Pergamon

Computers Math. Applic. Vol. 27, No. 4, pp. 11–22, 1994
Printed in Great Britain.
0898-1221/94 $6.00 + 0.00

0898-1221(93)E0006-R

Multidimensional Bisection: A Dual Viewpoint

W. BARITOMPA*
Department of Mathematics and Statistics, University of Canterbury
Christchurch, New Zealand

(Received July 1993; accepted August 1993)

Abstract—This paper provides an alternative viewpoint of multidimensional bisection global optimization methods of Wood. A dual coordinate representation of convex bodies is introduced which leads to an easy implementation and eliminates the need to see the geometry of intersecting simplexes. Although developed in the context of global optimization, the techniques deal more generally with regions represented as the union of convex bodies. With this dual framework the algorithm can be implemented efficiently using any multiattribute index data structure that allows for quick range queries. A C version using a "multi-key double linked skip list" based on Pugh's skip list has been implemented.

Keywords—Multidimensional bisection, Global optimization, Skip lists, Convex bodies.

1. INTRODUCTION

This paper provides a simple description and an easy implementation of multidimensional bisection global optimization methods of Wood [1]. This is achieved by a *dual coordinate representation* of the convex bodies used in the algorithms. This representation uses the right hand vector in the matrix inequality $Ax \geq s$ which specifies these bodies. The multidimensional bisection methods have the salient features of the usual "root finding" bisection, in that they produce a nested family of bracketing regions for the global minimum. They can be viewed as a geometric realization of Piyavkii's general approach [2,3] which uses a lower envelope of a function to estimate the global minimum, although as pointed out in [4], they can be used without building lower envelopes.

This paper builds on Wood's work. Its emphasis is on implementation. It provides the viewpoint behind the implementation used in [5] and supplies details of the techniques given in [1,6]. The theory, context and performance of the optimization techniques can be found in [1,4–6]. For completeness, Section 2 provides a brief review of multidimensional bisection. Section 3 presents the dual coordinate representation. Section 4 starts with an example of the main processes. It gives a simple description of the algorithms in this new framework and provides justifying theorems. Section 5 describes the required procedures, justifies key steps, and discusses data structure requirements needed for an implementation. Details of an available C-program are given. An Appendix contains computational details and extensions to the formulæ provided in [1,5] covering complete and spherical acceleration.

*Research was supported in part by an Erskine grant from the University of Canterbury.
The author wishes to thank Graham Wood for sharing his work on multidimensional bisection and his encouragement with the development of the implementation. He would also like to thank Zelda Zabinsky and Ivone Sasmitra for their helpful comments.

2. A REVIEW OF MULTIDIMENSIONAL BISECTION

A brief review extracted from [1] is given here. Refer to [1,5] for more details. The problem is the following: given $f : \mathbb{R}^n \to \mathbb{R}$ and K a compact domain in $\mathbb{R}^n$, find the points on the graph of f where $\min f(x)$ over $x \in K$ is realized. It is assumed that a constant M is known for which the function f belongs to $L(M)$, the set of Lipschitz continuous functions, or $SG(M)$ a class of functions described in [4].

The approach taken by Wood starts by bracketing all global minima in a simplex. At each iteration regions are cut away from this initial simplex in such a way as to leave a *system* of simplexes, the union of which gives an improved bracket.

Let $\bigtriangledown$ be the infinite cone of slope M with a simplex base, as shown in Figure 1. The two mathematical facts (see [1] for $L(M)$ and [4] for $SG(M)$) that insure a convergent algorithm are the following. For each (x, y) on the graph of f, (1) no point inside $(x, y) - \bigtriangledown$ and (2) no point above (x, y) can be a global minimum of f.

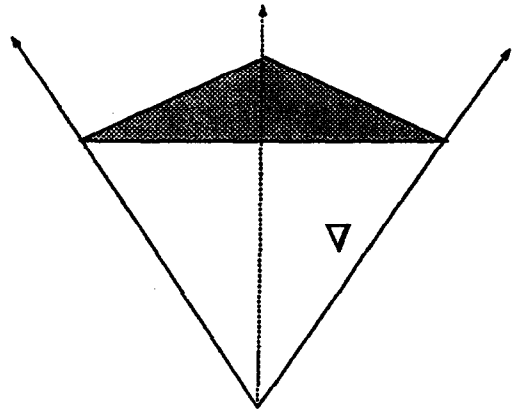

Figure 1. The simplex based cone $\bigtriangledown$.

These two facts allow an algorithm to be set up in a very simple way. A version of multidimensional bisection with complete reduction [1, p. 166] is described now.

At the outset the initial system consists of one standard simplex , T_0, which brackets all global minima over K. Here a *standard simplex* is a translate of a cap of the cone $\bigtriangledown$. For each function evaluation, cut from every simplex in the system the interior of $-\bigtriangledown$, with apex moved to the evaluation point on the graph of f. Also cap all the simplexes at the height of the lowest known evaluation. These processes are termed *reduction* and *elimination* in [1], or *cutting* and *capping* in [4]. When these are done to a standard simplex, at most $n + 1$ standard simplexes of smaller height than the original are left. After each iteration, the algorithm brackets all global minima over K in the union of the standard simplexes belonging to the current updated system. Figure 2 shows such a system for a function of two variables. All simplex tops, shaded in the figure, lie at the height of the least evaluation to date. The process continues until the maximum height of all simplexes in the system (the *variation*) is small enough. Properties (1) and (2) above guarantee no global minimum is removed. The key to the understanding of multidimensional bisection is what happens to one standard simplex as shown in Figure 3.

