

| Problem | Undirected | Heuristic & Random | Heuristic & Heuristic |
|---|---|---|---|
| 20x5 | 1268.76 (87.63) | 1263.42 (90.56) [M5] | 1267.66 (89.58) [M3] |
| 20x10 | 1590.32 (80.20) | 1588.32 (81.31) [M7] | 1592.44 (80.14) [T2] |
| 20x20 | 2284.18 (66.64) | 2284.58 (73.89) [M6] | 2287.56 (71.21) [T2] |
| 50x5 | 2814.04 (129.00) | 2805.04 (132.56) [T9] | 2809.58 (129.22) [T2] |
| 50x10 | 3174.40 (101.20) | 3162.00 (98.97) [T8] | 3162.66 (104.48) [M9] |
| 50x20 | 4033.84 (88.94) | 4021.42 (82.27) [M5] | 4028.04 (86.97) [M4] |
| 100x5 | 5401.38 (186.55) | 5379.84 (190.89) [T9] | 5388.70 (190.39) [T3] |
| 100x10 | 5844.22 (148.34) | 5817.38 (145.00) [T5] | 5826.96 (145.02) [T3] |
| 100x20 | 6828.34 (81.23) | 6808.98 (92.08) [T3] | 6818.50 (98.01) [M5] |
| 200x10 | 11073.30 (208.54) | 11024.76 (217.44) [T8] | 11033.40 (216.75) [T3] |
| 200x20 | 12243.10 (124.37) | 12207.40 (131.87) [T3] | **12197.06 (128.21)** [T9] |

Table 6.51: Evolutionary Algorithm Results — Swap and PMX Crossover

| Problem | Undirected | Heuristic & Random | Random & Heuristic | Heuristic & Heuristic |
|---|---|---|---|---|
| 20x5 | 1263.06 (90.78) | 1262.94 (91.54) [M4] | 1260.76 (92.37) [T6] | 1262.38 (90.82) [T2] |
| 20x10 | 1576.00 (78.94) | 1580.04 (79.90) [M4] | 1575.80 (79.62) [T3] | 1584.72 (77.28) [M3] |
| 20x20 | 2271.08 (69.95) | 2272.94 (74.36) [M4] | 2270.36 (69.92) [T2] | 2275.16 (73.77) [T7] |
| 50x5 | 2809.12 (130.13) | 2810.02 (129.63) [T4] | 2798.82 (133.38) [T8] | 2803.08 (131.50) [T2] |
| 50x10 | 3165.16 (103.64) | 3161.16 (104.29) [M4] | 3145.20 (99.19) [T8] | 3155.14 (101.41) [M6] |
| 50x20 | 4024.14 (82.78) | 4015.04 (86.37) [T2] | 4009.60 (85.66) [T5] | 4009.34 (86.06) [M8] |
| 100x5 | 5390.96 (185.53) | 5394.50 (186.07) [T2] | 5372.00 (187.50) [T9] | 5378.86 (190.34) [T3] |
| 100x10 | 5831.74 (144.03) | 5832.36 (149.05) [M5] | 5800.58 (145.83) [T6] | 5813.88 (148.60) [T3] |
| 100x20 | 6836.50 (86.46) | 6817.50 (98.39) [M5] | **6801.18 (90.38)** [T8] | **6804.46 (102.65)** [T2] |
| 200x10 | 11072.90 (216.02) | 11053.84 (219.50) [T8] | **11000.12 (209.69)** [T9] | 11030.34 (223.48) [T9] |
| 200x20 | 12267.60 (133.43) | 12252.32 (115.24) [T3] | **12227.52 (113.99)** [T8] | **12222.56 (119.92)** [M9] |

Table 6.52: Evolutionary Algorithm Results — Shift and Modified PMX Crossover

In the context of directing the neighbourhood, it was found that, for the swap neighbourhood, that selecting both jobs according to the heuristic was not as effective. This is presumably because the former method over-exploits the idle-time information. It is important to note that using idle-time to predict improving moves is only a rule-of-thumb, therefore it is prudent to 'hedge your bets' somewhat.

The results obtained using the shift neighbourhood were somewhat different — it appears that selecting the second job (that which defines the block to be shifted) is more important that selecting the first job (which defines which job is moved out of the may of the shifted block). This is probably connected to the reason why the shift operator is the operator of choice. If the movement of blocks is what is important, then how the block as a whole interacts with the other jobs will be important. As the idle time of the second selected job effectively measures how one end of the block 'fits in' with the rest of the schedule, it can be seen why it results in an effective measure for how blocks should be shifted.

On the basis of the results obtained here, the belief that moving jobs that have high idle times would lead to improved solutions appears to be valid. In addition, the method by which this belief is implemented does appear to have an effect, along with the choice of neigh-

| Problem | Undirected | Heuristic & Random | Heuristic & Heuristic |
|---|---|---|---|
| 20x5 | 1266.00 (88.95) | 1262.40 (90.95) [M4] | 1264.28 (90.63) [T2] |
| 20x10 | 1591.92 (85.28) | 1585.20 (78.65) [T6] | 1589.32 (83.12) [T2] |
| 20x20 | 2276.54 (73.91) | 2277.58 (74.55) [M4] | 2281.66 (71.04) [M3] |
| 50x5 | 2810.78 (131.24) | 2803.10 (133.16) [T4] | 2807.00 (130.95) [T2] |
| 50x10 | 3173.72 (104.94) | 3149.20 (104.64) [T6] | 3156.12 (101.39) [T3] |
| 50x20 | 4032.34 (89.43) | 4016.58 (90.23) [T6] | 4016.50 (93.76) [M3] |
| 100x5 | 5398.54 (190.18) | 5379.56 (187.17) [T9] | 5386.64 (189.87) [T4] |
| 100x10 | 5841.48 (141.79) | 5806.36 (141.00) [T7] | 5823.38 (151.62) [T5] |
| 100x20 | 6847.10 (91.11) | 6808.96 (103.70) [T6] | 6810.02 (92.94) [M4] |
| 200x10 | 11068.22 (204.37) | 11022.36 (209.19) [T9] | 11035.22 (215.51) [T4] |
| 200x20 | 12275.48 (124.43) | 12210.14 (132.29) [T4] | 12226.46 (133.85) [M4] |

Table 6.53: Evolutionary Algorithm Results — Swap and PPX Crossover

| Problem | Undirected | Heuristic & Random | Random & Heuristic | Heuristic & Heuristic |
|---|---|---|---|---|
| 20x5 | 1262.12 (93.06) | 1262.56 (90.45) [M3] | 1260.30 (91.29) [T3] | 1261.62 (91.60) [M3] |
| 20x10 | 1579.58 (80.38) | 1579.66 (77.52) [M7] | 1577.22 (76.93) [M3] | 1578.80 (78.05) [T2] |
| 20x20 | 2270.02 (70.75) | 2269.64 (72.23) [M6] | 2266.62 (73.52) [M9] | 2269.90 (71.66) [T2] |
| 50x5 | 2806.96 (128.70) | 2806.78 (132.93) [T2] | 2798.66 (132.24) [T6] | 2803.22 (131.08) [T2] |
| 50x10 | 3158.54 (109.64) | 3157.28 (101.07) [M5] | 3139.38 (102.69) [T6] | 3151.86 (101.16) [M3] |
| 50x20 | 4022.26 (82.40) | 4013.90 (88.71) [T3] | 3999.46 (82.70) [M6] | 4006.70 (90.12) [M6] |
| 100x5 | 5390.86 (190.86) | 5395.18 (188.47) [M4] | 5370.06 (187.87) [T7] | 5380.92 (186.85) [T6] |
| 100x10 | 5848.52 (150.18) | 5827.28 (148.80) [T2] | 5792.54 (147.10) [T7] | 5814.26 (147.82) [M3] |
| 100x20 | 6836.06 (98.81) | 6825.92 (105.78) [M5] | **6783.78 (90.39)** [T6] | **6796.98 (102.94)** [M6] |
| 200x10 | 11077.52 (221.53) | 11057.74 (213.48) [M4] | **10998.04 (213.61)** [T9] | 11024.00 (214.44) [M6] |
| 200x20 | 12283.90 (134.45) | 12256.26 (131.39) [M7] | **12180.82 (140.44)** [T7] | **12229.36 (115.27)** [M5] |

Table 6.54: Evolutionary Algorithm Results — Shift and PPX Crossover

bourhood operator. More importantly, other techniques (such as the constructive heuristic by [Rajendran & Chaudhuri 91]) have made use of idle time. This suggests that there may be some mileage in devising methods for extracting and 'sharing' useful knowledge from other domain-specific methods. This idea can be transferred to other similar problems, such as job-shop scheduling. Also, move selection heuristics can easily be formulated to deal with other optimisation criteria, for example, tardiness.

## 6.8 Transfer of Move Selection to a Candidate List Strategy

The examination of the transferability across optimisers of the move preference knowledge source has so far been restricted to SHC-based optimisers and evolutionary algorithms. This is because the 'directed mutation' approach is only applicable to those forms of neighbourhood search. Therefore to examine whether the above idle-time heuristic can be usefully used to direct search for FAHC and SAHC-based optimisers (and to evaluate the validity of the move selection design heuristic for these hillclimbing classes) it will be investigated whether the idle-time heuristic can be used as the basis of a useful candidate list strategy.

To this end, the candidate list strategy that is to be used will be first described, along with its implementation. The results of the experimental study will then be presented on an optimiser by optimiser basis before finishing with a summary and discussion of the results as a whole. For the reasons described earlier (Section 6.6), the comparative study that follows will use the same experimental methodology as for directed mutation experiments. Additional information on the optimiser set-up and implementation will also follow.

### 6.8.1 The Directed Candidate List Strategy

A common way to explore a neighbourhood is generate a candidate list of the possible moves and then to evaluate each one until an acceptance criterion is met. However, in the case of steepest-ascent, and to a lesser extent, first-ascent hillclimbers a problem arises in that the neighbourhoods to be examined can become quite large — in the case of the problem considered here, the size increases with $O(n^2)$.

A response to this, as noted in 2.4, is to use a **candidate list strategy** which selects a *subset* of the available moves (ie. a restricted neighbourhood is produced). The simplest of these strategies is to simply pick this subset at random [Reeves 93a] — a **random subset** candidate list strategy. Also, candidate lists have been used with a rule that excludes moves that are known to be non-improving, e.g. [Nowicki & Smutnicki 96], though such a rule will in most cases be heuristic in nature. A review of more sophisticated strategies is given in [Glover & Laguna 97], one of which is a **successive filter** strategy. This can be applied to moves that can be broken up into component parts. Instead of examining all of the possible combinations, a subset of the 'best' components is taken and only the combinations of these are examined.

**Algorithm 6** DIRECTED CANDIDATE LIST GENERATION

Let $M_u = \{m_1, m_2, \dots, m_n\}$; *// The unordered list of available moves*<br>
Let $M_c = \emptyset$; *// The directed candidate list (currently empty)*<br>
$M_o$ = ORDER$(M)$; *// Order the moves according to* ORDER() *to produce the list* $M_o$<br>
**repeat**<br>
    $m_s$ = SELECT$(M_o)$; *// Use* SELECT() *to obtain a move* $m_s$<br>
    $M_c$ = APPEND$(M_c, m_s)$; *// Use* APPEND() *to add* $m_s$ *to the end of* $M_c$<br>
    $M_o = M_o \setminus m_s$; *// Remove* $m_s$ *from the list* $M_o$<br>
**until** $M_o = \emptyset \vee$ (LENGTH$(M_c) < S$); *// Continue until the candidate list has been fully constructed*<br>
**return** $M_c$ as the candidate list;

This investigation introduces a **directed candidate list strategy** which is an analogue of the directed mutation approach, as well as incorporating features of the random subset and successive filter strategies. Algorithm 6 describes this approach. An unordered list, $M_u$, of the possible moves is ranked according to the heuristic function ORDER() to produce an ordered list of moves $M_o$. The candidate list $M_c$ is then built from $M_o$ by repeatedly calling SELECT() until a candidate list of the required size, $S$, is obtained.

This approach preserves the idea from directed mutation that moves that are more likely to be improving should be tried first — which should benefit first-ascent optimisers. Also, in the case of steepest-ascent optimisers which look at all the available moves, this strategy also allows, in principle, the construction of a 'concentrated' subset of moves to be which hopefully will contain the improving moves. In either case, given that the heuristic function ORDER() correctly predicts the improving moves, then the evaluation of unnecessary non-improving moves is avoided and search efficiency is improved as a result.

### 6.8.2 The ORDER() and SELECT() Functions

In fact, it is sufficient to define ORDER() in terms of a function, called PREFER_MOVE()[4] that, given two moves, returns which move is more likely to be improving. The partial ordering given by PREFER_MOVE() can then be used in conjunction with a sorting algorithm to implement ORDER(). In addition, ORDER() has to be formulated for this problem so as to take into account that the neighbourhood operators, $move(i, j)$, for this problem require two jobs to be selected. Therefore as for the directed mutation experiments, PREFER_MOVE() can compare moves either on the basis of the idle times of the first job of each move (we call this Heuristic & Random), the second jobs (Random & Heuristic — for a swap neighbourhood this is equivalent to Heuristic & Random), both jobs (Heuristic & Heuristic), or neither (Undirected). Algorithm 7 shows PREFER_MOVE() when the ordering of moves is based upon the idle times the first job in each move (Heuristic+Random). Similar versions of PREFER_MOVE() can be devised for the other cases.

In order to conform with the work described in [Reeves 93a], when a candidate list is created all cases are ordered such that all of the moves with the same job $i$ in $move(i, j)$ are placed together in a block; i.e. $\{move(a, \_), \ldots, move(b, \_), \ldots, move(c, \_), \ldots\}$. When a subset is

[4] The observant reader will note that this is a symbol-level implementation of the knowledge level description, $movepref(m_i, m_j, \Psi)$, of the move selection knowledge source.

**Algorithm 7** PREFER_MOVE() FOR IDLE TIMES (HEURISTIC & RANDOM)

**Require:** $move(J_1, J_2)$ and $move(J_3, J_4)$ // *The moves being compared*

**if** $I(J_1) < I(J_3)$ **then**
    **return** $J_1$;
**else if** $I(J_1) > I(J_3)$ **then**
    **return** $J_3$;
**else**
    **return** $NULL$
**end if**

selected, the parameter $p$ used refers to the number jobs with the same first job $i$ contained in the candidate list (therefore the number of moves in the candidate list is $S = p \times n$). This also has the advantage that Mohr's procedure [Taillard 90] can then be used to efficiently evaluate a shift neighbourhood, if desired. The value of $p$, the number of moves associated with a given first job in the subset candidate list (random and directed), was set to 7 in line with [Reeves 93a].

The only exception to this was the case of Random & Heuristic for a steepest-ascent optimisers with a shift neighbourhood, as the above arrangement would be equivalent to a random subset candidate list strategy. Therefore Algorithm 6 was modified to return the first $p$ moves $\{move(i, \_), \ldots, move(i, \_)\}$ associated with each of the $n$ first jobs denoted by $i$ (thus again giving a candidate list of size $S = p \times n$).

In line with the directed mutation experiments, it was decided that the function SELECT() would be randomised. Therefore **tournament selection** [Brindle 81] and **marriage selection** [Ross 96], a variant of tournament selection designed to soften its inherently high selection pressure, were used as in 6.7.2 earlier.

Finally, the effect of the move selection heuristic, restricting the neighbourhood size, and how the selection is carried out (ie. the effect of the selection methods and their parameters) will be investigated. Therefore the cases of PREFER_MOVE() examined were Undirected, Heuristic & Random, and Random & Heuristic.

### 6.8.3 First-Ascent Optimiser Results

The results for FAHC are given in Tables 6.55, 6.56, and 6.57, with the corresponding results for FATS are in Tables 6.58 6.59, and 6.60. Again, for all of the tables presented here, the mean and standard deviations of the makespan are given, with the selection type and pressure found to be most effective given in square brackets. The reader can find a full listing of the experiments and the statistical analysis in Appendix C.

Where the results in the table are highlighted in **bold**, this means that the directed result was significantly better than its corresponding undirected result. Results in *italics* indicate that the result in question performed significantly worse. In addition, a comparison between the undirected results for the full and restricted neighbourhoods was made. The results of this comparison is indicated in the undirected column of the results for the restricted neighbourhood.

For both neighbourhoods, the reduction of the neighbourhood using a random subset candidate list strategy brought no significant benefits — it would appear that, compared to steepest-ascent optimisers, the problems associated with large neighbourhoods is not so pronounced.

On the other hand, examination of the results for the swap neighbourhood did show that directing the neighbourhood did produce significant improvements for the larger problems, irrespective of whether FAHC or FATS was used. Significant improvements, for both FAHC and FATS, were also observed for the larger problems when a shift neighbourhood was used. In addition, the Random & Heuristic case was observed to generally do better than the Heuristic & Random case (though the differences were not found to be significant). The only trend observed in the Monte Carlo size for the directed strategy was that larger sizes were preferred in the cases where directing the neighbourhood had a beneficial effect.

Finally, in all cases, the use of the basic recency-based tabu search mechanism did not improve performance above that of hillclimbing. This would suggest, at the timescale of the search used here, that local optima do not present much of a problem. That said, running the search for longer could well necessitate some mechanism to deal with local optima as it then becomes more likely that such optima will be found.

### 6.8.4 Steepest-Ascent Optimiser Results

The results for SAHC are given in Tables 6.61 and 6.62, with the results for SATS in Tables 6.63 and 6.64. Comparison with the results for the FAHC-based optimisers above shows that the performance of SAHC-based optimisers is, at best, comparable to that FAHC and more

| | Full Neighbourhood | | Restricted Neighbourhood | |
|---|---|---|---|---|
| Problem | Undirected | Random & Heuristic | Undirected | Random & Heuristic |
| 20x5 | 1300.00 (12.21) | 1298.12 (7.60) [M3] | **1296.02 (5.48)** | 1296.24 (3.72) [M9] |
| 20x10 | 1640.56 (22.28) | 1638.04 (23.35) [T3] | 1635.20 (23.68) | 1632.54 (21.79) [T2] |
| 20x20 | 2383.40 (27.21) | 2382.10 (24.57) [T2] | 2376.82 (25.54) | 2381.72 (28.20) [T2] |
| 50x5 | 2753.20 (15.95) | **2748.32 (12.13)** [T3] | **2747.06 (14.87)** | 2748.66 (9.55) [M9] |
| 50x10 | 3162.76 (27.62) | 3166.72 (30.98) [M6] | 3160.68 (28.52) | 3158.32 (27.28) [M6] |
| 50x20 | 4065.66 (29.83) | 4060.30 (32.46) [M3] | 4061.52 (31.12) | 4060.72 (28.97) [M3] |
| 100x5 | 5531.66 (22.18) | **5523.78 (16.94)** [T6] | 5538.88 (28.04) | **5525.74** (16.62) [T3] |
| 100x10 | 6038.72 (61.47) | **5975.92 (45.31)** [M4] | 6028.24 (61.80) | **5979.70 (53.86)** [T6] |
| 100x20 | 6735.62 (56.59) | **6682.82 (45.95)** [M4] | 6723.20 (47.00) | **6678.72 (40.52)** [M4] |
| 200x10 | 11164.10 (62.66) | **11116.92 (42.36)** [M9] | 11158.16 (53.63) | **11121.04 (46.53)** [M6] |
| 200x20 | 12120.30 (69.41) | **12000.80 (61.53)** [T4] | 12136.46 (94.41) | **12001.14 (65.40)** [M6] |

Table 6.55: First-Ascent Hillclimbing Results — Swap Neighbourhood

| Problem | Undirected | Random & Heuristic | Heuristic & Random |
|---|---|---|---|
| 20x5 | 1294.58 (6.34) | 1296.46 (5.26) [M7] | 1296.18 (5.03) [M3] |
| 20x10 | 1621.52 (12.05) | 1616.74 (16.97) [T7] | 1617.44 (15.57) [T2] |
| 20x20 | 2353.36 (27.41) | 2349.36 (25.54) [M6] | 2358.40 (29.95) [T2] |
| 50x5 | 2742.54 (11.95) | 2740.54 (11.14) [T9] | 2743.30 (11.32) [M6] |
| 50x10 | 3163.36 (25.51) | 3155.96 (29.73) [T5] | 3162.76 (28.26) [M4] |
| 50x20 | 4068.56 (34.94) | 4063.30 (26.55) [T2] | 4062.50 (29.10) [M6] |
| 100x5 | 5539.72 (32.44) | **5527.14 (24.63)** [T4] | **5525.66 (20.58)** [T2] |
| 100x10 | 6038.72 (61.47) | **6004.94 (51.13)** [T6] | 6033.44 (50.54) [T2] |
| 100x20 | 6806.38 (60.89) | **6768.70 (55.10)** [T2] | **6777.18 (54.40)** [M3] |
| 200x10 | 11295.90 (77.61) | **11260.30 (86.38)** [M9] | 11278.18 (88.79) [T2] |
| 200x20 | 12326.00 (89.20) | **12292.46 (99.26)** [M7] | 12296.24 (93.99) [M3] |

Table 6.56: First-Ascent Hillclimbing Results — Full Shift Neighbourhood

| Problem | Undirected | Random & Heuristic | Heuristic & Random |
|---|---|---|---|
| 20x5 | 1296.12 (5.18) | 1294.66 (10.24) [T3] | 1295.12 (6.21) [T2] |
| 20x10 | **1615.72 (13.15)** | 1613.82 (16.30) [M3] | 1619.08 (16.10) [T2] |
| 20x20 | 2354.96 (27.76) | 2348.02 (26.15) [T6] | 2360.38 (28.69) [T2] |
| 50x5 | 2742.00 (10.74) | 2742.16 (12.86) [M7] | 2744.24 (11.79) [T2] |
| 50x10 | 3155.94 (30.21) | 3155.98 (26.05) [M8] | 3163.18 (26.74) [T2] |
| 50x20 | 4067.96 (35.33) | 4059.40 (27.52) [M7] | 4065.64 (28.74) [M3] |
| 100x5 | 5539.48 (27.33) | 5532.64 (20.77) [T2] | **5528.98 (19.51)** [T3] |
| 100x10 | 6022.72 (46.63) | **6005.40 (47.43)** [M3] | 6031.54 (44.11) [M3] |
| 100x20 | 6812.38 (55.78) | **6772.52 (55.21)** [T5] | **6780.24 (63.72)** [M6] |
| 200x10 | 11313.74 (84.74) | **11267.70 (87.61)** [M6] | **11274.48 (81.88)** [M3] |
| 200x20 | 12327.20 (96.39) | 12300.90 (102.98) [M9] | 12297.16 (95.04) [M3] |

Table 6.57: First-Ascent Hillclimbing Results — Restricted Shift Neighbourhood

often than not worse for all but the smaller problem instances. Again, in all cases, the use of the basic recency-based tabu search mechanism did not improve performance above that of hillclimbing — presumably for the reasons outlined above.

For the swap neighbourhood, the random subset candidate strategy was found to produce significant improvements over using the full neighbourhood, and predictably this was most pronounced for the larger problem instances. However, the use of a directed candidate list strategy almost always did not lead to a improvement in makespan.

Finally, similar results were observed for the shift neighbourhood with respect to the effect

| | Full Neighbourhood | | Restricted Neighbourhood | |
|---|---|---|---|---|
| Problem | Undirected | Random & Heuristic | Undirected | Random & Heuristic |
| 20x5 | 1298.12 (6.79) | 1298.42 (9.59) [T9] | 1297.00 (5.37) | 1296.44 (5.86) [M7] |
| 20x10 | 1628.60 (19.03) | 1624.64 (21.36) [T2] | **1620.38 (13.44)** | 1618.44 (11.45) [T2] |
| 20x20 | 2358.02 (24.97) | 2364.06 (25.37) [T3] | 2351.56 (22.06) | *2366.42 (21.38)* [T2] |
| 50x5 | 2753.60 (15.59) | **2748.44 (8.68)** [T6] | **2747.22 (15.60)** | 2748.00 (12.46) [M6] |
| 50x10 | 3160.36 (26.66) | 3166.46 (34.01) [M3] | 3158.40 (31.55) | 3156.36 (25.31) [M4] |
| 50x20 | 4073.28 (33.06) | 4066.82 (31.93) [T3] | 4066.08 (36.83) | 4056.96 (28.30) [T3] |
| 100x5 | 5532.54 (21.19) | **5523.40 (17.41)** [T6] | 5538.72 (26.97) | **5526.86** (16.53) [T3] |
| 100x10 | 6039.56 (60.06) | **5977.44 (42.65)** [M4] | 6027.02 (63.53) | **5980.20 (52.06)** [T6] |
| 100x20 | 6730.40 (65.55) | **6684.28 (50.50)** [M7] | 6719.78 (47.37) | **6678.98 (42.69)** [M7] |
| 200x10 | 11162.90 (62.58) | **11116.92 (42.36)** [M9] | 11161.24 (50.75) | **11123.00** (45.51) [M6] |
| 200x20 | 12122.70 (70.10) | **12001.18 (67.88)** [T3] | 12136.40 (94.04) | **12000.06** (70.28) [M6] |

Table 6.58: First-Ascent Tabu Search Results — Swap Neighbourhood

| Problem | Undirected | Random & Heuristic | Heuristic & Random |
|---|---|---|---|
| 20x5 | 1295.30 (5.92) | 1296.30 (4.71) [T7] | 1295.38 (6.12) [M6] |
| 20x10 | 1613.94 (14.63) | 1613.96 (14.05) [M5] | 1613.52 (11.29) [M9] |
| 20x20 | 2341.00 (21.71) | 2337.52 (25.90) [T6] | *2350.26 (30.77)* [T2] |
| 50x5 | 2742.96 (12.16) | 2741.82 (13.14) [M3] | 2744.02 (8.91) [M4] |
| 50x10 | 3164.14 (28.40) | **3152.54 (29.80)** [M3] | 3162.60 (28.32) [M4] |
| 50x20 | 4063.28 (31.75) | 4060.94 (37.95) [T4] | 4063.40 (29.54) [M4] |
| 100x5 | 5537.74 (32.79) | **5527.08 (24.21)** [T4] | **5526.64 (21.18)** [T2] |
| 100x10 | 6024.54 (43.80) | **6003.46 (43.70)** [T5] | 6031.66 (52.81) [T2] |
| 100x20 | 6802.86 (58.70) | **6766.82 (55.16)** [T2] | **6779.24 (55.19)** [M3] |
| 200x10 | 11295.90 (77.61) | **11260.80 (87.05)** [M9] | 11275.62 (89.66) [T2] |
| 200x20 | 12328.60 (88.14) | **12292.68 (99.50)** [M7] | **12292.98 (95.65)** [M5] |

Table 6.59: First-Ascent Tabu Search Results — Full Shift Neighbourhood

| Problem | Undirected | Random & Heuristic | Heuristic & Random |
|---|---|---|---|
| 20x5 | 1294.98 (6.74) | 1294.28 (7.27) [M6] | 1295.02 (6.38) [T2] |
| 20x10 | 1612.96 (11.98) | 1609.74 (11.66) [T6] | 1613.88 (11.72) [M5] |
| 20x20 | 2341.40 (21.89) | 2338.54 (23.29) [M5] | *2354.14 (26.10)* [T2] |
| 50x5 | 2742.68 (11.93) | 2742.42 (10.25) [T2] | 2743.60 (10.70) [T2] |
| 50x10 | 3158.00 (29.56) | 3156.88 (28.34) [M7] | 3160.54 (27.52) [T2] |
| 50x20 | 4067.08 (32.74) | 4059.18 (24.85) [M7] | 4067.12 (28.76) [T2] |
| 100x5 | 5541.00 (29.45) | 5533.20 (20.90) [T2] | **5529.66 (19.51)** [T3] |
| 100x10 | 6024.50 (45.28) | **6004.76 (47.01)** [M3] | *6032.94 (56.70)* [T2] |
| 100x20 | 6807.14 (57.40) | **6771.38 (60.96)** [T9] | **6782.28 (52.35)** [M3] |
| 200x10 | 11312.80 (83.77) | **11267.06 (78.64)** [M4] | **11272.58 (86.51)** [T2] |
| 200x20 | 12322.40 (93.55) | 12295.24 (102.96) [M7] | 12296.20 (95.37) [M3] |

Table 6.60: First-Ascent Tabu Search Results — Restricted Shift Neighbourhood

of using the random subset candidate list strategy and the Heuristic & Random case of the directed candidate list strategy. However, significant improvements were obtained using the Random & Heuristic case of the directed candidate list strategy.

