Journal of Global Optimization (2005) 31: 579–598
DOI 10.1007/s10898-004-9968-y

Matching Stochastic Algorithms to Objective Function Landscapes

W.P. BARITOMPA[1], M. DÜR[2], E.M.T. HENDRIX[3], L. NOAKES[4], W.J. PULLAN[5] and G.R. WOOD[6]
[1]*Department of Mathematics and Statistics, University of Canterbury, Christchurch, New Zealand*
[2]*Department of Mathematics, Darmstadt University of Technology, D-64289 Darmstadt, Germany*
[3]*Group Operations Research and Logistics, Wageningen University, Hollandseweg 1, 6706 KN Wageningen, The Netherlands*
[4]*School of Mathematics and Statistics, University of Western Australia, Nedlands WA 6907, Australia*
[5]*School of Information Technology, Griffith University, Gold Coast, Australia*
[6]*Department of Statistics, Macquarie University, North Ryde, NSW 2109, Australia (e-mail: gwood@efs.mq.edu.au)*

Abstract. Large scale optimisation problems are frequently solved using stochastic methods. Such methods often generate points randomly in a search region in a neighbourhood of the current point, backtrack to get past barriers and employ a local optimiser. The aim of this paper is to explore how these algorithmic components should be used, given a particular objective function landscape. In a nutshell, we begin to provide rules for efficient travel, if we have some knowledge of the large or small scale geometry.

Key words: Backtracking, Global optimisation, Local optimisation, Search region, Simulated annealing, Temperature

Mathematics Subject Classification. 90C65, 90C30, 65K05

1. Introduction

When selecting a stochastic algorithm to solve a practical problem it is useful to have some qualitative knowledge about the objective function landscape. In practice, this landscape knowledge may come from questioning context experts, a closed form of the objective function or as a by-product of using a particular algorithm on instances of the problem.

The aim of this paper is to explore how landscape knowledge can be used sensibly to tune a stochastic search algorithm so that it runs efficiently. Underpinning the work is the assumption that moves broadly downward, on some scale, will lead to a global minimum value (although we acknowledge that such a strategy will not always be appropriate for finding the global minimum). For example, a minimisation problem is analogous to descending a mountain shrouded in mist; following a ridge downward would seem a sensible strategy and this demands that we not step too far from the ridge path. Such strategies, which correspond to algorithm parameter settings, are of practical interest. Once, for example, simulated annealing with a given setting of parameters is known to be successful, then such settings may be used to solve variants of the problem efficiently.

There is an extensive array of stochastic global optimisation heuristics described in the literature, examples being [2, 4–6, 8, 10, 11]. Commonly, these use limited search regions, some amount of backtracking (acceptance of worse values) and local optimisation. Here we provide initial guidelines, by no means exhaustive, describing conditions under which each is worthwhile. That is, we show how to tailor them to the particular needs of an objective function, in order to efficiently reach the global optimum.

This aim is achieved in the following way. We set up a family of objective functions which are designed to capture the features which typically resist ready movement of an algorithm to the global minimum: high frequency and low frequency oscillation, multiple minima, saddles, valleys, and high dimension. These features are controlled using a small set of parameters. With these as the targets of our interest, we then set up a family of algorithms in corresponding fashion. In these we can control the size of the local search region, the ease of backtracking and the readiness to do a local search. Three parameters control these aspects of the algorithm.

The No Free Lunch Theorem [13] announced that all global optimisation algorithms have the same performance, when averaged across all objective functions. The vision of this paper has been a reaction to that result: given a landscape in the parametric family, to identify an algorithm in the parametric family well adapted to searching that landscape. Necessarily that vision has shrunk as work progressed; the paper now initiates the connection. It is hoped that this paper will stimulate future work which will progressively demonstrate the links between landscape and algorithm. The main contributions of this paper are to consider two simple landscapes (valleys and ridged slopes) and to describe algorithms which descend them efficiently. In doing this, the paper initiates a theory of algorithm design; at present the choices such design might render objective are made in an ad hoc way.

The paper is organised as follows. In Section 2 we set up the global optimisation environment, describing the objective functions and their parameterisation, together with a generic stochastic algorithm controlled by three parameters. In Section 3 we discuss the matching of algorithm parameter to landscape feature, using both simulation and theoretical analysis, focusing on search neighbourhood and backtracking. Section 4 then considers the role of local search. Section 5 concludes the paper with a summary.

2. Global Optimisation Environment

Global optimisation requires an objective function and an algorithm. In this section we capture critical features of objective functions in a parameterised family, then do the same for a family of algorithms.

2.1. OBJECTIVE FUNCTION LANDSCAPES

Our aim was to produce a family of functions that exhibited landscape features that affect the performance of a stochastic global optimisation algorithm. Features could be at a high level (representing the coarse overall features) or at a low level (which can be viewed as noise or perturbations not affecting the overall shape, but possibly interfering with an algorithm's performance). As outlined in the preliminary discussion paper, these features include the modality, the size of basins of attraction of the local minima and the magnitude of noise.

The family is built using functions defined on $[0,1]\times[0,1]$ (see Figure 1(a) for an example). This two-dimensional family captures some of the above-mentioned features by being built up of a one-dimensional high level base function of x (the first coordinate of a point in the domain) as illustrated in Figure 1(b), with perturbations, in the form of added fixed frequency oscillations, as illustrated in Figure 1(c). The second coordinate y comes into play to provide a valley which cuts through the oscillation barriers to provide a less obstructed narrow path, and as well provides a gentle curvature in the second direction.

Full details of the construction of the family are given in Appendix A. In brief, however, the functional form is

$$f(x,y) = f_{\text{base}}(x) + o(x)g(x,y) + \frac{(y-1/2)^2}{10}$$

where $f_{\text{base}}(x)$ is as pictured in Figure 1(b), parameterised by BasinRatio and LevelRatio*. The function $o(x)$ provides the low level perturbations in the form of oscillations. It is parameterised by OscillationNumber, OscillationWidth and OscillationHeight as illustrated in Figure 1(c). Pulse oscillations† were chosen so that width and height were independent; adding such a perturbation to the high level function creates obstructions that are independent of the base function. The function g is equal to one except over an oscillating curve in the plane where it goes to a value S_r less than one.