Proceeding more formally, 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 $\mathbb{R}^n$. Thus, $u_1 + \cdots + u_{n+1} = 0$ and $u_k \cdot u_l = -1/n$ for all distinct pairs k and l (see Appendix for details of finding these vectors). So $\bigtriangledown$ is the cone in $\mathbb{R}^{n+1}$ with apex the origin and cross-section $\text{co}\{u_1, \ldots, u_{n+1}\}$ at height M along the $(n+1)^{\text{st}}$ axis, where "co" denotes the convex hull. Formally, $\bigtriangledown = \text{pos}\{(u_k, M) : k = 1, \ldots, n+1\}$, where "pos" denotes all positive linear combinations.

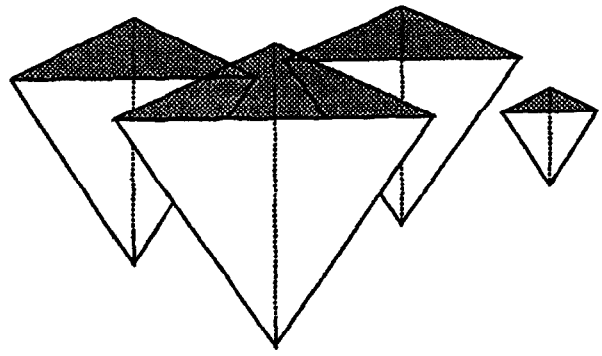

Figure 2. System of standard simplexes.

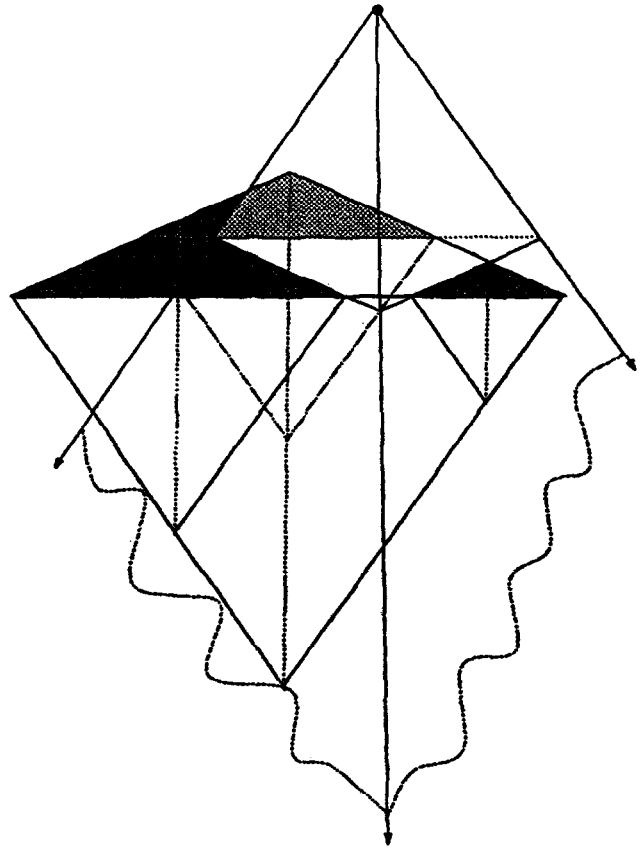

Figure 3. Cutting and capping a standard simplex.

The *standard simplex* in $\mathbb{R}^{n+1}$ with apex $(x, y) \in \mathbb{R}^{n+1}$ and height $h \in \mathbb{R}$ has the form

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

The *top* of $T(x, y, h)$ is the face opposite the apex. The *usual coordinate representation* of the standard simplex is (x, y, h). Points can be viewed as "degenerate simplexes" with height 0. The degenerate simplex corresponding to a function evaluation will be denoted by $E(x, f(x), 0)$.

A *system* of simplexes $\mathcal{S}$ in $\mathbb{R}^{n+1}$ is a finite set of standard simplexes. It is a *uniform system* if all tops lie in the same hyperplane of $\mathbb{R}^{n+1}$. The *variation* of $\mathcal{S}$, $V(\mathcal{S})$, is the difference between the highest and lowest points in the system.

The results in this paper primarily concern the bracket. The strategy for choosing the next function evaluation is important and covered in [1]. Here is an outline of "sequential deepest

point with complete reduction" (A^c in [1]) described in geometric terms:

Initialization: Choose an initial simplex T_0. $\mathcal{S}_0 = \{T_0\}$.

Get Next Point: Find lowest apex (x_i, y_i^*) of all simplexes in $\mathcal{S}$. Compute $f(x_i)$.

Capping: Lower all tops to the lowest value so far.

Cutting: From each simplex in $\mathcal{S}$, remove the interior of $(x_i, f(x_i)) - \bigtriangledown$.

Stopping Test: If $V(\mathcal{S})$ is small enough, terminate, otherwise go to Get Next Point step.

The formulæ for these reductions in terms of usual coordinates are included in the Appendix. These formulæ extend the results in [1,5]. In practice, the dual coordinates are used directly.

