Multidimensional Bisection: The Performance and the Context*

ZHANG BAOPING, G. R. WOOD, and W. P. BARITOMPA
Department of Mathematics, University of Canterbury, Christchurch, New Zealand.

(Received: 14 December 1990; accepted: 23 March 1992)

Abstract. Two aspects of the multidimensional bisection algorithms for the global optimisation of Lipschitz continuous functions are investigated. Firstly, for several test functions we examine the numerical performance of the deepest point algorithm and two acceleration procedures. Secondly, we phrase the branch and bound framework of Horst and Tuy in terms of covers, and show the algorithms to be included in this framework. A result of Basso on the convergence of localisations is extended to higher dimensions.

Key words. Bisection, global optimisation, branch and bound, numerical performance, simplex, localisation, deterministic, mathematical programming.

Mathematics Subject Classifications (1991). 90C30, 65K05.

1. Introduction

A generalisation of the familiar "interval-halving" bisection method to higher dimensions was introduced in [10]. The method can be viewed as a mechanism for finding all the global minima of a real-valued Lipschitz continuous function over a compact domain in R^n. Questions about the convergence, acceleration and optimality of the range of algorithms suggested were discussed in [9]. An extreme version generalises the Piyavskii–Shubert algorithm, [7, 8], to higher dimensions in a manner complementary to that of Mladineo, [3]. The algorithms can be coded using a particularly simple data structure, possess certain minimax properties, and can guarantee convergence to all global minima.

This paper explores two aspects of these multidimensional bisection (MB) algorithms: their numerical performance, and their relationship to branch and bound algorithms. In the interests of completeness we begin in §2 with an overview of the algorithms, at both an informal and a formal level. In §3 we address the specific question "How do the algorithms perform?" Numerical results are presented for three standard test functions, as well as for two functions drawn from a new non-differentiable family of test functions. Two acceleration methods are considered. Together they bring the algorithm closer to that of Mladineo, while retaining the simplicity of the simplex-based multidimensional

*The paper was presented at the II. IIASA-Workshop on Global Optimization, Sopron (Hungary), December 9–14, 1990.

Journal of Global Optimization **3**: 337–358, 1993.

bisection algorithm. In §4 we address the general question "How are the algorithms related to others in the literature?" The question is answered by showing that multidimensional bisection algorithms are included in a modification of the branch and bound framework of Horst and Tuy [6]. We conclude the paper in §5 with a result which simplifies and extends the results of Basso [2]: a strategy for choosing evaluation points is presented which ensures convergent localisations.

2. A Review of Multidimensional Bisection

Our problem is the following: given $f: R^n \to R$ and K a compact domain in R^n, find

$$\min f(x), \quad \text{for } x \in K$$

together with the points in K where this minimum is realised. We assume $f \in L(M)$, the set of Lipschitz continuous functions, with Lipschitz constant M.

Global optimisation algorithms which rest on the Lipschitz assumption use variations on two simple facts. Let

$$\mathcal{O} = \{(x, y): y \geq M\|x\|, \text{ for } x \in R^n \text{ and } y \in R\}$$

be the cone at the origin with axis of symmetry the y-axis and spherical cross-section of radius one at height M, and suppose that (x, y) lies on the graph of f over K. Then

(1) No point inside $(x, y) - \mathcal{O}$ lies on the graph of f, and
(2) No point above (x, y) can be a global minimum of f.

In the multidimensional bisection algorithm we approximate $\mathcal{O}$ by a cone ∇ with a simplex base, as shown in Figure 1. With $\mathcal{O}$ replaced by ∇, statements (1) and (2) remain true. Using ∇ in place of $\mathcal{O}$, an algorithm can be set up which proceeds in a very simple way. Furthermore, it can be viewed as a direct generalisation of the bisection method (see [10]). We firstly give a brief informal description of the MB algorithms, then follow it with a formal description, relegating the more technical details to Appendix 1.

2.1. AN INFORMAL REVIEW

At the end of each iteration, the algorithm brackets all global minima over K in a union of similar simplexes (known as *standard simplexes*) in R^{n+1}, each simplex being a translate of a cap of the cone ∇. Figure 2 shows such a bracket (or *system*) for a function of two variables. All simplex tops, shaded in the figure, lie at the height of the least evaluation to date.

How does this come about? At the outset, via $n + 1$ function evaluations, we are able to bracket all global minima over K in a standard simplex, T_0. Later

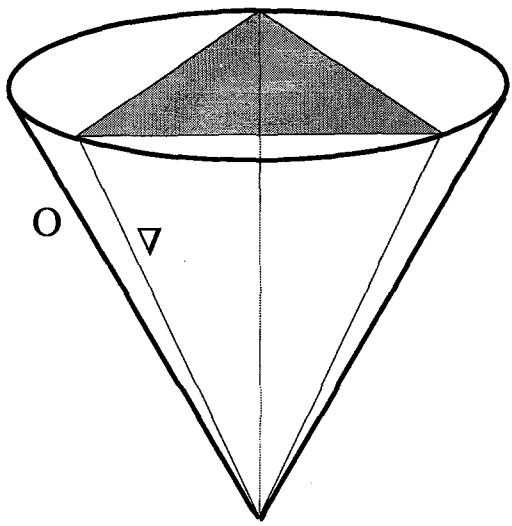

Fig. 1. The cone, $\mathcal{O}$, with spherical cross-section can be approximated by a cone, ∇, with simplicial cross-section.

iterations begin with a set of function evaluations. Each evaluation allows us to remove the interior of $-\nabla$, with apex moved to the evaluation point on the graph of f, from every simplex in the system. Here we are using property (1) above. When such an inverted cone is removed from a standard simplex it has the fortuitous effect of leaving at most $n+1$ standard simplexes, of smaller height than the original standard simplex. The idea is illustrated for $n=2$ in Figure 3. This process is termed *simplex reduction*. All such simplex reductions we term *system reduction*, $\mathcal{R}$. Following system reduction, the system is truncated at the height of the lowest evaluation to date, a process termed *system elimination*, $\mathcal{E}$. Thanks to property (2) above, this removes no global minima. We are then ready for the next iteration.

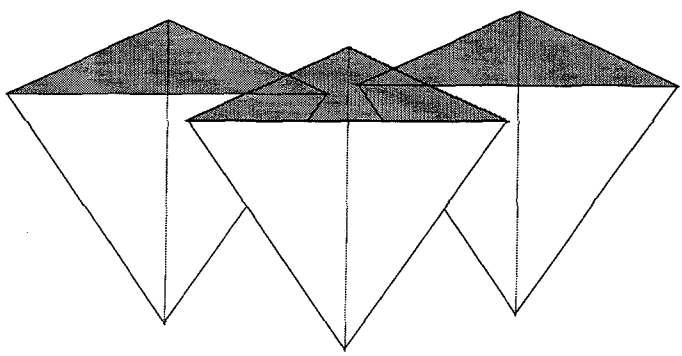

Fig. 2. Overlapping standard simplexes forming a bracket for the global minima, the situation at the end of each iteration.

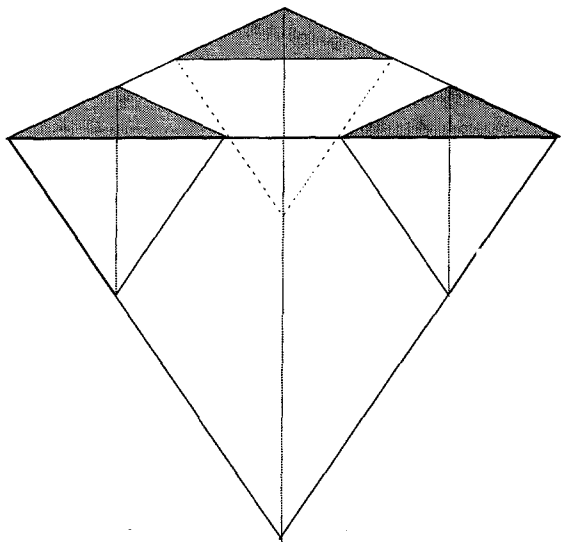

Fig. 3. Simplex reduction when $n = 2$: three small standard simplexes are left when the removal cone is withdrawn from the large standard simplex.

