CHARACTERISING AND SEARCHING FITNESS LANDSCAPES

Colin R Reeves
School of Mathematical and Information Sciences
Coventry University
Priory Street, Coventry CV1 5FB
UK
Phone: +44 1203-838579; Fax +44 1203-838080
Email: C.Reeves@coventry.ac.uk

ABSTRACT: Evolutionary algorithms and other heuristic search methods have been increasingly, and successfully, applied to combinatorial optimization problems. One of the most commonly-used metaphors to describe the process is that of a 'fitness landscape'. However, describing what we mean by such a term, and supplying an interpretation analogous to our everyday experience of 3-dimensional landscapes, is not as easy as a naive application of the word might suggest.

In this paper, firstly, we consider ways in which fitness landscapes can be characterized mathematically and empirically. Secondly we show how empirical results suggest the so-called 'big-valley conjecture', and introduce 'path-tracing' techniques whereby this property of fitness landscapes can be exploited by (for example) genetic algorithms. Finally, we mention some empirical studies which show the effectiveness of this approach to some scheduling problems.

KEYWORDS: Heuristic search, combinatorial optimization, evolutionary algorithms, fitness landscapes, path tracing

INTRODUCTION

The metaphor of a *landscape* is commonly found in descriptions of the application of heuristic methods for solving a combinatorial optimization problem (COP). We can define such problems as follows: we have a discrete search space $\mathcal{X}$, and a function

$$f : \mathcal{X} \mapsto I\!R.$$

The general problem is to find

$$\mathbf{x}^* = \max_{\mathbf{x} \in \mathcal{X}} f.$$

where $\mathbf{x}$ is a vector of *decision variables* and f is the *objective function*. (Of course, minimization can also be the aim, but the modifications are always obvious). In the field of evolutionary algorithms, such as genetic algorithms (GAs), the function f is often called the *fitness*, and the associated landscape is a *fitness landscape*. The vector $\mathbf{x}^*$ is a global optimum: that vector which is the fittest of all. (In some problems, there may be several global optima—different vectors of equal fitness.)

With the idea of a fitness landscape comes the idea that there are also many local optima or false peaks, in which a search algorithm may become trapped without finding the global optimum. In continuous optimization, notions of continuity and concepts associated with the differential calculus enable us to characterize quite precisely what we mean by a landscape, and to define the idea of an optimum. It is also convenient that our own experiences of hill-climbing in a 3-dimensional world gives us analogies to ridges, valleys, basins, watersheds etc that help us to build an intuitive picture of what is needed for a successful search, even though the search spaces that are of interest often have dimensions many orders of magnitude higher than 3.

However, in the continuous case, the landscape is determined only by the fitness function, and the ingenuity needed to find a global optimum consists in trying to match a technique to this single landscape. There is a major difference when we come to discrete optimization. Indeed, we really should not even use the term

Table 1: Local optima and basins of attraction for steepest ascent

Local optimum	0 1 0 1 0 (4100)	0 1 1 0 0 (3988)	0 0 1 1 1 (3803)	1 0 0 0 0 (3236)
Basin	0 0 0 0 0	0 0 1 0 0	0 0 1 1 0	1 0 0 0 0
	0 0 0 0 1	0 1 1 0 0	0 0 1 1 1	1 0 0 0 1
	0 0 0 1 0	1 1 1 0 0	1 0 1 1 0	1 0 0 1 0
	0 0 0 1 1		1 0 1 1 1	1 0 0 1 1
	0 0 1 0 1			1 0 1 0 0
	0 1 0 0 0			
	0 1 0 0 1			
	0 1 0 1 0			
	0 1 0 1 1			
	0 1 1 0 1			
	0 1 1 1 0			
	0 1 1 1 1			
	1 0 1 0 1			
	1 1 0 0 0			
	1 1 0 0 1			
	1 1 0 1 0			
	1 1 0 1 1			
	1 1 1 0 1			
	1 1 1 1 0			
	1 1 1 1 1			

'landscape' unless we can define the topological relationships of the points in the search space $\mathcal{X}$. Unlike the continuous case, we have some freedom in specifying these relationships, and in fact, that is precisely what we do when we decide to use a particular technique.

AN EXAMPLE

In practice, one of the most commonly used search methods for COPs is *neighbourhood search* (NS). This idea is at the root of modern 'metaheuristics' such as simulated annealing (SA) and tabu search (TS)—as well as being much more involved in the methodology of GAs than is sometimes realized.

A *neighbourhood structure* is generated by using an operator that transforms a given vector $\mathbf{x}$ into a new vector $\mathbf{x}'$. For example, if the solution is represented by a binary vector (as is often so for GAs, for instance), a simple neighbourhood might consist of all vectors obtainable by 'flipping' one of the bits. The 'bit flip' (BF) neighbours of (00000), for example, would be

$$\{(10000), (01000), (00100), (00010), (00001)\}.$$

Consider the problem of maximizing a simple function

$$f(z) = z^3 \Leftrightarrow 60z^2 + 900z + 100$$

where the solution z is required to be an integer in the range $[0, 31]$. Regarding z as a continuous variable, we have a smooth unimodal function with a single maximum at $z = 10$—as is easily found by calculus—and since the solution is an integer, this is undoubtedly the most efficient way of solving the problem.