3. THE DUAL COORDINATE REPRESENTATION

With dual coordinates, the bracket is represented efficiently and the geometry of the removal process described easily. Terminology and justification of the basic properties of this representation are given in this section.

The simplexes $T(x, y, h)$ of the system are convex bodies. The usual coordinates relate to its vertices. The dual viewpoint emphasizes the faces. The key observation is that there is a fixed family of functionals determining the faces of all the simplexes in the system. The dot product with vectors orthogonal to the faces of the usual standard simplex provides the dual coordinates. So inequalities with fixed left hand sides and the dot products on the right determine a simplex.

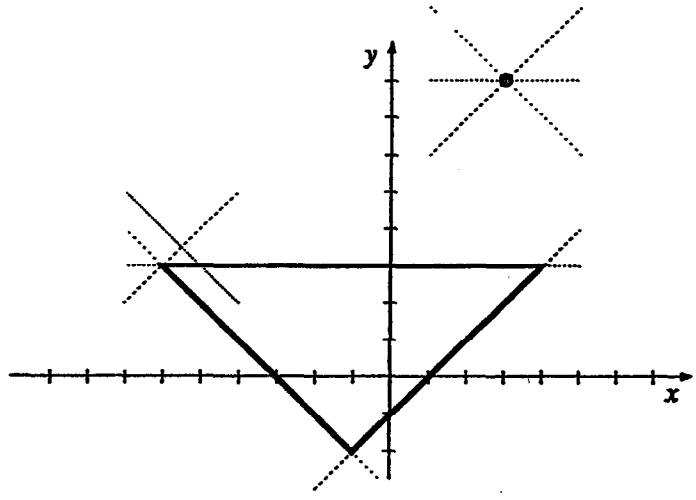

Figure 4. Representing regions by inequalities.

As an example consider Figure 4:

- The triangle with apex at $(-1, -2)$ and height 5 is specified by $x + y \geq -3$, $-x + y \geq -1$, and $-2y \geq -6$;
- The point $(3, 8)$ (a "degenerate" simplex) is specified by $x + y \geq 11$, $-x + y \geq 5$, and $-2y \geq -16$;
- Finally no point satisfies $x + y \geq -2$, $-x + y \geq 9$, and $-2y \geq -6$.

So the triangle has dual coordinates $(-3, -1, -6)$, the point has $(11, 5, -16)$, and the empty set has $(-2, 9, -6)$.

Formally a body is described by a vector inequality $\{v \mid Av \geq (s, s_{\text{top}})^\top\}$ where $s \in \mathbb{R}^{n+1}$ and $s_{\text{top}} \in \mathbb{R}$. (Note: The vector inequality means the inequality holds for each coordinate.) For

dimension n with constant M

$$A = \begin{pmatrix} u_1 & \frac{1}{nM} \\ \vdots & \vdots \\ u_{n+1} & \frac{1}{nM} \\ 0 & -\frac{n+1}{nM} \end{pmatrix}.$$

The rows of A are vectors orthogonal to the faces of the usual standard simplex. The $(n+2)$-tuple (s, s_{top}) gives the *dual coordinates*. $T(s, s_{\text{top}})$ will be used to refer to a body given in dual coordinates. At times it will be convenient to refer to the components of $s = (s_0, \ldots, s_n)$. Define $s_{\text{vari}} = s_0 + \cdots + s_n$.

THEOREM 3.1. *If specifying a nonempty set, the dual coordinates uniquely identify the convex body. Conversion between usual and dual coordinates is given by simple transformations (see the Appendix). In particular,*

- *the height of a simplex* $h = -\frac{nM}{n+1}(s_{\text{vari}} + s_{\text{top}})$;
- *the usual coordinate of the top of a simplex* $y + h = -\frac{nM}{n+1} s_{\text{top}}$.

Note by the above result, in a uniform system all simplexes have the same value of s_{top}.

4. THE ALGORITHM IN THE DUAL FRAMEWORK

The dual view leads to an easy implementation and eliminates the need to see the geometry of intersecting simplicial cones. The standard simplexes are kept as a list of the vectors. The geometric ideas of cutting and capping relate to simple modifications of the list.

Figure 5 gives an example of the geometric process in the one-dimensional case with $M = 1$. Cutting and capping remove the two regions at the evaluation and leave an improved bracket (shown outlined in bold). In this example the bracket begins as the union of a system of four simplexes (triangles outlined lightly). The geometric process is realized by removing the regions from each to get the updated system. Here the leftmost triangle is affected only by capping, the middle two produce four new smaller triangles, and the rightmost is completely removed.