If we denote the system at the kth iteration by $\mathcal{S}_k$, then we can summarize the algorithm as:

ALGORITHM 2.1. (Multidimensional Bisection).
Initial step: Form $\mathcal{S}_0$, the initial system.
Iterative step: Let $\mathcal{S}_{k+1} = \mathcal{E}(\mathcal{R}(\mathcal{S}_k))$. Repeat until a stopping criterion is satisfied.

2.2. A FORMAL REVIEW

We proceed now to a formal description of Algorithm 2.1. Let $\{u_1, \ldots, u_{n+1}\}$ comprise the unit vectors from the origin to the vertices of some regular simplex, with centroid the origin, in R^n. Thus $u_1 + \cdots + u_{n+1} = 0$ and $u_k \cdot u_l = -1/n$ for all distinct pairs k and l. Let ∇ be the cone in R^{n+1} with apex the origin and cross-section $\text{co}\{u_1, \ldots, u_{n+1}\}$ at height M along the $(n+1)$st axis, where "co" denotes the convex hull. Formally,

$$\nabla = \text{pos}\{(u_k, M)\colon k = 1, \ldots, n+1\},$$

where "pos" denotes all positive linear combinations. The following concepts allow us to describe the algorithm.

DEFINITION 2.1. (Basic ideas).
1. A *standard simplex* in R^{n+1} is a translate of a cap of the cone ∇ so has the form

$$T(x, y, h) = \text{co}\left\{(x, y), \left(x + \frac{h}{M} u_k, y + h\right)\colon k = 1, \ldots, n+1\right\},$$

where $(x, y) \in R^{n+1}$ is the apex, and $h \in R$ the height. By the *top* of $T(x, y, h)$ we shall mean the facet of $T(x, y, h)$ opposite the apex.

2. A *system of simplexes* (or *system*) $\mathcal{S}$ in R^{n+1} is (*a*) a finite set $\mathcal{T}$ of standard simplexes, plus (*b*) a point a in R^{n+1}, lying in a lowest top of the system.

3. A *uniform system* is a system $\mathcal{S}$ in which all tops lie in the same hyperplane of R^{n+1}. Alternatively, $y_j + h_j = y_k + h_k$ for all T_j, T_k in $\mathcal{T}$.

4. The *variation* of a system $\mathcal{S}$, $V(\mathcal{S})$, is the difference between the highest and lowest points in the system. That is,

$$V(\mathcal{S}) = \max_j\{y_j + h_j\} - \min_j\{y_j\} .$$

5. A *standard domain* in R^n has the form $c + rU$, where $c \in R^n$, $r \geq 0$ and U is the vector sum $U_1 + \cdots + U_{n+1}$, where U_k is the line segment from 0 to u_k.

A standard domain is a line segment when $n = 1$, a hexagon when $n = 2$ and a rhombic dodecahedron, the honeycomb cell, when $n = 3$. For this reason we sometimes call MB algorithms "beesection" algorithms.

THE INITIAL SYSTEM

Given a standard domain, $H = c + rU$, we can set up an initial standard simplex, T_0, which brackets all global minima over H, as follows:

(i) evaluate f at the $n+1$ "dual" vertices of H, $\{v_k = c - ru_k : k = 1, \ldots, n+1\}$,
(ii) remove the cones $(v_k, f(v_k)) - \nabla$ from R^{n+1}, for each k, and
(iii) truncate R^{n+1} at the level of the lowest evaluation.

This process yields an initial system in a natural way. That the initial system contains all global minima over H is proven in ([10], Proposition 6.1). A detailed description of the initial system is given in Appendix 1.

SIMPLEX AND SYSTEM REDUCTION

Simplex reduction is the key to the algorithm. Given a standard simplex, $T = T(x, y, h)$, an evaluation of f at x allows us to remove the interior of $(x, f(x)) - \nabla$ from T and truncate T at height $f(x)$, leaving a region which is again a union of at most $n + 1$ smaller standard simplexes. If $(x, f(x))$ is on or above the top of T we term it an upper reduction, while if $(x, f(x))$ is below the top of T we term it a lower reduction. We use the notation $\mathcal{T}$ for the set of simplexes comprising the reduction of T. For a detailed description of $\mathcal{T}$ we refer the reader to Appendix 1.

In *system reduction* we reduce some of the simplexes in the current system. In the following definition, J indexes all simplexes in the system, and I indexes the simplexes to be reduced.

DEFINITION 2.2. (System reduction). Let $\mathcal{S} = (\mathcal{T}, a)$ be a uniform system inside the initial simplex T_0, where $\mathcal{T} = \{T_j\}_{j \in J}$, and let I be a non-empty subset of J. A reduction of $\mathcal{S}$ is a system $\mathcal{R}(\mathcal{S}) = (\mathcal{T}', a')$ where

$$\mathcal{T}' = \bigcup_{j \in I} \mathcal{T}_j \bigcup_{j \in J \setminus I} \{T_j\}, \text{ where } \mathcal{T}_j \text{ is the reduction of simplex } T_j,$$

$$a' = \begin{cases} a, \text{ if no lower reductions occur, else} \\ (x, f(x)) \text{ such that } f(x) = \min_{j \in I}\{f(x_j)\}. \end{cases}$$

In order to ensure that the algorithm converges, we must make the following assumption:

ASSUMPTION 2.1. *The global minimum of f, over the projection of the top of* T_0 *onto* R^n, *occurs in H.*

With this assumption ([10], Proposition 4.1) shows that no global minima are removed during a system reduction. In [10] we analysed the case where $I = J$. Of particular interest in this paper is the case where I picks out the deepest simplex in the system at each stage, so $|I| = 1$. In §5 of this paper, stimulated by a result of Basso, an algorithm for which $1 \leq |I| \leq |J|$ is analysed.

SYSTEM ELIMINATION

Following a system reduction, parts of some simplexes may lie above the lowest function value recorded. We tighten up the system in the following way:

DEFINITION 2.3. (System elimination). The eliminated system associated with the system $\mathcal{S} = (\mathcal{T}, a)$ inside T_0 is $\mathcal{E}(\mathcal{S}) = (\mathcal{T}', a)$ where

$$\mathcal{T}' = \{T \cap P^- : T \in \mathcal{T}\},$$

with P^- the closed half-space of R^{n+1} below a.

Again, no global minima are removed during this process ([10], Proposition 4.2). This completes our formal description of the three critical aspects of Algorithm 2.1, namely the construction of the initial simplex, and the processes of system reduction and system elimination. We refer to the reduction of the I simplexes, together with elimination, as a full iteration. Further illustrations showing the ideas underlying the algorithm may be found in [9] and [10].

3. Performance

The character of the algorithm depends upon the choice of simplexes reduced in an iteration. In [10] all simplexes were reduced, so ensuring that the variation of the system was reduced by at least a factor of $n/(n+1)$ at each full iteration. In

[9] it was shown that the reduction of the deepest simplex at each iteration is sufficient to ensure that the variation converges to zero. This choice creates the multidimensional bisection analogue of the strategies employed by Piyavskii–Shubert and Mladineo.