### 6.8.5 Summary and Additional Remarks

Overall, the approach used here has largely been a success, and it has been shown that the idle time heuristic can be put to use in a candidate list strategy for FAHC and SAHC-based optimisers. Furthermore, it appears that the trends observed were unaffected upon moving

| | Full Neighbourhood | Restricted Neighbourhood | |
|---|---|---|---|
| Problem | Undirected | Undirected | Heuristic & Random |
| 20x5 | 1299.52 (12.60) | 1297.40 (6.36) | 1299.20 (8.38) [T3] |
| 20x10 | 1645.22 (25.80) | 1638.60 (22.91) | 1643.82 (26.72) [T2] |
| 20x20 | 2389.76 (29.98) | 2381.04 (26.15) | 2380.06 (25.39) [T2] |
| 50x5 | 2775.08 (23.42) | **2752.34 (18.07)** | 2747.78 (12.53) [T2] |
| 50x10 | 3352.96 (40.90) | **3177.08 (25.84)** | *3190.82 (25.89)* [M3] |
| 50x20 | 4341.72 (54.18) | **4103.58 (38.87)** | 4111.48 (43.04) [T2] |
| 100x5 | 5858.14 (118.98) | **5561.60 (33.86)** | **5546.76 (21.58)** [M3] |
| 100x10 | 6650.20 (108.27) | **6111.28 (58.59)** | 6094.02 (47.54) [T3] |
| 100x20 | 7540.60 (139.51) | **6867.38 (49.68)** | 6882.92 (56.40) [T2] |
| 200x10 | 12134.90 (197.88) | **11416.10 (79.17)** | 11439.42 (87.63) [T2] |
| 200x20 | 13407.00 (172.98) | **12638.58 (107.43)** | 12653.06 (111.02) [T2] |

Table 6.61: Steepest-Ascent Hillclimbing Results — Swap Neighbourhood

| | Full Neighbourhood | Restricted Neighbourhood | | |
|---|---|---|---|---|
| Problem | Undirected | Undirected | Random & Heuristic | Heuristic & Random |
| 20x5 | 1299.20 (8.52) | **1296.46 (6.21)** | 1295.52 (5.08) [M3] | 1297.20 (5.03) [T2] |
| 20x10 | 1630.40 (17.16) | **1622.56 (15.02)** | 1623.52 (19.89) [M3] | 1620.42 (15.45) [T2] |
| 20x20 | 2381.42 (28.65) | 2373.52 (26.85) | **2359.62 (28.06)** [M9] | 2367.86 (31.00) [T2] |
| 50x5 | 2936.68 (103.90) | **2748.32 (11.05)** | 2748.76 (12.29) [T2] | 2748.70 (12.36) [T5] |
| 50x10 | 3573.30 (95.47) | **3233.04 (37.58)** | **3211.22 (28.43)** [T2] | 3225.56 (32.53) [T4] |
| 50x20 | 4576.70 (99.95) | **4162.46 (37.97)** | **4147.86 (47.42)** [T2] | 4159.82 (35.55) [M3] |
| 100x5 | 6036.56 (147.63) | **5668.22 (72.76)** | *5593.38 (51.77)* [T3] | 5653.12 (65.34) [T3] |
| 100x10 | 6837.74 (127.17) | **6311.96 (95.11)** | **6232.24 (78.77)** [M9] | 6300.08 (81.23) [T5] |
| 100x20 | 7715.66 (164.52) | **7170.78 (100.88)** | **7082.78 (86.63)** [M5] | 7158.34 (95.35) [M7] |
| 200x10 | 12195.60 (199.38) | **11854.92 (164.26)** | **11761.82 (158.95)** [T2] | 11845.32 (161.67) [T6] |
| 200x20 | 13465.90 (172.81) | **13106.76 (161.12)** | **13005.08 (154.08)** [T6] | 13084.58 (155.10) [M4] |

Table 6.62: Steepest-Ascent Hillclimbing Results — Shift Neighbourhood

from hillclimbing to tabu search (though the lack of effect of the tabu recency mechanism make this conclusion somewhat more uncertain than is the case for the previous investigations).

Interestingly enough, one interesting trend was found when the idle-time heuristic was used with the directed candidate list strategy — the superiority of Random & Heuristic over Heuristic & Random was also observed with respect to stochastic hillclimbing based optimisers in Section 6.7 earlier. This was thought to be because the second job, in effect, defines the block that is to be shifted and how it interacts with the rest of the schedule. Therefore it would appear that there is some scope in transferring such information *across* hillclimbing classes.

That said, although the heuristic and the new directed candidate list strategy worked well in most cases, it was not always successful when the directed mutation indicated that it would be (for example with steepest ascent optimisers with a swap neighbourhood). However, given that the directed mutation results indicate strongly that the idle-time move selection heuristic successfully identifies improving moves, then the reason for this lies either in the form of hillclimbing used, or in the candidate list strategy itself.

| Problem | Full Neighbourhood Undirected | Restricted Neighbourhood Undirected | Restricted Neighbourhood Heuristic & Random |
|---|---|---|---|
| 20x5 | 1302.42 (15.05) | **1296.92 (2.22)** | **1299.20 (8.38)** [T3] |
| 20x10 | 1641.10 (22.67) | **1625.28 (12.58)** | *1643.82 (26.72)* [T2] |
| 20x20 | 2370.96 (26.69) | **2358.66 (21.77)** | *2380.06 (25.39)* [T2] |
| 50x5 | 2776.60 (26.18) | **2752.08 (17.45)** | 2747.78 (12.53) [T2] |
| 50x10 | 3352.62 (40.50) | **3178.74 (26.90)** | *3190.82 (25.89)* [M3] |
| 50x20 | 4341.28 (53.78) | **4099.38 (36.30)** | 4111.48 (43.04) [T2] |
| 100x5 | 5858.14 (118.98) | **5561.60 (33.86)** | **5546.76 (21.58)** [M3] |
| 100x10 | 6650.20 (108.27) | **6111.26 (58.59)** | 6094.02 (47.54) [T2] |
| 100x20 | 7540.60 (139.51) | **6867.28 (49.72)** | 6882.92 (56.40) [T2] |
| 200x10 | 12134.90 (197.88) | **11416.06 (79.13)** | 11439.42 (87.63) [T2] |
| 200x20 | 13407.00 (172.98) | **12637.90 (106.33)** | 12653.06 (111.02) [T2] |

Table 6.63: Steepest-Ascent Tabu Search Results — Swap Neighbourhood

| Problem | Full Neighbourhood Undirected | Restricted Neighbourhood Undirected | Restricted Neighbourhood Random & Heuristic | Restricted Neighbourhood Heuristic & Random |
|---|---|---|---|---|
| 20x5 | 1300.14 (10.57) | **1297.22 (3.66)** | 1295.34 (9.19) [M5] | 1297.20 (5.03) [T2] |
| 20x10 | 1626.62 (15.85) | **1615.18 (12.19)** | 1618.42 (14.48) [T2] | *1620.42 (15.45)* [T2] |
| 20x20 | 2377.74 (29.81) | **2360.70 (25.37)** | **2344.44 (22.04)** [T2] | 2367.86 (31.00) [T2] |
| 50x5 | 2937.24 (103.48) | **2748.10 (11.16)** | 2749.16 (12.28) [T2] | 2748.70 (12.36) [T5] |
| 50x10 | 3573.32 (95.48) | **3231.30 (37.66)** | **3211.90 (28.30)** [T2] | 3225.56 (32.53) [T4] |
| 50x20 | 4576.68 (100.09) | **4163.62 (39.06)** | **4148.02 (47.71)** [T2] | 4159.82 (35.55) [M3] |
| 100x5 | 6036.56 (147.63) | **5668.70 (72.88)** | 5593.58 (51.50) [T3] | **5653.12 (65.34)** [T3] |
| 100x10 | 6837.74 (127.17) | **6311.96 (95.11)** | **6233.28 (79.84)** [M9] | 6300.08 (81.23) [T5] |
| 100x20 | 7715.66 (164.52) | **7171.58 (100.79)** | **7084.22 (85.34)** [M5] | 7158.34 (95.35) [M7] |
| 200x10 | 12195.60 (199.38) | **11854.92 (164.26)** | **11761.82 (158.95)** [T3] | 11845.32 (161.67) [T2] |
| 200x20 | 13465.90 (172.81) | **13106.76 (161.12)** | **13005.00 (154.05)** [T6] | 13086.62 (154.56) [M7] |

Table 6.64: Steepest-Ascent Tabu Search Results — Shift Neighbourhood

Finally, as the move direction design heuristic only strictly applies to optimisers of the same hillclimbing class, it should be noted that the above differences should not be taken as being contrary to that heuristic. However, as noted above, the fact that the effect of the idle-time heuristic is largely transferable across hillclimbing classes is encouraging, though some further work in identifying *why* the idle-time heuristic was not successfully exploited in these cases would be of some interest.

## 6.9 Discussion and Conclusions

This chapter has examined many of the design heuristics proposed in Chapters 3 and 4 in the context of a well-studied sequencing problem in operational research. After an overview and review of the problem was given, each of the design heuristics considered here were then investigated in turn. These design heuristics concerned themselves with the transferability of landscapes between optimisers of the same hillclimbing class and EAs, the design of recombination operators, heuristic initialisation, and move selection.

Overall, the results obtained for these investigations supported the design heuristics. The few exceptions that were found related to comparisons of hypotheses that were in fact quite similar, and so search control effects could affect the outcome (such as precedence crossover and the random keys representation). As such these exceptions do not constitute a strong rebuttal of the above design heuristics.

In addition, some other useful trends were noted. Of these, the most significant was the proposal and subsequent validation of the effectiveness of the idle-time heuristic and its implementation into a both a directed mutation and a candidate list strategy. As noted earlier (Section 6.8), this indicates that knowledge from other methods (such as the idle-time based constructive heuristic due to [Rajendran & Chaudhuri 91]) can also be usefully transferred to the design of neighbourhood search optimisers.

Attention will now be turned to the next case study that will investigate whether the methodology described in this thesis can be applied to a real-world problem.

# Chapter 7

# Case Study Two: An Emergency Resource Redistribution System for the Developing World

This chapter details the second of the case studies considered here — the emergency redistribution of resources in the developing world. The focus here is less upon the explicit evaluation of the design heuristics, but rather on assessing the pragmatics and utility of the methodology in the context of constructing a 'proof of concept' prototype for a problem of real-world relevance.

This chapter is structured as follows. First the problem under consideration will be described in the wider context. Then the abstracted version of the problem which will be the subject of this study, and a suitable formulation of it as a non-linear multiple resource transportation problem will be presented and justified. Examination of the pragmatics of the problem and the constraints on the system's eventual deployment will then show that a further reformulation of the problem is necessary, reducing the problem to one of sequencing. The appropriate literature will be reviewed alongside the above discussion.

The initial version of the system to test this problem will then be described and evaluated. Also, the opportunity will be taken to again evaluate the second, third and fourth design heuristics proposed in Chapter 3. After this, the effect of some modifications to the problem formulation will be investigated. This part of this investigation will then close with an examination of how well the system performs on a dataset of a size that the system may have to deal with in practice.

The focus of the investigation will then turn further towards assessing the pragmatics of the methodology proposed here by addressing issues raised by the above examination of the system's performance. First of all, a procedure for finding an upper bound on the performance of the system, as well as locating bottlenecks in the redistribution plan, will be devised. This bounds procedure will then be used, in conjunction with the proposed methodology, to suggest possible improvements to the system. These improvements, investigated in turn, involve modifications to the plan builder itself (search space reduction), as well as the heuristic initialisation and move preference knowledge sources.

## 7.1 Introduction

Dealing with aid projects, such as treating epidemics and disease control, in the developing world can be made difficult by the fact that though an effective regime or plan exists the actual implementation of the plan is often more difficult. This problem is compounded further by poor communication and transportation links. These combine to give a low quality of information for the planners, and to make the implementation of any supply plan difficult.

For instance, situations may arise where regional imbalances may arise (possibly due to communication problems, or disruption of the original supply plan). Such imbalances can seriously compromise the effectiveness of a relief program; an example would be a relief site with plenty of diagnostic kits for a particular disease, but with no drugs to treat the diagnosed cases.

This can seriously jeopardise the success of such a programme. The concrete example used in this study, centres around efforts to fight the spread of *Mycobacterium Tuberculosis* (TB) in the People's Republic of China. This is a serious health problem in many parts of the developing world, not only because of the obvious fatalities, but also in that many infected persons also suffer a chronic lung condition that causes great health and economic hardship, whilst all that carry the infection are potential vectors of the disease. One part of the developing world that suffers from a severe TB problem is mainland China where it is estimated that 250,000 Chinese die from the disease every year, and as many as 600 out of every 100,000 Chinese are infected with the disease [WHO 98a].

TB is diagnosed via microscopic inspection of lung sputum samples. Once diagnosed the recommended treatment strategy is an intensive six-month course of antibiotic/antibacterial chemotherapy where the patient is observed taking the drugs by a health worker. This regime is known as Directly Observed Treatment: Short-course (DOTS) [WHO 97b]. When implemented successfully this regime can produce clear-up rates as high as 95%, even in the poorest countries, and the low cost of the treatment (as little as US $11 per patient) lead to the World Bank ranking DOTS as "one of the most effective of all health interventions" [WHO 97b].

The observation and montoring aspects of this regime are essential to its success to ensure compliance. Any interruption of the treatment means that the course has restarted anew. Furthermore, such interruptions can lead to antibiotic, or Multi-Drug Resistant (MDR), strains of TB thus making the situation worse than if no action was taken [WHO 97a].

Unfortunately, even with fully functional patient observation and treatment compliance protocols in place, insufficent resource levels can lead to the interruption of the DOTS regime. This can be a significant problem — in fact, [WHO 98b] states that the "establishment of a dependable, high-quality suppy of anti-TB drugs throughout the health system is an essential part of the DOTS strategy to ensure that the treatment of TB patients is never interrupted".

Since 1991 the World Health Organisation (WHO), with funding from the World Bank (US $50 millon of credit), has backed the Chinese Ministry of Health in its implementation of DOTS throughout mainland China. In light of the geographic size and high population of China, the logistics involved for such a programme are considerable, as stated in [WHO 98a]:

> "While China is an outstanding example of a successful DOTS strategy, the vastness of the country and its population still presents formidable challenges to expanding DOTS more widely."

Currently there are over 1200 treatment sites in China, all of which need to maintain stock levels sufficent to facilitate the treatment of current cases, to diagnose new cases, and to begin and sustain their treatment. This is especially relevant in the light of the WHO's targets of diagnosing 70% of the cases of sputum smear-positive TB and curing 85% of these cases.

The system that this chapter will describe primarily aims to deal with situations where some disruption/interruption of normal supply has occured. For example flooding is common in areas of mainland China — recently (1997) the Yangste river flooded, and cut off a large area of the countryside. In addition, the system may also be used to assist in the planning of periodic scheduled resupplies/redistributions.

It should be noted that the logistical aspects of disaster/aid relief programmes has received little attention of late. One notable exception is the SUMA[1] system, which began as a PAHO[2]/WHO technical cooperation project in 1990 [PAHO 98]. The system's aim is to assist in the management of disaster relief supplies all the way from donor pledges of supply, to their entry into the disater area, and then finally to their warehousing and distribution. Primarily the system ensures that aid workers and decision-makers are fully informed as to the current state of supply at any time or place, thus allowing planners to identify and effectively resolve gaps in supply. This can transform the management of the logistics of such programmes as evidenced by SUMA's deployment in disasters such as Hurricane Caesar in 1996, and the earthquake at Nacza, Peru in 1997. Needless to say, work such as SUMA is a necessary prerequisite for a system such as the one considered here to be successfully deployed, as SUMA is only concerning with monitoring and not redistribution.

Furthermore, there have been a number of previous studies in the literature on the application of both AI and OR methods to problems in developing countries. For further details the reader is directed to [Luck & Walsham 97] for overviews of OR interventions, and [King & Beck 90] for a review of appropriate medical AI systems for developing countries.

## 7.2 The Problem

Now that the wider context of the problem has been discussed the specific requirements of the system will now be outlined. This undertaking was assisted by the availability of a domain expert, with experience of AI/development issues and more specifically the WHO TB programme in mainland China. Previous to this work the domain expert, Mr Richard Wheeler, was primarily involved in the development of an AI system for the compliance monitoring of the DOTS regime in mainland China, and the detection of trends relevant to the strategic management of the programme (such as seasonal or regional variations in treatment effectiveness) [Wheeler 95]. The system produced from this undertaking has since been successfully pilot tested in Liaoning, China and since then a full version has been commissioned and field-tested,

[1] For further details on the SUMA system, go to their website: http://www.disaster.info.desastres.net/SUMA/.

[2] The Pan American Health Organisation.

as described in [Wheeler *et al.* 98]. The system is now awaiting deployment.

In addition, Mr Wheeler has also been involved in a number of AI/development projects including a system for teaching Anaemia/Malaria risk assessment skills[3] [Hackett 96]. As a result of this, Mr Wheeler has an extensive and detailed knowledge of the development field and aid programmes in general, and the mainland China TB programme in particular.

Following from initial interviews with Mr Wheeler, it became clear that the problem in reality was a complex one, and often ill-defined and situation dependent. Also, the aim of this case study is not only to demonstrate that a neighbourhood search approach is appropriate to this problem, but to evaluate the utility of the methodology presented in this thesis. Also, matters of operational confidentiality mitigated against a fully detailed description of the problem on the field in any case. In the light of this, it was decided to examine an abstracted version of the problem that was sufficiently challenging to provide a realistic test of the system and its design methodology whilst retaining sufficient realism to allow the system, in principle, to be extended to a deployable state.

The goal of any logistic programme of this nature is to provide maximum coverage and treatment, to the areas that need it. Therefore, the aim of the system is to generate a redistribution plan that maximises the number of resource targets met (referred to as resource-sites), whilst minimising the number of shipments. If more than one resource is moved between two sites then this is counted as one shipment, and the cost is the same regardless of the amount shipped.

After further interviews, the situation that the system will tackle was defined as follows: $N$ sites will be considered, each of which need to have minimum amounts of $M$ resources in order to operate effectively. Furthermore, each site has different requirements (minimum resource levels) of each resource, and shortages and surpluses of each resource occur at different sites. A resource management system has to give a list of recommendations of the form ‘move X amount of resource Z from site A (which has a surplus) to site B (which has a shortage)’. A simple example of such a redistribution plan is given in Figure 7.1.

This optimisation is subject to various constraints on feasibility, which are outlined below.

1. Obviously, no site can supply more of a resource than it has.

[3] For details, see the OUTLOOK group web page at: http://www.dai.ed.ac.uk/groups/outlook/outlook.html.

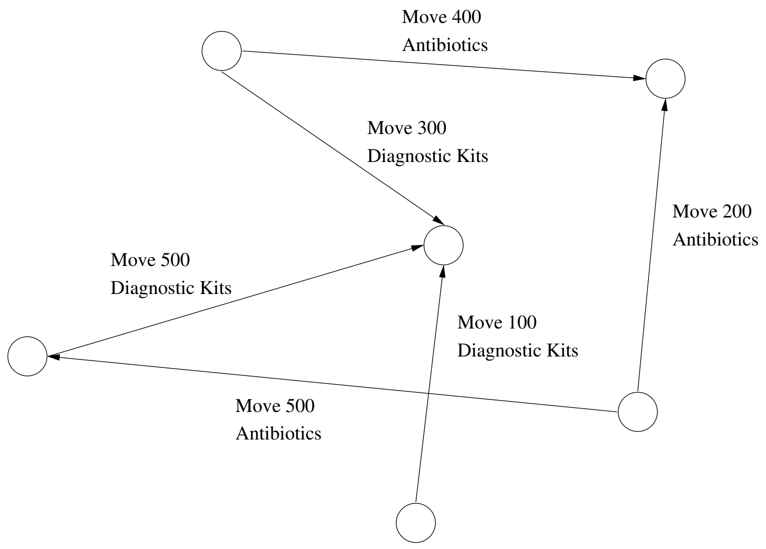


Figure 7.1: A Redistribution Plan For A Simple Domain

2. If a site has been supplied with a resource, then it cannot supply another site with that resource. This prevents 'daisy-chaining' which can lead to brittle plans, because if one of the shipments in the plan was not to occur, the later shipments then are likely to be disrupted.

3. Do not ship less than a certain minimum amount of a given resource (for reasons of efficiency).

4. If possible, a site should get resources from a site that is nearby (as the shipment is more likely to be completed, and with lower cost).

5. If possible, a site should get resources from a site that has a large surplus (as the site will be more likely to send the shipment).

6. A single large shipment is preferable to several smaller ones.

In addition to this, the nature of the shipments made should be clarified and repeated. First, a *single* shipment contains *all* the resources that are assigned to be moved from the source to the destination sites. Also, there is no capacity limit for any shipments — such a situation does not arise in practice as the quantities being moved are not bulky. This has two consequences: it is more economical of the available resources (as separate shipments need not be made);

more importantly it means that the problem cannot be split up into separate single-resource problems. As will be shown later, this has an impact on the problem formulation.

These constraints and objectives were selected, with consultation with the domain expert, as being those that would be universally present in problems of this type, and that would ensure that the shipment plan was implementable in practice. Given that, as noted in Chapter 2, heuristic methods are generally robust towards changes in the objective function this was felt to be a reasonable compromise. Also, though the interviews with the domain expert noted that the above constraints could in principle be relaxed, such situations would be rare and would not be left to an automated system to decide. Therefore the above constraints were assumed to be hard for the purposes of this study.

Finally, the hardware limitations are severe — the computer technology available in the third-world is hardly state-of-the-art. Therefore the final system aims to run on a 386-class PC compatible, with a 1 MB memory, and a 20 MB hard drive. The goal is to get good shipment plans for regions with at least 250 relief centres in, at most, several hours.

### 7.2.1 The Test Data

Two test problem datasets were provided by Mr Richard Wheeler in order to evaluate the system. The first is a small dataset consisting of 21 sites and 12 resources, and was intended for use in formative experiments and demonstrations[4]; the second is a large dataset consisting of 283 sites and 12 resources which is representative of the more difficult scenarios that the system is likely to face. Both datasets were derived from actual data taken from Mr Wheeler's work on the WHO's TB programme, though some modifications were made to maintain operational confidentiality — resources were given substituted names, as were counties and provinces.

Both datasets consist of, for each treatment site: the actual state of supply of the sites, a set of resource targets, and information on the relative location of the relief sites. The relative location data contains the sites where a supply route exists, and their distance measured in terms of the number of catchment areas that the shipment has to enter (which may not be the shortest possible route but the shortest *available*).

[4] Therefore a data set had to be available that returns a result in a few minutes, rather than overnight.

## 7.3 Problem Analysis and Formulation

The above problem can be viewed as a constrained optimisation problem with non-linear constraints and objective function. In other words, find a redistribution plan that minimises shortages and the number and cost of shipments required, whilst satisfying constraints such as those outlined earlier.

This section will look at a conventional OR formulation of the above situation as a multiple resource transportation problem, and discuss the shortcomings of this approach and why it leads to the adoption of a different formulation later in this chapter.

### 7.3.1 Direct Formulations

The obvious approach to designing an optimiser would be to propose a problem formuation solely in terms of the problem domain. Such **direct** or **actual-domain** formulations[5] have been used successfully to devise timetables (a classic constraint satisfaction problem). For example, [Corne *et al.* 93] adopts this approach to successfully tackle a real-world exam timetabling problem by encoding which exams are in which slots, and [Bruns 93] tackles the job-shop scheduling problem by operating directly on the Gannt chart. In fact, the majority of neighbourhood search optimisation applications are formulated in some terms of some view of the actual domain.

The main advantage of this approach to problem formulations is that it contains the entire space of possible redistribution plans, so the optimal solution is present and thus in principle able to be located. Furthermore since such formulations are closely related to the problem domain by their very nature, they can be readily enhanced by additional knowledge sources such as heuristic initialisation and move preference.

The remainder of this section will explore a direct formulation of this problem.

[5] Most of the EA literature uses the term 'direct representation'. However, this term (as noted earlier in Chapter 4 has a different meaning in the KBS domain; in addition the term formulation is more accurate.

### 7.3.2 The Transportation Problem

An extensive review of the AI and OR literatures revealed few studies that were directly analogous to this problem. The most relevant publication was (rather ironically) in the military domain [Kaplan 73] which examined the redistribution of military supplies. In this case the problem was formulated as a single resource transportation problem (multiple resource problems were left as further work).