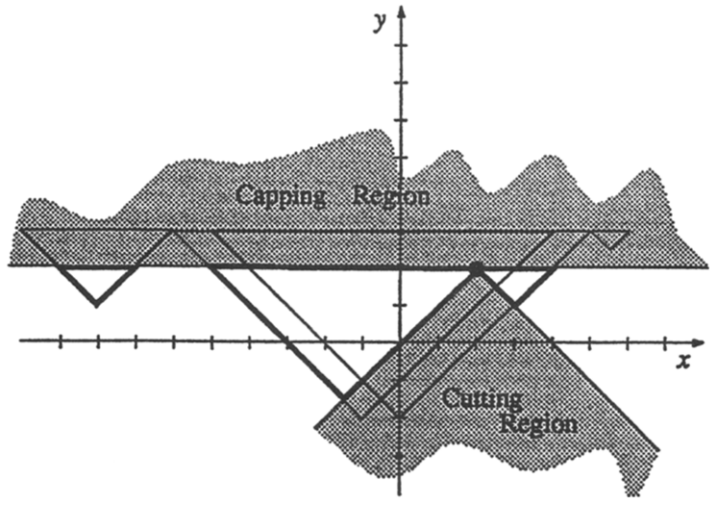

Figure 5. Cutting and capping of bracket.

In dual coordinates, the procedure is one of systematically changing coordinates in turn. Basically the dual coordinates always increase. Table 1 shows the dual coordinates of the system pictured in Figure 5 as the capping and cutting processes are applied. In this example $A = \begin{pmatrix} 1 & 1 \\ -1 & 1 \\ 0 & -2 \end{pmatrix}$, the evaluation at $(2,2)$ has dual coordinates $(r, r_{\text{top}}) = (4, 0, -4)$. Moving

from the first to the second row of Table 1 shows the result of capping. The new s_{top} changes to the larger value of r_{top}. Moving from the second to the third and fourth rows reflects the cutting process. Here the new s_0 is the larger of the old s_0 and r_0, and similarly for s_1. This produces eight representations of simplexes. However, only the bold faced ones are needed. The others represent either the empty set or a redundant simplex inside another.

Table 1. Cutting and capping by $E(4, 0, -4)$.

	$(-7, 9, -6)$	$(-3, -1, -6)$	$(-2, -2, -6)$	$(8, -3, -6)$
After Capping	$(-7, 9, -4)$	$(-3, -1, -4)$	$(-2, -2, -4)$	$(8, -3, -4)$
After Cutting	$(4, 9, -4)$	$(4, -1, -4)$	$(4, -2, -4)$	$(8, -3, -4)$
After Cutting	$(-7, 9, -4)$	$(-3, 0, -4)$	$(-2, 0, -4)$	$(8, 0, -4)$

The example illustrates the procedures needed in order to implement the algorithm. The effect of cutting and capping must be described and ways to eliminate inefficient representation must be handled. The following describes the effect of cutting and capping applied to one standard simplex as pictured in Figure 3.

THEOREM 4.1. *Let (r, r_{top}) be the dual coordinates of an evaluation, E. Let (s, s_{top}) be the dual coordinates of a simplex T. Capping of T by E changes the top coordinate to* $\max(s_{\text{top}}, r_{\text{top}})$. *Cutting gives $\mathcal{T} = \{T_k\}_{k=0,\ldots,n}$. The coordinates of the simplex T_k are found by further changing the k^{th} coordinate to* $\max(s_k, r_k)$.

PROOF. Note that geometrically cutting and capping of the standard simplex T removes the interior of the cone $(x_i, f(x_i)) - \nabla$ and the half-hyperplane above the function evaluation. In terms of matrix inequalities, the interior of the cone is $\{v \mid Av < (r, \infty)^T\}$. The half-hyperplane is $\{v \mid Av < (\infty, \ldots, \infty, r_{\text{top}})\}$. The complement of the former is $H_0 \cup \cdots \cup H_n$, where H_k is the half-hyperplane, $\{v \mid (k^{th} \text{ row of } A)v \geq r_k\}$. The complement of the later is H_{top} which equals $\{v \mid (\text{last row of } A)v \geq r_{\text{top}}\}$. Expanding $T \cap (H_0 \cup \cdots \cup H_n) \cap H_{\text{top}}$ gives the desired result. ∎

In order to implement the algorithm efficiently, some refinement is necessary. It is desirable to describe a system of simplexes with the minimal amount of storage requirements. Simply applying Theorem 4.1 provides $n + 1$ dually represented simplexes for each of the original. Upon closer inspection, some of these represent empty bodies and others represent redundant bodies properly contained inside other ones.[1] A *minimal uniform system*, which contains only representations of nonempty bodies and has the property that no body is properly contained in any other provides a more parsimonious description. The following provides a convenient test:

THEOREM 4.2.

(1) *$T(s, s_{\text{top}}) = \phi$ if and only if $s_{\text{vari}} > -s_{\text{top}}$.*
(2) *If (s, s_{top}) and (s', s'_{top}) represent nonempty simplexes, $T(s, s_{\text{top}}) \subseteq T(s', s'_{\text{top}})$ if and only if $s \geq s'$.*

PROOF. The first result follows from the formula for h in Theorem 3.1. The second follows from the inequality specification corresponding to the dual coordinates. ∎

The process of reduction only affects those simplexes that actually meet the removal cone. So Theorem 4.1 need not be applied. The following shows how to avoid this.

THEOREM 4.3. *$T(s, s_{\text{top}})$ meets the removal cone of $E(r, r_{\text{top}})$ if and only if $r > s$.*

PROOF. As in the proof of Theorem 4.1, the complement of the removal cone is $H_0 \cup \cdots \cup H_n$, where H_k is the half-hyperplane, $\{v \mid (k^{th} \text{ row of } A)v \geq r_k\}$. The following equivalences hold:

[1] Overlapping simplexes which lead to redundancies usually occur only when the dimension is greater than 1.