However, suppose we chose instead to represent z by a binary vector $\mathbf{x}$ of length 5. By decoding this binary vector as an integer it is possible to evaluate f, and we could then use NS, for example, to search over the binary hypercube for the global optimum.

This discrete optimization problem turns out to have 4 optima (3 of them local) when the BF operator is used. If a 'steepest ascent' strategy is used (i.e., the *best* neighbour of a given vector is identified before a move is made) the local optima and their basins of attraction are as shown in table 1. On the other hand, if a 'next ascent' strategy is used (where the next change which leads uphill is accepted without ascertaining if a still better one exists), the basins of attraction are as shown in table 2.

In fact, there are even more complications: in table 2, the order of searching the components of the vector is 'forward' (left-to-right). If the search is made in the reverse direction (right-to-left) the basins of attraction are different, as shown in table 3.

Table 2: Local optima and basins of attraction for next ascent (forward search) using single bit complement operator

Local optimum	0 1 0 1 0 (4100)	0 1 1 0 0 (3988)	0 0 1 1 1 (3803)	1 0 0 0 0 (3236)
Basin	0 0 1 0 1	0 0 1 0 0	0 0 1 1 1	0 0 0 0 0
	0 0 1 1 0	0 1 0 0 0	0 1 1 1 1	0 0 0 0 1
	0 1 0 0 1	0 1 1 0 0	1 0 1 1 1	0 0 0 1 0
	0 1 0 1 0	1 0 1 0 0	1 1 1 1 1	0 0 0 1 1
	0 1 0 1 1	1 1 0 0 0		1 0 0 0 0
	0 1 1 0 1	1 1 1 0 0		1 0 0 0 1
	0 1 1 1 0			1 0 0 1 0
	1 0 1 0 1			1 0 0 1 1
	1 0 1 1 0			
	1 1 0 0 1			
	1 1 0 1 0			
	1 1 0 1 1			
	1 1 1 0 1			
	1 1 1 1 0			

Table 3: Local optima and basins of attraction for next ascent (reverse search) using single bit complement operator

Local optimum	0 1 0 1 0 (4100)	0 1 1 0 0 (3988)	0 0 1 1 1 (3803)	1 0 0 0 0 (3236)
Basin	0 1 0 0 0	0 1 1 0 0	0 0 0 0 0	1 0 0 0 0
	0 1 0 0 1	0 1 1 0 1	0 0 0 0 1	1 0 0 0 1
	0 1 0 1 0	0 1 1 1 0	0 0 0 1 0	1 0 0 1 0
	0 1 0 1 1	0 1 1 1 1	0 0 0 1 1	1 0 0 1 1
			0 0 1 0 0	1 0 1 0 0
			0 0 1 0 1	1 0 1 0 1
			0 0 1 1 0	1 0 1 1 0
			0 0 1 1 1	1 0 1 1 1
				1 1 0 0 0
				1 1 0 0 1
				1 1 0 1 0
				1 1 0 1 1
				1 1 1 0 0
				1 1 1 0 1
				1 1 1 1 0
				1 1 1 1 1

Thus, by using BF with this binary representation, we have created local optima that did not exist in the integer version of the problem. Further, although the optima are still the same (as they are bound to be), the chances of reaching a particular optimum can be seriously affected by a change in search strategy.

However, the bit flip operator is not the only mechanism for generating neighbours. An alternative neighbourhood could be defined as follows: for $k = 1, \ldots, 5$, flip bits $\{k, \ldots, 5\}$. Thus, the neighbours of (0 0 0 0 0), for example, would now be

$$\{(11111), (01111), (00111), (00011), (00001)\}.$$

This creates a very different landscape. In fact, there is now only a single global optimum (01010); *every* vector is in its basin of attraction. This illustrates the point that it is not merely the choice of a binary representation that generates the landscape—the search operator needs to be specified as well.

Incidentally, there are two interesting facts about the CX operator. Firstly, it is closely related to the one-point crossover operator frequently used in genetic algorithms. (For that reason, it has been named [3] the complementary crossover or CX operator). Secondly, if the 32 vectors in the search space are re-coded using a *Gray* code, it is easy to show that the neighbours of a point in Gray-coded space under BF are identical to those in the original binary-coded space under CX. This is an example of an *isomorphism* of landscapes. More details of the mathematical background to these and similar phenomena can be found in [3].

MATHEMATICAL CHARACTERIZATION

We can define a landscape Λ for the function f as a triple $\Lambda = (\mathcal{X}, f, d)$ where d denotes a distance measure $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+ \cup \{\infty\}$ for which is required that

$$d(\mathbf{s}, \mathbf{t}) \geq 0; \qquad d(\mathbf{s}, \mathbf{t}) = 0 \Leftrightarrow \mathbf{s} = \mathbf{t}; \qquad d(\mathbf{s}, \mathbf{u}) \leq d(\mathbf{s}, \mathbf{t}) + d(\mathbf{t}, \mathbf{u}); \ \} \quad \forall \mathbf{s}, \mathbf{t}, \mathbf{u} \in \mathcal{X}.$$

This definition says nothing about how the distance measure arises. In fact, for many cases a 'canonical' distance measure can be defined. Often, this is symmetric, i.e. $d(\mathbf{s}, \mathbf{t}) = d(\mathbf{t}, \mathbf{s})\ \forall \mathbf{s}, \mathbf{t} \in \mathcal{X}$, so that d also defines a *metric* on $\mathcal{X}$. This is clearly a nice property, although it is not essential.