The single-resource transportation problem is well studied in the OR literature, and a brief description will be given here — the reader is directed to a standard OR textbook such as [Winston 93] for additional detail. Consider $m$ sources, and $n$ recipients of a single resource. Let $c_{ij}$ be the per-unit cost of transporting that resource between source $i$ and recipient $j$, $d_j$ the resource *deficit* of the recipient site $j$, and $s_i$ the amount of spare resource of the source site $i$. The aim is to find an assignment to a set of decision variables $x_{ij}$, which denote the amount of resource moved between $i$ and $j$, such that all the demands are met and the transportation cost is minimised.

The usual method for solving this problem is to assume that the objective function and problem constraints are linear functions/(in)equalities of the decision variables — a **linear programming** formulation. For transportation problems, the general form is[6]:

$$\begin{array}{lll}
 & \min \sum_{i=1}^{i=m} \sum_{j=1}^{j=n} c_{ij} x_{ij} & \\
\text{s.t.} & \sum_{i=1}^{i=m} x_{ij} \geq d_j \quad (i = 1, 2, \dots, m) & \text{(demand constraints)} \\
 & \sum_{j=1}^{j=n} x_{ij} \leq s_i \quad (j = 1, 2, \dots, n) & \text{(supply constraints)} \\
\text{where} & x_{ij} \geq 0 \quad (i = 1, 2, \dots, m;\ j = 1, 2, \dots, n) &
\end{array}$$

Though the linearity assumption is often not completely realisable in practice, it has the advantage that efficient polynomial-time solvers are available, and in many applications, the ap-

[6] This is for a single-resource problem, for a multiple resource problem an additional subscript $k$ is added to denote the resource and the summations are extended to cover all $r$ resources.

proximation is reasonable. In fact, the above framework is remarkably flexible and can readily be extended to deal with situations such as when not all the demands can be met — a dummy supply node, $d$, is used and a cost is incurred if a target is not met, denoted by $c_{dk}$. However, the linearity assumption is not appropriate given the diversity of possible extensions, and the fact that the value $c_{ijk}$ is dependent upon other values of $x_{ijk}$. This is because, as noted earlier, we aim to minimise the number of *shipments* which comprise of more than one resource. Therefore, if we have already decided to ship one resource between $i$ and $j$, transfers of other resources between these two sites incur no additional cost.

Furthermore, the amounts transported must be integer units and therefore an **integer programming** approach is necessary and thus such variants of the transportation problem are invariably NP-hard. As a result, the use of heuristics, such as neighbourhood search, need to be considered. Finally, the aim in this specific problem is to maximise as many *targets* as possible, rather than to meet as much demand as possible. Again, this introduces additional complications and non-linearity.

In the case of the problem at hand, a possible direct formulation would be as a 3D matrix of integer decision variables where one axis corresponds to the site being supplied, another to the site providing the supply, the third axis to the resource in question, with the values at each position representing the amount to move. As the values will be integers (and therefore locality is important), an appropriate forma analysis formalisation would use the dedekind cut approach used previously for real numbers [Surry & Radcliffe 96a], with its suggested operators.

The design of a suitable objective function would be a more complicated affair — therefore the practicalities of this formulation of the problem will be discussed in more detail below.

### 7.3.3 Shortcomings of this Formulation

First of all, even a cursory examination of example solutions in the search space will reveal that most of the search space consists of infeasible solutions. Recalling the issues discussed in Chapter 4 (where the reader is referred to for details), if these infeasible solutions were dealt with by the assignment of a low quality/objective score, then the problem of ‘barriers’ of infeasible solutions between regions of feasible solutions arises. The alternative is to devise operators, using the search space reduction knowledge source, that ensure that all solutions considered by the optimiser are feasible.

Not surprisingly, the latter approach was that adopted in previous work on the non-linear transportation problem by [Michalewicz 92]. Specialised mutation and recombination operators were devised to maintain feasible solutions for an EA system that outperformed a conventional non-linear IP solver — the Generalised Algebraic Modelling System (GAMS) with the MINOS solver — on single-resource problems.

So why not apply this approach to the problem at hand? The problem is simply one of scalability. The problems tackled were not only simpler in their form (single-resource, no daisy-chaining constraints), but the instances tackled were *much* smaller — the largest was $10 \times 10$ — than the problem instances the system will eventually have to deal with. The evolutionary system used in [Michalewicz 92] clearly outperformed GAMS, which suggests that a neighbourhood search approach is a good basis with which to make progress on this problem.

Finally, in any case, this formulation can prove expensive in memory terms (especially for an evolutionary algorithm) as large data structures need to be stored and manipulated — memory requirements increase as $O(N^2)$, where $N$ is the number of sites. As memory is somewhat at a premium, this is an additional reason to reject this approach.

## 7.4 An Alternative Formulation

From the discussion above, the pragmatics of the situation dictate against the straightforward formulation of the problem in terms of its native domain. However, the above is but one of two approaches that can be taken to the formulation of this problem. The alternative is an **indirect** or **referent-domain** [Glover & Laguna 97] formulation, which attempts to formulate the optimisation in a domain *other* than the problem's native domain. This is achieved by an **inter-domain mapping function** $refer(r, \Psi_{\mathcal{R}})$ that maps a solution in the search space of the referent domain, $r \in \mathcal{R}$, to a corresponding solution in the actual domain $\mathcal{S}$ (ie. $refer : \mathcal{R} \to \mathcal{S}$) with a feature basis set $\Psi_{\mathcal{R}}$. For notation purposes we will refer to the *encoding* space of the referent domain search space as $\mathcal{E}_{\mathcal{R}}$, and its encoding mapping function as $g_{\mathcal{R}}$ where $g_{\mathcal{R}} : \mathcal{E}_{\mathcal{R}} \to \mathcal{R}$.

In the context of the problem considered here, the mapping between the two domains is commonly and conveniently performed by a procedure that takes encoded instructions, or a set of decisions about the choices the plan construction procedure has to make, and uses these to construct redistribution plans (a **plan builder**). The search therefore no longer takes place in the space of redistribution plans but rather in the space of plan builder choices/instructions.

The above said, does this approach fit naturally into the KBS framework proposed in this work? The answer is yes, though the knowledge sources map slightly differently. First of all the mapping function comprises of the search space reduction knowledge source, as the plan building process constrains the solutions in the actual search space that can be produced — in other words the plan builder defined by $refer(r, \Psi_{\mathcal{R}})$, can be seen as a procedural implementation of the constraints specified by $legal(\Psi)$. Of course, since a procedural plan builder may be easily obtained, eg. by modifying dispatch/constructive heuristics in the literature, this approach can be used as a quick method of acquiring knowledge and placing it into the optimiser, albeit in a somewhat implicit form.

The remainder of the knowledge sources are defined in the referent domain so for instance the formae knowledge source, $\Psi_{\mathcal{R}}$, will refer to features of the referent domain, as well as the linkage knowledge source, which in this case is $legal(\Psi_{\mathcal{R}})$, and so on. Therefore to formulate sensible hypotheses concerning the referent domain, knowledge of the structure of how $\mathrm{refer}(r, \Psi_{\mathcal{R}})$ (and thus the plan builder) relates to the actual problem domain is required — this point will be illustrated later. In any case, since the same knowledge sources apply to this approach, the design heuristics proposed in this work should also apply here.

Closer to home, such formulations using plan/schedule builders have been shown to be successful for timetabling problems as well [Burke *et al.* 95]. The timetabling study used an ordering of exams that were allocated times so as to avoid any clashes. Another domain where this approach has been proven successful is job-shop scheduling [Fang 94] where the encoding is a 'pick list' of active jobs which are to be scheduled by a dispatch heuristic. It should be noted that in principle, an equivalent set of operators for a direct formulation can be found — the choice or formulation approach is an issue of the pragmatics of the situation that the optimiser is being designed for. The work, noted earlier, by [Bruns 93] in the Job-Shop scheduling domain is a good illustration of how a direct formulation can be augmented with knowledgeable operators to guarantee feasibility and compete with informed indirect formulations.

This approach to formulation has its advantages: first, often it is possible to ensure that all (or most) of the hard constraints are met so that the plans generated are feasible; this formulation can also dramatically reduce the size of the search space; finally, the representation is compact, saving on memory. However, a possible problem is that in cutting down the search space, the optimal solution may not be present; also, as the problem has been abstracted from its native domain, exploiting domain knowledge directly in the referent domain can be made potentially more difficult, though as will be shown later still possible.

For this system, an indirect formulation was adopted, for the above reasons. Its remaining drawback: no guarantee that the optimal solution exists in the search space, was not deemed to be a problem as we are looking for the best solution that we can find in the limited time available. This conclusion is supported by the scalability issues discussed earlier.

In this study, a possible indirect representation would be the order in which relief sites are considered for supply. As this would be a permutation, this would give us the additional advantage that we could reuse some of the work of the previous case study. The plan builder then tries to find a source for the required resources. The plan builder (and its constituent procedures) was constructed on the basis of informal observations of how a human scheduler would go about this task, in conjunction with the domain expert. It was agreed in these discussions that the sequential resolution of shortages on a site-by-site basis was a reasonable (and workable) approximation to the actual scheduling process, especially when the size of the problem was taken into account[7]. Furthermore, as the constraints will always be observed in construction of the plan, if the plan builder checks them as it goes, feasible solutions will be *guaranteed*. Therefore the procedures outlined below were devised to implement the plan building component of the system.

Finally, further consultation with the domain expert revealed that some form of explanation/rationale for the construction of the shipment plan would be desirable. The fact that the constructive plan builder approach above could also be extended to provide a basic explanation facility via a trace of the construction algorithm, and possibly the choice points that it rejects (eg. because of distance or lack of supply) was an additional reason for its adoption.

[7] A similar approach was used for a preliminary attempt on this problem by Mr Wheeler.

### 7.4.1 The Main Algorithm

The pseudo-code for the plan-builder algorithm is given below. As noted above, the problem is now transformed into a sequencing problem, and therefore the plan builder is given the order in which the relief sites are to be considered, the current situation as regards resource levels at all the sites, the desired level of supply for each site, information on which sites are accessible to each other, and the distance, measured in number of site catchment areas traversed, between the sites. The pseudo-code for the main algorithm is given by Algorithm 8 below.

**Algorithm 8** THE MAIN PLAN BUILDER ALGORITHM

```
Require: sequence of sites, empty plan, current states of resource, and each site's targets;
for each site not considered do
    assume that there is no preferred site for supply;
    for each resource in turn do
        if REQUIRES-SUPPLY(current-site, resource) then
            repeat
                supply = FIND-SUPPLY(current-site, resource);
                SUPPLY-SITE(supply, current-site, resource);
            until either the current site has sufficient resource or no site can supply it
        end if
    end for
end for
```

#### The Procedure REQUIRES-SUPPLY()

The procedure, REQUIRES-SUPPLY() is used to see if a site needs supplying with a given resource. It is described in full by Algorithm 9 below.

**Algorithm 9** THE PROCEDURE REQUIRES-SUPPLY()

```
Require: the current site, current states of resource, and the site's targets;
if the site has less of a given resource than the target set for it then
    it requires more of that resource (true);
else
    it has sufficient resource (false);
end if
return the result (true/false);
```

#### The Procedure CAN-SUPPLY()

The procedure CAN-SUPPLY() is used to find out whether a site can be used as a supply of a given resource. This procedure also enforces the daisy chaining constraint, by returning false if the supplying site has already been supplied with the resource. The pseudo-code is given by Algorithm 10 below.

**Algorithm 10** THE PROCEDURE CAN-SUPPLY()

**Require:** the site, the desired resource, its current state of that resource, and the site's targets;

**if** the site has not already been supplied with the given resource **then**
    the amount it can supply is the amount it has above its target;
    **return** this amount;
**else**
    **return** that this site is not a source of the resource (false);
**end if**

### The Procedure FIND-SUPPLY()

The procedure FIND-SUPPLY() uses CAN-SUPPLY() to find a site that can supply another site with a given resource. In addition, it was felt desirable to introduce some greedy behaviour into the plan builder. This was achieved here by examining the nearest sites first (which reduces the distance the shipments need to take). Then, of these sites, the site with the greatest surplus is selected as this both decreases the number of shipments required, and increases that chance that the shipment will be made (as noted in Section 7.2). This procedure is described fully in Algorithm 11.

**Algorithm 11** THE PROCEDURE FIND-SUPPLY()

**Require:** the site, the desired resource, also the current state of resource and the site targets of nearby sites;

**if** CAN-SUPPLY(site, preferred-site) **then**
    use that site for supply;
**else**
    **let** $distance$ = 1
    **repeat**
        CAN-SUPPLY() finds how much each site within $distance$ can supply the resource;
        the site which can supply the most of that resource is selected;
        **if** this site can supply more than the minimum transportable amount of that resource **then**
            a suitable site has been found;
        **end if**
        $distance = distance + 1$;
    **until** either a suitable site has been found **or** all sites have been considered
**end if**
**return** either the suitable site found **or** the fact that no such site exists;

**The Procedure** SUPPLY-SITE()

Finally, when a site that can supply a given resource is found, the procedure SUPPLY-SITE() moves the resource and notes that the shipment has been made in the system's internal book-keeping. This procedure is described in Algorithm 12.

**Algorithm 12** THE PROCEDURE SUPPLY-SITE()

```
Require: the receiving site, the supplying site, the desired resource, and their resource levels
    and targets;
calculate the amount of resource available from the supplying site;
calculate the amount of resource required from the receiving site;
if amount available < amount required then
    the amount moved is the amount available;
else if amount required < the minimum transportable amount then
    the amount moved is minimum transportable amount;
else
    the amount moved is the amount required;
end if
add the amount of resource moved to the receiving site;
subtract the amount of resource moved from the suppling site;
make a record of the move made;
```

### 7.4.2 Implementation and Methodology

Generic code to implement the optimisers (SA, TA, etc) for permutations was available and used (it was in fact adapted from the code used for the previous case study). At this stage, the problem-specific information was contained in the plan builder (described previously) which constructs a valid redistribution plan, and an evaluation function which when given a redistribution plan determines its quality.

Most of the experimental methodology used here follows closely from that described in Chapters 5 and 6 earlier. In fact, for all of the various optimiser tunable parameters (such as initial temperature for SA), informal formative experiments indicated that a range of 1-10 was sufficient for the tuning experiments (which are detailed as before in the Appendices). Additional experimental information will be given later when required.

As noted earlier (Section 7.2), discussion with the domain expert quickly revealed that in practice many of the constraints and the balance of the objectives were situation-dependent. Therefore, in recognition of the fact that the problem of specifying the actual problem in detail is extremely hard, and that the problem of providing high-quality feasible solutions to the very large transportation problem is itself more than enough to be getting on with it was decided to use an abstracted version of the problem which preserves many of its main features.

Therefore, a simple objective was used in this study — a linear combination of the number of targets met ($targets\_met$) and the number of shipments made ($shipments$). This is shown in the following equation.

$$fitness = 10 \times targets\_met - shipments$$

The number of targets met is weighted so that the system will ensure that as many targets as possible are met, before trying to cut down on the number of shipments. A factor of 10 was found to suffice, as it is greater than the number of sites that are accessible by any other (given the usual inter-site distance constraint). However, one potential problem that could arise here is that if a site with a shortage of size $S$ can only be supplied by a size with surplus $S/2-1$ then the objective function will favour no shipments being made. However, this 'stepped' objective function was deemed a reasonable behaviour by the domain expert, because there is always the possibility that some additional *outside* shipments can be made after the redistribution plan has been produced to 'mop up' any gaps in supply. Avoiding 'incomplete' shipments makes this post-planning easier to produce and implement.

## 7.5 An Initial Evaluation

The small domain dataset (consisting of 21 sites and 12 resources) was examined first as to exhaustively investigate a very large dataset over a range of optimisation techniques and their parameters would be computationally prohibitive. This dataset had the feature that there were roughly sufficient resources available to supply all of the resource targets (as is generally true in China).

This initial evaluation is in two parts. The first verifies and validates the plan builder and ensures that it works as planned, the second part is an informal assessment of the system's performance on the above dataset. This will ensure that the system is working sufficiently well for the later more exhaustive experiments to be worthwhile.

### 7.5.1 Example Output

The system's output was first examined to see whether the plan builder was performing as expected. The system outputted, for each site in turn, the shipments it should make to other nearby sites. An example extract of the system's output is given in Figure 7.2 below.

```
From NingXia-Jiaoqu make the following shipments:
        Move 83    of XRAYS        to NingXia-Chengqu
        Move 14    of BLISTER4     to NingXia-Chengqu
        Move 323   of MICROSLIDES  to NingXia-Yongning
        Move 13    of XRAYS        to NingXia-Yongning
        Move 19    of BLISTER1     to NingXia-Yongning
        Move 25    of BLISTER2     to NingXia-Yongning

From NingXia-Yongning make the following shipments:
        Move 901   of E400         to NingXia-Chengqu
```

Figure 7.2: An Extract of the System's Output

**NOTE**: for reasons of operational confidentiality the names of the shipment sites, though real, have been permuted in the dataset.

Manual verification of a number of example outputs confirmed that the plan builder was producing feasible shipment plans that did not violate any of the hard constraints (ie. 'daisy-chaining', shipping more resources than are there, and the minimum shipment size).

### 7.5.2 Performance

Stochastic hillclimbing with a shift neighbourhood was used to give a rough idea of the expected performance of the system. The result obtained was encouraging. The initial solution (a randomly generated permutation) met about 85-92% of resource targets when processed by the plan builder, compared to 32% of resource targets met before the system is run. It would appear that the plan builder on its own forms a very powerful heuristic.

Further optimisation increased this figure to 90-92%, but also managed to reduce the number of shipments required by 20% (from about 125 to about 105 shipments), in less than 1000 solution evaluations. This takes about 7 seconds on a SPARC 5 workstation, or 5 minutes on a 386-class PC. It would appear that the system can suggest a workable redistribution plan, and in a short enough time so that scaling up to the full problem appears possible.

## 7.6 The Effect of the Unary Neighbourhood Operator

The question is now one of which problem feature/neighbourhood operator is most likely to be most effective. An argument can be made for assuming that precedences are useful features to consider for this problem. The argument is as follows: consider two shortages of a given resource in (a possibly larger) problem. Let us label these sites $A$ and $B$. There are 2 sources nearby (called $X$ and $Y$), each of which can only supply one of the sites due to the levels of their stocks. However, though $X$ can supply $A$ or $B$, $Y$ can only supply $B$. If $B$ is processed before $A$ by the plan builder all is well. The problem arises when $A$ is processed before $B$ as it is possible, if $X$ is closer to $A$ than $Y$, that the plan builder will supply $A$ from $X$, thus leaving $A$ unsupplied. Figure 7.3 shows this more clearly — the movement of supplies prescribed by the plan builder is denoted by arrows, and the possible supply routes by dashed lines.

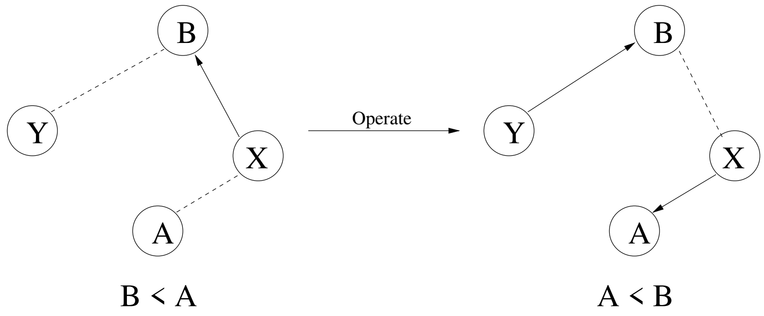


Figure 7.3: An Example of when a Precedence is Important

Therefore what appears to be important is whether $B$ is before $A$ in the sequence (a precedence), and not, for instance, their absolute positions in the sequence, or that $A$ and $B$ are adjacent (an edge). The analysis of the various permutation operators in Chapter 5 showed that the shift and the swap-adjacent operators were appropriate for this feature.

However, consider the case where $A$ and $B$ were very much apart with respect to their positions in the sequence, then if $B$ was $N$ positions in front of $A$, it would take at least $N$ adjacent-swaps to reverse their relative ordering. A single shift, or for that matter swap or reverse, operation could achieve the same result though with a corresponding degree of disruption to other precedences. On the other hand, the size of the neighbourhood is also a factor, especially for FAHC and SAHC-based optimisers. The question is therefore whether these factors are sufficiently important in this case to make shift the preferred neighbourhood operator.

### 7.6.1 The Representational Comparison

Therefore, the above points will be investigated in the context of a further evaluation of the second design heuristic carried out in the same manner as in the previous case study.

The measure of performance used here was the quality of solution obtained after 2000 evaluations for the small dataset described above, and informal formative experiments with SHC had shown that it converged (ie. not improved in quality of solution found) long before then. Fifty runs were taken in each case. Where differences in performance have been reported to be significant, this has been ascertained using the statistical analysis described in Appendix A.

As for the previous case study, the results obtained here were for the permutation neighbourhood operators: Shift, Swap, Reverse, Ordinal, Random Keys, and Swap-Adjacent; and the results of this comparative study are summarised in Tables 7.1 and 7.2. The mean of the solution quality obtained is quoted in all of the tables in this report, with its standard deviation in brackets, and the technique-dependent tunable parameter that was found to be most effective (with respect to quality) in square brackets.

| | Shift | Swap | Reverse |
|---|---|---|---|
| SHC | 2321.12 (1.76) | 2321.46 (1.42) | 2319.72 (2.00) |
| SA | 2321.64 (1.53) [7] | 2321.26 (1.59) [7] | 2319.96 (2.08) [3] |
| TA | 2321.82 (1.19) [1] | 2321.58 (1.30) [1] | 2319.66 (1.27) [2] |
| RTRT | 2321.58 (1.28) [1] | 2321.54 (1.47) [1] | 2320.00 (1.33) [1] |
| FAHC | 2319.08 (1.73) | 2319.22 (1.70) | 2317.54 (2.01) |
| FATS | 2319.16 (2.35) [3] | 2319.24 (1.69) [4] | 2317.90 (2.15) [10] |
| SAHC | 2316.84 (2.15) | 2319.34 (2.00) | 2317.58 (1.70) |
| SATS | 2318.28 (1.83) [1] | 2319.34 (2.00) [1] | 2317.74 (1.65) [6] |

Table 7.1: Comparison of Representations by Solution Quality (Part 1)

| | Swap-Adj | Ordinal | Random Keys |
|---|---|---|---|
| SHC | 2315.46 (4.42) | 2318.84 (1.49) | 2320.96 (1.39) |
| SA | 2315.32 (4.10) [4] | 2319.20 (1.60) [5] | 2321.10 (1.66) [5] |
| TA | 2317.08 (2.74) [2] | 2319.74 (1.48) [1] | 2321.92 (1.66) [1] |
| RTRT | 2316.88 (2.59) [2] | 2319.80 (1.54) [1] | 2321.42 (1.28) [1] |
| FAHC | 2311.36 (5.93) | 2317.66 (2.08) | 2320.10 (1.62) |
| FATS | 2316.00 (3.76) [7] | 2319.74 (1.66) [7] | 2321.24 (1.62) [9] |
| SAHC | 2310.96 (6.13) | 2317.92 (1.85) | 2320.12 (1.72) |
| SATS | 2316.48 (3.23) [10] | 2319.62 (1.40) [8] | 2320.88 (1.57) [7] |

Table 7.2: Comparison of Representations by Solution Quality (Part 2)

It should be noted that care must be taken when examining the differences between the means as the size of the standard deviations lead to some of the pairwise comparisons to be statistically insignificant. The details of the results of the statistical analysis are not given here — instead the reader is directed to Appendix C.

### Stochastic Hillclimbing Based Optimisers

Examining the trend for the SHC-based optimisers first of all, it is apparent that, as expected, the trends in relative operator performance are consistent across optimisers, with the shift and random keys neighbourhood operators performing best of all, closely followed by the swap operator which is known to operate in a similar metric space. The reverse and ordinal neighbourhoods then followed, with the swap-adjacent neighbourhood performing worse of all, thus demonstrating that the concerns raised earlier about the number of moves required to traverse the space were sufficient to cause problems. Use of the extended neighbourhood search optimisers (ie. SA, TA, and RTRT) generally resulted in improved performance, though the most suitable values for the optimiser parameters was low for the shift neighbourhood, which indicates that the height of local optima in the search spaces produced were not high, this could be because the plan builder is able to resolve many of the potential problems, thus leaving the optimiser with a much reduced task.

### First and Steepest-Ascent Hillclimbing Based Optimisers

Turning to the FAHC and SAHC-based optimisers, though the ordering of the relative performance of the operators is preserved, the situation has changed somewhat due to the greater effect exerted by the neighbourhood size for these optimisers — smaller neighbourhoods do relatively better. This is shown by the reversal of the some of the relative performance orderings obtained for SHC-based optimisers, for instance, ordinal and reverse for both FAHC and SAHC-based optimisers, and swap and shift for SAHC-based optimisers. The use of a tabu mechanism to avoid local optima often led to a significant improvement in performance, especially for the neighbourhoods that performed less well — presumably because they possess more scope for improvement due to their problems with, for example, local optima.

### General Points

The results obtained in the representation comparison support the second design heuristic in that relative landscape performance is preserved across optimisers of a given hillclimbing class. In addition, as in Chapter 6, it was found that the landscapes that were most suited to this problem also tended to have the lowest initial temperatures/thresholds which indicates that relatively little additional search control was required to attain a high standard of performance.

In further support of the above, it should also be noted that the relative performance of the neighbourhood operators here reflects that found for the 20-job FSSP problems in Chapter 6. One such example is that though the relative performance of some of the neighbourhood operators improves upon changing from a SHC-based to a FAHC or SAHC-based optimiser, in absolute terms, the results for SHC-based optimisers come out top, thus making any changes in relative performance obtained by switching to FAHC or SAHC-based optimisers rather academic.

Finally, as also found in Chapter 6, a clear pattern in relative performance emerges as SHC based optimisers come out on top, followed by FAHC and then SAHC-based optimisers — this is a clear demonstration that the gains possible by finding and taking the steepest path across the fitness landscape are not, at least in this case, are not enough to compensate for the additional evaluations required. This suggests that experience with one problem can possibly be transferred to another where there is reason to believe that there are structural similarities between them, such as in this case where it was thought that precedences/block moves were important for both problems.

## 7.7 Transfer to Evolutionary Algorithms

Attention will now turn towards the third and fourth design heuristics which suggest that stochastic-hillclimbing results can inform the selection of EA operators. Therefore, the results obtained above for the SHC-based optimisers suggest that crossover and mutation operators that explicitly manipulate precedences will do best of all — that is a combination of Precedence Preserving Crossover (PPX) and shift mutation. This combination will be compared with other combinations of the other unary neighbourhood operators considered in the hillclimbing experiments, as well as the recombination operators described in Chapter 5 and compared in the FSSP case study in Chapter 6.