No matter what reduction strategy we employ, there are two immediate failings of the algorithm:

(i) For n greater than one, an evaluation over one simplex frequently generates a removal cone which is capable of removing material from neighbouring simplexes. The algorithm, as described so far, does not effect this action, which we term complete reduction.

(ii) In reality, we can remove a spherically based cone at an evaluation point. Our simplex based cone merely approximates this cone, and the approximation worsens as n increases. A method is needed which retains the simplicity of simplexes, yet utilises the power of the spherical removal.

An implementation of the deepest point algorithm has been written in `matlab` which runs the raw "deepest point" algorithm (A), and either or both of two acceleration schemes which remedy the two points just mentioned.

In order to understand the idea behind the *complete reduction* algorithm (A^c), it is necessary to recognize that the notion of simplex reduction introduced in the previous section can be considerably generalised. So far we have only reduced a simplex when the evaluation occurs over its apex (or deepest point). This restriction is not necessary. Let z be any point in R^{n+1}. Given a standard simplex T, $z - \nabla$ can be used as a removal cone resulting in $T \backslash (z - \nabla)$ being a union of at most $n + 1$ standard simplexes. Figure 4 illustrates this statement. The `matlab` implementation of the algorithm is a "dual" implementation, in which each simplex is held by means of the dual coordinates of the sloping facets of the simplex. The dual coordinate of a facet is the inner product of any vector from the origin to a point on the facet with a unit vector orthogonal to the facet. This representation of the simplex allows us to cope with the technical problems of complete reduction in a straightforward fashion [1].

The *spherical reduction* algorithm (A^s) was described in detail in [9]. We provide a brief review now. The central idea is illustrated for the case where $n = 2$ in Figure 5. Pictured in the figure is the triangular top T of $T(x, y, h)$ and the cross-section D' of the removal cone $(x, f(x)) - \nabla$ through the plane of the simplex top. We are really permitted to remove $(x, f(x)) - \mathcal{C}$, a much larger volume, whose cross-section is shown as S in the diagram. It is now clear that we can remove a simplex based cone at an effective evaluation point higher than $(x, f(x))$. Its cross-section through the plane of T is shown as D, evidently the largest simplex dually oriented to the top of $T(x, y, h)$ whose intersection with T is contained in S. The technical details of spherical acceleration are presented in Appendix 1.

Spherical acceleration utilises the power of the spherically based removal cone,

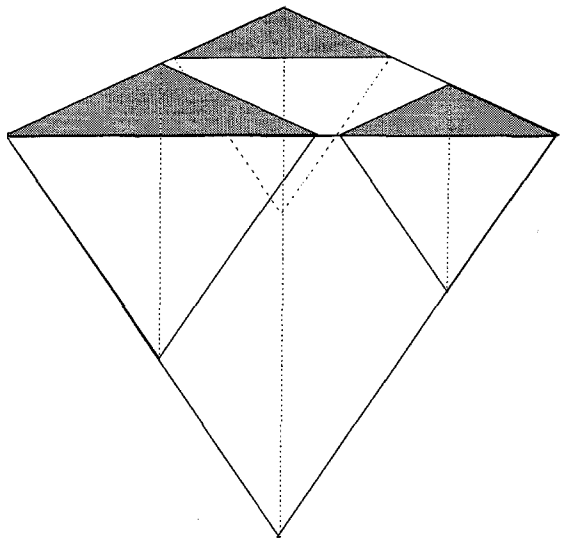

Fig. 4. Simplex reduction when the evaluation is not over the apex. Note that standard simplexes of unequal size are left after such a reduction.

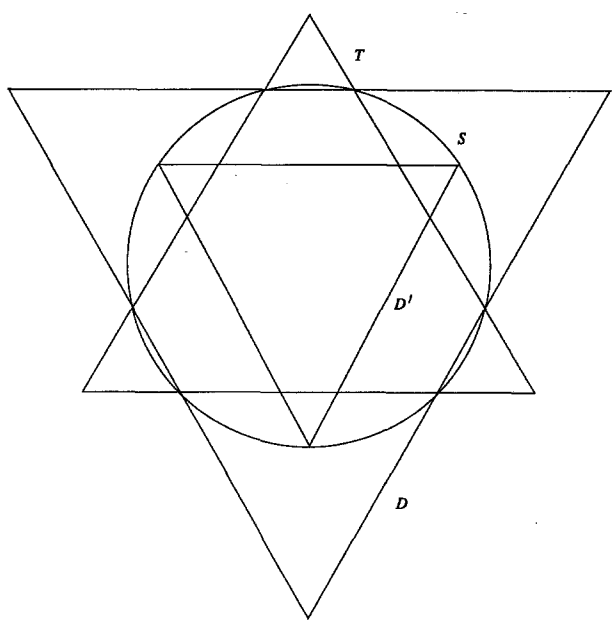

Fig. 5. The basis of spherical reduction: if T were the top of a simplex in the system, and S the cross-section of the spherical removal cone, then we could effectively remove a simplex based cone with cross-section D from the simplex with top T.

but only for removal from the simplex over which the evaluation was made. This acceleration can be combined with complete reduction to extend to all simplexes which intersect with the spherically based removal cone. We term this *complete spherical reduction* (A^{cs}), a technique we now describe. An evaluation over a point x determines a fixed spherically based removal cone. Consider a simplex in the system which meets this removal cone. Denote the top of this simplex, shaded in Figure 6, by T. Construct a dummy standard simplex, with top T', the smallest centered on x and containing T. Spherical reduction on this dummy simplex would generate a simplex based removal cone at an effective evaluation point higher than $(x, f(x))$. Its cross-section at the level of the tops is shown as D in Figure 6. Now remove this cone from the original simplex, as in the complete reduction procedure. This combined process allows us to remove more than would spherical or complete reduction alone.

We report now on the relative performance of the four schemes, using three test functions which are already in the literature, and two members of a family of test functions suggested by recent work of Mladineo [4]. All are described in the appendix. The Mladineo functions have the appearance of upside-down mountain ranges, being non-differentiable at the global minimum. Different runs were created by varying the location of the first evaluation following formation of the initial simplex. Thereafter the deepest point algorithm was used. Each run is terminated after 100 function evaluations. For each function we report the average, over a number of runs, of (i) the location of the least evaluation, and (ii) the ratio of the final variation to the initial variation.

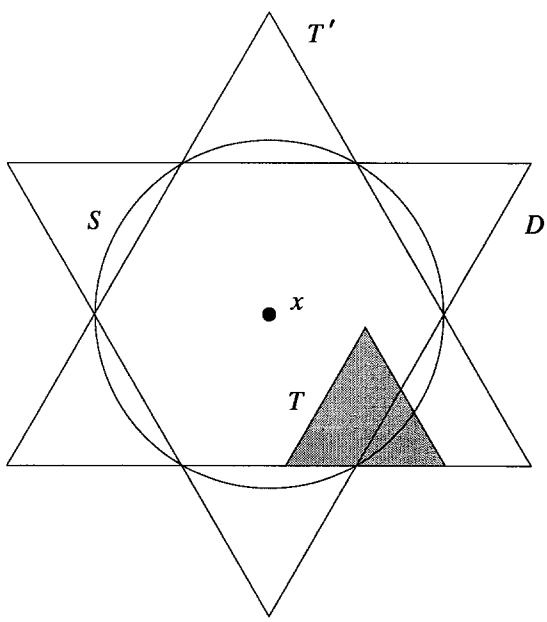

Fig. 6. The basis of complete spherical reduction.