What we have called a canonical distance measure is typically related to the neighbourhood structure. Every solution $\mathbf{x} \in \mathcal{X}$ has an associated set of *neighbours*, $N(\mathbf{x}) \subset \mathcal{X}$, called the neighbourhood of $\mathbf{x}$. Each solution $\mathbf{x}' \in N(\mathbf{x})$ can be reached directly from $\mathbf{x}$ by an operation called a *move*. Many different types of move are possible in any particular case, and we can view a move as being generated by the applying an operator ω to a vector $\mathbf{s}$ in order to transform it into a vector $\mathbf{t}$. The canonical distance measure d_ω is that induced by ω whereby

$$\mathbf{t} \in N(\mathbf{s}) \Leftrightarrow d_\omega(\mathbf{s}, \mathbf{t}) = 1.$$

The distance between non-neighbours is defined as the length of the shortest path between them (if one exists).

For example, if $\mathcal{X}$ is the binary hypercube $\mathbb{Z}_2^l$, the bit flip (BF) operator can be defined as

$$\phi(k) : \mathbb{Z}_2^l \to \mathbb{Z}_2^l \qquad \begin{cases} z_k \mapsto 1 \Leftrightarrow z_k \\ z_i \mapsto z_i & \text{if } k \neq i \end{cases}$$

where $\mathbf{z}$ is a binary vector of length l. It is clear that the distance metric induced by ϕ is the well-known Hamming distance.

LOCAL OPTIMA

We can now give a formal statement of a fundamental property of fitness landscapes: for a landscape $\Lambda = (\mathcal{X}, f, d)$, a vector $\mathbf{s} \in \mathcal{X}$ is *locally optimal* if

$$f(\mathbf{s}) > f(\mathbf{t})\ \forall\ \mathbf{t} \in N(\mathbf{s}).$$

Landscapes that have only one local (and thus also global) optimum are commonly called *unimodal*, while landscapes with more than one local optimum are said to be *multimodal*.

The number of local optima in a landscape clearly has some bearing on the difficulty of finding the global optimum. However, it is not the only indicator: the size of the basins of attraction of the various optima is also an important influence.

GRAPH REPRESENTATION

Neighbourhood structures are clearly just another way of defining a graph , , which can be described by its $(n \times n)$ *adjacency matrix* $\mathbf{A}$. The elements of $\mathbf{A}$ are given by $a_{ij} = 1$ if the indices i and j represent neighbouring vectors, and $a_{ij} = 0$ otherwise. For example, the graph induced by the bit flip ϕ on vectors of length 3 has

$$\mathbf{A}_{BF} = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

where the vectors are indexed in the usual binary-coded integer order (i.e., $0 = (000), 1 = (001)$ etc). By way of contrast, the adjacency matrix for the CX operator is

$$\mathbf{A}_{CX} = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

(Mathematical connoisseurs might like to verify that permuting the rows and columns so that they are in the order $0, 1, 3, 2, 6, 7, 5, 4$ reproduces the adjacency matrix $\mathbf{A}_{BF}$—another way of demonstrating the isomorphism mentioned earlier.)

The *graph Laplacian* $\mathbf{\Delta}$ is defined as

$$\mathbf{\Delta} = \mathbf{A} \Leftrightarrow \boldsymbol{D}$$

where $\boldsymbol{D}$ is a diagonal matrix such that d_{ii} is the degree of vertex i. Usually, these matrices are vertex-regular and $d_{ii} = k \ \forall i$, so that

$$\mathbf{\Delta} = \mathbf{A} \Leftrightarrow k\boldsymbol{I}.$$

This notion recalls that of a Laplacian operator in the continuous domain; the effect of this matrix, applied as an operator at the point $\mathbf{s}$ to the fitness function f is

$$\mathbf{\Delta} f(\mathbf{s}) = \sum_{\mathbf{t} \in N(\mathbf{s})} (f(\mathbf{t}) \Leftrightarrow f(\mathbf{s}))$$

so it functions as a kind of differencing operator. In particular, $\mathbf{\Delta} f(\mathbf{s})/|N(\mathbf{s})|$ is the average difference in fitness between the vector $\mathbf{s}$ and its neighbours. Grover [2] has shown that the landscapes of several COPs satisfy an equation of the form

$$\mathbf{\Delta} f + \frac{Cf}{n} = 0$$

where C is a problem-specific constant and n is the size of the problem instance. From this it can be deduced that *all* local optima are better than the mean ($\bar{f}$) over all points on the landscape. Furthermore, it can also be shown that under mild conditions on the nature of the fitness function, the time taken by NS to find a local optimum in a maximization problem is $\mathcal{O}(n \log_2[f_{max}/\bar{f}])$ where f_{max} is the fitness of a global maximum. (Modifications for minimization problems are obvious.)