* BasinRatio is the ratio of the width of the right highlevel basin to the width of the left highlevel basin. If this is zero, then the high level function has only one local minimum and is unimodal. LevelRatio is the ratio of the escape height from the right high level basin to the escape height from the left high level basin. If the ratio is less than one, the left one is global, if the ratio equals one there are two global minimisers and if the ratio is more than one, the right one is global.

†Smooth oscillations were tried and there was little difference in the experimental results.

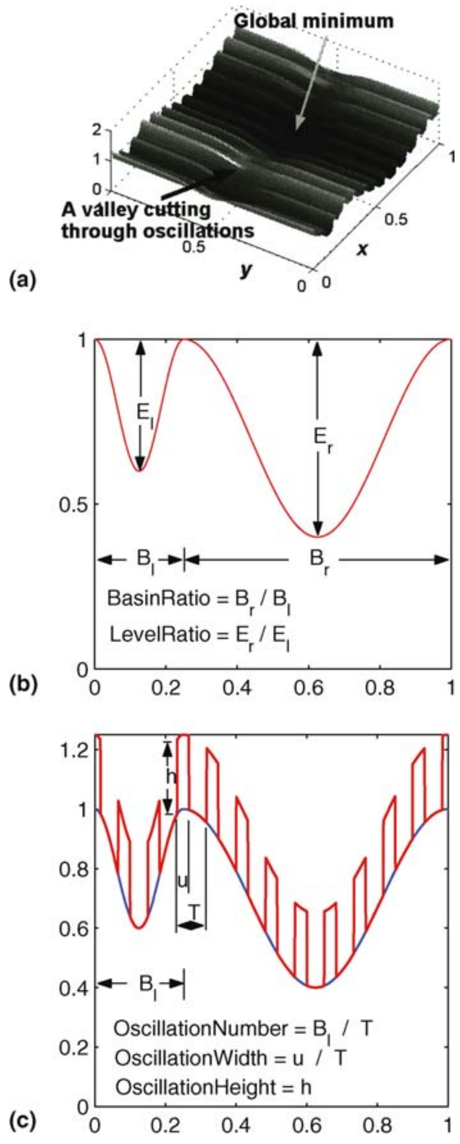

Figure 1. A typical landscape is shown in (a). The high level function of the first variable x is shown in (b), together with the added oscillation in (c). Parameters controlling the functions are also described.

The effect is to lower f over this region and provide a narrow valley which leads to the global optimum. For x values at the peaks of the oscillations, moving in the y direction is walking on a ridge which drops down to the bottom of the valley (thus creating a saddle). This valley function is parameterised by SaddleRatio, S_r, the ratio of the height of the pass to the height of the ridge, SaddleWidth, the width of the valley and SaddleDev which gives the relative deviation from the centre line. When SaddleDev is zero, the valley is straight along $y = 1/2$, and when SaddleDev is 0.5 it moves between $y = 0$ and 1.

One feature that is missing from our family is the dimension of the domain. This is known to be important, but it was thought that some of the "curse of dimensionality" could still be captured using the two-dimensional family. For example, using a very narrow valley which almost cancelled out very many high obstructions could mimic difficulties found in higher dimensional examples. This was observed to some extent, and it became apparent that valleys are an important phenomenon. In Section 3, however, we explore one very simple high dimensional example,

$$f(x_1, x_2, \dots, x_n) = \begin{cases} -sx_1, & \text{when } |x_i| < v,\ i = 2, \dots, x_n \\ \infty, & \text{otherwise} \end{cases}$$

In general, however, the two-dimensional definition can be extended to create functions on the unit hypercube using

$$f(x_1, y_1, x_2, y_2, \dots, x_n, y_n) = \sum_{i=1}^{n} f(x_i, y_i)$$

2.2. STOCHASTIC ALGORITHMS

Three fundamental properties of a stochastic global optimisation algorithm were chosen for study:

1. The size of the search region.
2. The ability to backtrack.
3. The use of local search.

The need to maintain simplicity at this early stage of the investigation made it desirable that each factor should be controlled by a single parameter. In future, a, t and p will control search region, backtracking and local search, respectively. A choice for each of these parameters determines a particular realisation of the algorithm. The role of each parameter is now described, followed by a formal description of the generic algorithm. As the parameters vary, the algorithm moves over an "algorithm space"; this will be pictured and discussed.

First, we describe the search region parameter a, which varies from zero to two. The search region on any iteration is chosen to be a hypercube, centred on the current point, of radius r. Recall that the objective function domain is also, in general, a hypercube, with width one in each direction. It would be simplest to use a radius r for the search region of $a/2$ at each iteration, but a richer one-dimensional mechanism was preferred. For the search parameter a, r is chosen uniformly on

$$\begin{aligned} &[0, a], \quad \text{for } 0 \leqslant a \leqslant 1 \\ &[a - 1, 1], \quad \text{for } 1 < a \leqslant 2 \end{aligned}$$

Note, the expected value of r is $a/2$. This process allows a wide range of search styles. When $a = 0$, the algorithm chooses an initial point and then remains there. When $a = 2$, the search region at each iteration is the entire domain. For intermediate values of a, values close to the current point are favoured but it is still possible to search broadly. Note that this flexibility is precluded if we always choose $r = a/2$.

Second, we discuss backtracking. Parameter $t \in [0, \infty]$, a temperature, controls this via a standard Boltzmann condition. Finally, parameter $p \in [0, 1]$ controls use of local search, in that at each iteration a local search is used with probability p. The algorithm is now presented formally. It is assumed that $a \in [0, 2]$, $t \in [0, \infty]$ and $p \in [0, 1]$ have been selected.