Table I. For each test function and algorithm type the table shows the true minimum, and average over several runs of the computed minimum and relative variation (the ratio of the final variation to the initial variation). Each run was terminated after 100 function evaluations. The differing number of runs within RCOS is due to the presence of more than one global minimum; we selected only the runs which converged to the specified global minimum

True minimum		Algorithm minimum	Rel. varn	No. runs
GOLDPR				
	A	(0.5014, 0.2565, 0.0000)	0.1788	10
(0.5000, 0.2500, 0.0000)	A^c	(0.4681, 0.2681, 0.0000)	0.1475	10
	A^s	(0.5014, 0.2565, 0.0000)	0.1788	10
	A^{cs}	(0.4687, 0.2681, 0.0000)	0.1272	10
RCOS				
	A	(0.5436, 0.1618, 0.0035)	0.1690	7
(0.5428, 0.1517, 0.0013)	A^c	(0.5281, 0.1553, 0.0032)	0.1240	4
	A^s	(0.5436, 0.1618, 0.0035)	0.1731	7
	A^{cs}	(0.5281, 0.1553, 0.0032)	0.1242	4
	A	(0.1308, 0.7840, 0.0027)	0.1695	3
(0.1239, 0.8183, 0.0013)	A^c	(0.1361, 0.7558, 0.0029)	0.1259	5
	A^s	(0.1308, 0.7831, 0.0027)	0.1695	4
	A^{cs}	(0.1361, 0.7558, 0.0029)	0.1242	5
	A	(0.9423, 0.1446, 0.0026)	0.1644	1
(0.9617, 0.1650, 0.0013)	A^c	(0.9665, 0.1805, 0.0035)	0.1241	2
	A^s	(0.9423, 0.1446, 0.0026)	0.1644	1
	A^{cs}	(0.9665, 0.1804, 0.0035)	0.1231	2
FUNCT2				
	A	(0.1249, 0.9629, −2.9916)	0.0855	10
(0.1427, 0.9757, −3.0000)	A^c	(0.1086, 0.9737, −2.9909)	0.0654	8
	A^s	(0.1285, 0.9745, −2.9927)	0.0790	10
	A^{cs}	(0.1286, 0.9699, −2.9940)	0.0588	8
MLADINEO(2, 3)				
	A	(0.0002, 0.8010, −1.7284)	0.0068	10
(0.0000, 0.8000, −1.7321)	A^c	(0.0000, 0.8001, −1.7317)	0.0004	10
	A^s	(0.0000, 0.8005, −1.7296)	0.0025	10
	A^{cs}	(0.0000, 0.8000, −1.7320)	0.0000	10
MLADINEO(4, 3)				
	A	(0.01, −0.01, 0.00, 0.73, −1.58)	0.3755	13
(0, 0, 0, 0, 0.8, −1.7321)	A^c	(0.01, −0.01, 0.00, 0.75, −1.60)	0.2997	13
	A^s	(0.01, −0.01, 0.00, 0.73, −1.58)	0.3680	13
	A^{cs}	(0.01, −0.01, 0.00, 0.75, −1.60)	0.2696	13

Note that the deviation from the true global minimum is available by comparing the final component of the located point with the final component of the true point in Table I.

REMARKS

(i) The algorithms work best for functions where the minima are well-defined, such as FUNCT2 and Mladineo(2, 3) and (4, 3). The effectiveness of simplex reduction in such cases causes the system variation to reduce rapidly.

(ii) For functions which are relatively flat over much of a neighbourhood around the global minimum the variation is slow to reduce. Thus while we may find a good solution early, it takes a lot of later evaluations to confirm that it is successful. GOLDPR and RCOS exhibit this behaviour.

(iii) The algorithms are converging to a global minimum, but it is evident by examining absolute error in Table I that they are more successful at finding the value of the function at the global minimum (the final coordinate) than the location of the global minimum (the first n coordinates).

(iv) Complete reduction produces roughly a 25% reduction in variation, though substantially more when the global minimum is sharply defined, as in Mladineo(2, 3). Spherical reduction makes almost no difference for functions which are flat around the global minimum, but does offer an improvement for FUNCT2 and Mladineo(4, 3), and a marked one for Mladineo(2, 3). Recall that spherical reduction comes into its own only when evaluations lie well above the simplex top. Note that A^{cs} is best overall.

(v) Accelerated methods require fewer function evaluations to reach a given variation, but the overheads per function evaluation are higher. For our current implementation the overall overheads to reach a given accuracy, measured in floating point operations, do not change very much from A to A^{cs}. No effort has been made so far, however, to use a streamlined data structure. An efficient implementation in C is planned. This will reveal the extent to which acceleration methods can reduce the overheads (as well as the function evaluations) involved in reaching a given accuracy.

(vi) The change from one run to the next in the located algorithm minimum and in the relative variation is illustrated by the following representative selection of means and standard deviations:

GOLDPR

A $(0.5014 \pm 0.0480,\ 0.2565 \pm 0.0223,\ 0.0000 \pm 0.0000);\ 0.1788 \pm 0.0120$

MLADINEO(2, 3)

A $(0.0002 \pm 0.0015,\ 0.8010 \pm 0.0024,\ -1.7284 \pm 0.0032);\ 0.0068 \pm 0.0032$

4. Context

In their text [6], Horst and Tuy present a general framework for "branch and bound" global optimisation algorithms. In an earlier paper [5], these authors showed that the algorithms of Pinter, and Zheng and Galperin are encompassed by this general framework. Somewhat surprisingly, the algorithm of Mladineo is also shown to sit beneath this umbrella, but the method used is not completely natural.

Our aim in this section is to slightly broaden the framework so that it more readily encompasses algorithms such as those discussed in this paper and their generalisation described in [1]. Motivation to alter the branch and bound framework springs from the observation that in the Piyavskii–Shubert algorithm, and

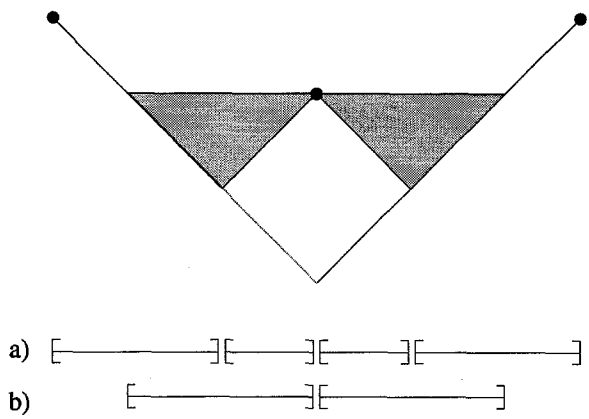

Fig. 7. After three evaluations (shown as heavy dots) the partition determined by the removal cones (used in [5]) is shown in (a), and the cover using simplex tops in (b).

multidimensional bisection, the natural cover at any stage is the projection onto the domain of the simplicial tops of the possibly overlapping simplex brackets. The idea is illustrated in Figure 7, for the case where $n = 1$. We acknowledge that every finite partition is a finite cover, and moreover that every finite cover gives rise to a (not necessarily unique) finite partition. The language of covers, however, allows us to express multidimensional bisection as a branch and bound algorithm more conveniently than the language of partitions.

4.1. A NEW BRANCH AND BOUND FRAMEWORK

We now present the branch and bound framework of Horst and Tuy in the language of covers, rather than partitions. We follow the format in ([6], pp. 114–116). A set C in R^n is termed *feasible* if $C \cap K \neq \emptyset$, and *uncertain* if it is not known whether it is feasible. We adopt the convention that the minimum taken over an empty subset of R equals $+\infty$.