Ideally, such mathematical characterizations could be used to aid our understanding of the important features of a landscape, and so help us to exploit them in designing search strategies. But beyond Grover's rather general results above, it is possible to carry out further analytical studies only for small graphs, or graphs with a special structure—such as Hamming graphs. While some interesting research is in progress, at the moment the mathematics does little more than make explicit the concepts, and so we turn to empirical studies that try to help us understand more about the properties of fitness landscapes.

EMPIRICAL STUDIES

NK-LANDSCAPES

One of the first and most well-known systematic studies of fitness landscapes was carried out by Kauffman [6]. His 'NK-landscapes' were formulated to enable'tuning' of landscape difficulty in a general sense. They work as follows: we define a binary vector $\mathbf{z}$ of length N, and for each bit position i associate K other bits with it. For each of the 2^{K+1} different assignments of 1s and 0s, we draw a random value from the distribution $U(0,1)$ and record these values in a table. In order to evaluate the function, we take each bit in turn, find the value of its K associated bits $\{j_1, \ldots, j_K\}$ and extract the value $v^i_{j_1 \ldots j_K}$ from the ith table. The function value f is then the average of the N values taken from tables $1, \ldots, N$.

By varying the value of K, it is possible to change the characteristics of the problem in a fairly crude way. Clearly, as K increases, the number and order of the interactions expressed in the function increases. The interesting question is, if we try to find the optimal value of f by NS using a specified operator, what sort of landscape is produced? The definition above says nothing about a distance measure—Kauffman called his creations NK-landscapes, on the assumption that the only interesting case is that of Hamming distance, but this is not necessarily so. However, the ways in which he was able to characterise his landscapes by empirical investigations are noteworthy.

Firstly, he noticed that if the distance of each local optimum from the global optimum was computed, he found that they were on average very much closer to the global optimum than were randomly chosen points, and closer to each other than random points would be. That is, the distribution of local optima was not *isotropic*, but that they were clustered in a *central massif*. This can most easily be demonstrated graphically by plotting a scatter graph of fitness against distance. Typically, graphs such as that shown later in figure 1 are obtained.

Secondly, if the basins of attraction of each optimum were found, the size of the basin was quite highly correlated with its quality: *the highest peaks are drained by the biggest valleys*. Experimental work also showed that as the value of K increased, the average level of all local optima became similar, and the central massif became less distinct—the differences in quality are smaller when the amount of interaction is larger. Other findings are also discussed in his books, but these empirical findings at least seem to be fairly general, as we shall now see from other evidence.

OPERATIONAL RESEARCH

Operational research (OR) has studied combinatorial optimization and heuristic search for over 40 years. In 1965, one of the most important papers on the *travelling salesman problem* (TSP) was published by Lin [7]. His '3-optimal' algorithm is one of the first high-quality NS approaches to the TSP, but perhaps more interesting is that he found what Kauffman was to discover nearly 30 years later: that the local optima were 'close' to each other. A little later, Nugent *et al.*[10] found a similar relationship held for the quadratic assignment problem (QAP).

Later, Boese *et al.*[1] extended this analysis for the TSP, plotting fitness-distance plots in the style of Kauffman, and examined the case of graph bi-partitioning (GBP). Again the same phenomenon was observed, as it was also by Merz and Freisleben [9]. Because in such problems the emphasis is usually on minimization, the phenomenon has become known as the 'big valley', but it is of course just the central massif inverted.

Not all COPs have obvious distance measures as is the case for problems like the TSP and GBP. This led Reeves [12] to experiment with different distance measures in the case of the permutation flowshop scheduling problem (PFSP), which will be discussed later in this paper.

SEARCHING THE LANDSCAPE

Knowing that there is (or might be) a 'big valley' is all very well, but is of little consequence unless we can actually exploit it. In the first place, we can say that the big valley conjecture (BVC) explains the success of some search methods developed before any evidence for it was widely recognised. Before [1] for example, several researchers [5, 8, 18] had suggested solving the TSP by repeatedly generating new start points for search by perturbing a previous local optimum rather than by choosing a random point in the search space. Increasing sophistication in the implementation of such 'perturbation' methods mean that they currently appear to be the best available for the TSP. If new local optima are indeed close to each other, it both saves computational effort, and concentrates the search in the most likely region in which to find high-quality solutions.

However, it is not known whether this phenomenon is general, or whether it is specific to the cases previously examined. In extending it to other COPs, some interesting questions arise as to the way in which 'closeness'

can be measured, and as to how the significance of the observed behaviour can be assessed. In this paper, the well-known $n/m/P/f$ flowshop sequencing problem will be used as an example in an attempt to shed further light on this question.

FLOWSHOP SEQUENCING

The permutation flowshop sequencing problem $n/m/P/f$, is one in which n jobs have to be processed (in the same order) on m machines. The object is to find the permutation of jobs that will minimize the function f, which is unspecified at this point, as several different objectives are possible.