GENERIC STOCHASTIC ALGORITHM – FOR a, t AND p

```
Step 1. Set $i = 0$. Generate $X_0$ uniformly on the domain and evaluate
        $f(X_0)$.
Step 2. (i)   Choose a search radius $r$, using search parameter $a$.
        (ii)  Select $X_{\text{new}}$ uniformly in the intersection of the domain with
              a hypercube of radius $r$ centred on $X_i$.
        (iii) With probability $p$, run a local search from $X_{\text{new}}$, replacing
              $X_{\text{new}}$ with the local minimiser.
        (iv)  Set $X_{i+1} = X_{\text{new}}$ with probability $\min\left\{1, \exp\left(\frac{f(X_i) - f(X_{\text{new}})}{t}\right)\right\}$
              else set $X_{i+1} = X_i$.
Step 3. If a stopping criterion is met, stop. Otherwise, increment $i$ and
        return to Step 2.
```

For different a, t and p the algorithm takes on different character, as summarised in the "cheese" of algorithms pictured in Figure 2.

As a increases (so as we move from left to right in the cheese), local generation of the next point gives way to global generation of the next point. As t increases (so as we move from the front face to the back face), the propensity to backtrack increases. Finally, as p increases (so as we move from low to high in the cheese), more local search is added. Note that when $a = 2$ and p is fixed, the algorithms produced as t varies are essentially equivalent; the raw domain sequences are the same, since X_{new} is independent of X_i. Certain vertices and edges are broadly familiar. In particular, the following table explains the marked points in Figure 2.

The aim now will be to tailor the algorithm to the geometry of the objective function. We conclude this section by remarking that our simplified algorithms do not allow parameters a, t and p to vary with the iteration; this should be allowed in real algorithms.

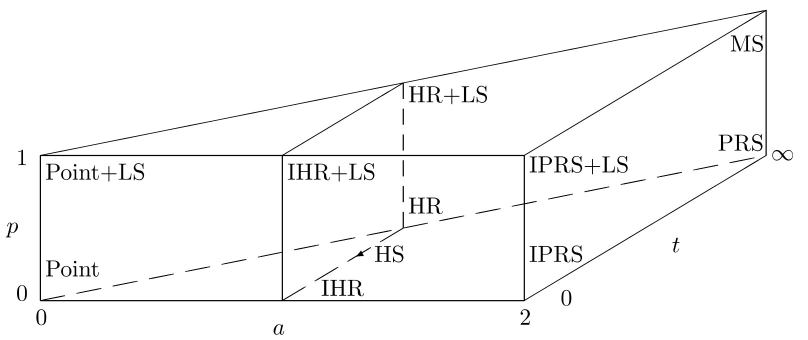

Figure 2. The space of algorithms: parameter a controls the extent of the search, parameter t controls backtracking using a Boltzmann condition and parameter p is the probability with which a local search is used. Familiar algorithms can be recognised within the "cheese"; for example, multistart corresponds to the top right corner where $(a, t, p) = (2, \infty, 1)$.

Abbreviation	Explanation
Point	Single point, repeated
Point + LS	Single point followed with a local search, repeated
IHR	Improving hit-and-run [14]
IHR + LS	Improving hit-and-run with local search
HS	Hide-and-seek [7]
HR	Hit-and-run [9]
HR + LS	Hit-and-run with local search
IPRS	Improving pure random search
IPRS + LS	Improving pure random search with local search
PRS	Pure random search
MS	Multistart [2]

3. Matching Algorithms to Functions: Role of Search Size and Temperature

Our stochastic optimisation method uses a homogeneous (the distribution does not depend on the iteration number) Markov (the distribution depends only on the current location) random walk to explore the domain. When $t = 0$, only steps which decrease the objective function value are used. When $t = \infty$, however, the random walk in the domain takes no account of the landscape, and movement into a particular region is a consequence only of the starting position and the random walk mechanism.

Regardless of the overall assumption that the landscape leads in the right direction, just how good is the algorithm at downward movement? Our aim in this section is to determine, for some typical landscape types, how parameters a, t and p should be set in order to move downwards efficiently. The original aim (during the workshop week) was to match points of the objective function space to points in the algorithm space. We have chosen a lesser objective since that time, namely to match certain non-global aspects of objective function space to values of parameters. In particular, we consider two situations: movement down a valley and movement down a slope with obstructions. The pattern is to develop some theoretical results, then to illustrate them with simulation. Throughout this section we assume that $p = 0$, so there is no local search.

3.1. DESCENDING IN A VALLEY

High dimensional problems, subject to constraints, can require us to follow a low dimensional valley to the global minimum. We address the question of how to follow this valley, once we are in it, beginning by visualising a steep-walled but straight valley. We are able to move to a point in our local neighbourhood and scan for the global minimum. If we choose too small a neighbourhood then we will not move far at each stage. On the other hand, if our search neighbourhood is too large, we will spend a lot of time scanning points outside the valley; there is evidently an optimal search radius. We will find, for a valley as shaded in Figure 3, that the optimal half-width (or radius) of the search box is precisely the width of the valley. As the dimension of the domain increases, the optimal radius reduces. Recall that the expected value of the half-width, in the generic algorithm, is $a/2$, so we need to keep in mind that the optimal algorithm parameter is twice this optimal half-width.