Initial step ($k = 0$): The algorithm begins with a compact set C_0 covering K, or a subset of K where $\min f(K)$ is realised. Set $\mathscr{C}_0 = \{C_0\}$, the initial cover. Associated with C_0 are bounds $\beta_0 = \beta(C_0)$ and $\alpha_0 = \alpha(C_0)$ such that

$$\beta_0 \leq \min f(K) \leq \alpha_0 = \min f(S_{C_0})$$

where S_{C_0} is the possibly empty set of evaluation points made in K. If $\alpha_0 < \infty$, then choose x^0 such that $f(x^0) = \alpha_0$. If $\alpha_0 - \beta_0 \leq \epsilon$, then stop, else proceed to the iterative step.

Iterative step ($k = 1, 2, \ldots$): At the outset we have the finite cover of closed sets, $\mathscr{C}_{k-1}$, of the subset of C_0 still of interest. For every $C \in \mathscr{C}_{k-1}$ we have bounds

$\beta(C)$ and $\alpha(C)$ satisfying

$$\begin{array}{ll} \beta(C) \leqslant \min f(C \cap K) \leqslant \alpha(C) = \min f(S_C) & \text{if } C \text{ is known to be feasible,} \\ \text{and } \beta(C) \leqslant \min f(C) & \text{if } C \text{ is uncertain} \end{array}$$

where S_C is the possibly empty set of evaluation points in $C \cap K$, and the overall lower and upper bounds are defined as

$$\beta_{k-1} = \min\{\beta(C): C \in \mathscr{C}_{k-1}\}, \quad \alpha_{k-1} = \min\{\alpha(C): C \in \mathscr{C}_{k-1}\}$$

satisfying $\beta_{k-1} \leqslant \min f(K) \leqslant \alpha_{k-1}$. We now describe the four-stage "branch and banish, bound and banish" iterative step, based on the presentation in [6].

1. *Branch*: Select a subset $\mathscr{P}_k$ of $\mathscr{C}_{k-1}$ and finitely recover each member of $\mathscr{P}_k$. Let $\mathscr{P}'_k$ be the set of all newly formed cover sets.
2. *Banish*: Delete $C \in \mathscr{P}'_k$ if it lies outside K, or if it is known that $\min f(K)$ cannot occur over C. Let $\mathscr{C}'_k$ be the collection of cover sets remaining.
3. *Bound*: Assign to each $C \in \mathscr{C}'_k$ which is known to be feasible bounds $\beta(C)$ and $\alpha(C)$ satisfying

   $$\beta(C) \leqslant \min f(C \cap K) \leqslant \alpha(C) = \min f(S_C)$$

   and to each uncertain $C \in \mathscr{C}'_k$ a bound $\beta(C)$ satisfying $\beta(C) \leqslant \min f(C)$. Here S_C is the set of evaluation points in $C \cap K$. We assume that $\beta(C) \geqslant \beta(B)$ if $C \subseteq B \in \mathscr{C}_{k-1}$.
   Let $\beta_k = \min\{\beta(C): C \in \mathscr{C}'_k\}$ and $\alpha_k = \min\{\alpha(C): C \in \mathscr{C}'_k\}$, the overall bounds, and if $\alpha_k < \infty$ let x^k in K be such that $f(x^k) = \alpha_k$.
4. *Banish*: Delete all $C \in \mathscr{C}'_k$ not containing x^k which are fathomed, that is, $\alpha_k \leqslant \beta(C)$. Let $\mathscr{C}_k$ be the collection of cover sets remaining.
   If $\alpha_k - \beta_k \leqslant \epsilon$, then stop, else re-run the iterative step.

At this stage, $\beta_k \leqslant \min f(K) \leqslant \alpha_k$.

REMARKS. (i) The cycling of the steps, placing "fathoming" last instead of first, does not alter the algorithm, and suits our purpose in the next section. It has, however, obliged us to add the phrase "not containing x^k" in Step 4 in order to exclude the possibility that all C's are fathomed.

(ii) We have assumed compactness throughout to ensure that the minima exist. This can be generalised to non-compact C_0 and infima, as in [6].

4.2. MULTIDIMENSIONAL BISECTION WITHIN THE BRANCH AND BOUND FRAMEWORK

We now show that the multidimensional bisection algorithm, Algorithm 2.1, falls into this new branch and bound framework.

The natural domains K for MB algorithms are the standard domains described

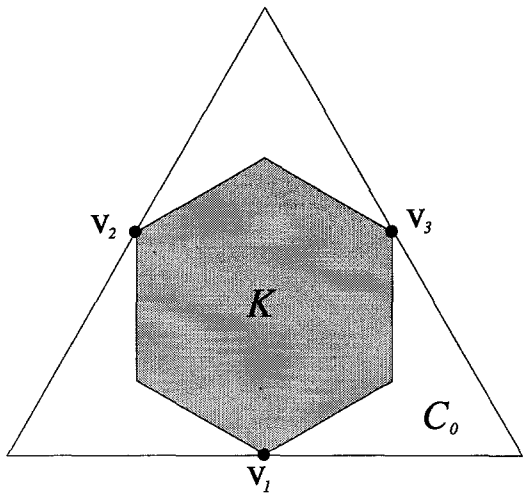

Fig. 8. The initial feasible set K, contained in the relaxed feasible set C_0. Shown here is the most conservative case, where all initial evaluations are equal. Variation in the function evaluations causes shrinkage of the initial cover set, C_0.

in Definition 2.1(5). The algorithm begins by placing the minimum over such a K in a simplex C_0, the top of the initial simplex bracket, projected onto the domain, R^n. This simplex contains the region of K still of interest. Figure 8 shows the situation for $n = 2$. Evaluations of f at the dual vertices $S_{C_0} = \{v_1, \ldots, v_{n+1}\}$ of K (see [9], §2.2) provide a simplex in R^{n+1} with top at height $\alpha_0 = \min\{f(v_i): i = 1, \ldots, n+1\}$, and base at height $\beta_0 \leq \min f(K)$. Here α_0 is always finite, so we immediately have an iteration point $x_0 \in K$ for which $f(x^0) = \alpha_0$. It is one of the vertices v_i for which the evaluation is least.

At the start of the kth iteration a global minimum over K is bracketed in a finite set of similar simplexes. The projection of these simplex tops onto the domain forms the cover $\mathscr{C}_{k-1}$. For the simplex in the system with projected top C, $\beta(C)$ equal to the level of the simplex base, and $\alpha(C) = \min f(S_C)$ (where S_C is the set of evaluation points in $C \cap K$) form lower and upper bounds for $\min f(C \cap K)$. A property of the algorithm is that for all feasible $C \in \mathscr{C}_{k-1}$ we have $\beta(C) \leq \min f(C \cap K) \leq \alpha(C)$. The first inequality follows since over K the graph of f remains on or above the sloping facets of the simplexes in the system.

We now describe how the deepest point MB iteration slots into the four stages of the branch and bound iterative step.