$r \not> s \Leftrightarrow r_k \leq s_k$ for some $k \Leftrightarrow T \subset H_k$ for some $k \Leftrightarrow T \subset H_0 \cup \cdots \cup H_n \Leftrightarrow T$ is disjoint from the removal cone. ∎

Such simplexes in the system are the only ones necessary to look at when implementing the cutting process. The following terminology (inspired during a visit to the salmon ladder at Seattle's aquarium) is useful. Given the representation of an evaluation (r, r_{top}), we say a member of the system is a *spawner* of E if $r \geq s$. Note for technical reasons equality is allowed for a spawner. The *spawn* of a spawner are all the nonempty simplexes produced when Theorem 4.1 is applied. Concerning finding the deepest point of the system, in a uniform system, the deepest simplex will have the smallest value for s_{vari}.

It is now possible to describe multidimensional bisection with this dual viewpoint. The algorithm will work with a uniform system, so s_{top} is viewed as being a global variable associated with all simplexes in the system.

Initialization: Choose initial simplex T_0. Let s_{top} be its top. Let $\mathcal{S}_0 = \{T_0\}$.

Get Next Point: Find the simplex with minimum s_{vari}. Convert to usual coordinates to get the lowest apex (x_i, y_i^*). Compute $f(x_i)$. Convert $(x_i, f(x_i), 0)$ to the dual coordinates of the evaluation (r, r_{top}).

Capping: Let $s_{\text{top}} = \max(s_{\text{top}}, r_{\text{top}})$.

Remove Empties: Remove any simplexes from $\mathcal{S}$ with $s_{\text{vari}} > -s_{\text{top}}$.

Cutting: Find all the spawners of (r, r_{top}). Compute their spawn, remove any empty simplexes, and make it minimal by eliminating any that belong to another body in the spawn.

Stopping Test: If the smallest value of s_{vari} is large enough, terminate, otherwise loop.

The method of spherical acceleration was introduced in [1] and described geometrically in detail in [5]. Basically it is a way to take advantage of the fact that cones of spherical cross section could be removed. In practice this means, when an evaluation is above the system, an even higher value can be used. The following was the basis for the trials in [5] and describes this acceleration in the dual viewpoint. It is the basis for the formula in the Appendix which extends Definition 3.1 in [1].

THEOREM 4.4. *Let (r, r_{top}) be the dual coordinates of an evaluation and (s, s_{top}) be the dual coordinates of a simplex T where $s_{\text{top}} > r_{\text{top}}$. Spherically accelerated cutting of T proceeds as follows: s is a spawner of r if $s \leq r_{\text{accel}}$ and its accelerated spawn is the spawn of $T(s, s_{\text{top}})$ and the evaluation $(r_{\text{accel}}, r_{\text{top}})$ as in Theorem 4.1. Here*

$$r_{\text{accel}} = (r_0 + s_{\text{accel}}, \ldots, r_n + s_{\text{accel}})$$

where

$$s_{\text{accel}} = \frac{A(\rho)[d(n+1) + r_{\text{top}} - s_{\text{top}}] + r_{\text{top}} - s_{\text{top}}}{n+1},$$

$$d = \max\{r_i - s_i \mid i = 0, \ldots, n\},$$

$$\rho = \frac{s_{\text{top}} - r_{\text{top}}}{d(n+1) + r_{\text{top}} - s_{\text{top}}},$$

and the function $A(\rho)$ is as described in [1].

Note d used in the above theorem is a measure of the distance between the apex of T and $x \in K$. When this is zero, the formula corresponds to the spherical acceleration specified in [1]. As d goes to infinity, the effect of acceleration goes to zero. As r_{accel} must be calculated for each simplex in the system, the decision to use spherical acceleration must be be weighed against the increase in overheads.

5. IMPLEMENTATION DETAILS

In order to realize the implementation, it is necessary to set up the required procedures and appropriate data structure.

Required Procedures

The steps of the algorithm can be coded effectively if the system is stored in a structure that allows range queries using the keys $s_0, \ldots, s_n$ and s_{vari}. The procedures (with self explanatory names) discussed here relate to the outline in Section 4. Here $\mathcal{S}$ represents a system, s_{top} is a global variable for simplexes in the system, and E represents the evaluation.

$[E, s_{\text{top}}, s_{\text{known spawner}}] = \texttt{GetRegionAndCap}(\mathcal{S}, s_{\text{top}})$
- performs Get Next Point and Capping steps. In addition, returns $s_{\text{known spawner}}$, the dual coordinate of the simplex used to determine the evaluation.

`ConvertUsualToDual` and `ConvertDualToUsual`
- performs coordinate conversions (required by `GetRegionAndCap`). These can be done with only $2(n+1)$ multipications due to the structure of the matrices involved (see Appendix).

$[\mathcal{S}_{\text{empties gone}}] = \texttt{RemoveEmpties}(\mathcal{S})$
- retains all simplexes in $\mathcal{S}$ satisfying $s_{\text{vari}} \leq -s_{\text{top}}$.

$[\mathcal{S}_{\text{not affected}}, \mathcal{S}_{\text{spawners}}] = \texttt{NewSpawners}(\mathcal{S}, E(r, r_{\text{top}}), s_{\text{known spawner}})$
- separates $\mathcal{S}$ into those simplexes satisfying the range query $s_k \leq r_k$ for $k = 0, \ldots, n$ and those that do not.