The effect of crossover probability (with mutation applied otherwise) and population size was also investigated. In each case, the EA was run 50 times for each crossover operator over a range of crossover probabilities (0.0 to 1.0) and population sizes (10 to 100). The results for an EA of population size 100 are summarised below in Table 7.3 is a similar fashion to the hillclimbing experiments. The detailed results are presented in full in Appendix C, and the the statistical analysis is presented in Appendix C.

| | Shift/Default | Swap | Reverse | Swap-Adj |
|---|---|---|---|---|
| PPX | 2320.44 (1.27) [0.3] | 2320.10 (1.12) [0.4] | 2319.94 (1.35) [0.6] | 2319.00 (1.47) [0.7] |
| PMX | 2320.42 (1.17) [0.5] | 2319.98 (1.10) [0.5] | 2319.40 (1.25) [0.5] | 2319.04 (1.50) [0.9] |
| RRR | 2320.30 (1.20) [0.0] | 2319.76 (1.10) [0.0] | 2318.62 (1.02) [0.0] | 2318.10 (1.36) [0.4] |
| Edge | 2320.30 (1.20) [0.0] | 2319.76 (1.33) [0.0] | 2318.62 (1.02) [0.0] | 2317.58 (1.61) [0.5] |
| Ord | 2318.56 (1.12) [0.2] | – | – | – |
| Keys | 2320.24 (1.21) [0.0] | – | – | – |

Table 7.3: Comparison of Crossover and Mutation Operators (Solution Quality)

As predicted, the results obtained here tie in well with the predictions suggested by the hill-climbing experiments — the best performance was achieved by an EA with PPX crossover and shift mutation which therefore confirms the hypothesis that precedences are a suitable feature/building block for this problem. Looking at the results in more detail shows that the trends in the relative performance of the mutation operators reflect those obtained for the hillclimbing experiment with the shift neighbourhood (with the usual caveat about statistical significance). The trends in crossover performance also follow the general trend suggested by the hillclimbing experiments, irrespective of the mutation operator used: PPX and Modified PMX perform best of all giving comparable performance, followed by Position RRR, because of its more disruptive nature, and Edge Crossover — although these differences only become significant for the reverse and swap-adjacent neighbourhoods.

The observed trends are further highlighted by an examination of the crossover probabilities that were observed to be most effective. As was also observed in the previous case study (Chapter 6) the general trend was for higher crossover probabilities to be observed when 'effective' crossover operators (such as PPX) were used in conjunction with less effective mutation operators (such as reverse), and for low crossover probabilities to be observed when ineffective operators such as edge crossover were used with other more effective mutation operators. These trends are readily explicable as the evolutionary algorithm will give its best performance when it is operating is the most tractable metric space, and therefore, given the choice, a operator that induces a more tractable metric space will be used more often than one that is not.

### 7.7.1 A Closer Look

However, the full effect of the differences between the performance of the operators is not fully reflected in the tables above as the run with the best performance is considered. For example in the case of edge crossover, the best results are usually obtained when it is not used at all, which brings its performance close to the results for the other crossover operators. To resolve this problem and show the performance differences more clearly, plots of the average solution quality (over 50 runs) obtained by an EA with population 100 in 2000 evaluations against a range of crossover probabilities were obtained from the results in Appendix C.

Figure 7.4 shows a plot against crossover probability for the various crossover operators; for the permutation crossover operators, shift mutation was used but the results in the appendix are sufficient to construct similar plots with respect to a different permutation mutation operator. As is quite evident upon inspection of the plots, PPX does produce a consistent improvement over PMX, for all but the highest crossover probabilities[8].

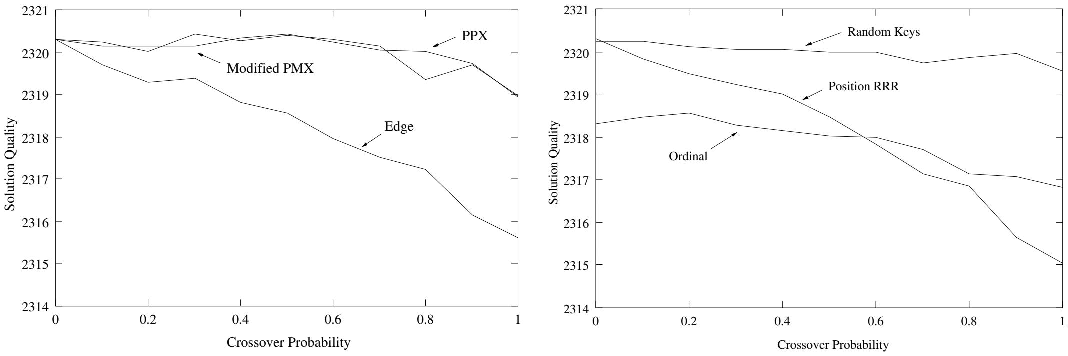


Figure 7.4: A Plot of the Effect of Crossover Operators and Probabilities (Shift Mutation)

Figure 7.5 shows a similar plot for the various mutation operators with respect to the Modified PMX crossover operator (the plots for random keys and ordinal representations are repeated for easy comparison).

[8] This change in relative operator performance at high crossover would be expected, as the disruptive effect of Modified PMX on precedences would be analogous to mutation.

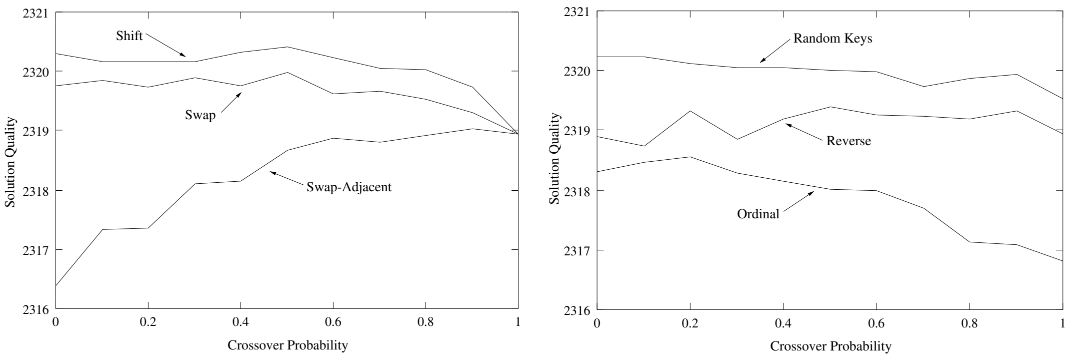


Figure 7.5: A Plot of the Effect of Mutation Operators and Probabilities (Modified PMX)

## 7.8 Optimising The Shipment Distance

So far the assumption has been made that the shipment can travel over any distance — this is in fact not true. For the problem we are considering, traversing over the catchment area of more than 3 sites is usually infeasible. Removing this assumption may reduce the ability of the system to supply all of the sites as some may have to be supplied from quite a distance. Therefore before we proceed any further (e.g. onto the full-sized problem) we need to assure ourselves that this is not the case, and that no anomalous behaviour results from this (as this could affect the test of the design heuristics and the methodology proposed in this thesis).

Also, one of the factors that has not been explicitly dealt with so far, but is never the less of importance if the system was to be used in the real world is how the approach presented here responds to reasonable changes in the objective function and other constraints imposed by the problem. Issues arising from this will be discussed and addressed in this section using hillclimbing experiments.

### 7.8.1 Reducing The Maximum Shipment Distance

As in the previous study, these further investigations where confined to the two most effective permutation neighbourhoods (shift and swap). Also, as one aim of this investigation is to see whether an effective system can be produced, some additional optimisation methods were run on the swap and shift neighbourhoods so to find the most effective optimiser. This choice turned out to be a simple iterated stochastic hillclimber which randomly restarts the optimiser if no improvement in solution quality has been found after a given number of evaluations (the

**iteration gap**).

Therefore, the investigation began by running an iterated stochastic hillclimber (iteration gap 250, shift neighbourhood) 50 times on the unconstrained problem. A histogram of the average number of shipments made at a given distance (measured as the number of site catchment areas that have to be traversed) against distance is given in Figure 7.6.

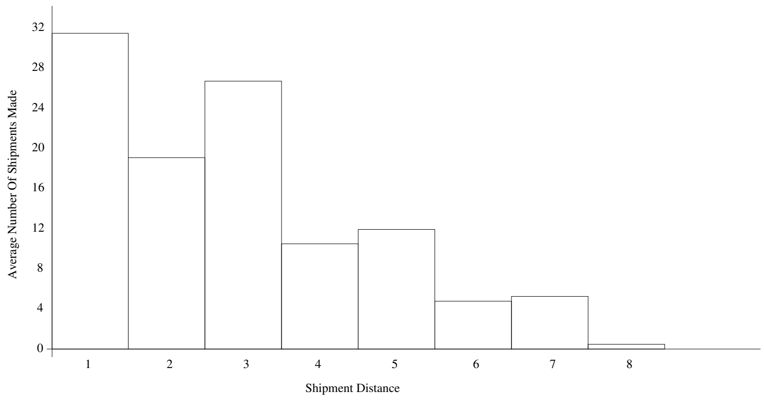


Figure 7.6: Graph of Average Number of Shipments Made Against Distance (Metric 1)

As expected, nearby sites were used more often. However, it would appear that shipments still have to stretch over quite a considerable distance — this means that the high coverage attained so far will (probably) not be achieved when distance constraints are imposed. This is not really a problem with the optimisation techniques, but rather, the new constraints render the previously high-coverage solutions infeasible.

Additionally, there appears to be a 'ripple' superimposed on the downward trend with periodicity 2. This was unexpected and raises question of why it arose. Is this an anomalous behaviour that we need to concern ourselves with? The test data was examined to see if this was an artifact of the data. A plot was made of the number of intersite links of a given distance, against distance (Figure 7.7).

The overall trend is different, peaking at distance 3, but appears plausible on the basis of geometric arguments. The point to note here is that the 'ripple' is superimposed upon the overall trend also (observe distances 1, 3 and 5). Therefore, the hypothesis that the ripple observed in the redistribution plan was an artifact of the data-set appears correct.

The hypothesis that imposing a constraint on shipment distance would affect the coverage

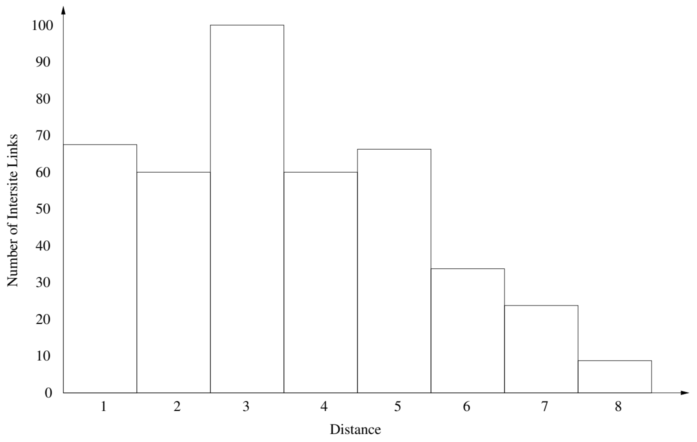


Figure 7.7: Graph of Number of Intersite Links Against Distance

attainable, was tested by again running an iterated stochastic hillclimber (iteration gap 250, shift and swap neighbourhoods) 50 times, with the constraint that the shipments cannot be made further than a maximum value. The components of the fitness function[9] are given in Tables 7.4 and 7.5.

As expected from the arguments presented earlier, tightening the maximum distance constraint, degrades the solution quality attainable. The choice of neighbourhood made no difference to the results obtained. Taking the maximum allowed shipment distance to be 3, then the coverage attainable appears to be around 90%. An impressive figure, but not as high as without the constraint (96%). Though, this appears to be a result of the solutions attainable in the search space being of lower quality, and not a problem with the optimisation algorithm which still performs well.

### 7.8.2 Using Shipment Distance As An Optimisation Criterion

Obviously, it would be desirable for the distance of the shipments to be minimised as well. This is performed to a certain extent by the plan builder which uses the closest site available; but this preference is not expressed in the evaluation function. Therefore, the optimiser may favour plans which use long shipments to reduce the number of shipments, rather than use two short shipments which may be preferred.

[9] The number of resource targets met is our main objective, followed by the cost of a shipment measured by either the number of shipments made, or the total distance of the shipments made as a secondary objective.

| Distance | Targets Met | Shipments | Distance |
|---|---|---|---|
| 1 | 159.90 (0.30) | 43.92 (0.27) | 43.92 (0.27) |
| 2 | 188.56 (0.50) | 65.28 (2.96) | 88.54 (4.30) |
| 3 | 227.00 (0.00) | 91.82 (0.82) | 188.62 (2.16) |
| 4 | 231.90 (0.46) | 97.68 (1.36) | 231.26 (7.26) |
| 5 | 241.90 (0.30) | 105.66 (2.07) | 295.42 (9.27) |
| 6 | 243.00 (0.00) | 107.88 (1.24) | 303.08 (8.76) |
| 7 | 243.00 (0.00) | 108.06 (1.36) | 312.22 (9.80) |
| 8 | 243.00 (0.00) | 108.04 (1.15) | 313.78 (9.56) |

Table 7.4: Performance Components with Maximum Shipment Distance (Shift)

| Distance | Targets Met | Shipments | Distance |
|---|---|---|---|
| 1 | 159.72 (0.45) | 43.72 (0.45) | 43.72 (0.45) |
| 2 | 188.50 (0.54) | 65.22 (2.75) | 88.50 (4.21) |
| 3 | 227.00 (0.00) | 92.50 (0.98) | 189.86 (2.81) |
| 4 | 231.88 (0.38) | 98.86 (1.44) | 234.82 (7.54) |
| 5 | 241.98 (0.14) | 106.44 (1.70) | 300.86 (7.36) |
| 6 | 243.00 (0.00) | 108.52 (1.27) | 303.74 (8.97) |
| 7 | 243.00 (0.00) | 108.42 (1.36) | 316.46 (10.67) |
| 8 | 243.00 (0.00) | 108.64 (1.16) | 317.60 (11.23) |

Table 7.5: Performance Components with Maximum Shipment Distance (Swap)

The evaluation function used in the prototype was modified to be a linear combination of the number of targets met ($targets\_met$) and the total *distance* covered by the shipments made ($distance$). This is shown in the following equation.

$$fitness = 10 \times targets\_met - distance \tag{7.1}$$

This modification should hopefully lead to solutions that have a greater proportion of low-distance shipments, and also a lower sensitivity to the maximum shipment distance being reduced than previously.

The first point was investigated by running a stochastic hillclimber (iteration gap 250, shift neighbourhood) 50 times. A histogram of the average number of shipments made at a given distance against distance is given in Figure 7.8.

As can be observed, the shipments made have been shifted to shorter distances, as expected — the total distance travelled was significantly lower than before. To see if the sensitivity of performance to the maximum shipment distance was reduced as a consequence of the new fitness function, a stochastic hillclimber (iteration gap 250, shift and swap neighbourhoods) was run 50 times. The performance attained is given in Table 7.6.

In these cases, the coverage attained is 90% when the constraint of a maximum shipment

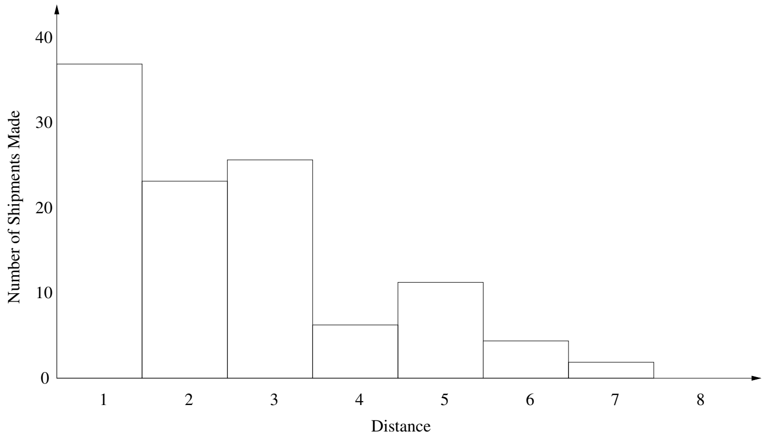


Figure 7.8: Graph of Average Number of Shipments Made Against Distance (Metric 2)

| | Metric 1 | | Metric 2 | |
|---|---|---|---|---|
| Distance | Shift | Swap | Shift | Swap |
| 1 | 1555.08 (2.76) | 1553.48 (4.04) | 1555.08 (2.76) | 1553.48 (4.04) |
| 2 | 1820.32 (2.71) | 1819.78 (3.40) | 1797.52 (2.16) | 1797.64 (2.23) |
| 3 | 2178.18 (0.82) | 2177.50 (0.98) | 2082.84 (2.92) | 2081.54 (4.27) |
| 4 | 2221.32 (3.83) | 2219.94 (3.49) | 2096.32 (5.92) | 2095.96 (4.73) |
| 5 | 2313.34 (3.10) | 2313.36 (2.07) | 2145.68 (4.76) | 2144.38 (4.85) |
| 6 | 2322.12 (1.24) | 2321.48 (1.27) | 2159.20 (8.39) | 2158.64 (6.73) |
| 7 | 2321.94 (1.36) | 2321.58 (1.36) | 2138.38 (4.27) | 2137.46 (4.64) |
| 8 | 2321.96 (1.15) | 2321.36 (1.16) | 2138.86 (5.06) | 2138.12 (4.66) |

Table 7.6: Performance as a Function of Maximum Shipment Distance

distance of 3 is imposed. Again, this compares well with the 96% coverage obtained without the constraint. Further comparison with the results obtained using the number of shipments as a fitness criterion (see Tables 7.7 and 7.8 and compare them with Tables 7.4 and 7.5 earlier), shows that the coverage attained using the two criteria are not significantly different.

However, as the fitness function would prefer two short shipments rather than one long one, does this affect the total number of shipments made significantly? This question was answered by comparing the above with Tables 7.4 and 7.5 for each of the two fitness functions. A significant increase in the number of shipments made was observed along with a decrease in the total distance travelled — which confirms the suspicion that reducing the total distance travelled can only be achieved at the expense of additional shipments.

Finally, since the performance with respect to the primary objective was unaffected by the change in the evaluation function, it would appear that this approach is reasonably robust to changes in the objective function.

| Distance | Targets Met | Shipments | Distance |
|---|---|---|---|
| 1 | 159.90 (0.30) | 43.92 (0.27) | 43.92 (0.27) |
| 2 | 188.50 (0.50) | 65.18 (2.36) | 87.48 (3.58) |
| 3 | 226.90 (0.36) | 92.72 (1.11) | 186.16 (1.75) |
| 4 | 230.88 (0.86) | 97.94 (1.55) | 212.48 (5.02) |
| 5 | 239.98 (0.42) | 105.82 (1.98) | 254.12 (5.66) |
| 6 | 242.12 (0.38) | 107.02 (2.68) | 262.00 (10.54) |
| 7 | 243.00 (0.00) | 111.56 (2.27) | 291.62 (4.27) |
| 8 | 243.00 (0.00) | 111.46 (2.53) | 291.14 (5.06) |

Table 7.7: Performance Components with Maximum Shipment Distance (Shift)

| Distance | Targets Met | Shipments | Distance |
|---|---|---|---|
| 1 | 159.72 (0.45) | 43.72 (0.45) | 43.72 (0.45) |
| 2 | 188.48 (0.50) | 64.96 (1.98) | 87.16 (3.05) |
| 3 | 226.86 (0.40) | 92.96 (1.18) | 187.06 (1.99) |
| 4 | 231.02 (0.58) | 98.00 (1.59) | 214.24 (5.72) |
| 5 | 240.08 (0.39) | 106.32 (2.02) | 256.42 (6.57) |
| 6 | 241.98 (0.42) | 106.78 (2.00) | 261.16 (7.73) |
| 7 | 243.00 (0.00) | 111.58 (1.94) | 292.54 (4.64) |
| 8 | 243.00 (0.00) | 111.96 (1.79) | 291.88 (4.66) |

Table 7.8: Performance Components with Maximum Shipment Distance (Swap)

## 7.9 Experiments on Larger Datasets

The experiments so far have been performed on a relatively small domain (21 treatment sites, 12 resources). A practical system will have to deal with more than this —- a minimum figure of 250 sites at any one time. As one of the objectives of this chapter is to produce a system that could be used for this problem, the following subsections will detail the arguments for why we can be optimistic that the system will scale up, and then some experiments to see whether or not this is the case will be described.

### 7.9.1 Why Should The System Scale Up?

It can be argued that the method proposed here will not be able to scale up to problems of this size. This is because the small domain had a search space of size 21! (about $5 \times 10^{19}$) — not small by any standards! But consider a search space of 250! and it could be argued that such an exponential increase in the size of the search space would make the system too slow to be useful.

However, we do have reasons to be optimistic that size (at least in this case) does not matter! First, the experiments in Section 7.8 have shown that the shipments between sites are relatively local, especially when a constraint on the maximum shipment distance is added. This means that the redistribution plan builder only has to consider a relatively few number of sites, which should remain (roughly) constant as the number of sites considered increases as there are only so many sites that can be within $N$ catchment areas of a given site. Therefore we can reasonably expect the time complexity to be at best linear, or at worse to a power of 2, with respect to the number of sites. This does not address the problem of combinatorial explosion, but at least assures that it will not be compounded.

Another way that the plan-builder based approach taken here benefits from the locality of the shipments is that we can reasonably expect the redistribution plans generated with random sequences to produce solutions of similar quality to the small domain considered so far, as what to do about shortages of a particular site will only affect nearby sites significantly, therefore the amount of interaction (epistatis) between sites is local and bounded. This is one possible reason why the landscape for this problem is highly tractable for hillclimbing and contains few local optima. The tractability of this landscape is a major justification for supposing that this approach can scale up.

### 7.9.2 An Evaluation Using A Large Dataset

To test the above, a larger data-set was provided, consisting of 283 sites and 12 resources, based on actual data as before.

Stochastic hillclimbing was used in this evaluation, using the first of the evaluation criteria examined. and with the constraint that a shipment cannot go further than 3 site catchment-areas. Before the system was run, only 19% of the resource targets were met (660 out of a maximum of 3398 resource-sites) The initial (randomly generated) solution met about 54% (1830 resource-sites) of resource targets when processed by the plan builder.

Further optimisation was then used for 2000 evaluations (which took about 10.5 minutes on a SPARC 5 workstation, or 7-8 hours on a 386-class PC). This increased the number of resource targets met to 57% (1959 resource-sites), however improvements were still being made at the end of this period which indicates that better quality solutions did exist. This was tested by extending the number of evaluations made to 20,000. The best solution obtained (after 19734 evaluations) met 58% (2002 out of a maximum of 3398) for the resource targets; though it still appeared that further improvements were being made — a locally optimal solution would appear to take much (i.e. prohibitively) longer to find.

During the optimisation, the number of moves made rose slightly (from 979 to 987) over the 2000 evaluations, but fell again after 20,000 evaluations to 966 moves. These figures appear not to be greatly different, but as the main criterion was to optimise the number of resource targets met above all else, it was hardly surprising that they were not optimised, when the main criterion was not yet fully optimised at those points in the search.

The percentage of resource targets met does seem somewhat disappointing in the light of the results obtained for the 21 site data-set. This could be taken as a failure of the system to scale up to larger problems; but examination of the large dataset indicates strongly that this is not the case. What is of importance here is the distribution of sites with surpluses in the data set; it was found that these sites were heavily concentrated (spatially) at one end of the dataset — in fact [Wheeler 98] notes that, given the vast size of China, it is entirely possible for situations to arise where areas the size of the UK have insufficient levels of a resource! When this observation was combined with the maximum shipment distance constraint, it became immediately apparent that at least 30% of sites in the data set were inaccessible to any of the sites with surpluses. Therefore the lower performance appears to be a result of the dataset, and the constraints upon it, rather than any fault of the system (i.e. given the constraints on the problem, reaching 90% of the resource targets is impossible).

## 7.10 Calculating an Upper Bound

There are still two knowledge sources described in Chapter 4 that have not been considered (heuristic initailisation and move selection). In order to formulate heuristics for these two knowledge sources, some indication of what the difficult to solve shortages are would be desirable.

In addition, the observation that many of the sites were heavily concentrated (spatially) will now be examined quantitatively to get a more accurate idea of the number of sites that could be supplied. Therefore, the following procedure for estimating an upper bound of the number of resource sites that can be supplied, and which shortages may be difficult to resolve, was devised.

### 7.10.1 The Bounds Procedure

For this procedure, each site is considered in turn, then each resource, $i$, is considered by the following procedure:

1. If the current site has a shortage of the resource under consideration, and a supply can be found, then this site is a **supplyable shortage** of that resource ($num\_sup\_short_i$).
2. If the current site has a shortage of the resource under consideration, but *no feasible* supply can be found (according to, for example, the 3-site distance limit), then this site is a **unsupplyable shortage** of that resource ($un\_short_i$).
3. If the current site has a surplus of the resource under consideration, and a site that has a shortage of that resource can be found that the current site can supply, then this site is a **supplyable surplus** of that resource ($ok\_surpluses_i$).
4. If the current site has a surplus of the resource under consideration, and *no* site that has a shortage of that resource can be found that the current site can supply, then this site is a **unsupplyable surplus** of that resource ($un\_surpluses_i$).
5. Otherwise, the current site has a **satisfactory** level of that resource ($satisfactory_i$).

A tally is kept up of the the size of the supplyable surpluses and shortages ($sup\_surplus_i$ and $sup\_short_i$) of each resource and the number of sites in each category. The upper bound can then be calculated as follows:

$$bound = short\_supplied + \sum_{i=1}^{resources} satisfactory_i + un\_surpluses_i + ok\_surpluses_i$$

$$short\_supplied = \sum_{i=1}^{resources} min\{1.0, sup\_surplus_i / sup\_short_i\} \times num\_sup\_short_i$$

The upper bound is effectively the number of resource-sites that are already satisfied, plus the number of resource-sites with supplyable shortages, weighted by the amount of resources available to supply those shortages.

### 7.10.2 An Initial Evaluation

| | Small Dataset | Large Dataset |
|---|---|---|
| Upper Bound: | 242 (cf. 252) | 2623 (cf. 3396) |
| Supplyable Surpluses: | 82 | 605 |
| Unsupplyable Surpluses: | 0 | 13 |
| Satisfactory Resource Levels: | 0 | 42 |
| Supplyable Shortages: | 166 | 1963 |
| Unsupplyable Shortages: | 4 | 773 |

Table 7.9: Results of the Analysis by the Bound Calculation Procedure

The analysis of the two test problems by the procedure is shown in Table 7.9. It indicates that it is not possible to deal with all of the shortages, and that the distribution of supplies across the data is very uneven. In addition, the upper bound allows us to evaluate the performance attained by this system in a more positive light as we are comparing the performance of the system against a more attainable target.