1. *Branch*: Select the $C \in \mathscr{C}_{k-1}$ corresponding to the simplex which is to be reduced, and any other cover sets influenced by elimination in this iteration. These sets constitute $\mathscr{P}_k$. The deepest point evaluation then yields a cover of C which will be one of three types, according as the evaluation is above, on or below the simplex top. Figure 9(a)–(c) shows these cases when $n = 2$. Cover

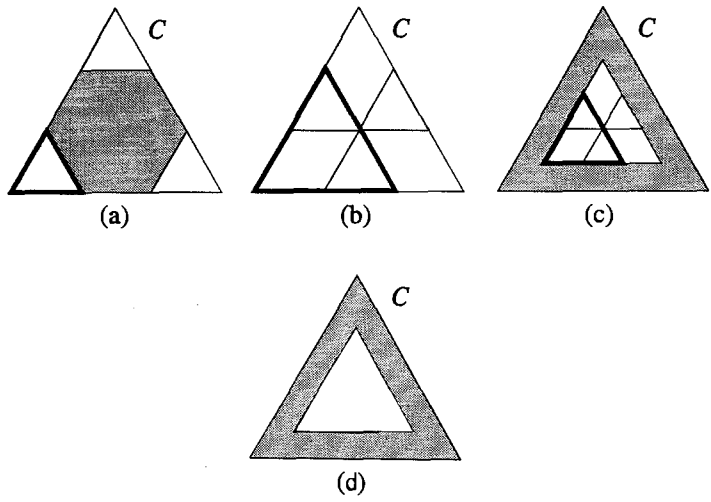

Fig. 9. The refinements of the cover set C when the evaluation over the base is (a) above the top, (b) on the top, and (c) below the top. In cases (b) and (c) the refinement is a cover rather than a partition. Case (d) shows a cover set C, associated with the elimination phase, recovered using two sets.

sets influenced by elimination will have their associated $n+1$ dimensional simplexes truncated, so give rise to the recovering shown in Figure 9(d). This comprises two sets, shown shaded and unshaded.

2. *Banish*: The shaded regions in Figure 9, in (a) and (c), or their analogues for larger n, can now be deleted. This is possible since it is known that min $f(K)$ cannot occur over these regions (see §2.2, System reduction). This leaves us with $\mathscr{C}_k'$. The geometry of multidimensional bisection would allow us to remove cover sets C in $\mathscr{C}_0 \backslash K$. In practice we do not expend the effort, since Assumption 2.1 ensures that such sets are eventually fathomed in Step 4.
3. *Bound*: We assign to each new simplex C in the cover the level of the base, $\beta(C)$. In MB this choice of $\beta(C)$ ensures that $\beta(C) \leq \min f(C \cap K)$ for each feasible set C, whether or not we know it to be feasible. That this inequality may not hold for infeasible C is not a concern, since infeasible cover sets for which it does not hold will be more quickly fathomed in Step 4, a desirable outcome. With the annular shaded region in (d) we associate a β value of α_k, the level of the lowest evaluation to date. We know that min $f(K)$ cannot occur on such a C, so this ensures that it is fathomed in Step 4. To each cover set we can assign $\alpha(C) = \min f(S_C)$, where S_C is the set of evaluation points in $C \cap K$. Certainly it follows that min $f(C \cap K) \leq \alpha(C)$. The evaluation point x^k in K is chosen such that $f(x^k) = \alpha_k$.

These three steps together correspond to the reduction step, $\mathscr{R}$. They are viewed simultaneously in MB, but can be linearly ordered in the way just described.

4. *Banish*: This fourth step corresponds to the elimination step, $\mathscr{E}$. All shaded regions of the type shown in Figure 9(d) are removed.

This completes the demonstration that Algorithm 2.1 with the deepest point strategy is an example of the new framework. A fine point should be acknowledged here: any (necessarily singleton) $C \neq \{x^k\}$ for which $\beta(C) = \alpha_k$ would be eliminated in Step 4. Thus we are capturing a very slight modification of MB. The algorithm just described, however, still is such that $f(x^k) \downarrow \alpha$.

Incorporation of spherical reduction would alter only the refinement of $\mathscr{P}_k$. Any complete reduction, or a reduction strategy which involved more than a single evaluation (such as that in the next section) would necessitate overlaying the covers generated by reduction and elimination. These algorithms still follow the pattern of the branch and bound format.

A final remark: we have shown in this section that MB can be expressed in branch and bound language. This language distinguishes the domain R^n and the range R. Our intuition is that a more natural framework for MB is waiting to be phrased in R^{n+1}.

5. Convergence

We turn now to the convergence of the multidimensional bisection algorithms. We review the convergence of the branch and bound algorithm presented in §4.1, and show how this relates to the convergence of the MB algorithm with the deepest point reduction strategy. We then investigate a stronger reduction strategy, initially proposed for univariate functions by Basso. This strategy ensures that the localisation converges to the set of global minimisers.

For the α_k and β_k of §4.1, evidently α_k is non-increasing and β_k non-decreasing. Thus $\lim \alpha_k = \alpha$ and $\lim \beta_k = \beta$ necessarily exist, and $\beta \leq \min f(K) \leq \alpha$. Following Horst and Tuy we say that an infinite procedure (one for which $\alpha_k \neq \beta_k$ for all k) converges if $\alpha_k - \beta_k \to 0$, as $k \to \infty$, whence

$$\alpha = \lim_k f(x^k) = \beta = \min f(K) .$$

We now restate in appropriate form the convergence conditions for an infinite branch and bound procedure, given in their original form in ([6], pp. 123–125). These will ensure that $\alpha_k - \beta_k \to 0$.

DEFINITIONS. 1. A bounding procedure is termed *consistent* if at the start of each iteration every non-degenerate cover set can be refined, and any decreasing sequence C_{k_q} coming from successively refined covers satisfies

$$\lim_q (\alpha_{k_q} - \beta(C_{k_q})) = 0 .$$

2. A selection procedure is termed *complete* if for every $C \in \bigcup_{p=1}^{\infty} \bigcap_{k=p}^{\infty} \mathscr{C}_k$ we have $\min f(C \cap K) \geq \alpha$.

3. A selection procedure is termed *bound improving* if, for each k, there exists an $l \geq k$ for which

$$\operatorname{argmin}\{\beta(C): C \in \mathscr{C}_k\} \cap \mathscr{P}_l \neq \emptyset .$$

That is, at each iteration one covering element where β_k occurred is later selected for refinement.

The following theorem, which we restate from ([6], pp. 124–127) then follows, with the proofs as in [6].

THEOREM 5.1. *In an infinite branch and bound procedure, suppose that the bounding operation is consistent. It follows that*

(*i*) *if the selection is complete, then*

(*a*) $\alpha = \min f(K)$,

(*b*) *if f is continuous, then every accumulation point x of $\{x^k\}$ is such that* $f(x) = \min f(K)$.

(*ii*) *if the selection is bound improving, then the procedure is convergent, so*

$$\alpha = \min f(K) = \beta .$$

We now use this theorem to discuss the convergence of MB with deepest point reduction. Condition (C1) discussed in [9] required that all deepest simplexes in the system at the end of each MB iteration be eventually reduced. This is readily shown to be equivalent to the bound improving condition. For MB, a selection procedure which is bound improving also is such that the bounding procedure is consistent. This is shown in ([9], Theorem 4.1(i)). The key to this is the variation reducing result ([9], Lemma 4.1). When the bounding in the algorithm is consistent, the selection is complete ([6], p. 127). In [9] the location of the least evaluation to date is chosen as the iteration point, x^k. Assumption 2.1 ensures that eventually all such points are feasible, so that the accumulation points of this sequence coincide with those of the sequence of least feasible evaluation points, used in §4.1. It follows from Theorem 5.1 that MB, with the deepest point strategy, is "range" convergent, or $f(x^k) \to \min f(K)$.

It was pointed out in [2], however, that the deepest point reduction strategy does not ensure "domain" convergence of the algorithm. We clarify this statement using the following notation. Denote by

(i) A, the accumulation points of the iteration points,

(ii) E, the points in K where $\min f(K)$ is realised,

(iii) L_∞, the set $\cap_{k=0}^{\infty} L_k$, where $L_k = \cup\{C: C \in \mathscr{C}_k\}$ is the "localisation" at the kth iteration.