Suppose the processing time is $t(i,j)$ for job i on machine j, and the job permutation is denoted by $\boldsymbol{\pi} = \{\pi_1, \pi_2, \cdots, \pi_n\}$. Then the job completion times $C(\pi_i, j)$ are as follows:

$$C(\pi_i, j) = \max\{C(\pi_{i-1}, j), C(\pi_i, j \Leftrightarrow 1)\} + t(\pi_i, j), \quad \text{for } i = 1, \ldots, n;\ j = 1, \ldots, m$$

where $C(k,l) \stackrel{\text{def}}{=} 0$ if at least one of (k,l) is undefined. If we define the *makespan* as

$$C_{max}(\boldsymbol{\pi}) = C(\pi_n, m),$$

the *makespan version* of the PFSP is then to find a permutation $\boldsymbol{\pi}^*$ such that

$$C_{max}(\boldsymbol{\pi}^*) \leq C_{max}(\boldsymbol{\pi}) \ \ \forall\ \boldsymbol{\pi} \in \boldsymbol{\Pi}_n.$$

This problem has received most attention, but other objectives have also been investigated. Perhaps the second most common is to minimize the mean *flow-time* (the time a job spends in process), which amounts to minimizing

$$C_{sum}(\boldsymbol{\pi}) = \sum_{i=1}^{n} C(\pi_i, m)$$

if all jobs are available at the time origin. We call this the *flowsum version* of the PFSP.

DISTANCE MEASURES

In order to find good solutions to instances of this problem, there are many possible NS operators, each generating a different landscape, and each needing a suitable metric for measuring the distance between solutions. In principle, the distance between two solutions $\boldsymbol{\pi}$ and $\boldsymbol{\pi}'$ on a landscape should be measured by the minimal number of applications of the operator which would convert $\boldsymbol{\pi}$ into $\boldsymbol{\pi}'$. However, there seems to be no way of finding this number other than by actually tracing a path from $\boldsymbol{\pi}$ to $\boldsymbol{\pi}'$. In other words, unlike the case of Hamming distance for binary vectors, it is not possible to look at two permutations and find the operator-distance d_ω by a simple calculation.

Thus, the question arises: are there operator-*independent* distance measures for such cases? In [12], 4 metrics were tested and 2 were found to be useful, in the sense that the BVC appeared to hold. These were

- the **precedence-based metric** which counts the number of times job j is *preceded* by job i in both $\boldsymbol{\pi}$ and $\boldsymbol{\pi}'$. To get a 'distance', this quantity is subtracted from $n(n \Leftrightarrow 1)/2$;
- the **position-based metric** which takes the principle one stage further, by comparing the actual *positions* in the sequence of job j in each of $\boldsymbol{\pi}$ and $\boldsymbol{\pi}'$. For a sequence $\boldsymbol{\pi}$ we can define the 'inverse' permutation $\boldsymbol{\sigma}$ where the position of job π_i is given by $\sigma_{\pi_i} = i$. The position-based metric is then just

$$\sum_{j=1}^{n} |\sigma_j \Leftrightarrow \sigma'_j|.$$

NS OPERATORS FOR SEQUENCING

Several operators have been proposed in previous work on permutation or sequencing problems. In [12] we investigated 6 ways in which one solution can be transformed into another. Details can be found in that paper; here we simply mention those that appeared to be the most powerful.

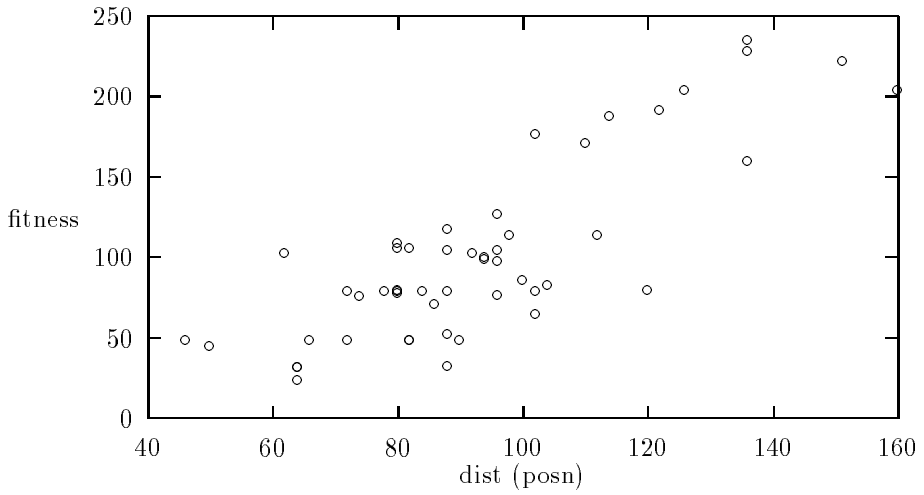

Figure 1: 50 local optima plotted in terms of their distances from a global optimum (x-axis), against their relative objective function values (y-axis). The operator used was forward shift, the strategy was next ascent, each repeated from 50 different initial random start points. In this example the distance metric used was position-based.

We define the *forward shift operator* for the case where $\mathcal{X}$ is $\mathbf{\Pi}_n$—the space of permutations $\boldsymbol{\pi}$ of length n. The operator, denoted by $\mathcal{FSH}(i,j)$ (where we assume $i<j$), is

$$\mathcal{FSH}(i,j) : \mathbf{\Pi}_n \rightarrow \mathbf{\Pi}_n \qquad \begin{cases} \pi_k \mapsto \pi_{k-1} & \text{if } i < k \leq j \\ \pi_i \mapsto \pi_j & \\ \pi_k \mapsto \pi_k & \text{otherwise} \end{cases}$$

An analogous backward shift operator $\mathcal{BSH}(i,j)$ can similarly be described; the composite of $\mathcal{BSH}$ and $\mathcal{FSH}$ is denoted by $\mathcal{SH}$.