In fact, the system finds a plan that deals with 75% of the shortages after 2000 evaluations. This is quite respectable, especially when one considers that previous approaches to transportation problems with non-linear constraints and objective function could not deal with problems of even a tenth of this size.

In addition, it should be noted that the upper bound presented here is still an overestimate as it does not take into account that supplyable sites might be clustered with respect to their sources of supply, and therefore the method used above of working out the proportion of targets met by using the ratio of supplyable resource surplus and demand can mislead ([Wheeler 98] notes that this is likely to be the case).

To illustrate this, consider the situation where there are a large number of supplyable sites that can be serviced by a site $A$ which only has sufficient amount of resource to supply a few of these. Furthermore, the few remaining sites that have supplyable shortages are clustered around a site $B$ which has a surplus sufficient to not only deal with the sites it can supply but also some of $A$'s sites as well (if such a transfer could be made). Then, considering the two clusters together in the manner of the above bounds procedure will effectively assume that site $B$'s surplus *can* be used to supply shortages that are only supplyable by site $A$. This is clearly false and therefore an overestimate of the number of targets that can be met occurs.

However, though this problem could in principle be circumvented by separating out the clusters and dealing with each individually, the above procedure is sufficient for our purposes. Attention will now turn to look at ways of exploiting the above information to improve the efficiency of the system.

## 7.11 Improving the Plan Builder

Most of the domain knowledge is held in the plan builder and therefore it is one of the most important aspects of the system to get right (especially as most of the CPU time is spent in there). In particular, there is additional scope for using the search space reduction knowledge source described in Chapter 4 (as the bounds procedure gives us additional information on which shipment plans are not worth considering). Two approaches to improving the plan builder will now be investigated, before our attention turns to the last two knowledge sources.

### 7.11.1 Exploiting the Constraints to Improve the Plan Builder

Needless to say, if sites that cannot be supplied are a regular occurance then there would be no point in considering them in the search. Therefore we could exploit these constraints to reduce both the search space and the amount of effort involved in constructing a shipment plan.

To reduce the search space, the following reasoning was employed: as the search space is the order in which the sites have their shortages dealt with, the only sites we need deal with are those with shortages. Furthermore using the information obtained by the bound calculation procedure, we can further restrict this to the sites that can be somehow supplied. Therefore each site was considered in turn, and if it contained no shortages which could be supplied, it was removed from the search space. Apart from reducing the search space, we are reducing the number of redundant moves which should speed up search. To see why, consider a site that has no supplyable surpluses, in this case the plan builder will not be able to do anything for that site, and therefore there is nothing to be gained in considering it.

In addition, the bound calculation procedure shows us which sites are supplies for which resources. This information could be used to prevent the plan builder having to examine sites that we already know are not sources of supply for that resource.

### 7.11.2 Modifying the Plan Builder

As described in full in Section 7.4, when a shortage is encountered, the plan builder looks for a site that can supply that resource, starting with the sites closest to the shortage.

The effect of changing this so that instead of all of this sites at a particular distance being considered to find the one with the highest surplus, each site is considered in turn and the first one that is able to supply is chosen was investigated. This change from the original procedure effectively implements a *partial* search of the possibilities for supply. Hopefully this will reduce the computational effort involved in plan construction, though at the possibility of reducing the quality of solutions generated by the plan builder.

### 7.11.3 Results Obtained

When the constraints were exploited by the system, no reduction in the search space was produced for the small dataset, and a modest reduction was obtained for the large dataset (from 283! to 278!). A closer examination of the datasets showed that shortages of different resources occur in different regions of the datasets, and thus very few sites are without some kind of a shortage[10]. Fortunately, the reduction in the number of resource-sites that have to be considered by the plan builder was more encouraging: from 252 to 82 for the small dataset, and from 3396 to 605 for the large dataset.

To see whether this, and the modification of the plan builder, resulted in an actual improvement in performance, stochastic hillclimbing with a shift-neighbourhood was run 20 times for 2000 evaluations. The following performance indicators were recorded: actual (wall-clock) time to perform 2000 evaluations on a SPARC 5, the number of evaluations until the best solution was found, the overall solution quality, and the number of resource-sites that met their targets. The results obtained are shown in Table 7.10, with standard deviations given in brackets. Where differences in performance have been reported to be significant, this has been ascertained using a t-test.

It was found that both search space reduction (Std vs Red) and the partial search for supplies (Full vs Part) produced a significant improvement in the wall clock time. Search space reduc-

[10] Although the datasets were drawn from real examples, they were selected to be challenging — most situations are not as difficult as this.

| | Time | Evaluations | Quality | Resource-Sites |
|---|---|---|---|---|
| Full & Std | 704s | 1975.65 (21.65) | 18619.45 (56.30) | 1958.30 (5.95) |
| Full & Red | 671s | 1966.10 (27.92) | 18796.40 (50.42) | 1975.10 (5.36) |
| Part & Std | 611s | 1968.70 (23.25) | 18594.15 (61.41) | 1957.85 (6.34) |
| Part & Red | 597s | 1962.80 (34.62) | 18593.55 (42.43) | 1958.35 (4.79) |

Table 7.10: Results Obtained when the Plan Builder was Modified

tion was also found to be effective in increasing the quality of the final solution, both in terms of overall fitness, and the number of resource-sites that met their targets (1975 resource-sites is just over 75% of the upper bound). However, not making a thorough search for possible sources of supply led to a degradation in solution quality.

In summary, it would appear that exploiting the constraints was a good idea for this problem, though efforts to reduce the amount of work performed by the plan builder were not successful.

## 7.12 Directing the Neighbourhood Operators

Attention will now, in this section, turn towards investigating whether the above bounds procedure can help us to devise a suitable heuristic for the move selection knowledge source.

The rationale behind this is as follows. A possible reason that a site may be experiencing a shortage may be due to it being too late in the sequence to the plan builder (as earlier sites may have used up its source(s) of supply). So preferentially applying operations which push sites with shortages closer to the front may speed the search by making moves that are more likely to improve performance more common. To this end, the neighbourhood operators were directed (an idea pioneered in evolutionary timetabling [Ross *et al.* 94]) to see if this had any effect.

Directed mutation was used implement a move selection heuristic in the fashion described in Chapters 4 and 6 earlier for flowshop scheduling, which also used a permutation encoding and shift and swap neighbourhood operators. That study used tournament and marriage selection to select the sites to be moved on the basis of a heuristic value of ‘blame’ for each site. The difference lies in the heuristic used — in this case the ‘blame’ was the number of unsupplied resources that the bound calculation procedure considers supplyable for a site.

Stochastic hillclimbing (over 2000 evaluations) and the large dataset were used to evaluate the effectiveness of this approach

Examination of the results above show that the directed neighbourhood operator appears to degrade performance — but why? Is it because the directed mutation/move selection heuristic is making it more likely that *worse* solutions are being found first, or that the directed mutation leads to improved performance at the early stages of the run, but these early gains cause problems later?

One way to investigate this would be to do a plot of solution fitness against evaluations for both with and without directed mutation. Now if the first of the possibilities outlined above is occurring we would expect the plain optimiser to do better than the directed optimiser throughout the run. In the case that the second of the possibilities is occurring, we would expect to see the directed optimiser doing better at the beginning of the run, and worse later on.

However, just 'eye-balling' the plots is unsatisfactory as there is no way to tell if any differences are significant. Therefore, Figures 7.9 and 7.10 below show both plots of average fitness (over 20 runs) and plots of the t-statistic to see if any differences are, in fact, significant.

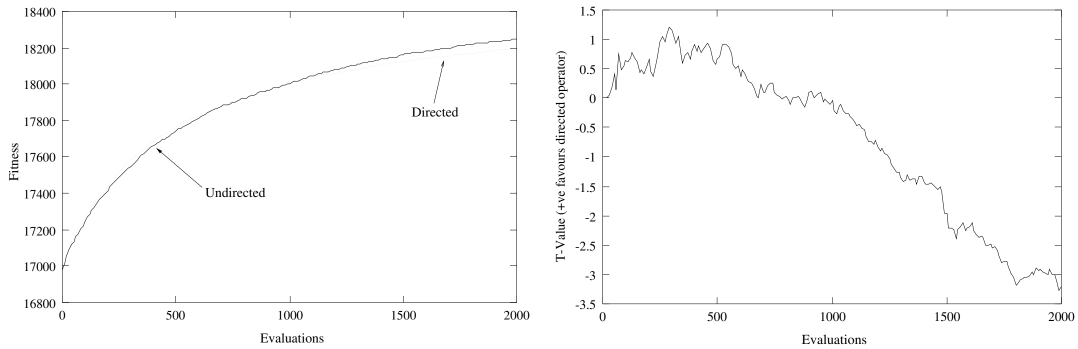


Figure 7.9: Plots of the Effect of Directing the Neighbourhood (Shift Neighbourhood)

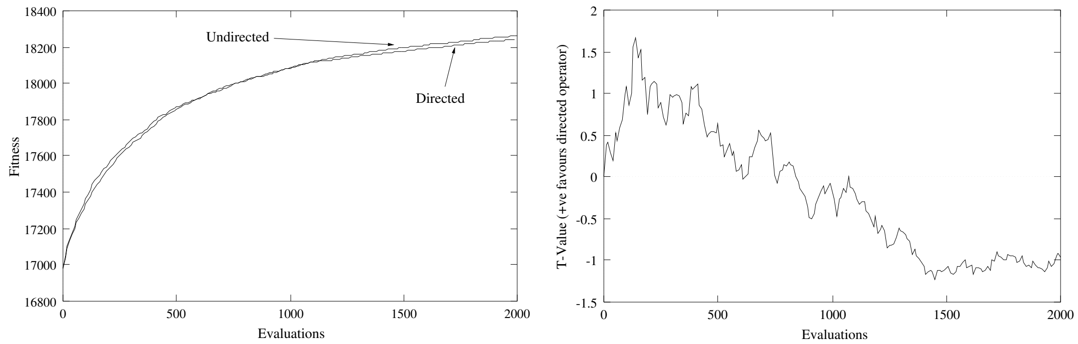


Figure 7.10: Plots of the Effect of Directing the Neighbourhood (Swap Neighbourhood)

The figure above indicates that the directed neighbourhood operator does lead to early, though modest, gains in performance — especially for the swap neighbourhood. These gains are then lost later in the run when the performance of the directed optimiser becomes significantly worse. It would appear that the move selection heuristic is partly right — providing some gains early on in the run, but can cause problems later presumably either because the search has been misled, or that the heuristic has the opposite effect at later stages of the search.

## 7.13 Intelligent Initialisation

One remaining knowledge source, described in Chapter 4, that has not been considered so far is heuristic initialisation. This section will show how the bounds procedure described above can be exploited to produce an implementation of this knowledge source.

The reasoning that sites with the most problems should be earlier in the sequence, may be exploited in the initialisation procedure. If we can identify which sites have the most shortages and place then at the start of the initial solution, we may obtain a better quality solution. Therefore the following procedure was used: generate a random solution, calculate a heuristic value for each site, then apply a sorting procedure (in this case quick-sort) to place sites with the most supplyable shortages at the start of the sequence. The following two heuristics were used:

1. Using the bounds procedure, obtain the number of shortages at each site that are supplyable and use this as the heuristic.
2. Evaluate a randomly generated string using the plan builder and then, with the help of the bounds procedure, find the number of supplyable shortages at each site assuming that the plan has been implemented.

Both methods employ the belief that the sites causing trouble (ie. that have shortages) should be placed earlier in the sequence. However, the second method aims to overcome the possibility that the first heuristic could be mislead by virtue that, in the initial situation, although a site may have a lot of shortages these shortages are easily resolved by the plan builder, whereas a site with fewer but more difficult to supply shortages would be placed later by the first heuristic. Of course the second heuristic overcomes this problem at the cost of an extra evaluation, but this may be worth the effort.

The first experiment was to ascertain if the initialisation procedures do, in fact, produce a better quality initial solution. Therefore each of the 2 initialisation methods were used 1000 times and the quality of their solutions measured and compared against 1000 randomly-generated solutions.

| | Random | Using Bounds | Using Plan Builder |
|---|---|---|---|
| Quality | 16988.93 (123.31) | 16995.66 (124.73) | 16993.86 (126.05) |
| Resource-Sites | 1793.03 (13.08) | 1793.67 (13.18) | 1793.56 (13.32) |

Table 7.11: Comparison of Initialisation Methods over 1000 Initialisations

Table 7.11 summarises the results obtained for the large dataset. There appears to be some increase in the quality of solutions generated by the 2 heuristics but a t-test indicates that this difference is not significant. In addition, it also appears that using the plan builder does not result in any improvement above using the bounds procedure alone, despite its higher computational cost.

Attention was then turned to whether these initialisation methods would result in any gains when the optimiser was run. Again stochastic hillclimbing and the large dataset were used in this evaluation. In each case, stochastic hillclimbing was run 20 times for 2000 evaluations. Figures 7.11, 7.12, 7.13 and 7.14 summarise the results obtained.

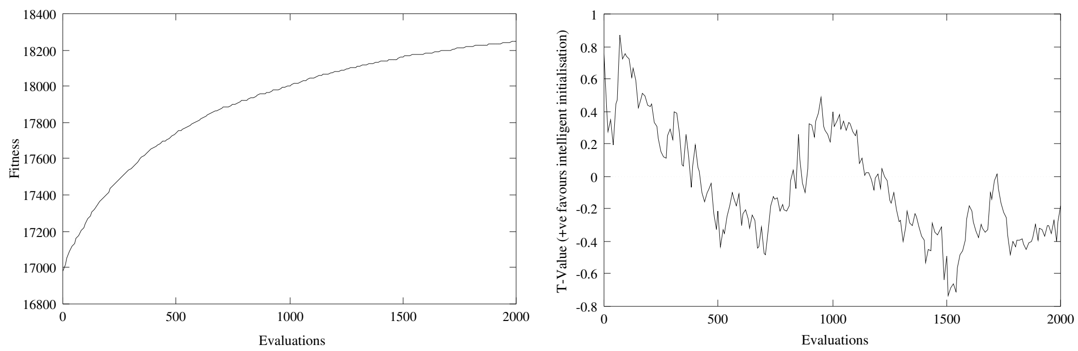


Figure 7.11: The Effect of Initialisation using the Bounds Procedure (Shift Neighbourhood)

For the shift neighbourhood, it is evident from an inspection of the t-value plots that initialisation using the bounds procedure is not effective — presumably due to the concerns about this method discussed earlier. On the other hand, initialisation using the evaluation function

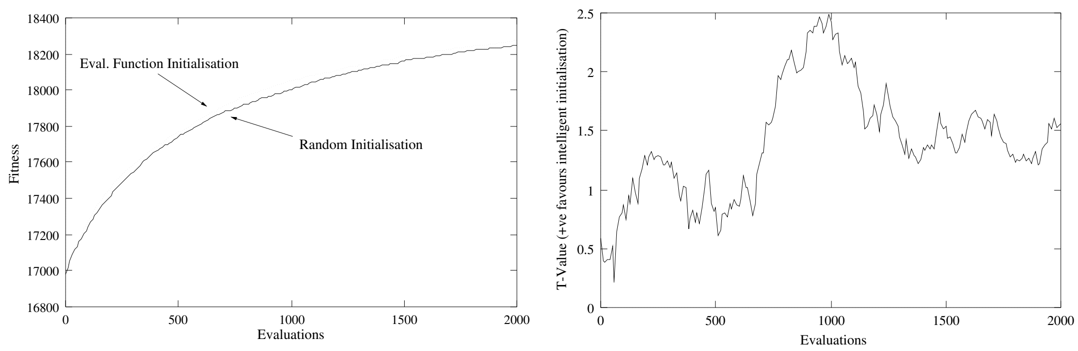


Figure 7.12: The Effect of Initialisation using the Evaluation Function (Shift Neighbourhood)

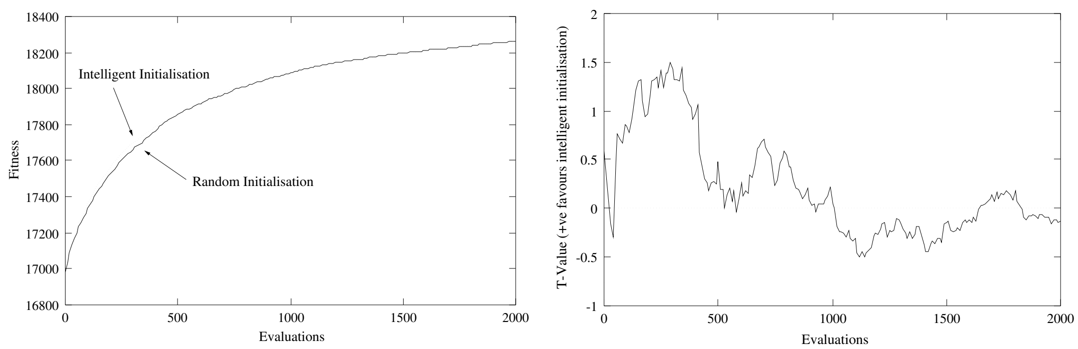


Figure 7.13: The Effect of Initialisation using the Bounds Procedure (Swap Neighbourhood)

was effective in improving solution quality, though the effect seemed to diminish in the later stage of the run which would not be unexpected as the rate of progress of the hillclimber also diminishes with time, thus allowing the randomly initialised hillclimber to 'catch up' on the intelligently initialised hillclimber. This was independent of the neighbourhood operator used.

Furthermore, it would appear, on the basis of these results that improvements (or lack of them) in initial solution quality is a poor predictor of initialisation method effectiveness when the optimiser is run for some period of time.

## 7.14 Conclusion

The system outlined above was constructed as a test of the applicability of some modern optimisation techniques to a real-world problem of resource management. The prototype constructed was able to quickly produce a workable redistribution plan for both small and full-sized versions of the problem which increased dramatically the number of resource targets

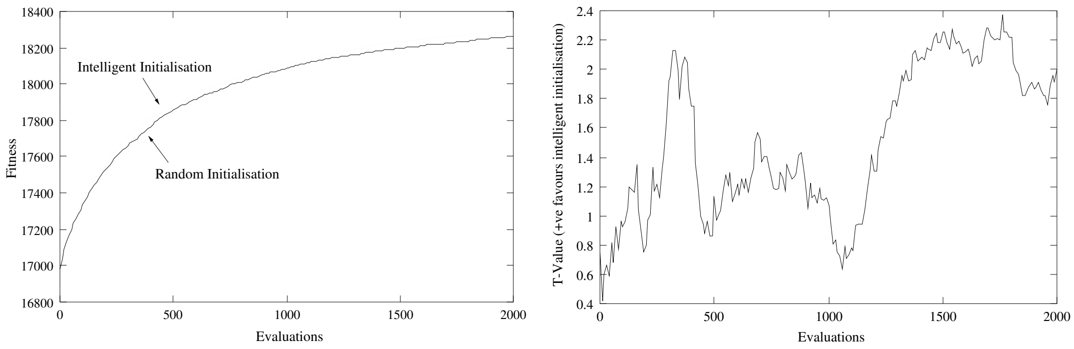


Figure 7.14: The Effect of Initialisation using the Evaluation Function (Swap Neighbourhood)

met. The system was also found to respond well to changes in the objective function. Optimisation improved upon this still further; but the size of the search space meant that this took some time for large problems. Fortunately the 'anytime' characteristic of the system means that this is not so much of a problem (the user can run the system for as long as they can and take the best solution found so far).

The experiments with the full-sized dataset indicated that the constraints upon the problem may often prevent all the resource targets being met, even *if* there are sufficient resources available. In this light, the performance of the system on the large dataset is respectable considering both the scale and the complexity of the problem.

In any case the fact that feasible redistribution plans are produced by the system at all, let alone guaranteed on an anytime basis, compares extremely well with conventional approaches to this problem which would not scale up to problems of this size. Given that the pragmatics of the situation dictate that plan feasibility is more important than achieving plan optimality, the approach taken here is a workable compromise — any increases in plan performance during optimisation can be regarded as a welcome benefit rather than a necessity.

Furthermore, the investigations in this chapter have shown how the methodology proposed here can be used in an informal manner, to both suggest useful performance enhancements to the system. This aspect is of value as the formal machinery, though undoubtedly useful, may be unwieldy or unnecessary in some situations. This is especially for non-specialists in optimisation who want to get a working system up and running with the minimum of time and effort.

The design heuristics proposed in this work were not as extensively validated in this case study as for the FSSP case study earlier. However, where they where validated they were also shown to apply to this problem, allowing experience and knowledge gained with a hillclimber to be transferred to other optimisers. As these design heuristics comprise a major component of this work, this test is an important one. Overall the proposed methodology has come out well of this case study.

Finally, the comparison of the meta-heuristics showed that stochastic hillclimbing was the method of choice for the large dataset given the specified time/solution quality tradeoff. In this context, the performance of the Evolutionary Algorithm was particularly disappointing, although whether this lies with the type of recombination operator used, or with population-based search in general requires further investigation. Though if for some reason an EA was deemed desirable, a population with some degree of hillclimbing character would be recommended.

# Chapter 8

# Conclusion: Putting it all Together

Hopefully the reader has, by now, been convinced of the advantages of viewing the design of neighbourhood search techniques as the construction of a knowledge based system, the importance of the role of having a clear and explicit model of the problem domain and the nature of the desired solution, and the need for structured experimental protocols. Therefore, to finish this thesis, this chapter will summarise the contributions of the work presented in this thesis and outline routes for future work.

## 8.1 Contributions of This Approach

The main contributions of the the work presented here can be summarised under the following headings below.

- **Relates Design to the Problem Domain**. The design process suggested here begins by designing the optimiser in terms of the problem domain by finding a 'good' fitness landscape by specifying, and then continuing to evaluate, other knowledge sources that are also related to hypotheses about the problem domain. This ensures that the final system's behaviour can be justified in terms that the end-user and/or domain expert is able to understand, and thus increases the final system's chances of being adopted.
- **The 'Right' Level of Description**. The design process proposed here works at the knowledge level and in a declarative manner. That is so as to separate *what* the optimiser knows (ie. the hypotheses about the structure of the problem it embodies) from *how* they

are implemented (represented) by the optimiser. This again helps to focus the design process onto the problem domain.

- **Directs the Search for Domain Knowledge**. The approach of specifying the roles that knowledge plays and decomposing them into separate knowledge sources that relate to concrete aspects of a model of search by trial and error allows sensible and well-formulated questions to be devised for the domain expert to answer.

- **Formalisation**. The forma analysis work by [Radcliffe 94, Surry 98] was used to play a central role in suggesting a suitable formalism for the above knowledge sources and in providing a formal declarative semantics for the fitness landscape. This allows the designer's hypotheses about the problem domain to be made explicit, unambiguous, and independent of the particular way that they may be represented by the optimiser.

- **Exploits Technique Commonalities**. The design heuristics proposed here allow their common aspects to be considered. This is because of the knowledge-level analysis undertaken here which separates what the optimiser is trying to do from how it is doing it. This has the obvious advantage that more of the experience gained by one optimiser can now be transfered to another even if they are superficially quite different.

- **Suggests Useful Hybridisations**. As noted in Chapter 3, hybrid methods, that is combinations of neighbourhood search methods with for example domain-specific heuristics, have been suggested by some practitioners as the way forward. However, Chapter 3 objects to this by arguing that this is a truism that is of little use unless the issue of *how* to combine two or more methods effectively. The KBS view described here provides such a route because the user can now hybridise methods in a principled fashion by examining these domain-specific heuristics to find the problem structure that they exploit, and then transfer this knowledge to the most convenient knowledge source for the neighbourhood search optimiser.

- **An Effective Experimental Protocol**. The design heuristics presented here and in Chapter 3 consider the interaction between knowledge roles and sources, and provide guidance on which should be considered first. More importantly, they allow hypotheses about the problem domain to be evaluated, using hillclimbing experiments, with results that are both transferable to other optimisers and to the design of EA recombination operators. The combination of these two benefits demonstrate how the proposed design heuristics structure experimentation in an effective manner, extracting maximal benefit from the limited time and resources for experimentation available.

- **Tuning Issues**. As the search control aspects of the optimiser are considered only after hypotheses concerning the problem domain theory have been exhausted, the difficulties of optimiser choice/tuning are avoided for as long as possible. Especially, as noted in Chapter 3, these techniques aim to produce a system that can obtain a 'good enough' answer in the time available — therefore it is possible that a suitably informed hillclimber could provide sufficient performance.

- **Provides Structure to the Literature**. As a result of the above, the literature can be usefully categorised in terms of the knowledge roles and sources that they refer to and *then* the different search control options can then be considered. Applications can also be similarly organised in terms of the different ways that they fill the knowledge roles/sources so as to highlight interesting similarities and differences.

- **Pedagogical Utility**. This structure can then find its way to teaching as the above organisation focuses the student's attention unto the design aspects of these techniques and what to look for when designing these optimisers in practice. In addition, leaving the description of the various search control options to the end will emphasise both the common features of these technique, and the need to design optimisers in terms of the problem domain theory.

- **Future Research Directions**. Finally this work suggests a number of new research directions, such as further integrating the work here with KBS design methods. These and other possible research topics will be discussed in more detail later in Section 8.2.

Attention will now turn to outlining the constributions made by the two case studies, and other studies the author has been involved with.

### 8.1.1 Contributions of the Case Studies

It should be noted that the case studies have produced a number of useful contributions aside from the (obviously) important ones of sucessfully evaluating the proposed design heuristics, and affirming the intuitions raised in Chapter 3 of how neighbourhood size affects relative search performance. Though the reader is directed to the conclusion of each of the two case studies for details, some points are worth repeating. In the case of the FSSP case study in Chapter 6 the methodology was able to produce an effective idle-time move preference heuristic that was successfully implemented in both a directed mutation and candidate list strategy. In the case of the resource redistribution study in Chapter 7, this methodology was able to produce a system that was able to deal with problem instances of a real problem much larger than the current state-of-the art.