More formally, to define a valley of half-width v in $\mathbf{R}^n$ consider the landscape of objective function $f : \mathbf{R}^n \longrightarrow \mathbf{R} \cup \{\infty\}$, depending only on the first coordinate, given by

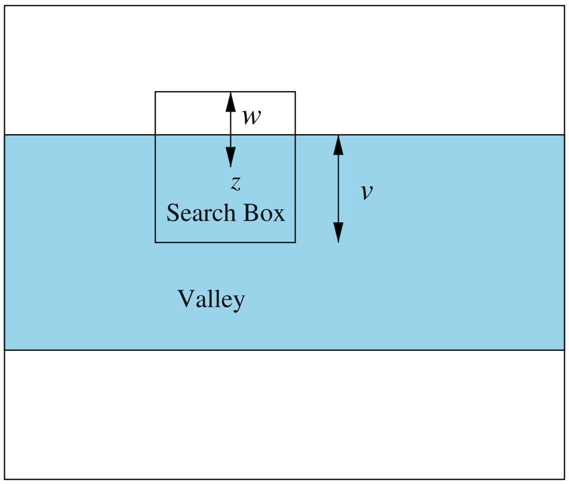

Figure 3. The valley, of half-width v, is shown shaded, for $n = 2$. The search region, centred on z in the valley, has half-width w.

$$f(x) = \begin{cases} g(x_1), & \text{when } |x_i| < v,\ i = 2, \ldots, n \\ \infty, & \text{otherwise} \end{cases}$$

where g is any function from **R** to **R**. The points of the domain having finite value form what we term a valley of half-width v. This may seem artificial, but it is typical of how a constrained optimisation problem with a low dimensional feasible region essentially comes from a high dimensional unconstrained optimisation problem on a box, via use of a penalty function. In the special case where g is linear (with $g(x_1) = -sx_1$ with $s > 0$), we shall calculate the expected movement to the right, conditional on staying in the valley (which we assume is the direction of decreasing g). It is reasonable to suppose that the time before getting into the valley is inversely proportional to the relative size of the valley and that this does not vary over a wide range of a values. We find, unsurprisingly, that $t = 0$ is best, as higher temperatures will only lower the expected improvement. Finding the optimal a is more interesting.

We begin with a proposition that describes the expected movement down a valley (recall that this is determined by an objective function whose domain is $\mathbf{R}^n$).

PROPOSITION 3.1. *Let v be the half-width of the valley and w be the half-width of the search box. Without loss of generality, let $z = (0, z_2, \ldots, z_n)$ be the current point in the algorithm, where z_i is assumed to come from a uniform distribution on $[-v, v]$ for $i = 2, \ldots, n$. Then the expected movement in the first coordinate at the next iteration is*

$$V^{n-1} \frac{1}{2w} \int_{-w}^{w} x \min\{1, \exp((g(0) - g(x_1))/t)\} \mathrm{d}x$$

where

$$V = \left(\frac{1}{2vw} \int_0^v L(x) \mathrm{d}x \right) = \begin{cases} 1 - \frac{w}{4v} & \text{when } w \leqslant 2v \\ \frac{v}{w} & \text{otherwise} \end{cases}$$

and

$$L(x) = \begin{cases} 2w, & w \leqslant v - x \\ w + v - x, & v - x < w \leqslant v + x \\ 2v, & v + x < w \end{cases}$$

Proof. The expected movement in the first coordinate is the product of the probability that we hit the valley and the expected improvement given that we do hit the valley. We now show that these quantities correspond to the two terms in the displayed expression in the proposition statement. We begin by considering the probability of hitting the valley. For fixed i, $i = 2, \ldots, n$,

$$P[\ |x_i| < v \mid \text{current point } i\text{th coordinate is } z_i]$$

is $L(z_i)/(2w)$ since $L(x)$ is the length in the valley of a search box of half-width w, centred at x. Since z_i can be assumed to be drawn uniformly on $[0, v]$, we define

$$P[|x_i| < v] = \frac{1}{v}\int_0^v P[|x_i| < v| \text{ current point } i\text{th coordinate is } z_i]\mathrm{d}z_i$$
$$= \frac{1}{2vw}\int_0^v L(z_i)\mathrm{d}z_i = V$$

The integral giving V was calculated in Maple. Now, since x_1, the value of the first coordinate after the iteration, can be any value in $[-w, w]$

$$P[\text{Hit valley}] = P[|x_2| < v] \times \cdots \times P[|x_n| < v] = V^{n-1}$$

Finally, for the expected value of the movement, conditional upon hitting the valley, we have

$$E[x_1|\text{Hit valley}] = \int_{-w}^{w} x_1 \frac{\min\{1, \exp((g(0) - g(x_1))/t)\}}{2w}\mathrm{d}x_1$$

since the density of X_1 is $\min\{1, \exp((g(0) - g(x_1))/t)\}/(2w)$, the uniform distribution on the interval $[-w, w]$ tempered by the acceptance probability. □

We turn now to explore the extreme cases where $t = 0$ and $t = \infty$, and then the intermediate case.

Case $t = 0$.
We begin with an immediate consequence of Proposition 3.1.

COROLLARY 3.1

(i) $P(\text{Hit valley}) = V^{n-1} = \begin{cases} \left(1 - \frac{w}{4v}\right)^{n-1} & \textit{when} \quad w \leqslant 2v \\ \left(\frac{v}{w}\right)^{n-1} & \textit{otherwise} \end{cases}$

(ii) *When* $g(x_1) = -sx_1$ (*with* $s > 0$) *and* $t = 0$ *the expected movement, conditional upon hitting the valley, is* $w/4$.

Proof. The proof involves routine integrations. □

Note that in (ii), for the linear valley, the expected improvement does not depend on s or the value of the first coordinate of the starting point. Graphs of the expected movement for this special case (which we call the linear valley) were created in Maple, using the expression given in Proposition 3.1, and they agree closely with results produced by simulation (see Figure 4). The remainder of this section deals only with the linear valley.