FITNESS-DISTANCE CORRELATIONS

In order to try to understand better what the landscape of such problems looks like, operators were applied 50 times from different random initial vectors to a number of problem instances of various sizes, resulting in 50 distinct local optima in each case.

The distances of these local optima from a given point can be measured in terms of the operator-independent metrics introduced above, and then the local optima can also be compared in terms of their fitness values. If the results of Boese *et al.*[1] carry over to the flowshop problem, we would expect to find that these distances are in some sense correlated with each other. This can be examined in any particular case by plotting a graph of local optima distance from the global optimum (in terms of one of the distance metrics) against distance in terms of objective function values. Figure 1 provides an example of such a graph for 50 local optima generated by repeated restarts of a next descent procedure using the first of the 20-job, 5-machine set of problem instances generated by Taillard [16]. These problem instances were chosen for investigation partly because they have become well-known cases of flowshop sequencing, but also because the global optima are known for most if not all of the smaller instances, thus providing a fixed reference point in each case.

This graph appears to confirm that a relationship such as that found by Boese *et al.*[1] for the TSP also exists in this case of the flowshop sequencing problem—local optima seem to be 'close' to each other, and the closer they are to the global optimum, the higher the level of fitness (at least, on average).

However, it is clearly becomes tedious to plot such a graph for every operator, metric and problem instance. What is needed in order to assess any particular case is a simple measure of the relationship between the local optima. We would like to know whether such a relationship exists or not, and if so, how significant it is.

The obvious measure would be the usual correlation coefficient between the two sets of values, whose significance can be assessed in the conventional way. In the case above, that is simple enough to do. However, if the global optimum is unknown (and of course that is the normal situation!), it is difficult to measure the fitness of local optima relative to a fixed point. We could still check that the closeness of the local optima *to*

each other (in the sense of distance) is associated with their closeness in fitness, by computing a correlation coefficient from the corresponding entries in the two matrices found from the distance metric and the fitness values respectively. While such a value can easily be calculated, interpreting its significance would be difficult, as the sample clearly does not consist of independent random samples (for example, if local optima A and B are close in terms of their fitness values, and B is also close to C, then so are A and C). In these circumstances, we cannot carry out a standard hypothesis test of the correlation coefficient. As is shown in [12], it is possible to use randomization tests instead.

In the experiments carried out in [12], the correlations were almost always significant for both single and composite shift operators, and for both precedence-based and position-based metrics, so that the existence of a big valley was indicated in all the problems analysed—the 20/5, 20/10, 20/20, 50/5 and 50/10 sets of problem instances defined by Tailard [16].

Other aspects of the landscapes were also investigated. Perhaps the most interesting finding was that local optima under one operator could often be improved by switching to a different operator. This is not unexpected, given the argument we have made initially, that the landscape is not invariant, but is an artefact of the operator used. However, the proportion of local optima that could be improved, even for the $\mathcal{SH}$ operator (the best overall), and the average amount of improvement, were both somewhat larget than anticipated.

APPLICATION TO EVOLUTIONARY ALGORITHMS

As already stated above, the existence of a big valley is one reason to adduce for the good performance of perturbation methods. It is also tacitly assumed by such methods as simulated annealing and tabu search, which would lose a great deal of their potency if local optima were isotropically distributed. The BVC also motivated the development of 'adaptive multi-start' techniques described in [1]. Recent work on path-tracing algorithms (reviewed in [14, 15] also rests on the assumption that the BVC is true.

It is perhaps not quite so obvious how a big valley relates to the application of evolutionary algorithms. In the first place, the landscape induced by a GA, for example, is not the same as that induced by an operator such as $\mathcal{FSH}$ in a simple NS. Although many commonly-used GA operators for permutation problems have a similar effect to operators such as $\mathcal{FSH}$, the introduction of recombination complicates matters, in the sense of repeatedly transforming the base landscape (see [4] for further discussion on this point). Secondly, the moves made on these 'GA-landscape(s)' lack a sense of directionality—they are usually more or less random moves, rather than improving ones as in NS.

Nevertheless, it is possible to exploit the BVC in a GA, by embedding a path-tracing technique as a novel way of performing crossover. The details have already been published elsewhere [13, 17], so a simple sketch will suffice here.

If we consider the case of crossover of vectors in $\mathbb{Z}_2^l$, it is easily seen that any 'child' produced from two 'parents' will lie on a path that leads from one parent to another. Figure 2 demonstrates this fact.

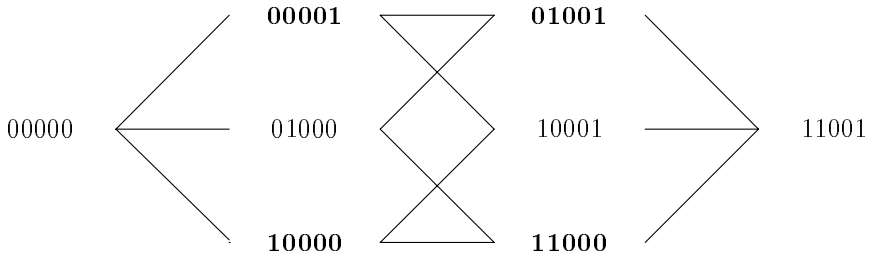

Figure 2: The diagram shows the set of paths that could be traced between the parents 00000 and 11001. Only those intermediate vectors indicated by bold type can be generated by one-point crossover, but all can be generated by uniform crossover.