For a consistent and bound improving MB algorithm we have $A \subseteq E \subseteq L_\infty$. The first inclusion follows from Theorem 5.1 or ([9], Theorem 4.1(iii)) and the second

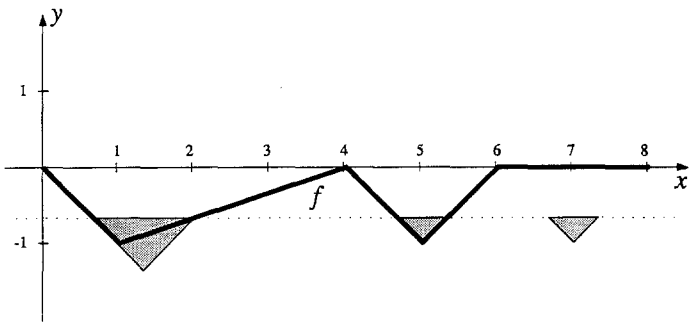

Fig. 10. For the f shown, the solution set $E = \{1, 5\}$. The Piyavskii–Shubert algorithm has accumulation points $A = \{1\}$ and final localisation $L_\infty = \{1, 5, 7\}$, showing that in general equality is not the case for the inclusions $A \subseteq E \subseteq L_\infty$. A typical system is shaded in the figure.

from ([9], Proposition 4.1). We conclude the paper by finding conditions under which we have domain convergence, that is, $A = E = L_\infty$.

The function pictured in Figure 10, adapted from Basso, illustrates that for the deepest point algorithm we cannot guarantee equality in the inclusions, $A \subseteq E \subseteq L_\infty$. A typical system is sketched over the function. Iteration points stay in the neighbourhood of $x = 1$, with the points $(5, -1)$ and $(7, -1)$ remaining in all systems. Note that $5 \in E$ but is not an accumulation point of the iteration points, and $7 \in L_\infty$ but is not in E. Thus both inclusions are proper.

The central result in ([2], Theorem 2 and Corollary 2) shows that if the deepest point algorithm is modified, and in Basso's terminology, a block-sequential reduction strategy is used, then $A = E = L_\infty$. We now extend Basso's work to show that the MB algorithm, in all four forms described in §3, and equipped with block-sequential reduction, guarantees that the localisations converge to the solution set E. In block-sequential reduction the deepest point in each connected component of the localisation is evaluated in each iteration. We prove the following:

THEOREM 5.2. *For the multidimensional bisection algorithm, with block-sequential reduction and any combination of complete and spherical reduction, then* $A = E = L_\infty$.

Proof. Block-sequential reduction is certainly bound-improving, so it follows as before that $A \subseteq E \subseteq L_\infty$. Hence it suffices to take $x \in L_\infty$ and show that $x \in A$.

From Theorem 5.1 we know that $\alpha_k \downarrow \alpha$ and $\beta_k \uparrow \alpha$, where $\alpha = \min f(K)$. Thus the only point over x eventually remaining in the bracket is (x, α). We consider two cases.

Case 1: (x, α) is eventually only in simplexes with apex at the point (x, α). Since $\alpha_k \downarrow \alpha$ as $k \to \infty$, eventually (x, α) lies in an isolated system simplex, of variation as small as we please. If $f(x) > \alpha$, an evaluation at x, ensured by the assumption of block-sequential reduction, would remove (x, α) from the system. Since $x \in L_\infty$ this provides a contradiction so $f(x) = \alpha$, and $x \in E$. It is also evident

that eventually the system must include the degenerate simplex $\{(x, \alpha)\}$. Block sequential reduction ensures that this isolated simplex will be evaluated in every later full iteration. Hence $x \in A$.

Case 2: (x, α) is always in some simplex with apex at height less than α. Since β_k increases to α, (x, α) must lie in a strictly nested sequence of simplexes, $T_k = T(x_k, y_k, h_k)$, as the algorithm progresses. Note that h_k must decrease to zero.

For incomplete reduction, with or without spherical reduction, any raising of the apex of T_k must occur through an evaluation at x_k. Thus $x \in A$.

For complete reduction, with or without spherical reduction, the apex of T_k can be raised through an evaluation at a point other than x_k. We now show, however, that given a neighbourhood U of x, and for T_k's with projected top inside U, there can be at most finitely many evaluations outside U which raise the level of these simplexes. Thus $x \in A$.

Suppose that $\{z_l\}$ is a sequence of evaluation points outside U, each of which raises the level of one of these simplexes. Then there exists an $\epsilon > 0$ such that $f(z_l)$ exceeds $\alpha + \epsilon$ for each l. Since $C_0 \backslash U$ is compact, the sequence of evaluation points $\{z_l\}$ has an accumulation point, z. Since f is continuous, $f(z) > \alpha$. But $z \in A \subseteq E$, so $f(z) = \alpha$, a contradiction. This completes the proof.

REMARKS. (i) In trials of the algorithms using deepest point reduction, we have noted the occurrence of points of the type occurring at $x = 7$ in the example of Figure 10. With block-sequential reduction such a point would disappear in an early iteration. The result of Theorem 5.2 suggests that every so often a block-sequential iteration should be run to remove such stray points.

(ii) The proof of Theorem 5.2 reveals that the behaviour exhibited in Figure 10 at $x = 1$ and $x = 5$ illustrates the only two ways in which a point can remain forever in the system.

Appendix 1: Algorithm Details

The initial system is described in the following definition.

DEFINITION A.1. For $H = c + rU$, a standard domain in R^n, and function f in $L(M)$, the initial system $S_0 = (\mathcal{T}_0, a_0)$ with $\mathcal{T}_0 = \{T_0 = T(x_0, y_0, h_0)\}$ is given by

$$x_0 = c + \frac{1}{M(n+1)} \sum_{k=1}^{n+1} (f(v_k) - m) u_k ,$$

$$y_0 = \frac{1}{n+1} \sum_{k=1}^{n+1} f(v_k) - Mnr ,$$

$$h_0 = Mnr - \frac{1}{n+1} \sum_{k=1}^{n+1} (f(v_k) - m) ,$$

$$a_0 = (c - ru_l, m) ,$$

with $v_k = c - ru_k$, $k = 1, \ldots, n+1$, the dual vertices of H, $m = \min_k\{f(v_k)\}$, $l =$ an index associated with the lowest evaluation, that is, $f(v_l) = m$.

The details of simplex reduction are given in the next definition.

DEFINITION A.2. Let $T(x, y, h)$ be a standard simplex. Two cases occur:
1. *Upper reduction* (when $h \leq f(x) - y$). If $h \leq f(x) - y \leq (n+1)h$, then the reduction of T is

$$\mathcal{T} = \left\{ T\left[x + \frac{f(x)-y}{M(n+1)}\, u_k,\ y + \frac{f(x)-y}{n+1},\ h - \frac{1}{n+1}\,(f(x)-y) \right] : \right.$$
$$\left. k = 1, \ldots, n+1 \right\},$$

else if $(n+1)h < f(x) - y$, then $\mathcal{T}$ is the empty set.

2. *Lower reduction* (when $f(x) - y < h$). If $0 \leq f(x) - y < h$, then the reduction of T is

$$\mathcal{T} = \left\{ T\left[x + \frac{f(x)-y}{M(n+1)}\, u_k,\ f(x) - \frac{n}{n+1}\,(f(x)-y),\ \frac{n}{n+1}\,(f(x)-y) \right] : \right.$$
$$\left. k = 1, \ldots, n+1 \right\},$$

else if $f(x) - y < 0$, then $\mathcal{T}$ is the empty set.

The key to spherical reduction is the acceleration function A which relates the radius of S to the radius of D, as shown in Figure 5. We standardise by taking T in Figure 5 to have unit radius, whence A will be a function from $[0, 1]$ to $[0, n]$. The following theorem, whose proof was given in ([9], Theorem 3.1), describes A.