As the expressions for the probability of hitting the valley and the

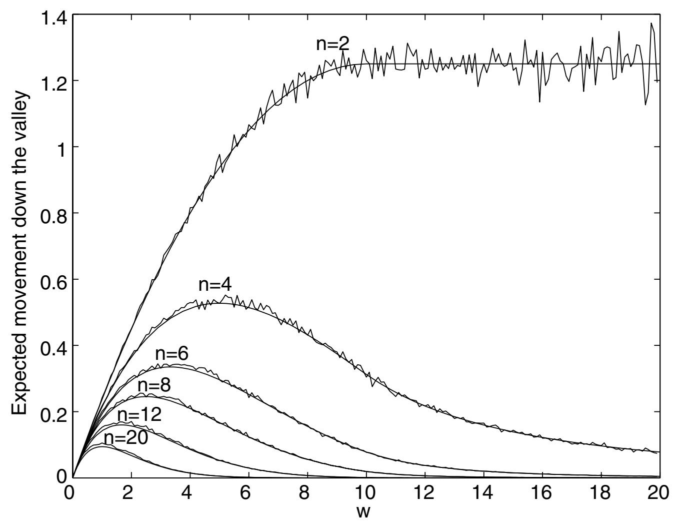

Figure 4. The smooth curves plot the expected movement (Proposition 3.1) of the valley objective function when $g(x_1) = -sx_1$, as w changes, for a given dimension n. The dimension takes the values 2, 4, 6, 8, 12 and 20 (from top to bottom), with $v = 5$ and $t = 0$. Note that the optimal search radius w reduces as the dimension increases, as does the optimal movement. Overlaid on each curve is a simulation of the same situation.

expected movement conditional upon hitting the valley are explicitly known (Corollary 3.1), we have the following result which characterises the optimal w shown in Figure 4.

PROPOSITION 3.2. *For dimension greater than two, $g(x_1) = -sx_1$ (with $s > 0$) and $t = 0$, the optimal w is $4v/n$ which gives expected movement of*

$$\frac{v}{n}\left(\frac{n-1}{n}\right)^{(n-1)} \approx \frac{v}{(n-1/2)e}$$

For dimension two, the maximum expected movement is first achieved at this optimal w value and is constant for larger values.

Proof. For dimension greater than two, the optimum half-width is in the region where the probability of hitting the valley is $((v - w/4)/v)^{n-1}$ and the expected movement conditional upon hitting the valley is $w/4$. The product gives the expected movement in the valley. This is maximised at the value stated in the proposition. □

More generally, it is possible to define a valley of dimension d $(1 \leqslant d \leqslant n)$ as that part of the domain with finite objective function value for

$$f(x) = \begin{cases} g(x_1, \ldots, x_d), & \text{when } |x_i| < v,\ i = d+1, \ldots, n \\ \infty, & \text{otherwise} \end{cases}$$

where g is any function from $\mathbf{R}^d$ to $\mathbf{R}$. We then have the following result.

PROPOSITION 3.3. *For dimension n greater than d and the simple slope function $g(x_1, \ldots, x_d) = -sx_1$, the optimal w is $4v/(n-d+1)$, giving expected movement in the first coordinate of*

$$\frac{v}{n-d+1}\left(\frac{n-d}{n-d+1}\right)^{n-d} \approx \frac{v}{(n-d+0.5)e}$$

For dimension $n = d$, the maximum expected movement occurs at this optimal w value and is constant for greater values.

Proof. This proof is the same as for Proposition 3.2, except that $n-1$ is replaced by $n-d$. □

Case $t = \infty$.

Consider the case where $n = 1$. When $t = 0$, the landscape is homogeneous and the random walk is only in one direction, so the expected movement is a reasonable measure of performance; the expected number of iterations to first move a unit distance to the right is inversely proportional to the expected movement and hence w (see [1]). When $t = \infty$, however, we have an unbiased walk, with expected movement of zero! For such a walk, the expected number of iterations to first move a unit distance to the right is inversely proportional to the variance and hence w^2.

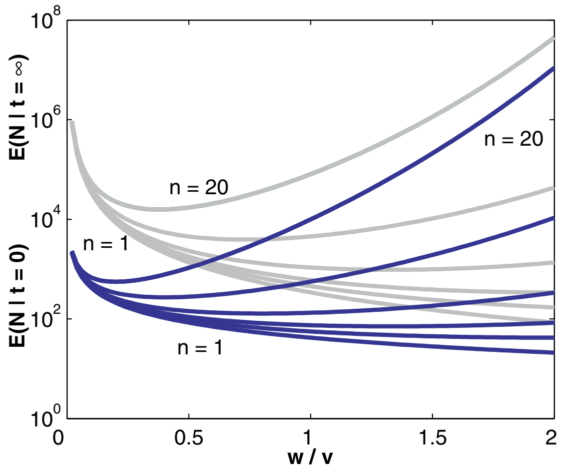

Figure 5. The expected number of iterations N to move unit distance to the right, for the linear valley (with $s = 1$), against w/v, as the dimension n varies. The dark curves correspond to the case where $t = 0$ and the light curves to $t = \infty$. For each temperature, n takes the values 1, 2, 3, 5, 10 and 20. For $n = 2$ and $t = 0$ note that N reduces as w increases, consistent with the increased movement shown in Figure 4 as w increases. The graphs were produced using $v = 0.1$ (so w ranged from 0 to 0.2), representative of small v, and allowing the boundary effects in the discrete approximation to be ignored.

These curves, found by discretely approximating the continuous walk of our algorithm on **R**, are shown in Figure 5 and labelled with $n = 1$. (Note, to avoid boundary effects, points that would go outside the interval are clumped with the appropriate end point.) The curves in Figure 5 for higher dimensions were found by dividing the value of the curve for $n = 1$ by the probability of hitting the valley.

Note the functional form of these curves gives, for $t = 0$, the optimal performance when $w/v = 4/n$ for $n \geqslant 2$ (as in Proposition 3.2 also). For $t = \infty$, optimal performance occurs when $w/v = 8/(n+1)$ for $n \geqslant 3$. (The expected number of iterations to move unit distance to the right is inversely proportional to w^2, while the probability of hitting the valley is $((v - w/4)/v)^{n-1}$. Taking the product, differentiating and solving gives the stated result.)

Two results here are noteworthy for the two sets of graphs in Figure 5. First, for fixed t, as n increases the optimal w decreases. Thus a narrower search is better as the dimension increases. Second, for n fixed, $t = 0$ has a smaller optimal w than $t = \infty$. Thus a narrower search is also better as backtracking decreases.

Case $0 \leqslant t \leqslant \infty$.