### 8.1.2 Other Case Studies

In addition to this, the author has supervised a number of MSc and undergraduate projects during the course of his PhD studies. For instance, the work in [Nakata 97] has validated the design heuristics in the context of the travelling salesman problem; also [Ramos 97] has tested these design heuristics on a problem of finding an optimal variable ordering for the logic minimisation of FPGAs (Field Programmable Gate Arrays), producing clear improvements over the current state-of-the art in the process. Furthermore, the knowledge level decompositon has been successfully used to suggest improvements in a number of problem-orientated projects, examples include: FORTRAN array optimisation to maximse cache usage [Altmann 97]; musical composition [Phon-Amnuaisuk *et al.* 99]; as well as the design/shape optimisation of structural beams and aircraft annuluses [Baron *et al.* 97a, Baron *et al.* 97b, Baron 97, Baron *et al.* 99].

In summary, once work outside of this thesis has been taken in account, there is a fair body of empirical evidence that the arguments made in this thesis do in fact stand up.

### 8.1.3 Where Now?

Given that we have established that the work presented here provides a number of real contributions, especially to the principled design of neighbourhood search optimisers, the remainder of this chapter will comprise of an outline of ideas for future research that have been suggested by the points raised in this thesis.

## 8.2 Possibilities For Further Work

Recent work in the philosophy of science argues that the effectiveness of a scientific theory/research programme is primarily measured by its 'fertility' — the amount of future research that it suggests [Chalmers 98]. The arguments presented so far show that the methodology above does indeed suggest a rethink of the current research programme and brings previously separate lines of research 'into the fold'. In addition to this, the methodology presented above suggests some further ideas for future research — some of these are now outlined below.

### 8.2.1 Further Formalisation

A number of knowledge sources have been proposed in this work and the issue of formalisation has been addressed, albeit in differing degrees, for each of the knowledge sources. This is because, the process of formalisation allows assumptions and hypotheses about the nature of the problem to be made explicit and unambiguous, and also makes possible the use of mathematical (formal) methods for the structured implementation of systems that embody these hypotheses. In addition, formalisation is also important in an engineering context as practitioners would find a library of suitable operator and features definitions for a number of problem domains very useful. This is because, apart from the convenience of using 'off the shelf' components, a codified repository of 'best practice' should have the benefit of suggesting suitable formulations and hypotheses about the problem at hand.

Therefore, there is a need for the further formalisation of the knowledge sources described here, which should take two related directions. First of all, with the exception of forma analysis (which was already well-developed), the emphasis has been upon arguing the roles of the knowledge sources, and the justification of their design heuristics. Therefore the formalisations presented here should be considered *suggestions*, and thus require further refinement. This is especially with regard to the process of mapping the knowledge level descriptions to operator specifications, which was described here in at best a semi-formal manner. Also, in order to codify the best practice for a representative number of problem domains, forma analysis needs to be extended to a wider range of domains with particular attention needing to be given to problems that do not naturally correspond to string/array-like data structures such as the LISP S-expressions used in genetic programming [Koza 92]. In addition, systematic procedures for producing operator refinements (such as linkage specialisation) need to be established, and formal 'operator taxonomies' of application domains need to be constructed (such as a formal version of the work in [Mattfeld *et al.* 96]).

Finally, it should be noted with interest that formal operator and feature definitions should, in principle, allow for the *automated* synthesis of suitable and formally verified program code for neighbourhood operators, thus making the task of constructing a correct working optimiser much easier and more reliable. For example, work by [Wiggins *et al.* 91] shows how a Prolog program can be produced from a verification proof produced by an automated theorem prover (in short, this is because every verification proof corresponds to a computer program).

### 8.2.2 Integration with KBS Design Methods

The most obvious gain a KBS view of neighbourhood search brings is that once the KBS framework and knowledge-level analysis have been produced, the way is open to adapt and make use of KBS design methods to assist in the design and implementation of neighbourhood search optimisers.

To this end, KBS methodologies can be employed in a number of ways. For example, KBS methods exist to assist in the process of acquiring/eliciting domain knowledge — this is termed **knowledge acquisition/elicitation** [Musen 89]. One specific example should illustrate this. Section 4.2 earlier notes that the problem solving knowledge sources can, in principle be phrased as questions that the domain expert could be expected to understand and answer. This bears many similarities to the **probing questions** method [Kline & Dolins 89, Inder *et al.* 90] which uses a structured set of questions to elicit the required knowledge from the domain expert — which suggests that this approach should be extensible to neighbourhood search optimisers.

Furthermore, as noted in [Motta 97], there are a number of other potential aids to assist in the development of effective KBSs. These include **domain and task analysis** [Steels 90, Chandrasekaran *et al.* 92], **knowledge modelling** systems such as the **CommonKADS** framework [Wielinga *et al.* 92] and **VITAL** [Domingue *et al.* 93], work on **system specification** [Jonker & Spee 92], and **KBS validation and verification** [Fensel & Schoenegge 97]. In addition, like neighbourhood search optimisers, KBSs are more often than not constructed 'from scratch' each time one is built which is potentially rather wasteful of manpower and other resources. In the KBS community this is seen as a problem and the issue of **knowledge sharing** [Neches *et al.* 91] between KBSs, and the move to **reuse-centred** models of KBS design is a currently active topic in the KBS community [Motta 97][1]. Therefore, in summary, it would appear that there is much useful future work to be performed in integrating and adapting these KBS design methods to neighbourhood search optimisation techniques.

### 8.2.3 Improved Search Frameworks and Theory

Further research into general frameworks for neighbourhood search in necessary. The most obvious reason for this is that such symbol level frameworks will then provide a stable and formal implementational platform with a well-defined interface for knowledge-level descriptions to be mapped (represented) onto. In other words, the knowledge and symbol level descriptions used need to be well matched to be maximally effective. There is another reason for such frameworks, however, where further work in developing a better theory of the search dynamics plays a more useful role.

Given that the underlying theory for the search control knowledge role is rather incomplete, knowledge acquisition in the context of this knowledge role is currently largely a process of experimentation (tuning). Also, as the various search control extensions appear to be at first glance quite different, then it is likely that experimental results for one optimisation method would be difficult to use with another. That said there are commonalities between the techniques that could, in principle, be exploited to make the above process more efficient.

This key to this would be to devise experimental protocols to *characterise* the landscape, with emphasis upon identifying the features of the fitness landscape that are impeding the search and devising modifications to the search control to overcome them. Therefore the knowledge level entity in this approach would be the designer's hypotheses concerning the structure of the fitness landscape. An example of this would be if, at the knowledge level, the landscape was thought to be quite rugged but otherwise correlated, then some mechanism to ignore the local peaks would be a good idea. So, if simulated annealing was used the temperature term would perform this function, with the required temperature increasing as the ruggedness increases —

[1] In fact PSMs are one of the methods that are used to this end, and the design heuristics proposed here could also be viewed as being relevant.

this could be transferred to threshold accepting. With tabu search, this knowledge would be represented in the tabu list, aspiration criteria, etc. so as to strike the correct balance between exploration and exploitation.

As a result of these considerations, a KBS view of optimiser traversal rule design suggests that there may be scope in reusing experience and knowledge gained in examining one optimisation method to the design of another. From this, one important role of future work on search frameworks and the search dynamics of neighbourhood search optimisers will be to produce a categorisation of search control methods and fitness landscapes that highlight similarities and mappings between the features of the fitness landscapes, suitable search control extensions, and their symbol-level representations.

## 8.3 Summary

> "So next time you're faced with an NP-hard problem, don't just roll over and expose the jugular. Fight back!" — Randy Helzerman

This thesis has attempted to tackle the problem of the principled design of neighbourhood search optimisers head on. To this end, a KBS view of optimiser design has been proposed, centered around a semi-formal knowledge-level decomposition of neighbourhood search. Concurrently, a number of design heuristics were proposed that exploit hillclimbing experiments to allow the designer's knowledge of the problem domain to be evaluated in an effective manner that is transferrable between different optimisers.

Two case studies were investigated to see whether the proposed methodology would be a suitable candidate. The first of the case studies, the flowshop sequencing problem successfully validated the design heuristics proposed here, and produced an idle-time based move preference heuristic that was found to produce gains in optimiser performance. The second case study, tackling a real-life resource redistribution problem, was able with this methodology to produce an optimisation system that guaranteed workable, good-quality shipment plans for problems much larger than the current state-of-the-art.

For a summary of the publications that have arisen thus far from the work in this thesis, the reader is directed to Appendix B.

Finally, this thesis ends with the note that it is, in reality, the foundation rather than the culmination of a research agenda. Much still needs to be done before the design of neighbourhood search optimisers can truely be considered a science rather than an art.

# Bibliography


[Aarts & Korst 89] E. H. L. Aarts and J. H. M. Korst. *Simulated Annealing and Boltzmann Machines*. Wiley, Chichester, 1989.

[Abramson 91] D. Abramson. Constructing school timetables using simulated annealing: sequential and parallel algorithms. *Management Science*, 37:98–113, 1991.

[Adenso-Diaz 92] B. Adenso-Diaz. Restricted neighborhood in the tabu search for the flowshop problem. *European Journal of Operational Research*, 62:27–37, 1992.

[Allahverdi & Aldowaisan 98] A. Allahverdi and T. Aldowaisan. Job lateness in flowshops with setup and removal times separated. *Journal of the Operational Research Society*, 49(9):1001–1006, 1998.

[Altenberg 94] Lee Altenberg. The evolution of evolvability in genetic programming. In Kenneth E. Kinnear, Jr., editor, *Advances in Genetic Programming*, chapter 3, pages 47–74. MIT Press, 1994.

[Altenburg 95] L. Altenburg. The Schema Theorem and Price's Theorem. In D. Whitley and M. Vose, editors, *Foundations of Genetic Algorithms 3*, pages 23–49. San Mateo: Morgan Kaufmann, 1995.

[Altenburg 97] L. Altenburg. Fitness Distance Correlation Analysis: An Instructive Counterexample. In T. Bäck, editor, *The 7th International Conference on Genetic Algorithms*, pages 57–64. San Mateo: Morgan Kaufmann, 1997.

[Althofer & Koschnick 91] I. Althofer and K. U. Koschnick. On the convergence of 'threshold accepting'. *Applied Mathematics and Optimisation*, 24:183–195, 1991.

[Altmann 97] N. A. Altmann. An Investigation of Neighbourhood Search to Improve Cache Use. Unpublished M.Sc. thesis, Department of Artificial Intelligence, Edinburgh University, 1997.

[Anderson 96] E. J. Anderson. Mechanisms for local search. *European Journal of Operational Research*, 88:139–151, 1996.

[Antonisse 89] J. Antonisse. A new interpretation of schema notation that overturns the bibary coding constraint. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, San Mateo, CA, 1989. Morgan Kaufmann.

[Ashour 70] S. Ashour. A branch-and-bound algorithm for flow shop sequencing problems. *AIIE Transactions*, 2:172–176, 1970.

[Bäck 97] T. Bäck. Binary Strings. In T. Bäck, D. B. Fogel, and Z. Michalewicz, editors, *Handbook of Evolutionary Computation*, chapter C 1.2. IOP Publishing Ltd and Oxford University Press, 1997.

[Bäck *et al.* 97] T. Bäck, D. B. Fogel, and Z. Michalewicz. *Handbook of Evolutionary Computation*. IOP Publishing Ltd and Oxford University Press, 1997.

[Baron 97] P. Baron. Evolutionary Shape Optimisation using a Voxel Representation. Unpublished M.Sc. thesis, Department of Artificial Intelligence, Edinburgh University, U.K., 1997.

[Baron *et al.* 97a] P. Baron, R. Fisher, F. Mill, A. Sherlock, and A. Tuson. A Voxel-based Representation for the Evolutionary Shape Optimisation of a Simplified Beam: A Case-Study of a Problem-Centred Approach to Genetic Operator Design. In *The 2nd On-line World Conference on Soft Computing in Engineering Design and Manufacturing (WSC2)*, 1997.

[Baron *et al.* 97b] P. Baron, R. Fisher, F. Mill, A. Sherlock, and A. Tuson. A Voxel Based Approach To Evolutionary Shape Optimisation. In *The Fourth AISB Workshop on Evolutionary Computing*, 1997.

[Baron *et al.* 99] P. Baron, A.L. Tuson, R. Fisher, A. Sherlock, and F. Mill. An Investigation of a Voxel-based Representation for Evolutionary Shape Optimisation. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AI-EDAM)*, 6(2), 1999.

[Bean 94] J. C. Bean. Genetic Algorithms and Random Keys for Sequencing and Optimization. *ORSA Journal on Computing*, 6(2), 1994.

[Beck 92] H. Beck. Constraint monitoring in TOSCA. Technical report, AIAI, Edinburgh University, UK, 1992.

[Beck 93] H. Beck. The Management of Job-Shop Scheduling Constraints. Technical report, AIAI, Edinburgh University, UK, 1993.

[Benjamins 93] R. Benjamins. *Problem Solving Methods for Diagnosis*. Unpublished PhD thesis, Department of Social Science Informatics, University of Amsterdam, 1993.

[Bersini 91] H. Bersini. Immune network and adaptive control. In F.J Varel and P. Bourgine, editors, *Proceedings of First European Conference on Artifical Life*. MIT Press, 1991.

[Bertoni & Dorigo 93] A. Bertoni and M. Dorigo. Implicit parallelism in genetic algorithms. *Artificial Intellignce*, 61:307–14, 1993.

[Boese 96] K. D. Boese. *Models for Iterative Global Optimisation*. Unpublished PhD thesis, University of Califoria, Los Angeles, 1996.

[Box & Jenkins 71] G. E. P. Box and G. M. Jenkins. *Time Series Analysis, Forecasting and Control*. Holden Day, 1971.

[Bridges & Goldberg 87] C. Bridges and D. E. Goldberg. An analysis of reproduction and crossover in a binary-coded genetic algorithm. In J. J. Grefenstette, editor, *Proceedings of the Second International Conference on Genetic Algorithms*, Hilldale, NJ, 1987. Erbaum.

[Brindle 81] A. Brindle. *Genetic Algorithms for Function Optimization*. Unpublished PhD thesis, University of Alberta, 1981.

[Bruns 93] R. Bruns. Direct chromosome representation and advanced genetic operators for production scheduling. In Stephanie Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 352–359. San Mateo: Morgan Kaufmann, 1993.

[Burke *et al.* 95] E. Burke, D. Elliman, and R. Weare. The Automated Timetabling of University Exams using a Hybrid Genetic Algorithm. In *The AISB Workshop on Evolutionary Computing*, 1995.

[Campbell *et al.* 70] H. G. Campbell, R. A. Dudek, and M. L. Smith. A heuristic algorithm for the n-job, m-machine sequencing problem. *Management Science*, 16(10):630–637, 1970.

[Carlton & Barnes 95] W.B. Carlton and J.W. Barnes. A Note on Hashing Functions and Tabu Search Algorithms. *European Journal of Operational Research (submitted)*, 1995.

[Cartwright & Harris 93] Hugh M. Cartwright and Stephen P. Harris. Analysis of the distribution of airborne pollution using genetic algorithms. *Atmospheric Environment*, 27:1783–1791, 1993.

[Cartwright & Tuson 94] Hugh M. Cartwright and Andrew L. Tuson. Genetic algorithms and flowshop scheduling: towards the development of a real-time process control system. In Terry C. Fogarty, editor, *Selected Papers: AISB Workshop on Evolutionary Computing, Lecture Notes in Computer Science No 865*, pages 277–290. Springer Verlag, 1994.

[Chalmers 98] A. F. Chalmers. *What is this thing called Science?* Open University Press, 1998.

[Chandrasekaran *et al.* 92] B. Chandrasekaran, T. R. Johnson, and J. W. Smith. Task-Structure Analysis for Knowledge Modelling. *Communications of the ACM*, 35(9):124–137, 1992.

[Charon & Hurdy 93] I. Charon and O. Hurdy. The noising method. a new method for combinatorial optimization. *Operations Research Letters*, 14:133–137, 1993.

[Cheng & Smith 95] C.-C. Cheng and S. F. Smith. Applying Constraint Satisfaction Techniques to Job Shop Scheduling. Technical report, Carnegie Mellon University, 1995.

[Cleveland & Smith 89] Gary A. Cleveland and Stephen F. Smith. Using Genetic Algorithms to Schedule Flow Shop Releases. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms and their Applications*, pages 160–169. San Mateo: Morgan Kaufmann, 1989.

[Cohen 95] P. R. Cohen. *Empirical Methods for Artificial Intelligence*. MIT Press, 1995.

[Cohen 96] P. R. Cohen. Getting what you deserve from data. *IEEE Expert*, pages 12–14, October 1996.

[Collins *et al.* 88] N. E. Collins, R. W. Eglese, and B. L. Golden. Simulated annealing — an annoted biblography. *AJMMS*, 8:209–307, 1988.

[Cook 71] S. A. Cook. The complexity of theorem-proving procedures. In *Proceeedings of the 3rd Annual ACM Symposium on Theory of Computing*, pages 151–158, New York, 1971.

[Corne *et al.* 93] D. Corne, H.-L. Fang, and C. Mellish. Solving the Module Exam Scheduling Problem with Genetic Algorithms. In Paul W. H. Chung, Gillian Lovegrove, and Moonis Ali, editors, *Proceedings of the Sixth International Conference in Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, pages 370–373. Gordon and Breach Science Publishers, 1993.

[Corwin & Esogbue 74] B. D. Corwin and A. O. Esogbue. Two machine flowshop scheduling problems with sequence dependant set-up times: a dynamic programming approach. *Naval Research Logistics Quarterly*, 21:695–705, 1974.

[Culberson 95] J. Culberson. Mutation-Crossover Isomorphisms and the Construction of Discriminating Functions. *Evolutionary Computation*, 2(3):279–311, 1995.

[Culberson 96] J. Culberson. On the Futility of Blind Search. Technical Report TR96-18, University of Alberta, Canada, 1996.

[Cunningham & Dutta 72] A. A. Cunningham and S. K. Dutta. Scheduling Jobs with Exponetially Distributed Processing Times on Two Machines of a Flow Shop. *Naval Research Logistics Quarterly*, 19, 1972.

[Dannenbring 77] D. G. Dannenbring. An Evaluation of Flowshop Sequencing Heuristics. *Manag. Sci.*, 23:1174–1182, 1977.

[Darwin 59] C. Darwin. *On the Origin of Species*. John Murray, London, 1859.

[Davidor 90] Y. Davidor. Epistatis Variance: Suitability of a representation to genetic algorithms. *Complex Systems*, 4:369–383, 1990.

[Davis 89] L. Davis. Adapting Operator Probabilites in Genetic Algorithms. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms and their Applications*, pages 61–69. San Mateo: Morgan Kaufmann, 1989.

[Davis 91] Lawrence Davis. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York, 1991.

[DeJong 75] K. A. DeJong. *Analysis of Behavior of a Class of Genetic Adaptive Systems*. Unpublished PhD thesis, The University of Michigan, 1975.

[Desain & Honing 92] P. Desain and H. Honing. *Music, Mind and Machine*. Thesis Publishers Amsterdam, 1992.

[Domingue *et al.* 93] J. Domingue, E. Motta, and Watt. The Emerging Vital Workbench. In N. Aussenac, G. Boy, B. Gaines, M. Linster, J.-G. Ganascia, and Y. Kodratoff, editors, *Knowledge Acquisition for Knowledge-Based Systems — 7th European Workshop (EKAW'93)*, pages 320–339, Berlin, 1993. Springer-Verlag.

[Donha *et al.* 97] D Donha, D. Desanj, and M. Katebi. Automatic Weight Selection for $H_\infty$ Control Design. In *The Second International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA 97)*, 1997.

[Dowsland 91] K. A. Dowsland. Hill-climbing, simulated annealing, and the Stenier problem in graphs. *Eng. Opt.*, 17:91–107, 1991.

[Dudek *et al.* 92] R. A. Dudek, S. S. Panwalker, and M. L. Smith. The Lessons of Flowshop Sequencing Research. *Operations Research*, 1992.

[Dueck & Scheuer 90] G. Dueck and T. Scheuer. Threshold Accepting: A General Purpose Optimisation Algorithm Superior to Simulated Annealing. *Journal of Computation Physics*, 90:161–175, 1990.

[Dueck 90] G. Dueck. New Optimisation Heuristics: The Great Deluge Algorithm and the Record-to-Record Travel. Technical report, IBM Germany, Heidelburg Scientific Center, 1990.

[Eiben 96] A.E. Eiben. Evolutionary Exploration of Search Spaces. In *Foundations of Intelligent Systems (LNAI 1079)*, pages 178–188. Springer Verlag, 1996.

[Eiben *et al.* 95] A. E. Eiben, E. H. L. Aarts, K. M. van Hee, and W. P. M. Nuijten. A unifying approach to heuristic search. *Annals of Operations Research*, 55:81–99, 1995.

[English 96] T. English. Evaluation of Evolutionary and Genetic Optimizers: No Free Lunch. In L. Fogel, P. Angeline, and T. Bäck, editors, *Evolutionary Programming V: Proceedings of the Fifth Annual Conference on Evolutionary Programming*, pages 163–169. MIT Press, 1996.

[English 98] T. English. Information Is Conserved in Optimization. *IEEE Transactions on Evolutionary Computation (under review)*, 1998.

[Faigle & Kern 92] U. Faigle and W. Kern. Some Convergence Results for Probabilistic Tabu Search. *ORSA Journal on Computing*, 4:32–37, 1992.

[Fang 94] Hsiao-Lan Fang. *Genetic Algorithms in Timetabling and Scheduling*. Unpublished PhD thesis, Department of Artificial Intelligence, University of Edinburgh, 1994.

[Fensel & Schoenegge 97] D. Fensel and A. Schoenegge. Specifying and Verifying Knowledge-Based Systems with KIV. In *Proceedings of the European Symposium on the Validation and Verification of Knowledge Based Systems — EUROWAV'97*, Leuven, Belgium, June 1997.

[Fensel *et al.* 97] D. Fensel, E. Motta, S. Decker, and Z. Zdrahal. The use of Ontologies for Specifying Tasks and Problem Solving Methods: A Case Study. In *10th European Workshop on Knowledge Acquisition, Modeling, and Management*, 1997.

[Feo *et al.* 91a] T. A. Feo, J. F. Bard, and R. F. Claflin. An overview of GRASP methodology and applications. Technical report, University of Texas at Austin, 1991.

[Feo *et al.* 91b] T. A. Feo, K. Venkatraman, and J. F. Bard. A GRASP for a Difficult Single Machine Scheduling Problem. *Computers Ops. Res.*, 17(8):635–643, 1991.

[Fogel & Ghozeil 97] D. B. Fogel and A. Ghozeil. A Note on Representations and Variation Operators. *IEEE Transactions on Evolutionary Computation*, 1(2):159–161, 1997.

[Fogel *et al.* 66] L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial Intelligence Through Simulated Evolution*. John Wiley and Sons, Inc., 1966.

[Fonseca & Fleming 95] C. M. Fonseca and P. J. Fleming. An Overview of Evolutionary Algorithms in Multiobjective Optimization. *Evolutionary Computation*, 3(1):1–16, 1995.

[Fox 93] B.L. Fox. Integrating and accelerating tabu search, simulated annealing and genetic algorithms. *Annals of Operations Research*, 41:46–67, 1993.

[Garey & Johnson 79] Michael R. Garey and David S. Johnson. *Computers and Intractability: a Guide to the Theory of NP-Completeness*. Freeman, 1979.

[Giffler & Thompson 60] B. Giffler and G. L. Thompson. Algorithms for solving production scheduling problems. *Operations Research*, 8:487–503, 1960.

[Glib 98] T. Glib. *Principles of Software Engineering Management*. Addison-Wesley, 1998.

[Glover & Laguna 97] F. W. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, 1997.

[Glover 66] F. Glover. An Algorithm for Solving the Linear Integer Programming Problem over a Finite Additive Group, with Extensions to Solving General and Certain Non-linear Integer Program. Technical report, CRC 66-29, University of California at Berkeley, 1966.

[Glover 68] F. Glover. Surrogate Constraints. *Operations Research*, 16:741–749, 1968.

[Glover 89] F. Glover. Tabu Search — Part I. *ORSA Journal on Computing*, 1:190–206, 1989.

[Glover 90a] F. Glover. Tabu Search — Part II. *ORSA Journal on Computing*, 2, 1990.

[Glover 90b] F. Glover. Tabu Search: A Tutorial. *Interfaces*, 4:445–460, 1990.

[Glover 94] F. Glover. Tabu search fundamentals and uses. Technical report, University of Colorado at Boulder, 1994.

[Glover 96] F. Glover. Ejection Chains, Reference Structures and Alternating Path Methods for Travelling Salesman Problems. *Discrete Applied Mathematics*, 65:223–253, 1996.

[Glover *et al.* 93] F. Glover, E.D. Taillard, and D. de Werra. A user's guide to tabu search. *Annals of Operations Research*, 41, 1993.

[Goldberg & Deb 91] David E. Goldberg and Kalyanmoy Deb. A comparative analysis of selection schemes used in genetic algorithms. In G. J. E. Rawlins, editor, *A comparative analysis of selection schemes* *used in genetic algorithms*, pages 69–93. Morgan Kaufmann, 1991.

[Goldberg 89a] D. E. Goldberg. Genetic algorithms and Walsh functions: Part I, a gentle introduction. *Complex Systems*, 3(129-152), 1989.

[Goldberg 89b] D. E. Goldberg. Genetic algorithms and Walsh functions: Part II, deception and its analysis. *Complex Systems*, 3(153-171), 1989.

[Goldberg 89c] David E. Goldberg. *Genetic Algorithms in Search, Optimization & Machine Learning*. Reading: Addison Wesley, 1989.

[Goldberg 95] D. E. Goldberg. The existential pleasures of genetic algorithms. In G. Winter, J. Periaux, M. Galan, and P. Cuesta, editors, *Genetic Algorithms in Engineering and Computer Science*, pages 32–31. John Wiley and Sons, 1995.

[Gonzales-Hernandez 95] L. Gonzales-Hernandez. Evolutionary Divide and Conquer for the Set-Covering Problem. Unpublished M.Sc. thesis, Department of Artificial Intelligence, University of Edinburgh, 1995.

[Graves 81] S. C. Graves. A Review of Production Scheduling. *Operations Research*, 1981.

[Grefenstette 87] J.J. Grefenstette. Incorporating problem specific knowledge into genetic algorithms. In Lawrence Davies, editor, *Genetic Algorithms and Simulated Annealing*, pages 42–60. Morgan Kaufmann, 1987.

[Grefenstette 93] J. J. Grefenstette. Deception considered harmful. In L. Darrell Whitley, editor, *Foundations of Genetic Algorithms 2*, pages 75–91. San Mateo: Morgan Kaufmann, 1993.