In an earlier paper [11], we described such points as 'intermediate vectors': intermediate in the sense of being an intermediate point on the Hamming landscape. In the case of non-binary strings, the distance measure may be more complicated, but the principle is still relevant. It is this that we implemented for both the makespan and the flowsum versions of the PFSP. Figure 3 displays a template for performing crossover by path tracing, while figure 4 shows in a 2-dimensional diagram the idea behind it.

- Let $\boldsymbol{\pi}_1$, $\boldsymbol{\pi}_2$ be the parents. Set $\mathbf{x} = \mathbf{q} = \boldsymbol{\pi}_1$.

do

- For each neighbour $\mathbf{y}_i \in N(\mathbf{x})$, calculate $d(\mathbf{y}_i, \boldsymbol{\pi}_2)$.
- Sort $\{\mathbf{y}_i\}$ in ascending order of $d(\mathbf{y}_i, \boldsymbol{\pi}_2)$.

do

1. Select $\mathbf{y}_i$ from $N(\mathbf{x})$ with a probability inversely proportional to the index i.
2. Calculate $f(\mathbf{y}_i)$ if it is unknown (or retrieve it if previously calculated).
3. Accept $\mathbf{y}_i$ with probability 1 if $f(\mathbf{y}_i) \geq f(\mathbf{x})$, and with probability $P_c(\mathbf{y}_i)$ otherwise.
4. Change the index of $\mathbf{y}_i$ from i to n, and the indices of $\mathbf{y}_k$ $(k \in \{i+1, \ldots, n\})$ from k to $k-1$.

until $\mathbf{y}_i$ is accepted.

- Set $\mathbf{x} = \mathbf{y}_i$.
- If $f(\mathbf{x}) > f(\mathbf{q})$ then set $\mathbf{q} = \mathbf{x}$.

until some termination condition is satisfied.

- $\mathbf{q}$ is used for the next generation.

Figure 3: Generalized crossover using path tracing for a maximization problem. The path starts from $\boldsymbol{\pi}_1$ and is directed towards $\boldsymbol{\pi}_2$. The idea is that neighbours closer to $\boldsymbol{\pi}_2$ are probabilistically preferred, so encouraging a path towards $\boldsymbol{\pi}_2$ to form. Rather than simply selecting any point on a path from $\boldsymbol{\pi}_1$ to $\boldsymbol{\pi}_2$, points are chosen in an attempt to improve the solutio. Thus, better neighbours (larger f values) are always accepted, otherwise $\mathbf{y}_i$ may be accepted with probability $P_c(\mathbf{y}_i) = \exp(\Leftrightarrow\Delta f/c)$, where $\Delta f = f(\mathbf{y}_i) \Leftrightarrow f(\mathbf{x})$. This is similar to the approach of simulated annealing, corresponding to annealing at a constant temperature $T = c$. A suitable termination condition might relate to the number of trials without an improvement, or simply a fixed limit.

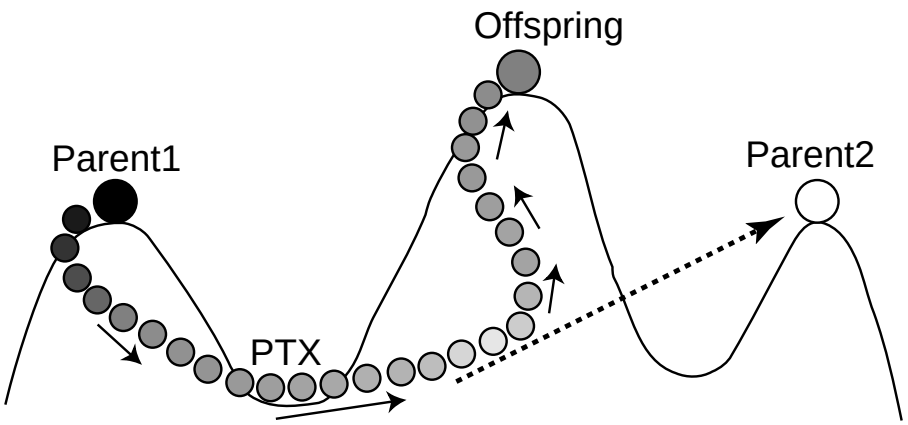

Figure 4: Path tracing crossover combined with local search: a path is traced from one parent in the direction of the other. In the 'middle' of the path, solutions may be found that are not in the basins of attractions of the parents. A local search can then exploit this new starting point by climbing to the top of a hill (or the bottom of a valley, if it is a minimization problem)—a new local optimum. The acronym PTX signifies 'path-tracing crossover'.

In this way, the concept of recombining two solutions together can be integrated with traditional NS methods, and the results obtained (see [13, 17] for the details) were gratifyingly good. For the makespan problem, embedded path tracing helped the GA to achieve results of outstandingly high quality: several new best solutions were discovered for Taillard's benchmarks. For the flowsum version, optimal solutions are not known, but the path-tracing GA consistently produced better solutions than other proposed techniques. It is also worth pointing out that in principle it is not difficult to implement, the coding effort required being essentially what is needed to couple together existing software for GA and NS in a different way.