In this case the expected number of iterations to move unit distance to the right runs monotonically between the two extremes given by the curves in Figure 5. The transition, however, is quite sudden, as illustrated in Figure 6. Over the interval from $t = 0$ to $t = t_0$, where $t_0 = s/30$, the expected number of iterations to move one unit to the right is comparable

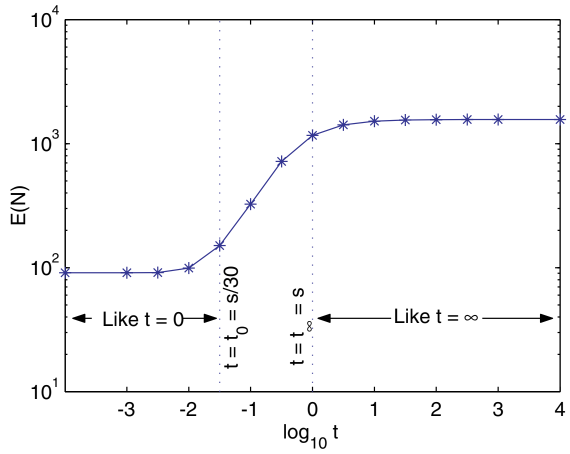

Figure 6. The graph shows the expected number of iterations to move one unit to the right, against the temperature t on a log scale. As temperature increases, the algorithm becomes less efficient, but it does so by moving from a $t = 0$ plateau to a $t = \infty$ plateau between $t = t_0 = s/30$ and $t = t_\infty = s$. This phenomenon was noted for a range of s and w values. Here $n = 1$, $w = 0.04$ and $s = 1$.

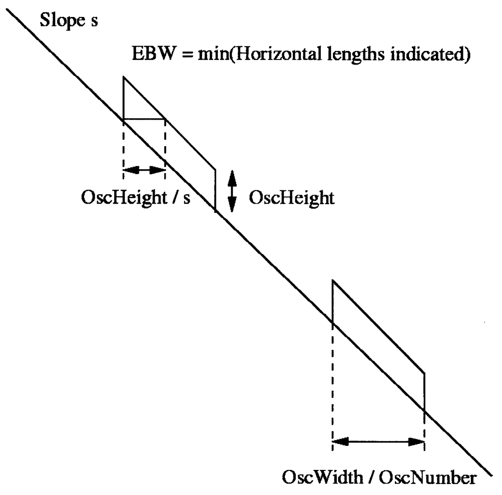

Figure 7. A cross-section, down the fall-line, of a slope with obstructions.

to that for $t = 0$; for $t \geqslant t_\infty$, where $t_\infty = s$, the behaviour is comparable to that at $t = \infty$.

3.2. DESCENDING A SLOPE WITH OBSTRUCTIONS

We turn now to explore landscapes with obstructions on a linear slope; the expected number of iterations required to improve by a unit amount in the first coordinate will be used to compare algorithm configurations.

We begin by defining a linear slope with obstructions. The slope runs linearly from top left to bottom right, and then obstructions of the type shown in Figure 1(c) are added, OscNumber of them. The result is pictured in Figure 7.

We allow ourselves two ways to get past obstructions: raising the temperature t or using a large enough a. Each option has its merits, we shall find, depending on the character of the landscape.

A critical measure of the character of the landscape which we shall use is the "effective barrier width" (EBW), defined as min(OscWidth/OscNumber,OscHeight/s). This is most easily seen in Figure 7. In this context, OscWidth/OscNumber provides the horizontal width of the barrier while OscHeight/s provides the horizontal distance from the left of the barrier to the point to the right where the barrier again falls to the level at the left of the barrier. Thus EBW measures the smallest jump that must be made, starting at the left of the obstruction, moving from left to right, before returning to a lower level.

In the previous section we saw that for an unobstructed valley $w = 4v/n$ is best with $t = 0$. For the obstructed landscape, if this width is enough to pierce the obstruction at the same time it should be used. That is, if w is bigger than EBW, then the performance is the same as if the function was

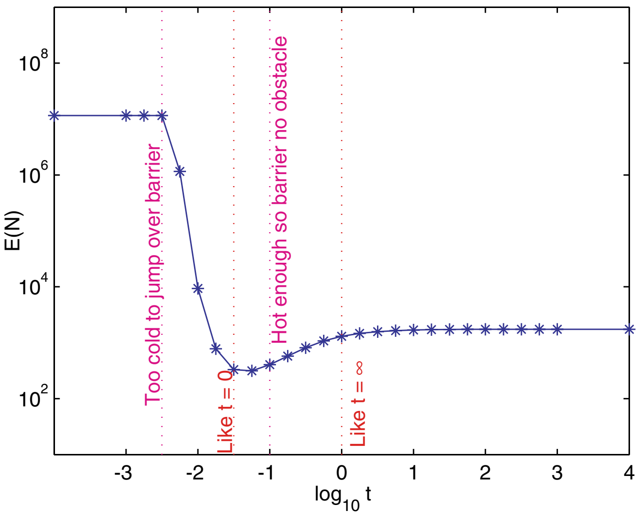

Figure 8. The relationship between expected number of iterations to move a unit horizontal distance down a slope with obstructions, and temperature. Here $n = 1$, $s = 1$ and both the width and height of the obstruction are 0.1, so EBW is 0.05, since OscNumber is 2. The half-width of the search region is $w = 0.04$, less than EBW. On the left, the temperature is too cold to jump the obstructions, so the number of iterations is high. As temperature climbs, it reaches an optimal value for the algorithm. As temperature climbs further the algorithm becomes less effective, taking little note of the landscape.

not obstructed, so $t = 0$ is best. On the other hand, if w is less than EBW then the situation is more complicated.

We begin by exploring the relationship between the expected number of iterations to move unit distance to the right, on a slope with obstructions, and temperature (Figure 8). We will then bring in the factor of obstruction height (Figure 9). All results in this section are found using a discrete approximation to the continuous walk, though we consider that the qualitative conclusions we draw will hold over a broad range of similar landscapes.