[Grefenstette *et al.* 85] J. J. Grefenstette, R. Gopal, B. Rosmaita, and D. Van Gucht. Genetic Algorithm for the TSP. In J. J. Grefenstette, editor, *Proceedings of the International Conference on Genetic Algorithms and their Applications*, pages 160–168. San Mateo: Morgan Kaufmann, 1985.

[Hackett 96] D. P. Hackett. An ITS To Teach Malaria Stratification For the WHO - A Prototype. Unpublished M.Sc. thesis, Department of Artificial Intelligence, Univeristy of Edinburgh, 1996.

[Hajek & Sasaki 89] B. Hajek and G. Sasaki. Simulated Annealing — to Cool or Not. *Systems and Control Letters*, 12:443–447, 1989.

[Hancock 92] P. J. B. Hancock. *Coding strategies for genetic algorithms and neural nets*. Unpublished PhD thesis, Department of Computing Science and Mathematics, University of Stirling, UK, 1992.

[Hancock 94] P. J. B. Hancock. An empirical comparison of selection methods in evolutionary algorithms. In T. C. Fogarty, editor, *Evolutionary Computing: AISB Workshop, Selected Papers, Lecture Notes in Computer Science 865*, pages 80–94. Springer Verlag, 1994.

[Hansen 86] P. Hansen. The steepest ascent, mildest decent heuristic for combinatorial programming. In *The Congress on Numerical Methods in Combinatorial Optimisation*, 1986.

[Harrison 99] A. Harrison. Personal Communication, 1999.

[Hart & Tuson 98] E. Hart and A. L. Tuson. An investigation of a heuristic representation for the flowshop sequencing problem. Unpublished Work, 1998.

[Hart *et al.* 68] P. E. Hart, N. J. Nilsson, and B. Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions of Systems, Science, and Cybernetics*, 4(2):100–107, 1968.

[Hasan & Osman 95] M. Hasan and I. H. Osman. Local search algorithms for the maximal planar layout problem. *International Transactions in Operations Research*, 2:89–106, 1995.

[Hertz & deWerra 87] A. Hertz and D. de Werra. Using tabu search techniques for graph coloring. *Computing*, 29:345–351, 1987.

[Hertz *et al.* 97] A. Hertz, E. D. Taillard, and D. de Werra. Tabu Search. In E. Aarts and J. K. Lenstra, editors, *Local search in combinatorial optimization*. J. Wiley & Sons Ltd, 1997.

[Hjorring 95] C. A. Hjorring. *The Vehicle Routing Problem and Local Search Metaheuristics*. Unpublished PhD thesis, The University of Auckland, New Zealand, 1995.

[Ho & Chang 91] J.C. Ho and Y.-H. Chang. A new heuristic for the n-job, M-machine flow-shop problem. *European Journal of Operational Research*, 52:194–202, 1991.

[Hofmann 93] R. Hofmann. Examinations on the Algebra of Genetic Algorithms. Diploma Thesis, Technical University of Munich, 1993.

[Holland 75] John H. Holland. *Adaptation in Natural and Artificial Systems*. Ann Arbor: The University of Michigan Press, 1975.

[Hordijk 96] W. Hordijk. A Measure of Landscapes. *Evolutionary Computation*, 4(4):335–360, 1996.

[Ignall & Schrage 65] E. Ignall and L. Schrage. Application of the branch-and-bound technique to some flow shop scheduling problems. *Operations Research*, 13:400–412, 1965.

[Inder *et al.* 90] R. Inder, R. Aylett, D. Bental, T. Lydiard, and R. Rae. Study on the Evaluation of Expert Systems Tools for Ground Segment Infrastructure: Final Report. Technical Report AIAI-TR-84, Artificial Intelligence Applications Institute, Edinburgh University, October 1990.

[Ingber & Rosen 92] L. Ingber and B. Rosen. Genetic algorithms and very fast simulated annealing - a comparison. *Mathematical and Computer Modeling*, 16(11):87–100, Nov 1992.

[Jelasity & Dombi 96] M. Jelasity and J. Dombi. Implicit Formae in Genetic Algorithms. In H.-M. Voigt et al, editor, *Parallel Problem-solving from Nature - PPSN IV*, LNCS, pages 154–163. Springer-Verlag, 1996.

[Johnson & McGeoch 96] D. S. Johnson and L. A. McGeoch. The Travelling Salesman Problem: A Case Study in Local Optimsation. In E. H. L. Aarts and J. K. Lenstra, editors, *Local Search in Combinatorial Optimisation*, New York, 1996. Wiley and Sons.

[Johnson 54] S. M. Johnson. Optimal two and three stage production schedules with set-up times included. *Naval Res. Logist. Q.*, 61, 1954.

[Johnson 90] D. S. Johnson. Local Optimisation and the Travelling Salesman Problem. In *Proceedings of the 17th Interational Colloqium on Automata, Languages and Programming*, pages 446–460, 1990.

[Jones & Forrest 94] T. Jones and S. Forrest. Genetic algorithms and heuristic search. Research Paper, Santa Fe Institute, December 1994.

[Jones & Forrest 95] T. Jones and S. Forrest. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In L. J. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, San Francisco, Ca., 1995. Morgan Kaufmann.

[Jones 95] T. Jones. *Evolutionary Algorithms, Fitness Landscapes and Search*. Unpublished PhD thesis, University of New Mexico, 1995.

[Jonker & Spee 92] W. Jonker and J. W. Spee. Yet Another Formalization of KADS Conceptual Models. In Th. Wetter, K.-D. Althoff, J. Boose, B. R. Gaines, M. Linster, and F. Schmalhofer, editors, *Current Developments in Knowledge Acquisition — EKAW'92*, LNAI 599, pages 112–132, Berlin, 1992. Springer Verlag.

[Kan 76] A. H. G. Rinnooy Kan. *Machine Sequencing Problems: Classification, complexity and computations*. Martinus Nijhoff, The Hague, 1976.

[Kaplan 73] S. Kaplan. Readiness and the Optimal Redeployment of Resources. *Naval Logistics Research Quarterly*, 20:625–638, 1973.

[Kappler *et al.* 96] C. Kappler, T. Bäck, J. Heistermann, A. van de Velde, and M. Zamparelli. Refueling of a nuclear power plant: comparison of a naive and a specialized mutation operator. In H-M. Voigt, W. Ebeling, I. Rechenberg, and H-P. Schwefel, editors, *Parallel Problem-Solving from Nature—PPSN IV*, pages 829–838. Springer-Verlag, Berlin, 1996.

[Karimi & Ku 88a] I. A. Karimi and H. M. A. Ku. A Modified Heuristic for an Initial Sequence in Flowshop Scheduling. *Ind. Eng. Chem. Res.*, 27:1654–1658, 1988.

[Karimi & Ku 88b] I. A. Karimi and H. M. A. Ku. Scheduling in Multiproduct Batch Processes with Finite Interstage Storage: A Mixed Linear Program Formulation. *Ind. Eng. Chem. Res.*, 27:1840–1848, 1988.

[Kauffman 89] S. A. Kauffman. *Origins of Order: Self-Organization and Selection in Evolution*. Oxford University Press, Oxford, 1989.

[Keeney & Raiffa 76] R. Keeney and H. Raiffa. *Decisions with Multiple Objectives, Preferences, and Value Tradeoffs*. John Wiley, New York, 1976.

[King & Beck 90] K. King and H. Beck. Medical AI systems as appropriate technology for developing countries. *Knowledge Engineering Review*, 5(4), 1990.

[Kirkpatrick *et al.* 83] S. Kirkpatrick, C.D. Gelatt, Jr., and M.P. Vecchi. Optimization by Simulated Annealing. *Science*, 220:671–680, 1983.

[Kline & Dolins 89] P. Kline and S. Dolins. *Designing Expert Systems: A guide to selecting implementation techniques*. John Wiley and Sons, London, 1989.

[Kolodner 93] J. L. Kolodner. *Case-based reasoning*. Morgan Kaufmann Publishers, 1993.

[Koza 92] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, 1992.

[Koza *et al.* 96a] J. R. Koza, F. H. Bennett III, D. Andre, and M. A. Keane. Automated wywiwyg design of both the topology and component values of analog electrical circuits using genetic programming. In J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, editors, *Genetic Programming 1996: Proceedings of the First Annual Conference, July 28-31, 1996, Stanford University*, pages 123–131, Cambridge, MA, 1996. MIT Press.

[Koza *et al.* 96b] J. R. Koza, F. H. Bennett III, D. Andre, and M. A. Keane. Four problems for which a computer program evolved by genetic programming is competitive with human performance. In *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation*, pages 1–10. IEEE Press, 1996.

[Ku *et al.* 87] H. M. A. Ku, D. Rajagopalan, and I. A. Karimi. Scheduling in Batch Processes. *Chem. Eng. Prog.*, 83(8), 1987.

[Laguna *et al.* 94] M. Laguna, T. Feo, and H. Elrod. A Greedy Randomized Adaptive Search Procedure for the 2-Partion Problem. *Operations Research*, 42(4):667–687, 1994.

[Laguna *et al.* 95] M. Laguna, J.P. Kelly, J.L. Gonzalez-Velarde, and F. Glover. Tabu Search for the Multilevel Generalized Assignemnt Problem. *European Journal of Operations Research*, 82:176–189, 1995.

[Langdon 98] W. B. Langdon. *Data Structures and Genetic Programming: Genetic Programming + Data Structures = Automatic Programming!* Kluwer Academic Publishers, 1998.

[Lenat & Feigenbaum 91] D. B. Lenat and E. Feigenbaum. On the thresholds of knowledge. *Artificial Intelligence*, 47(1-3):185–250, 1991.

[Levesque 86] H. J. Levesque. Making believers out of computers. *Artificial Intelligence*, 30(1):81–108, 1986.

[Lin & Kernighan 73] S. Lin and B. W. Kernighan. An Effective Heuristic Algorithm for the Travelling-Salesman Problem. *Operations Research*, 31:498–516, 1973.

[Lomnicki 65] Z. Lomnicki. A branch-and-bound solution for the exact solution of the three-machine scheduling problem. *Operational Research Quarterly*, 1965.

[Luck & Walsham 97] E. Luck and R. Walsham. *Selected Readings in OR for Developing Countries*. Operational Research Society, 1997.

[Lundy & Mees 86] M. Lundy and A. Mees. Convergence of an annealing algorithm. *Math. Prog.*, 34:111–124, 1986.

[Maini *et al.* 94] H. Maini, K. Mehrotra, C. Mohan, and S. Ranka. Knowledge-Based Nonuniform Crossover. *Complex Systems*, 8:257–293, 1994.

[Manderick *et al.* 91] B. Manderick, M. De Weger, and P. Spiessens. The Genetic Algorithm and the Structure of the Fitness Landscape. In R. K. Belew and L .B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 143–150. Morgan Kaufmann, 1991.

[Marcus 98] S. Marcus. *Automatic Knowledge Acquistion for Expert Systems*. Kluwer Academic, Boston, MA, 1998.

[Marcus *et al.* 98] S. Marcus, J. Stout, and J. McDermott. VT: An Expert Elevator Designer the uses Knowledge-Based Backtracking. *AI Magazine*, 9(1):95–112, 1998.

[Mattfeld *et al.* 96] D. Mattfeld, C. Bierwirth, and H. Kopfer. On Permutation Representations for Scheduling Problems. In H.-M. Voigt et al, editor, *Parallel Problem-solving from Nature - PPSN IV*, LNCS, pages 310–318. Springer-Verlag, 1996.

[McGeoch 86] C. McGeoch. *Experimental Analysis of Algorithms*. Unpublished PhD thesis, Carnegie Mellon University, 1986. Also available as CMU-CS-87-124.

[McMahn & Burton 67] G. B. McMahn and P. G. Burton. Flow shop scheduling with the branch-and-bound method. *Operations Research*, 15:473–481, 1967.

[Michalewicz 92] Z. Michalewicz. *Genetic algorithms + data structures = evolution programs*. Artificial Intelligence. Springer-Verlag, New York, 1992.

[Miller 94] G. F. Miller. Exploiting mate choice in evolutionary computation: Sexual selection as a process of search, optimization, and diversification. In T. C. Fogarty, editor, *Proceedings of the 1994 AISB Workshop on Evolutionary Computing*. Springer-Verlag, 1994.

[Minton *et al.* 90] S. Minton, A. Phillips, M. Johnston, and P. Laird. Solving Large Scale CSP and Scheduling Problems with a Heuristic Repair Method. In *Proceedings of AAAI-90*, 1990.

[Mitchell 80] T. Mitchell. The need for biases in learning generalizations. Technical Report CBM-TR-117, Rutgers University, 1980.

[Mitchell 96] M. Mitchell. *An Introduction to Genetic Algorithms*. MIT, 1996.

[Moccellin 95] J. V. Moccellin. A New Heuristic Method for the Permutation Flow Shop Sequencing Problem. *Journal of the Operational Research Society*, 46:883–886, 1995.

[Mott 90] G. F. Mott. Optimising Flowshop Scheduling Through Adaptive Genetic Algorithms. Chemistry Part II Thesis, Oxford University, 1990.

[Motta 97] E. Motta. *Resusable Components in Knowledge Modelling*. Unpublished PhD thesis, Open University, UK, November 1997.

[Mühlenbein 89] H. Mühlenbein. Parallel genetic algorithms, population genetics, and combinatorial optimisation. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 416–421, San Mateo, CA, 1989. Morgan Kaufmann.

[Musen 89] M. A. Musen. *Automated Generation of Model-Based Knowledge Acquisition Tools*. Research Notes in Artificial Intelligence. Pitman, London, 1989.

[Nakata 97] S. Nakata. A Study of Modern Combinatorial Optimisation Approaches to the Travelling Salesman Problem. AI4 Project Report, Department of Artifical Intelligence, Edinburgh University, 1997.

[Nawaz *et al.* 83] M. Nawaz, Jr. E.E. Emscore, and I. Ham. A heuristic algorithm for the m-machine, n-job flowshop sequencing problem. *OMEGA*, 11:91–95, 1983.

[Neches *et al.* 91] R. Neches, R. Fikes, T. Finin, T. Gruber, , R. Patil, T. Senator, and W. Swartout. Enabling technology for knowledge sharing. *AI Magazine*, 12(3):37–56, 1991.

[Newell & Simon 76] A. Newell and H. A. Simon. Computer Science as empirical enquiry: Symbols and search. *Communications of the ACM*, 9(3):113–126, 1976.

[Newell 82] A. Newell. The Knowledge Level. *Artificial Intelligence*, 18(1):87–127, 1982.

[Nix & Vose 91] A. Nix and M. D. Vose. Modelling genetic algorithms with Markov chains. *Annals of Mathematics and Artificial Intelligence*, 5:79–88, 1991.

[Nowicki & Smutnicki 96] E. Nowicki and C. Smutnicki. A fast tabu search algorithm for the permutation flow-shop problem. *European Journal of Operational Research*, 91:160–175, 1996.

[Ogbu & Smith 90] F. A. Ogbu and D. K. Smith. The application of the simulated annealing algorithm to the solution of the $n/m/P/C_{max}$ flowshop problem. *Computers and Ops. Res.*, 1990.

[Oliveira & Stroud 89] S. Oliveira and G. Stroud. A parallel version of tabu search and the path assignment problem. *Heuristics for Combinatorial Optimisation*, 4:1–24, 1989.

[Osman & Christofides 94] I. H Osman and N. Christofides. Capacitated clustering problems by hybrid simulated annealing and tabu search. *International Transactions in Operations Research*, 1:317–336, 1994.

[Osman & Kelly 96] I. H. Osman and J. P. Kelly. *Meta-Heuristics. Theory and Applications*. Kluwer Academic Publishers, 1996.

[Osman & Laporte 95] I. H. Osman and G. Laporte. Metaheuristics for combinatorial optimisation problems: An annotated bibliography. *Annals of Operational Research*, 63:513–628, 1995.

[Osman & Potts 89] I. H. Osman and C. N. Potts. Simulated annealing for permutation flow-shop scheduling. *OMEGA*, 17:551–557, 1989.

[Osman 95] I. H. Osman. An introduction to Meta-Heuristics. In M. Lawrence and C. Wilsdon, editors, *Operational Research Tutorial Papers*, pages 92–122. Operational Research Society, 1995.

[PAHO 98] PAHO. SUMA — Humanitarian Supply Management System. Pan-American Health Organisation Information WWW Page (obtainable from `http://www.disaster.info.desastres.net/SUMA/`), 1998.

[Palmer 65] D. S. Palmer. Sequencing jobs through a multi-stage process in the minimum total time — a quick method of obtaining a near optimum. *Operational Research Quarterly*, 16:101–107, 1965.

[Papadimitriou & Steiglitz 82] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimisation: Algorithms and Complexity*. Prentice-Hall, 1982.

[Phon-Amnuaisuk *et al.* 99] S. Phon-Amnuaisuk, A. Tuson, and G. Wiggins. Evolving Musical Harmony. In *The Fourth International Conference on Artificial Neural Networks and Genetic Algorithms (to appear)*, 1999.

[Pirlot 93] M. Pirlot. General local search heuristics in combinatorial optimisation: a tutorial. *Belgian Journal of OR, Statistics and Computer Science*, 32:7–67, 1993.

[Poli & Logan 96] R. Poli and B. Logan. The evolutionary computation cookbook: Recipes for designing new algorithms. In *Proceedings of the Second Online Workshop on Evolutionary Computation*, Nagoya, Japan, March 1996.

[Polya 57] G. Polya. *How to Solve it: A New Aspect of Mathematical Method*. Doubleday, 1957.

[Price 70] G. R. Price. Selection and co-variance. *Nature*, 227:520–521, 1970.

[Prügel-Bennet & Shapiro 94] A. Prügel-Bennet and J. Shapiro. Analysis of genetic algorithms using statistical mechanics. *Physics Review Letters*, 72:1305, 1994.

[Radcliffe & Surry 94a] N. J. Radcliffe and P. D. Surry. Fitness Variance of Formae and Performance Prediction. In L. D. Whitley and M. D. Vose, editors, *Foundations of Genetic Algorithms III*. Morgan Kaufmann, 1994.

[Radcliffe & Surry 94b] N.J Radcliffe and P. D. Surry. Formal memetic algorithms. In T. C. Fogarty, editor, *Proceedings of the AISB workshop on Evolutionary Computation*. Springer-Verlag, 1994.

[Radcliffe & Surry 96] N. J. Radcliffe and P. D. Surry. Fundamental Limitations on Search Algorithms: Evolutionary Computing in Perspective. In J. van Leeuwen, editor, *Computer Science Today: Recent Trends and Developments, LNCS 1000*. Springer Verlag, 1996.

[Radcliffe 91a] N. J. Radcliffe. Equivalence Class Analysis of Genetic Algorithms. *Complex Systems*, 5(2):183–205, 1991.

[Radcliffe 91b] N. J. Radcliffe. Forma analysis and random respectful recombination. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 222–9. San Mateo: Morgan Kaufmann, 1991.

[Radcliffe 92] N. J. Radcliffe. Non-Linear Genetic Representations. In R. Maenner and B. Manderick, editors, *Parallel Problem Solving from Nature 2*, pages 259–268. Elsevier Science, 1992.

[Radcliffe 94] N.J Radcliffe. The Algebra of Genetic Algorithms. *Annals of Maths and Artificial Intelligence*, 10:339–384, 1994.

[Rajendran & Chaudhuri 91] C. Rajendran and D. Chaudhuri. An efficient heuristic approach to the scheduling of jobs in a flowshop. *European Journal of Operational Research*, 61:318–325, 1991.

[Ramos 97] S. Ramos. A Neighbourhood Search Approach to FPGA Design. Unpublished M.Sc. thesis, Department of Artificial Intelligence, Univeristy of Edinburgh, 1997.

[Ratford 96] M. Ratford. The Single Chromosome's Guide To Dating. Unpublished M.Sc. thesis, Department of Artificial Intelligence, University of Edinburgh, 1996.

[Ratford *et al.* 97] M. Ratford, A. Tuson, and H. Thompson. The Single Chromosome's Guide To Dating. In *Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms*, 1997.

[Rayward-Smith 94] V. J. Rayward-Smith. A Unified Approach To Tabu Search, Simulated Annealing and Genetic Algorithms. In *The Proceedings of the UNICOM Seminar on Adaptive Computing and Information Processing*, 1994.

[Rechenburg 73] I. Rechenburg. *Evolutionstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution*. Frommann-Holzberg, 1973.

[Reeves & Wright 95a] C. R. Reeves and C. C. Wright. An experimental design perspective on genetic algorithms. In D. Whitley and M. Vose, editors, *Foundations of Genetic Algorithms 3*. San Mateo: Morgan Kaufmann, 1995.

[Reeves & Wright 95b] C. R. Reeves and C. C. Wright. Epistasis in genetic algorithms: an experimental design perspective. In I. J. Eshelman, editor, *The 6th International Conference on Genetic Algorithms*, pages 217–224. San Mateo: Morgan Kaufmann, 1995.

[Reeves & Wright 97] C. R. Reeves and C. C. Wright. Genetic algorithms and the design of experiments. In D.Whitley, editor, *Proc. IMA Fall Workshop on Evolutionary Algorithms*, Minneapolis, October 1997.

[Reeves 92] C. R. Reeves. A genetic algorithm approach to stochastic flowshop sequencing. In *Proc.IEE Colloquium on Genetic Algorithms for Control and Systems Engineering*. Digest No.1992/106. IEE, London, 1992.

[Reeves 93a] C. R. Reeves. Improving the efficiency of tabu search in machine sequencing problems. *J.Opl.Res.Soc.*, 44:375–382, 1993.

[Reeves 93b] C. R. Reeves. *Modern Heuristic Techniques for Combinatorial Problems*. Blackwell Scientific Publications, 1993.

[Reeves 94a] C. R. Reeves. Genetic algorithms and neighbourhood search. In Terry C. Fogarty, editor, *Selected Papers: AISB Workshop on Evolutionary Computing, Lecture Notes in Computer Science No 865*. Springer Verlag, 1994.

[Reeves 94b] C. R. Reeves. Non-biological metaphors in genetic algorithms. In *Proc. of the 2nd Finnish Workshop on Genetic Algorithms*. University of Vaasa, Finland, 1994.

[Reeves 95a] C. R. Reeves. A genetic algorithm for flowshop sequencing. *Computers & Ops. Res.*, 22:5–13, 1995.

[Reeves 95b] C. R. Reeves. Restricted neighbourhood search: an implicit tabu method. In F.Glover, J.P.Kelly, and I.H.Osman, editors, *Proc. 1$^{st}$ International Conference on Metaheuristics*. University of Colorado, 1995.

[Reeves 98] C. R. Reeves. Landscapes, operators and heuristic search. *Annals of OR*, 1998.

[Ridley 94] M. Ridley. *The red queen : sex and the evolution of human nature*. Penguin, 1994.

[Rodammer & White 88] F. A. Rodammer and K. P. White. A Recent Survey of Production Scheduling. *IEEE Transactions on Systems, Man, and Cybernetics*, 18(6):841–851, 1988.

[Ronald 98] S. Ronald. More distance functions for order-based encodings. In D. Fogel, editor, *Proceedings of the 3rd IEEE Interational Conference on Evolutationary Computation (ICEC'98)*, pages 558–563. IEEE, IEEE Press, 1998.

[Rose *et al.* 96] H. Rose, W. Ebeling, and T. Asselmeyer. The density of states — a measure of the difficulty of optimisation problems. In H-M. Voigt, W. Ebeling, I. Rechenberg, and H-P. Schwefel, editors, *Parallel Problem-Solving from Nature—PPSN IV*, pages 208–217. Springer-Verlag, Berlin, 1996.

[Ross & Corne 95] P. Ross and D. Corne. Comparing Genetic Algorithms, Simulated Annealing, and Stochastic Hillclimbing on Timetabling Problems. In T. C. Fogarty, editor, *Evolutionary Computing; AISB Workshop, Sheffield 1995, Selected Papers. Springer-Verlag Lecture Notes in Computer Science 993*, 1995.

[Ross 96] Peter Ross. Personal communication, 1996.

[Ross 98] P. Ross. Personal Communication, 1998.

[Ross *et al.* 94] P. Ross, D. Corne, and H.-L. Fang. Improving evolutionary timetabling with delta evaluation and directed mutation. In Y. Davidor, H-P. Schwefel, and R. Manner, editors, *Parallel Problem-solving from Nature — PPSN III*, LNCS, pages 566–565. Springer-Verlag, 1994.

[Ross *et al.* 96] P. Ross, E. Collingwood, and D. Corne. Predicting the Success of a GA. In *East-West Joint Conference on Evolutionary Computing*, 1996.

[Royce 70] W. W. Royce. Managing the development of large software systems: concepts and techniques. In *Proceedings IEEE WESCON*, pages 1–9, 1970.

[Russell & Norvig 95] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Series in Artificial Intelligence. Prentice Hall, Englewood Cliffs, NJ, 1995.

[Salomon 96] R. Salomon. The influence of different coding schemes on the computational complexity of genetic algorithms in function optimization. In H-M. Voigt, W. Ebeling, I. Rechenberg, and H-P. Schwefel, editors, *Parallel Problem-Solving from Nature—PPSN IV*, pages 227–235. Springer-Verlag, Berlin, 1996.

[Sarin & Lefoka 93] S. Sarin and M. Lefoka. Scheduling Heuristic for the n-Job m-Machine Flow Shop. *OMEGA*, 21(2):229–234, 1993.

[Schaffer 84] J. D. Schaffer. *Some experiments in machine learning using vector evaluated gentic algorithms*. Unpublished PhD thesis, Vanderbilt University, 1984.

[Schaffer 85] J. D. Schaffer. Multiple objective optimisation with vector evaluated genetic algorithms. In J. J. Grefenstette, editor, *Proceedings of the International Conference on Genetic Algorithms and their Applications*, pages 93–100. San Mateo: Morgan Kaufmann, 1985.

[Schwefel 97] H.-P. Schwefel. Challenges to and future developments of evolutionary algorithms. In T. Bäck, D. B. Fogel, and Z. Michalewicz, editors, *Handbook of Evolutionary Computation*, chapter H 1.3. IOP Publishing Ltd and Oxford University Press, 1997.

[Selman *et al.* 92] B. Selman, H. Levesque, and D. Mitchell. A new method for solving hard satisfiability problems. In *Proceedings of the 10th National Conference on Artificial Intelligence*, pages 440–446, San Jose, California, USA, July 1992. AAAI, AAAI Press.

[Semet & Taillard 93] F. Semet and E. Taillard. Solving real-life vehicle routing problems effectively using taboo search. *Annals of Ops. Res.*, 41, 1993.