$\mathcal{S}_{k\text{-spawn}} = \texttt{Spawn}_k(\mathcal{S}_{\text{spawners}}, E(r, r_{\text{top}}))$
- produces the part of the spawn (denoted T_k in Theorem 4.1) found by changing the k^{th}coordinate to r_k.

$\mathcal{S}_{\text{minimal}} = \texttt{MakeMinimal}(\mathcal{S})$
- loops through a system, starting with the simplex with the smallest value for s_{vari}, loops over those with higher vari-coordinates, and removes any redundant simplexes inside it. Theorem 5.3 shows this process creates a minimal system.

$[\mathcal{S}_{\text{non-redundant}}, \mathcal{S}_{\text{redundant}}] = \texttt{RemoveRedundant}(\mathcal{S}, T_{\text{to be kept}})$
- removes all simplexes (except T itself) with coordinates all bigger or equal than those of T (used by `MakeMinimal`).

The basic loop of the algorithm becomes

$[E, s_{\text{top}}, s_{\text{known spawner}}] = \texttt{GetRegionAndCap}(\mathcal{S}, s_{\text{top}})$
$[\mathcal{S}_{\text{empties gone}}] = \texttt{RemoveEmpties}(\mathcal{S})$
$[\mathcal{S}_{\text{not affected}}, \mathcal{S}_{\text{spawners}}] = \texttt{NewSpawners}(\mathcal{S}, E(r, r_{\text{top}}), s_{\text{known spawner}})$
$\mathcal{S}_{\text{spawn}} = \bigcup \texttt{MakeMinimal}(\texttt{RemoveEmpties}(\texttt{Spawn}_k(\mathcal{S}_{\text{spawners}}, E(r, r_{\text{top}}))))$
$\mathcal{S}_{\text{new minimal system}} = \mathcal{S}_{\text{spawn}} \cup \mathcal{S}_{\text{not affected}}$.

The following two results limit the size of the set to be searched to find spawners. A known spawner's coordinates can be used to give a two sided range query.

THEOREM 5.1. *Given a minimal system $\mathcal{S}$ and a spawner $T(s_{\text{known spawner}}, s_{\text{top}})$ of an evaluation $E(r, r_{\text{top}})$. Then all other spawners of E can be found in*

$$\bigcup_{k=0,\ldots,n} \{T(s, s_{\text{top}}) \mid k^{th} \text{ coordinate of } s_{\text{known spawner}} < s_k \leq r_k\}.$$

PROOF. Let $T'(s', s_{\text{top}})$ be a spawner of E, so for all k, $s'_k \leq_k$. If $T' \in \mathcal{S}$ but not in the union, s'_k is less than or equal to the k^{th} of $s_{\text{known spawner}}$ for all k. But $s' \leq s_{\text{known spawner}}$ contradicts the minimality of $\mathcal{S}$. ∎

More generally, a two sided query can be used.

THEOREM 5.2. *Given a minimal system $\mathcal{S}$ and an evaluation $E(r, r_{\text{top}})$. Then all other spawners of E can be found in*

$$\bigcup_{k=0,\ldots,n} \{T(s, s_{\text{top}}) \mid s_{\text{vari}} - r_{\text{vari}} + r_k \le s_k \le r_k\}.$$

PROOF. Let s be a spawner. By definition of a spawner for each k, $s_k \le r_k$. Taking the sum over all but index k gives $s_{\text{vari}} - s_k \le r_{\text{vari}} - r_k$ and rearrangement gives $s_{\text{vari}} - r_{\text{vari}} + r_k \le s_k$. ∎

The next two results justify the procedures.

THEOREM 5.3. *The routine* `MakeMinimal` *creates a minimal system.*

PROOF. Suppose two simplexes with coordinates (s, s_{top}) and (s', s_{top}) were left after running the procedure and $s \ge s'$. Then since $s'_{\text{vari}} \le s_{\text{vari}}$, the procedure would eliminate the simplex with coordinates (s, s_{top}) when it started looking for redundant simplexes to $T(s'_{\text{vari}}, s_{\text{top}})$. This contradicts them both being left at the end. ∎

THEOREM 5.4. *The scheme of separating the spawners from the nonspawners and recombining the minimal spawn creates the required minimal system.*

PROOF. $\mathcal{S}_k$ = `MakeMinimal(RemoveEmpties(Spawn`$_k$`(`$\mathcal{S}_{\text{spawners}}, E(r, r_{\text{top}})$`)))`. We need to show $\mathcal{S}_{\text{not affected}} \cup \mathcal{S}_0 \cup \cdots \cup \mathcal{S}_n$ is minimal. Suppose to the contrary that $T(s, s_{\text{top}}) \subseteq T'(s', s_{\text{top}})$ both belonged, so by Theorem 4.2 $s \ge s'$. Consider these four cases:

(1) T and T' belong to the same set of the union. This contradicts the minimality of that set.
(2) $T \in \mathcal{S}_k$ and $T' \in \mathcal{S}_{\text{not affected}}$. Since $T \in \mathcal{S}_k$, it came from a spawner so $r \ge s \ge s'$ which means T' is a spawner of E, which contradicts $T' \in \mathcal{S}_{\text{not affected}}$.
(3) $T' \in \mathcal{S}_k$ and $T \in \mathcal{S}_{\text{not affected}}$. T' came from a spawner $T''(s'', s_{\text{top}})$, say. So $s'' \le s' \le s$. This means $T \subseteq T''$ which contradicts $\mathcal{S}$ being minimal.
(4) $T' \in \mathcal{S}_k$ and $T \in \mathcal{S}_j$. Without loss of generality, $T' \in \mathcal{S}_0 \setminus \mathcal{S}_1$ and $T \in \mathcal{S}_1 \setminus \mathcal{S}_0$. So $s' = (r_0, s'_1, \ldots)$ and $s = (s_0, r_1, \ldots)$. Since $s' \le s$, $r_0 \le s_0$, but $T \notin \mathcal{S}_0$ means $s_0 < r_0$ which is a contradiction. ∎

Note at step (2) in the above proof, the technical condition of equality in the definition of spawner was required.

Data Structure Requirements

The procedures can be efficiently coded if range queries can be done quickly. The use of an inverted list can take advantage of Theorems 5.1 and 5.2 by first finding the index with the smallest number satisfying the query. There are a number of multikey structures [7–9] that are variations of K-d trees that are more efficient and could be utilized.

Implementation in C

An implementation in C is available from the author. This implementation includes a Multikey Double linked Skip List package based on modification of Pugh's Skip List code (available via anonymous ftp [10]). The multidimensional bisection routines are described using this package.

The skip list has many of the nice properties of balanced trees, but is easier to use. In a skip list the data nodes have pointers which form a linked list. Additionally, a certain proportion of the nodes (randomly chosen) have pointers which point a little further, thus skipping over their immediate successors. Of those, a certain proportion have nodes skipping even further. This scheme allows for $\log(N)$ searching (on average) while making insertion and deletion relatively easy. Although worst case behavior could be poor, it is highly unlikely and not data dependent (as nodes with multiple pointers are randomly allocated independently of incoming data).

The data structure used here maintains a skip list pointer scheme for each key. In the basic skip list, the simple unidirectional nature of the list was no barrier to fast insertion, as the predecessors are easily remembered during the search process. However, with multiple keys, locating the pointers appropriate to one key is no help for the other keys. For that reason, double linkages are used for all keys at all levels. In other words, a "Multi-key Double linked Skip List." Range queries are handled by the inverted list approach.

Note since the dual coordinates of all simplexes in the system come from the evaluations, the program maintains a list of the evaluations. Rather than storing the dual coordinates, only pointers to the evaluations that give rise to them is stored.

The program dynamically allocates storage to hold the system. Typically the system size builds up to a certain point and then stabilizes until the variation gets quite small. At that point the behavior becomes that of trying to minimize the constant function, and the storage requirements increase exponentially.

6. FUTURE WORK

Not only does this dual approach lead to a simplification of multidimensional bisection, but also implements generalizations (see [4]). Capping can be done with convex regions other than hyperplanes. Also various choices for the inequality matrix A lead to algorithms which built up the bracket from convex bodies other than standard simplexes. In these cases, even though the bodies are not uniformly of the same shape, the dual viewpoint provides a uniform representation which can be capitalized upon. In particular, cones over any polygon can be used. When seen in this context, at one end of a spectrum is Wood's method using cones over the simplex, the simplest polygon. Mladineo's [11] is on the other end using cones over the sphere, the limiting "polygon."

Although developed in the context of global optimization, the techniques apply more generally to regions represented as the union of convex bodies based on a fixed form of vector inequality. The matrix A used to describe simplexes is particularly nice and gave simple tests for empty and redundant representation. Work is in progress to deal with the more general situation. There the tests are more complicated, and entail "presolving" linear programming problems with a fixed matrix A.

Another area for further explorations concerns the optimal data structure suited to this algorithm. The bulk of the work of the algorithm is in finding the spawners. This is usually a small number out of all simplexes which can grow exponentionally. Some variation of the hB-tree [7] looks promising.

APPENDIX A

Vectors used to describe the standard simplexes can be constructed as follows.

REMARK A.1. Inductive construction of $\{u_1, \ldots, u_{n+1}\}$, the vertices of a regular simplex in $\mathbb{R}^n$.

- *Dimension* 1. Let $u_1 = -1$ and $u_2 = 1$.
- *Dimension* $n > 1$. Let the first $(n-1)$ coordinates of $\{u_1, \ldots, u_n\}$ be $\sqrt{1 - 1/n^2}$ times the vectors for dimension $(n-1)$, and let the last coordinate be $-1/n$. Let u_{n+1} be the unit vector in the n^{th} coordinate direction.

REMARK A.2. For dimension n, the vectors $\{u_1, \ldots, u_{n+1}\}$ can be described in terms of the n constants $(c_k^n = (n/(n+1)) \prod_{i=k+2}^{n} \sqrt{1 - 1/i^2})$ where $(k = 0, \ldots, n-1)$ as follows:

$$\frac{n}{n+1}\left(u_1^T \ldots u_{n+1}^T\right) = \begin{pmatrix} -c_0^n & c_0^n & & & \\ \frac{-c_1^n}{2} & \frac{-c_1^n}{2} & c_1^n & & \\ \vdots & \vdots & \vdots & & \\ \frac{-c_{n-2}^n}{(n-1)} & \cdots & \frac{-c_{n-2}^n}{(n-1)} & c_{n-2}^n & \\ \frac{-c_{n-1}^n}{n} & \cdots & \cdots & \frac{-c_{n-1}^n}{n} & c_{n-1}^n \end{pmatrix}.$$

Details of coordinate conversions of Theorem 3.1 between usual and dual coordinates is described here.