Figure 8 shows how the ability to move down a slope with obstructions varies with temperature, when w is less than EBW. Observe in Figure 8 that there are two key temperatures which depend on barrier height. Temperature $t < t_c$, $t_c = \text{OscHeight}/30$ is too cold to jump the barriers and so $E(N)$ is infinite (the graphic was made using Matlab, programmed to plot "infinity" at 10^7). Temperature $t > t_h$, $t_h = \text{OscHeight}$, is hot enough to jump barriers as if no obstruction is present and $E(N|\text{obstructed function}) = E(N|\text{unobstructed function})$ (as on the right in Figure 6).

Figure 9 builds on the graphic just discussed. All configurations are unchanged, but the height of the obstruction is allowed to vary, taking values of 0, 0.1 (as before), 1 and 10, so t_c and t_h move with respect to t_0 and t_∞. The lowest curve shows the performance in the unobstructed

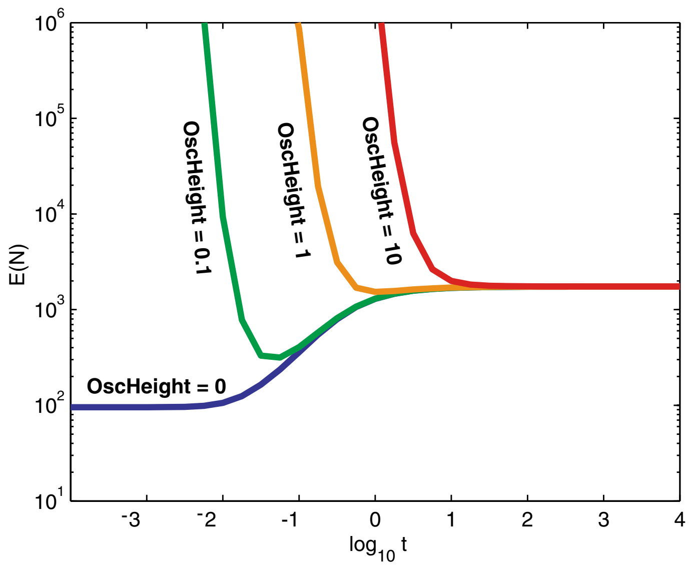

Figure 9. This graphic shows the effect of increasing obstacle height on performance; all parameters, bar OscHeight, are as in the simulation of Figure 8. For zero obstacle height, zero temperature is best. For small obstacle height (0.1 in the graphic, as seen in Figure 8), an optimal temperature exists. As the obstacle height increases, $t = \infty$ offers the best performance.

case. The effect of an obstruction is to force this curve to go to infinity as t goes to zero. The obstructed curves asymptotically approach the unobstructed curve. The region of contact varies with the height of the obstruction. For very low obstructions this contact is in the region $t < t_0$, for moderate obstruction size this contact is in the transitional region and for large obstructions this contact is in the region $t > t_\infty$.

4. Matching Algorithms to Functions: Role of Local Search

In this section we provide an initial result indicating when local search is worthwhile. Given a local minimiser (this could be a theoretical optimiser that finds the lowest point in the "geometric" basin, or a practical method using a given step size), we define the "flattened" function as

$$f^c(x) = f(x_{\text{local}}(x))$$

where $x_{\text{local}}(x)$ is the local minimiser found when starting at x. This idea has been used by others, for example by Wales and Doye in applying simulated annealing to the solution of Lennard–Jones problems [12].

We have seen in the previous section that width of valleys and size of oscillation barriers are some landscape features which need to be considered when choosing reasonable values for the parameters a and t. The decision as to whether to use a local method (that is, setting p to be non-zero) depends on properties of f^c. If the local method is used at each step, we are really applying the basic stochastic optimisation method to f^c. For this reason, it is the size of the valleys and barriers of the flattened function that are the important features. Of course evaluation of f^c is more costly than evaluation of f, so it is a question of cost versus benefit.

A result in this area is possible and presented now. For a given function, we can give the condition that determines whether multistart or pure random search will be more efficient.

THEOREM 4.1. *Let α be the probability measure of the level set of the acceptable global minima (that is, the set of points that are acceptable answers). Let β be the probability measure of the level set of the acceptable global minima of f^c. Let one evaluation of f^c be equivalent to say N_c evaluations of f. Then multistart is better than pure random search if and only if* $N_c \leqslant \frac{\log(1-\beta)}{\log(1-\alpha)} \approx \beta/\alpha$.

Proof. For a given number of function evaluations, say N, we can do N iterations of pure random search. The probability of finding a point in the acceptable level set around the global minimum is $1-(1-\alpha)^N$. For this number of function evaluations, we can do N/N_c iterations of multistart, which will have probability of success $1-(1-\beta)^{N/N_c}$. The second probability is greater than the first precisely when the condition of the theorem holds. $\square$

This result can be conveniently illustrated using the objective functions of Section 2. A wide barrier width favours multistart, since then the catchment (measured by α) for the local search is large. So if we use an objective function with parameters L to F, as ordered in Appendix A, of (1.1, 3, 3, 1, 0.9999, 1, 0.5, 0.5, 3) and $N_c = 20$ then we find that pure random search, so the generic algorithm with parameters $(a, t, p) = (2, \infty, 0)$, performs worse than multistart, the generic algorithm with parameters $(a, t, p) = (2, \infty, 1)$.

On the other hand, narrow barrier widths favour pure random search. So for the objective function with parameters (1.1, 3, 3, 1, 0.1, 1, 0.5, 0.5, 3) and $N_c = 20$, pure random search performs better than multistart. Here we have altered OscWidth; for values less than 0.9 pure random search is superior, while multistart is superior for values higher than 0.9.

5. Summary and Future Directions

The central aim of this paper has been to understand how a stochastic global optimisation algorithm can be tuned to the landscape of the objective function. The initial vision of the research group was to find a mapping from a parameterised family of objective functions (exhibiting typical recalcitrant features) to a parameterised family of stochastic search algorithms (exhibiting typical behaviour). As a step towards this end, simple such families were set up in Section 2 of this paper. As the difficulty of the goal was realised, however, the aim narrowed to one of establishing some links from objective function features to algorithm type. In particular, in Section 3 we investigated how to tune an algorithm to efficiently move down a slope, first with no obstructions, then with obstructions. In each case we found there are optimal algorithm parameter settings. Finally, in Section 4, we explored the role of local search.