[Sen *et al.* 96] A. Sen, A. Bagchi, and R. Ramaswamy. Search Graphs with A*: Applications to Job Scheduling. *IEEE Transactions on Systems, Man, and Cybernetics*, 26(1):168–173, 1996.

[Sinclair 93] M. Sinclair. Comparision of the performance of modern heuristics for combinatorial problems on real data. *Computers and Operations Research*, 20:687–695, 1993.

[Smith & Fogarty 97] J. E. Smith and T. C. Fogarty. Operator and Parameter Adaptation in Genetic Algorithms. *Soft Computing*, 1(2):81–87, 1997.

[Standler & Happel 92] P. F. Standler and R. Happel. Correlation structure of the landscape of the graph-bipartition problem. *J. Phys. A.: Math. Gen.*, 25:3103–10, 1992.

[Standler & Schnabl 92] P. F. Standler and W. Schnabl. The landscape of the travelling salesman problem. *Physics Letters*, 161A:337–44, 1992.

[Starkweather *et al.* 91] T. Starkweather, S. McDaniel, K. Mathias, C. Whitley, and D Whitley. A Comparison of Genetic Sequencing Operators. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 69–76. San Mateo: Morgan Kaufmann, 1991.

[Steels 90] L. Steels. Components of Expertise. *AI Magazine*, 11(2):29–49, 1990.

[Stefik 95] M. Stefik. *Introduction to Knowledge Systems*. Morgan Kauffmann, 1995.

[Stewart *et al.* 94] B. S. Stewart, C.-F. Liaw, and C. C. White. A bibliography of heuristic search research through 1992. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(2):268–293, 1994.

[Storn & Price 97] R. Storn and K. Price. Differential Evolution: Numerical Optimisation Made Easy. *Dr Dobb's Journal*, pages 18–24, April 1997.

[Surry & Radcliffe 96a] P. D. Surry and N. J. Radcliffe. Directed Recombination and Real Parameter Evolutionary Algorithms. In *Foundations of Genetic Algorithms IV*, 1996.

[Surry & Radcliffe 96b] P. D. Surry and N. J. Radcliffe. Formal Algorithms + Formal Representations = Search Strategies. In *Parallel Problem Solving from Nature, LNCS*. Springer Verlag, 1996.

[Surry & Radcliffe 96c] P. D. Surry and N. J. Radcliffe. Inoculation to initialise evolutionary search. In T. C. Fogarty, editor, *Proceedings of the 3rd AISB workshop on Evolutionary Computation*. Springer-Verlag (LNCS), 1996.

[Surry 98] P. D. Surry. *A Prescriptive Formalism for Constructing Domain-Specific Evolutionary Algorithms*. Unpublished PhD thesis, University of Edinburgh, UK, 1998.

[Surry *et al.* 95] P. D. Surry, N.J. Radcliffe, and I.D. Boyd. A Multi-Objective Approach to Constrained Optimisation of Gas Supply Networks: The COMOGA Method. In *The 2nd AISB Workshop on Evolutionary Computing (LNCS)*. Springer Verlag, 1995.

[Szwarc & Gupta 87] W. Szwarc and J. N. D. Gupta. A Flow-Shop Problem with Sequence-Dependent Additive Setup Times. *Naval Research Logistics*, 34:619–627, 1987.

[Szwarc 71] W. Szwarc. Elimination methods in the $m \times n$ sequencing problem. *Naval Research Logistics Quarterly*, 18:295–305, 1971.

[Szwarc 79] W. Szwarc. The critical path approach in the flow-shop problem. *OPSEARCH*, 16(2 & 3):98–102, 1979.

[Taillard 90] E. Taillard. Some efficient heuristic methods for the flow-shop sequencing problem. *European Journal of operations research*, 47:65–74, 1990.

[Taillard 93] E. Taillard. Benchmarks for basic scheduling problems. *European Journal of operations research*, 64:278–285, 1993.

[Tanese 87] R. Tanese. Parallel genetic algorithms for a hypercube. In J. J. Grefenstette, editor, *Proceedings of the Second International Conference on Genetic Algorithms*, Hilldale, NJ, 1987. Erbaum.

[Thangiah 95] S. R. Thangiah. An adaptive clustering method using a geometric shape for vehicle routing problems with time windows. In L. J. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 536–544, San Francisco, Ca., 1995. Morgan Kaufmann.

[Tsang 93] E. P. K. Tsang. *Foundations of Constraint Satisfaction*. Academic Press, London and San Diego, 1993.

[Tsang 96] C. K. Tsang. A Genetic Algorithm for RNA Secondary Structure Prediction. AI4 Project Report, Department of Artificial Intelligence, Edinburgh University, July 1996.

[Turner & Booth 87] S. Turner and F. D. Booth. Comparison of heuristics for flow shop sequencing. *OMEGA*, 15:75–78, 1987.

[Tuson & Ross 98] A. Tuson and P. Ross. Adapting operator probabilites in genetic algorithms. *Evolutionary Computation (to appear)*, 6(2), 1998.

[Tuson 95] A. L. Tuson. Adapting Operator Probabilities In Genetic Algorithms. Unpublished M.Sc. thesis, Department of Artificial Intelligence, Univeristy of Edinburgh, 1995.

[VanLaarhoven & Aarts 88] P.J. M. VanLaarhoven and E. H. L. Aarts. *Simulated Annealing: Theory and Applications*. Kluwer, Dordrecht, 1988.

[vanVilet 93] H. van Vilet. *Software Engineering: Principles and Practice*. John Wiley and Sons, 1993.

[Vassilev 97] V. K. Vassilev. An information measure of landscapes. In T. Bäck, editor, *The 6th International Conference on Genetic Algorithms*, pages 49–56. San Mateo: Morgan Kaufmann, 1997.

[Vonk 87] R. Vonk. *Prototyping of Information Systems*. Academic Service, 1987. (in Dutch).

[Vose & Liepins 91] M. Vose and D. Liepins. Schema disruption. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 237–243. San Mateo: Morgan Kaufmann, 1991.

[Walker 95] T. Walker. Is marker based encoding of genetic algorithms an effective technique for neural network generation? Unpublished M.Sc. thesis, Department of Artificial Intelligence, Univeristy of Edinburgh, 1995.

[Watanabe 69] S. Watanabe. *Knowing and Guessing - A Formal and Quantative Study*. John Wiley and Sons, 1969.

[Weinburger 90a] E. D. Weinburger. Correlated and uncorrelated fitness landscapes and how to tell the difference. *Biological Cybernetics*, 63:325–36, 1990.

[Weinburger 90b] E. D. Weinburger. Fourier and taylor series on fitness landscapes. *Biological Cybernetics*, 65:321–30, 1990.

[Wheeler 95] R. Wheeler. Hybrid Tools for Intractable Problems: The Development of the WHO/TB Integrated System. Unpublished M.Sc. thesis, Department of Artificial Intelligence, Univeristy of Edinburgh, 1995.

[Wheeler 98] R. Wheeler. Personal Communication, 1998.

[Wheeler *et al.* 98] R. Wheeler, Levy, and Zhao. The Role of Automated Micro-Management in Disease Control Systems: Tuberculosis in China. *Accepted for publication in The International Journal of Tuberculosis and Lung Disease*, 1998.

[Whitley & Rana 97] D. Whitley and S. Rana. Representation, Search, and Genetic Algorithms. In *14th National Conference on Artificial Intelligence (AAAI-97)*. AAAI Press/MIT Press, 1997.

[Whitley & Rana 98] D. Whitley and S. Rana. Search, Binary Representations, and Counting Optima. In *Proceedings of a Workshop on Evolutionary Algorithms*. Institute for Mathematics and its Applications, 1998.

[Whitley 89] D. Whitley. The GENITOR Algorithm and Selective Pressure. In Stephanie Forrest, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 116–121. San Mateo: Morgan Kaufmann, 1989.

[Whitley *et al.* 89] D. Whitley, T. Starkweather, and D'A. Fuquay. Scheduling Problems and Traveling Salesman: The Genetic Edge Recombination Operator. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms and their Applications*, pages 133–140. San Mateo: Morgan Kaufmann, 1989.

[Whitley *et al.* 98] D. Whitley, S. Rana, and R. Heckendorn. Representation Issues in Neighborhood Search and Evolutionary Algorithms. In D. Quagliarella, J. Periaux, C. Poloni, and G. Winter, editors, *Genetic Algorithms and Evolution Strategies in Engineering and Computer Science*, pages 39–57. John Wiley, 1998.

[WHO 97a] WHO. *Anti-tuberculosis Drug Resistance in the World*. World Health Organisation, 1997.

[WHO 97b] WHO. *Treatment of Tuberculosis: Guidelines for National Programmes*. World Health Organisation, 1997.

[WHO 98a] WHO. Global Tuberculosis Programme — Country Profiles: China. World Health Organisation Information Sheet (obtainable from `http://www.who.ch/`), 1998.

[WHO 98b] WHO. Global Tuberculosis Programme — DOTS: Directly Observed Treatment, Short-course. World Health Organisation Information Sheet (obtainable from `http://www.who.ch/`), 1998.

[Wielinga *et al.* 92] B. Wielinga, W. Van de Velde, G. Screiber, and H. Akkermans. The CommonKADS Framework for Knowledge Modelling. In *Proceedings of the 7th Banff Knowledge Acquisition Workshop*, Banff, Alberta, Canada, 1992.

[Wiggins *et al.* 91] G. A. Wiggins, A. Bundy, I. Kraan, and J. Hesketh. Synthesis and transformation of logic programs through constructive, inductive proof. In K-K. Lau and T. Clement, editors, *Proceedings of LoPSTr-91*, Workshops in Computing Series, pages 27–45. Springer Verlag, 1991.

[Winston 93] W. L. Winston. *Operations Research: Applications and Algorithms (3rd Ed.).* Duxbury Press, 1993.

[Wolpert & Macready 95] D.H. Wolpert and W.G. Macready. No free lunch theorems for search. Technical report, SFI-TR-95-02-010, Santa Fe Institute, 1995.

[Woodruff 94] D. L. Woodruff. Simulated Annealing and Tabu Search: Lessons from a Line Search. *Computers and Operations Research*, 21(8):823–839, 1994.

[Wright 32] S. Wright. The roles of mutation, inbreeding, crossbreeding, and selection in evolution. In D. F. Jones, editor, *Proc. 6th Int. Congr. on Genetics*, volume 1, pages 356–66. Ithaca, NY, 1932.

[Yamada & Reeves 97] T. Yamada and C. Reeves. Permutation Flowshop Sequencing by Genetic Local Search. In *The Second International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA 97)*, 1997.

[Zdrahal & Motta 95] Z. Zdrahal and E. Motta. An In-Depth Analysis of Propose & Revise Problem Solving Methods. In B. R. Gaines and M. Musen, editors, *Proceedings of the 9th Banff Knowledge Acquistion for Knowledge Based Systems Workshop*, pages 1–20, Banff, Canada, 1995.

[Zegordi & Enkawa 95] S. H. Zegordi and K. I. T. Enkawa. Minimizing makespan for flow shop scheduling by combining simulated annealing with sequencing knowledge. *European Journal of Operational Research*, 85:515–531, 1995.

[Zweben & Davis 92] M. Zweben and E. Davis. Learning to Improve Iterative Repair Scheduling. Technical report, NASA Ames Research Centre, Report FIA-92-14, 1992.

[Zweben *et al.* 92] M. Zweben, E. Davis, B. Daun, and M. Deale. Scheduling and Rescheduling with Iterative Repair. Technical report, NASA Ames Research Centre, Report FIA-92-16, 1992.

# Appendix A

# Overview of the Statistical Methods Used

This appendix provides an overview and justification for the statistical methods used in this thesis. For further details the reader is directed to a statistics textbook, of which [Cohen 95] is especially suitable for the computer scientist.

## A.1 The Student's *t*-Test

The usual method for making pairwise comparisons is the **t-test**. For this we need the two sample sizes, means, and variances, which are $N_1$, $N_2$, $\bar{x}_1$, $\bar{x}_2$, $\hat{\sigma}_1^2$, and $\hat{\sigma}_2^2$ respectively. They are then used to test the following hypotheses: the **null** hypothesis that the two samples are drawn from populations with equal means ($H_0 : \mu_1 = \mu_2$), and the **alternative** hypothesis that the samples were drawn from populations with different means ($H_1 : \mu_1 \neq \mu_2$). The form of the alternative hypothesis dictates that a **two-tailed** test is required. We can then calculate the **t-statistic**, $t_{\bar{x}_1 - \bar{x}_2}$, as follows:

$$t_{\bar{x}_1 - \bar{x}_2} = \frac{\bar{x}_1 - \bar{x}_2}{\hat{\sigma}^2_{\bar{x}_1 - \bar{x}_2}}$$

where the pooled, estimated variance of the sampling distribution of the difference of means, $\hat{\sigma}^2_{\bar{x}_1 - \bar{x}_2}$, is given by:

$$\hat{\sigma}^2_{\bar{x}_1 - \bar{x}_2} = \sqrt{\hat{\sigma}^2_{pooled}\left(\frac{1}{N_1} + \frac{1}{N_2}\right)}$$

and where the pooled, estimated variance of the difference of the means, $\hat{\sigma}^2_{pooled}$ is simply a weighted average of $\hat{\sigma}_1^2$, and $\hat{\sigma}_2^2$:

$$\hat{\sigma}^2_{pooled} = \frac{(N_1 - 1)\hat{\sigma}_1^2 + (N_2 - 1)\hat{\sigma}_2^2}{N_1 + N_2 - 2}$$

**NB.** The above pooling of the sample variances is on the grounds that $\sigma^2_{x_1 - x_2} = \sigma^2_{x_1} + \sigma^2_{x_1}$ [Cohen 95].

The $t$-statistic and the number of **degrees of freedom** is then indexed to the **t distribution** to give a probability, $p$, of the null hypothesis, $H_0$, being correct. The number of degrees of freedom in this case is given by $N_1 + N_1 - 2$. Therefore in order to say that a result is significant, we have to define a threshold for $p$, which is decided by the investigator (denoted $\alpha_c$) — for the purposes of the studies here, this was set at $p \leq 0.1$. This means, as we are using a two-tailed test, that the **rejection region** covers 90% of the $t$ distribution, equally divided between the two extremes of the distribution. This equates to a 90% probability that the two sample means, $\bar{x}_1$ and $\bar{x}_2$, are in fact drawn from populations with different underlying means (the alternative hypothesis $H_1$).

The $t$-test does assume that the distribution from which the sample was drawn is normal. Strictly speaking, as there is a lower and upper limit of the solution quality attainable in the search space, this assumption is incorrect. Fortunately, as noted in [Cohen 95], the $t$-test is very robust and can be trusted in all but the most extreme cases where $N$ very small, the $t$ score is marginal, and the underlying distribution is very skewed.

**NOTE:** in applications of the $t$-test the rejection region is usually set to be one of 95%, 99%, or 99.9%. The value of 90% was selected for this study as finding a difference contrary to the design heuristics will lead us to doubt their validity — this is unlike most applications where showing there is a difference confirms the experimenter's hypothesis. Therefore it is more scientifically conservative to favour detecting differences which could lead to us rejecting the validity of the proposed design heuristics.

## A.2 Shortcomings of Pairwise Comparisons

Unfortunately, in the analysis performed here, there are really two types of null hypothesis. To illustrate this, consider the following example (taken from [Cohen 95]). An experimental analysis requires that 105 pairwise comparisons of 15 sample means be made, but all of the means (unknown to the experimenter) were drawn from the same population. Now if we set $\alpha_c$ to 0.05, then the probability of all of the $m$ comparisons correctly reporting that the means were drawn from the same population would be $(1 - \alpha_c)^m$, and therefore the probability of one spurious (incorrect) result would be $1 - (1 - \alpha_c)^m$, which is equal to 0.9954 in this case. Therefore we are almost certain to incorrectly conclude that there are differences in the means.

Therefore, in addition, to the **comparison** null hypothesis we have considered so far, there is an additional **experiment** null hypothesis that *all* of the means to be compared are drawn from the same population, with its own error rate $\alpha_e$. Needless to say that if the experiment null hypothesis is true, then so must the comparison null hypotheses. This therefore suggests that the experiment-wise error rate is the important one.

A problem then arises on how to protect the experiment null hypothesis. The two error rates, $\alpha_c$ and $\alpha_e$ can be related by the equation $\alpha_e \approx 1-(1-\alpha_c)^m$, so why not just lower $\alpha_c$ to give an acceptable error rate for $\alpha_e$? This is fine in principle, but unfortunately the danger exists that we would have to make $\alpha_c$ too stringent to detect *any* pairwise differences, even those that actually exist! In short, we are forced to favour the needs of either $\alpha_c$ or $\alpha_e$ — increasing our ability to obtain the correct result for one, decreases the corresponding ability for the other.

There is still an additional problem if it is decided that the experiment null hypothesis is to be ignored — given that $\alpha_c$ is still set to 0.05, then the comparison null hypothesis will be expected to be in error approximately $105 \times 0.05 = 5.25$ times. This problem is compounded by the observation made in [Cohen 95], that it is not known which of the comparisons is wrong.

Some method to address these issues is therefore required, and there are three alternative approaches. The first is to test for the experimental null hypothesis first of all, and if it concludes that the means are different, to then use pairwise comparisons to discover which means are different. The second approach is to use two types of pairwise comparison, one method which is rather conservative and protects the experiment null hypothesis, and another, more sensitive method, that favours the comparison null hypothesis. The results of these are then contrasted and any differences and noted and resolved on an individual basis. Finally, if the desired comparisons to be made are known, it may be possible to construct a set of comparisons that are statistically independent of each other, and ask focused questions about the data with a known probability of error — this approach is known as **planned comparisons**.

The first and second approaches will be described here and their suitability assessed. The final approach was not considered here as it was not thought possible to say in advance which comparisons would be important.

## A.3 Analysis of Variance

Analysis of variance (ANOVA) can be used to test the experimental null hypothesis. Let us say that we have $J$ groups (each containing $K$ samples) for which we would like to examine the experimental null hypothesis. Therefore we have access to all of the datums, $x_{jk}$, and the means for each group, $\bar{x}_j$, as well as the **grand mean**, the mean over all groups, $\bar{x}_G$. Now it turns out, given some assumptions, that the **grand variance** can be decomposed into two parts, as the sums of squared deviations are additive:

$$\sum_j \sum_k (x_{jk} - \bar{x}_G)^2 = \sum_j \sum_k (x_{jk} - \bar{x}_j)^2 + \sum_j (\bar{x}_j - \bar{x}_G)^2$$

The first of these two components, $\sum_j \sum_k (x_{jk} - \bar{x}_j)^2$, is the total amount of error *within* the groups (i.e. background noise); and the second component, $\sum_j (\bar{x}_j - \bar{x}_G)^2$ measures the variance *between* the groups, which is an indication of the effect of being in one group or another. The above is useful because if all of our group means are drawn from the same population, we would expect these two components of the grand variance to be of comparable size. This is the basis of the **F-test** for the experimental null hypothesis.

### A.3.1 The F-Test

The above intuition needs to be turned into a statistical test. Fortunately this is rather straightforward. First divide the within and between-group sums of squares above by their degrees of freedom (which are $J \times K - J$ and $J - 1$ respectively). This gives us the within-group and between-group variances, or mean square deviations, $MS_{within}$ and $MS_{between}$. The quotient of these two gives us the **F-statistic**:

$$F = \frac{MS_{between}}{MS_{within}}$$

Now, if the experiment null hypothesis is correct, these two terms should be equal. Therefore we should expect the F-statistic to be one, and large values to be evidence against the null hypothesis. As for the *t*-distribution, the F-statistic can be indexed to the **F-distribution** to give a probability of the null hypothesis being true — in this case, with $J - 1$ and $J \times K - J$ degrees of freedom.

Therefore, as alluded to earlier, the F-test can be used to establish whether the experiment null hypothesis is true (thus protecting it). If the null hypothesis is rejected, we can then use a pairwise comparison that is sensitive to the comparison null hypothesis (such as a *t*-test) to identify where the differences are. It should be noted, however, that the use of the F-test is applicable if the following assumptions are met [Cohen 95]:

- The population distributions from which groups are drawn are normal.
- These distributions have the same variance.
- The error components of the group data are independent.

However, more care needs to be taken in this case with the validity of the normality assumption for the population distributions when variances are compared. In should also be noted that, on examination of the data in the appendices, the same variances were often quite different thus possibly reducing the validity of the second assumption. In addition, since the F-test considers the data-set as a whole, it may *over-protect* the experiment null hypothesis in the sense that it *implicitly* assumes that all of the comparisons are equal interest. This illustrated by the fact the F-test does not tell us *which* means are different when it rejects the experiment null hypothesis.

In other words, though accepting the null hypothesis may be *statistically* conservative, it may not be *scientifically* conservative in that, as noted in [Cohen 96], tests may fail despite the actual veracity of the null hypothesis. This is especially true if we have doubts about the **power** of a test, the proportion of correct conclusions for a test, such as those we have above. This was found particularly true in this case especially as many of the F-tests performed gave middling values (e.g. around 70%) that though not rejecting the null hypothesis, did not strongly support it either. Therefore these objections, when taken together, lead to the adoption of an alternative approach of using two types of comparison method.

Finally, it should be noted that, for the purposes of explanation, the case of when the data was grouped in one way was considered (a **one-way** ANOVA). The experiments that we will be analysing have data that can be analysed in two groupings (i.e. algorithm and representation). The approach taken here was to consider the two separately and to 'collapse' the other grouping when a particular grouping was being considered. That said, though the use of a **two-way** ANOVA would allow us to look at the interaction between these two factors, it was not felt to relevant to the question being considered and so it was not used.

## A.4 The Scheffé and LSD Tests

An alternative to the above is to use the second approach noted above and advocated in [Cohen 96]. This is to use two tests, which favour either the experiment or comparison null hypotheses and to contrast the results and resolve any differences on a case-by-case basis.

### A.4.1 The LSD Test

The Least Significant Difference, or LSD, test is designed to favour the comparison null hypothesis. The test statistic used for the LSD test is:

$$F_{LSD} = \frac{(\bar{x}_1 - \bar{x}_2)^2}{MS_{within}\left(\frac{1}{N_1} + \frac{1}{N_2}\right)}$$

where $MS_{within}$ is the within-group variance from the ANOVA described above. The F-test has 1 and $N - J$ degrees of freedom, where $N$ is the sum of the group sizes.

Finally it should be noted that this test is related to the *t*-test. Taking the square root of $F_{LSD}$ gives an equation that is identical to that of the *t*-test, albeit with the substitution of $\hat{\sigma}^2_{pooled}$ for $MS_{within}$. So why not just use a *t*-test? The reason is, as noted in [Cohen 95] that $MS_{within}$ is a better estimator of the population variance than $\hat{\sigma}^2_{pooled}$ as it is based on all of the groups and not just the two being compared.

### A.4.2 The Scheffé Test

The contrasting, conservative, pairwise comparison method used to favour the experiment null hypothesis is the Scheffé test. The test statistic used for the Scheffé test is:

$$F_S = \frac{(\bar{x}_1 - \bar{x}_2)^2}{MS_{within}\left(\frac{1}{N_1} + \frac{1}{N_2}\right)(J - 1)}$$

where the F-test used in this case has $J - 1$ and $N - J$ degrees of freedom, where $N$ is the sum of the group sizes.

The main difference, degrees of freedom aside, between this and the LSD test is the inclusion of the extra term in the divisor. This protects the experiment null hypothesis by making the differences in $\bar{x}_1$ and $\bar{x}_2$ approximately $1/\sqrt{J-1}$ times as large as would be required by an LSD test to show a significant difference in the means.

## A.5 Summary and Closing Remarks

This appendix should close with the following note. As we are using two statistical tests this does introduce some additional complications — namely, when the two tests (Scheffé and LSD) disagree. The approach taken here was to examine the statistics for the LSD and *t*-tests on an individual basis to see whether to side with the Scheffé or LSD result.

# Appendix B

# Publications Arising From This Thesis

The following publications have been drawn from the work in this thesis.

- A.L. Tuson (1998). Optimisation with Hillclimbing on Steroids: A Tutorial on Neighbourhood Search. In the *Keynote Papers of the 10th Young OR Conference*, Operational Research Society, (ISBN 0903440199, 16 pages)
- P. Ross and A.L. Tuson (1997). Directing the Search of Evolutionary and Neighbourhood-Search Optimisers for the Flowshop Sequencing Problem with an Idle-Time Heuristic. In the proceedings of the *1997 AISB Workshop on Evolutionary Computing*, Lecture Notes in Computer Science, Springer Verlag, (ISBN 354063476, 13 pages).
- A.L. Tuson, R. Wheeler, and P. Ross (1997). Emergency Resource Redistribution In The Developing World: Towards a Practical Evolutionary/Meta-Heursitic Scheduling System. In the proceedings of the *Second International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA 97)*, IEE, (ISBN 0852966938, 6 pages).
- A.L. Tuson, R. Wheeler, and P. Ross (1997). An Evolutionary/Meta-Heursitic Approach To Emergency Resource Redistribution In The Developing World. In *Artificial Neural Nets and Genetic Algorithms: Proceedings of the Third International Conference On Artificial Neural Networks And Genetic Algorithms (ICANNGA 97)*, G.D. Smith et al (eds.), Springer Verlag, (ISBN 3211830871, 4 pages).
- A.L. Tuson, R. Wheeler, and P. Ross (1996). A Prototype Emergency Resource Redistribution System For Disease Control Programmes. In the *15th Workshop of the UK Planning and Scheduling Special Interest Group*, (ISSN 1368-5708, 10 pages)

Copies of the papers are included in this appendix in the order given above.

# Appendix C

# Detailed Results and Statistical Analysis

For a number of reasons, though mostly because it would double the length of this thesis, the detailed results of the experiments and the statistical analysis of the results has not been included. Instead a web page has been set up where this information can be obtained electronically.

`http://www.soi.city.ac.uk/~andrewt/thesis-appx.html`

Though little use was made in the case studies of the tuning experiments, the tables of results are given in full as PostScript files on the above web page — this is to allow other researchers to fully investigate them, if desired, without having to repeat the experiments. The average of 50 runs is given in all cases, and the standard deviation is given in parentheses. Both the solution quality obtained at the end of the run, and the number of evaluations taken to find that solution are given.

The purpose of the statistical analysis was to identify statistically significant differences in performance. This allows us to see whether there were any differences that contradict the design heuristics proposed in this thesis. For details of the tunable parameters adopted, the reader is directed to Chapters 5 and 6. The spreadsheets are given in the web page in Microsoft Excel format, the statistical analysis they performed was described earlier in Appendix A.