Theorem A.1. *Let T be a regular n-simplex of unit radius and S be an n-sphere with the same centre, and radius r, $0 \leq r \leq 1$. Then the radius, $A(r)$, of the largest regular n-simplex, D, dually oriented to T, sharing the common centre and such that $T \cap D \subseteq S$, is given by the piecewise formula, with n parts:*

$$A(r) = \begin{cases} r\,, & \text{for } 0 \leq r \leq \sin\theta_{n,n-1}\,, \\ \vdots & \\ \text{maximum value of } A \text{ on previous interval} + \dfrac{\sqrt{r^2 - \sin^2\theta_{n,i}}}{\prod_{j=i+1}^{n} \tan\theta_{j,i}}\,, & \text{for } \sin\theta_{n,i} \leq r \leq \sin\theta_{n,i-1}\,, \\ \vdots & \\ \text{maximum value of } A \text{ on previous interval} + \dfrac{\sqrt{r^2 - \sin^2\theta_{n,1}}}{\prod_{j=2}^{n} \tan\theta_{j,1}}\,, & \text{for } \sin\theta_{n,1} \leq r \leq 1 \end{cases}$$

where $\theta_{j,k}$, for $j \geq k$, is the angle between the vertex to centroid line, and vertex to centroid of a k-dimensional face line, in a j-simplex.

The accelerated upper reduction procedure is then given in the following definition. This replaces (1) in Definition A.2.

DEFINITION A.3. Let $T(x, y, h)$ be a standard simplex. If $h \leqslant f(x) - y \leqslant 2h$, then the upper reduction of this simplex is

$$\mathcal{T} = \left\{ T\left[x + \frac{F(x) - y}{M(n+1)} u_k,\ y + \frac{F(x) - y}{n+1},\ h - \frac{1}{n+1}(F(x) - y) \right] : \right.$$
$$\left. k = 1, \ldots, n+1 \right\},$$

else if $2h < f(x) - y$, then $\mathcal{T}$ is the empty set. Here $F(x) = hA\left[\frac{f(x) - (y+h)}{h}\right] + (y+h)$ is the effective evaluation of f at x.

Appendix 2: Test Functions

For each of the five functions explored in §3, we present the function f, the location of the global minima x^*, the Lipschitz constant M which we adopted, the initial feasible domain H (in terms of the centre c and radius r, see Definition 2.1(5)) and the variation of the initial simplex, V_0.

1. **Goldstein and Price** (GOLDPR)

$$f(x_1', x_2') = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \cdot$$
$$[30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]/\delta$$

where $x_1 = 4x_1' - 2$, $x_2 = 4x_2' - 2$ and $\delta = 1{,}015{,}000$. Then $x^* = (0.5000, 0.2500, 0.0000)$; $M = 50$; $c = (0.5, 0.5)$, $r = 0.7098$; $V_0 = 70.3$. Our initial domain cuts the corners from $[0, 1] \times [0, 1]$ in order to avoid a second minimum.

2. **Branin** (RCOS)

$$f(x_1', x_2') = [a(x_2 - bx_1^2 + cx_1 - d)^2 + e(1 - f)\cos x_1 + e]/g$$

where $a = 1$, $b = 5.1/(4\pi^2)$, $c = 5/\pi$, $d = 6$, $e = 10$, $f = 1/(8\pi)$, $g = 308.1$ and $x_1 = 15x_1' - 5$, $x_2 = 15x_2'$. There are three global minima, at $(0.5428, 0.1517, 0.0013)$, $(0.1239, 0.8183, 0.0013)$ and $(0.9617, 0.1650, 0.0013)$, and $M = 10$; $c = (0.5, 0.5)$, $r = 0.7887$; $V_0 = 15.63$. Here H is the smallest hexagon containing $[0, 1] \times [0, 1]$.

3. **Mladineo** (FUNCT2)

$$f(x_1, x_2) = -[\sin(4x_1 + 1) + 2\sin(6x_2 + 2)].$$

For the hexagonal initial domain we use there are two global minima, at

$(0.1427, 0.9759, -3.0000)$ and $(0.1427, -0.0715, -3.000)$. Also $M = 12.65$; $c = (0.5, 0.5)$, $r = 0.7887$; $V_0 = 19.24$.

We now define a class of functions on which the type of algorithm we are using thrives. They have the appearance of down-under mountain ranges, and were suggested by recent work of Mladineo. For $i = 1, \ldots, m$, let M_i be a positive real number, and c_i be a point in R^n. Define

$$f(x) = \min\{-M_i \exp(-\|x - c_i\|) : i = 1, \ldots, m\} \quad \text{on } R^n .$$

Then $x^* = (c_{i_0}, -M_{i_0})$, where i_0 is such that $M_{i_0} = \max\{M_i : i = 1, \ldots, m\}$.

4. **Mladineo(2, 3).** Two variables and three inverted peaks.
We choose $n = 2$, $m = 3$ and let c_1, c_2 and c_3 be $(-0.5, -0.5)$, $(0.6, -0.4)$ and $(0, 0.8)$ respectively, with $M_i = \sqrt{i}$, for $i = 1, 2, 3$. Then $x^* = (0.8000, -1.7321)$; $M = \sqrt{3}$; $c = (0, 0)$, $r = 1$; $V_0 = 3.435$.

5. **Mladineo(4, 3).** Four variables and three inverted peaks.
We choose $n = 4$, $m = 3$ and let c_1, c_2 and c_3 be $0.7u_1 + 0.5u_2 + 0.6u_3 + 0.8u_4$, $-0.6u_1 - 0.7u_2 - 0.8u_3 + 0.5u_4$ and $0.8u_5$ respectively, with $M_i = \sqrt{i}$, for $i = 1, 2, 3$. Here $u_1, \ldots, u_5$ are the directions in R^n defining the vertices of the simplex top, see §2.2. Then $x^* = (0, 0, 0, 0.8000, -1.7321)$; $M = \sqrt{3}$; $c = (0, 0, 0, 0)$, $r = 1$; $V_0 = 6.897$.

References

1. Baritompa, W. (1990), A Dual View of Multidimensional Bisection – Extensions and Implementation, Research Report, University of Canterbury, Christchurch, New Zealand.
2. Basso, P. (1982), Iterative Methods for the Localization of the Global Maximum, *SIAM J. Numerical Analysis* **19**, 781–792.
3. Mladineo, R. H. (1986), An Algorithm for Finding the Global Maximum of a Multimodal, Multivariate Function, *Mathematical Programming* **34**, 188–200.
4. Mladineo, R. H. (1992), Convergence Rates of a Global Optimization Algorithm, *Mathematical Programming* **54**, 223–232.
5. Horst, R. and Tuy, H. (1987), On the Convergence of Global Methods in Multiextremal Optimization, *J. Optimization Theory and Applications* **54**, 253–271.
6. Horst, R. and Tuy, H. (1990), *Global Optimization (Deterministic Approaches)*, Springer, Berlin.
7. Piyavskii, S. A. (1972), An Algorithm for Finding the Absolute Extremum of a Function, *USSR Computational Mathematics and Mathematical Physics* **12**, 57–67.
8. Shubert, B. O. (1972), A Sequential Method Seeking the Global Maximum of a Function, *SIAM J. Numerical Analysis* **9**, 379–388.
9. Wood, G R. (1991), Multidimensional Bisection and Global Optimisation, *Computers and Mathematics with Applications* **21**, 161–172.
10. Wood, G. R. (1992), The Bisection Method in Higher Dimensions, *Mathematical Programming*, **55**, 319–337.