Much remains to be researched. Questions for the future include:

- Is it really possible to capture key features of objective functions in a parameterised family?
- Is is possible to capture key features of a wide range of stochastic algorithms in a parameterised family?
- Are there universal rules linking the two?

This paper has scratched the surface of a challenging problem in the area of global optimisation.

Acknowledgement

The authors would like to thank the Marsden Fund of the Royal Society of New Zealand for support of this research.

Appendix A

This appendix provides a detailed description of the parametric family of objective functions introduced in Section 2.1.

Input parameter	Definition
L	LevelRatio: Ratio of escape height out of right high level basin to escape height out of left high level basin
B	BasinRatio: Ratio of width of right high level basin to width of left high level basin
O_r	OscRatio: Number of oscillations in the left high level basin
O_h	OscHeight: Amplitude of pulse oscillations
O_w	OscWidth: Width of pulse oscillations
S_r	SaddleRatio: Ratio of height of saddle to height of ridge
S_w	SaddleWidth: Width of saddle
S_d	SaddleDev: Relative deviation of saddle from centre line
F	FunctionType: 1 for a smooth function, 2 for piecewise linear oscillations, 3 for pulse oscillations

Derived constants	Definition
μ	$\frac{S_w^2}{4\log(100)}$
Z_f	$\frac{O_r}{2(B+1)}$
a	$1-\frac{1}{(1+L)}$
b	$\frac{1}{(1+L)}$
c	$\frac{1}{(1+B)}$

Derived functions	Definition
$x_1(x)$	$2\pi x O_r$
$x_2(x)$	$\frac{4x_1(x)}{2\pi}+1$
$x_3(x)$	$4\left(\frac{x_2(x)}{4}-\left\lfloor\left(\frac{x_2(x)}{4}\right)\right\rfloor\right)$
$x_4(x)$	$4\left(\frac{x_2(x)+3}{4}-\left\lfloor\left(\frac{x_2(x)+3}{4}\right)\right\rfloor\right)+1$
$S_p(x)$	$\frac{S_d}{2\cos(2\pi Z_f x)}+0.5$

The functional form is defined as follows:

$$f(x,y)=f_{\text{base}}(x)+o(x)g(x,y)+\frac{\left(y-\frac{1}{2}\right)^2}{10}$$

where

$$f_{\text{base}}(x)=\begin{cases}(1-a)\frac{1+\cos\left(\frac{2\pi x}{c}\right)}{2}+a & x\leqslant c\\ (1-b)\frac{1+\cos\left(\frac{2\pi(x-c)}{1-c}\right)}{2}+b & x>c\end{cases}$$

$$o(x)=\frac{O_h(j(F,x)+1)}{2}$$

$$g(x,y)=1-(1-S_r)\exp\left(\frac{-(y-S_p(x))^2}{\mu}\right)$$

$$j(F,x)=\begin{cases}\cos(x_1(x)) & F=1\\ k(x) & F=2\\ l(x) & F=3\end{cases}$$

$$k(x)=\begin{cases}x_3(x) & x_3(x)\leqslant 1\\ 2-x_3(x) & 1<x_3(x)\leqslant 3\\ x_3(x)-4 & 3<x_3(x)\leqslant 5\end{cases}$$

$$l(x)=\begin{cases}-1 & 1+2O_w<x_4(x)<5-2O_w\\ 1 & \text{otherwise}\end{cases}$$

References

1. Baritompa, W. and Steel, M. (1996), Bounds on absorption times of directionally biased random sequences. *Random Structures and Algorithms* 9, 279–293.
2. Boender, C.G.E. and Romeijn, H.E. (1994), Stochastic Methods, in: Horst, R. and Pardalos, P. (eds.), *Handbook of Global Optimization*, pp. 829–869. Kluwer Academic Publishers.
3. Hendrix, E.M.T., Ortigosa, P.M. and Garcia, I. (2001), On success rates for controlled random search, *Journal of Global Optimization* 21, 239–263.
4. Pardalos, P. and Romeijn, E. (2002), *Handbook of Global Optimization*, Vol. 2, Kluwer Academic Publishers.
5. Pronzato, L., Walter, E., Venot, A. and Lebruchec, J.F. (1984), A general purpose global optimizer: Implementation and applications. *Mathematics and Computers in Simulation* 24, 412–422.
6. Pullan, W.J. (1996), A direct search method applied to a molecular structure problem. *Australian Computer Journal* 28, 113–120.
7. Romeijn, H.E. and Smith, R.L. (1994), Simulated annealing for constrained global optimization. *Journal of Global Optimization* 5, 101–126.
8. Schoen, F. (1991), Stochastic techniques for global optimization: A survey of recent advances. *Journal of Global Optimization* 1, 207–228.
9. Smith, R.L. (1984), Efficient Monte Carlo procedures for generating points uniformly distributed over bounded regions. *Operations Research* 32, 1296–1308.
10. Törn, A., Ali, M.M. and Viitanen, S. (1999), Stochastic global optimization: Problem classes and solution techniques. *Journal of Global Optimization* 14, 437–447.
11. Törn, A. and Zilinskas, A. (1989), *Global Optimization.* Lecture Notes in Computer Science, Vol. 350. Springer.
12. Wales, D.J. and Doye, J.P.K. (1997), Global optimization by basin-hopping and the lowest energy structures of Lennard–Jones clusters containing up to 110 atoms. *Journal of Physical Chemistry A* 101, 5111–5116.
13. Wolpert, D.J. and Macready, W.G. (1997), No free lunch theorems for optimization, *IEEE Transactions on Evolutionary Computing* 1, 67–82.
14. Zabinsky, Z.B., Smith, R.L., McDonald, J.F., Romeijn, H.E. and Kaufman, D.E. (1993), Improving Hit and Run for global optimization, *Journal of Global Optimization* 3, 171–192.