CONCLUSION

We have discussed some ways in which the landscapes associated with heuristic search methods can be characterized, empirically studied, and exploited in order to find high-quality solutions to hard COPs. We have argued that the existence of a 'big valley' provides an explanation for the good performance of perturbation techniques in particular, and is also a factor in other methods. Finally we have briefly alluded to an approach for exploiting it in the context of GAs.

The results obtained by this approach are of high quality, but perhaps more interesting is that it is based in a fundamental way on the concept of the fitness landscape, and it thus becomes natural to develop algorithms that generalize concepts of an existing technique.

Several interesting future research questions are suggested. For instance, can we provide a formal definition of what it means for a 'big valley' structure to exist, and can we relate it to mathematical constructs associated with neighbourhood structures? Does the big valley exist almost everywhere, or are there types of problem for which it does not occur? In the area of implementation, is there scope for further work in refining the path tracing methodology and its integration into GAs?

As our understanding of the nature of search landscapes and how to exploit them develops, this promises to become an important area of research into the theory and application of heuristic techniques such as genetic algorithms.

References

[1] K.D.Boese, A.B.Kahng and S.Muddu (1994) A new adaptive multi-start technique for combinatorial global optimizations. *Operations Research Letters*, **16**, 101-113.

[2] L.K.Grover (1992) Local search and the local structure of NP-complete problems. *Operations Research Letters*, **12**, 235-243.

[3] C.Höhn and C.R.Reeves (1996) The crossover landscape for the *onemax* problem. In J.Alander (Ed.) (1996) *Proceedings of the 2^{nd} Nordic Workshop on Genetic Algorithms and their Applications*, University of Vaasa Press, Vaasa, Finland, 27-43.

[4] C.Höhn and C.R.Reeves (1996) Are long path problems hard for genetic algorithms? In H-M Voigt, W.Ebeling, I.Rechenberg and H-P Schwefel (Eds.) (1996) *Parallel Problem-Solving from Nature—PPSN IV*, Springer-Verlag, Berlin, pages 134-143.

[5] D.S.Johnson (1990) Local optimization and the traveling salesman problem. In G.Goos and J.Hartmanis (Eds.) (1990) *Automata, Languages and Programming*, Lecture Notes in Computer Science **443**, Springler-Verlag, Berlin, 446-461.

[6] S.Kauffman (1993) *The Origins of Order: Self-Organization and Selection in Evolution.* Oxford University Press.

[7] S.Lin (1965) Computer solutions of the traveling salesman problem. *Bell Systems Tech. J.* ,**44**,2245-2269.

[8] O.Martin, S.W.Otto and E.W.Felten (1992) Large step Markov chains for the TSP incorporating local search heuristics. *Operations Research Letters*, **11**, 219-224.

[9] P.Merz and B.Freisleben (1998) Memetic algorithms and the fitness landscape of the graph bi-partitioning problem. In A.E.Eiben, T.Bäck, M.Schoenauer, H-P.Schwefel (Eds.) (1998) *Parallel Problem-Solving from Nature—PPSN V*, Springer-Verlag, Berlin, 765-774.

[10] C.E.Nugent, T.E.Vollman and J.E.Ruml (1968) An experimental comparison of techniques for the assignment of facilities to locations. *Opns.Res.*, **16**, 150-173.

[11] C.R.Reeves (1994) Genetic algorithms and neighbourhood search. In T.C.Fogarty (Ed.) (1994) *Evolutionary Computing: AISB Workshop, Leeds, UK, April 1994; Selected Papers.* Springer-Verlag, Berlin, 115-130.

[12] C.R.Reeves (1999) Landscapes, operators and heuristic search. *Annals of Operational Research*, **86**, 473-490.

[13] C.R.Reeves and T.Yamada (1998) Genetic algorithms, path relinking and the flowshop sequencing problem. *Evolutionary Computation*, **6**, 45-60.

[14] C.R.Reeves and T.Yamada (1999) *Embedded Path Tracing and Neighbourhood Search Techniques in Genetic Algorithms*. In K.Miettinen, M.M.Mäkelä, P.Neittaanmäki and J.Périaux (Eds.) (1999) *Evolutionary Algorithms in Engineering and Computer Science*, John Wiley & Sons, Chichester, 95-111.

[15] C.R.Reeves and T.Yamada (1999) *Goal-Oriented Path Tracing Methods*. To appear in D.A.Corne, M.Dorigo and F.Glover (Eds.) (1999) *New Methods in Optimization*, McGraw-Hill, London.

[16] E.Taillard(1993) Benchmarks for basic scheduling problems. *Euro.J.OR*, **64**, 278-285.

[17] T.Yamada and C.R.Reeves (1998) Solving the C_{sum} permutation flowshop scheduling problem by genetic local search. In *Proc. of 1998 International Conference on Evolutionary Computation*, 230–234, IEEE Press.

[18] G.Zweig (1995) An effective tour construction and improvement proedure for the traveling salesman problem. *Operations Research*, **43**, 1049-1057.