REMARK A.3. The dual coordinates can be found by taking dot products, $s_i = (u_{i+1}, 1/(nM)) \bullet (x, y)$ for $i = 0, \ldots, n$ and $s_{\text{top}} = -(y+h)(n+1)/(nM)$. As a matrix transformation $(s, s_{\text{top}})^T = S(x, y, h)^T$ where

$$S = \left(\begin{array}{c|c} A & \begin{array}{c} 0 \\ \vdots \\ -\frac{n+1}{nM} \end{array} \end{array} \right) = \begin{pmatrix} u_1 & \frac{1}{nM} & 0 \\ \vdots & \vdots & \vdots \\ u_{n+1} & \frac{1}{nM} & 0 \\ 0 & -\frac{n+1}{nM} & -\frac{n+1}{nM} \end{pmatrix}.$$

Finding the usual coordinates from the dual ones uses the inverse transformation:

$$S^{-1} = \frac{n}{n+1} \begin{pmatrix} u_1^T & \ldots & u_{n+1}^T & 0 \\ M & \ldots & M & 0 \\ -M & \ldots & -M & -M \end{pmatrix}.$$

The special structure mentioned in the previous remark can be utilized to perform these transformations with only $2(n+1)$ multiplications.

REMARK A.4. An efficient conversion from dual to usual coordinates is:

$$x_k = c_k^n \left(\frac{s_{k+1} - (s_0 + \cdots + s_k)}{(k+1)} \right) \text{ for } k = 0, \ldots, n-1$$
$$y = Mn \frac{(s_0 + \cdots + s_n)}{(n+1)}$$
$$h = -Mn \frac{s_{\text{top}}}{(n+1)} - y.$$

An efficient conversion from usual to dual coordinates is:

$$s_0 = 0$$
$$s_k = \frac{x_{k-1}}{c_{k-1}^n} + \frac{(s_0 + \cdots + s_{k-1})}{k} \text{ for } k = 1, \ldots, n$$
$$\Delta s = \frac{y}{(Mn)} - \frac{(s_0 + \cdots + s_n)}{(n+1)}$$
$$s_k = s_k + \Delta s \text{ for } k = 1, \ldots, n$$
$$s_{\text{top}} = \frac{-(y+h)(n+1)}{(nM)}.$$

The details of complete simplex reduction (with spherical acceleration) in usual coordinates is given here. This extends the formulæ in [1,5]. The proofs entail taking the dual coordinate formula of Theorem 4.1 and converting to the usual coordinates via the previous conversion.

REMARK A.5. Let $T(x, y, h)$ be a standard simplex. Let (x', y') be the usual representation of an evaluation.

Upper Reduction: If the evaluation is above the top of the simplex, the effect of the simplex by this evaluation is:

$$\left\{ T((x,y,h) + \frac{n}{n+1}\left(u_k, \frac{1}{nM}\right) \bullet [(x',y') - (x,y)](u_k, M, -M) \mid k = 1, \ldots, n+1 \right\}.$$

In terms of dual coordinates this is:

$$\left\{ T((x,y,h) + \frac{n}{n+1}[r_k - s_k](u_{k+1}, M, -M) \mid k = 0, \ldots, n \right\}.$$

Lower Reduction: If the new value will also lower the simplex (i.e., $y' < y + h$), then adding $y' - y - h$ to the height gives the result.

NOTE. If the final height is negative, the simplex is empty.

In the case of upper reduction, spherical acceleration can be applied using the above formula with y'' in place of of y'. Here $y'' = hA(\rho) + y' + h$ is the "effective" evaluation where

$$\rho = \frac{y' - y - h}{h + \max\{u_i \bullet (x' - x) \mid i = 1, \ldots, n+1\}},$$

and the function $A(\rho)$ is as described in [1].

REFERENCES

1. G.R. Wood, Multidimensional bisection and global optimisation, *Computers Math. Applic.* **21** (6/7), 161–172 (1991).
2. S.A. Piyavskii, An algorithm for finding the absolute extremum of a function, *USSR Computational Mathematics and Mathematical Physics* **12**, 57–67 (1972).
3. B.O. Shubert, A sequential method seeking the global maximum of a function, *SIAM J. Numerical Analysis* **9**, 379–388 (1972).
4. W. Baritompa, Customizing methods for global optimisation—A geometric viewpoint, *J. Global Optimisation* **3** (2), 193–212 (1993).
5. B.-P. Zhang, G.R. Wood and W. Baritompa, Multidimensional bisection: The performance and the context, *J. Global Optimisation* **3** (3), 337–358 (1993).
6. G.R. Wood, The bisection method in higher dimensions, *Mathematical Programming* **55**, 319–337 (1992).
7. D.B. Lomet and B. Salzberg, The *hB*-Tree: A multiattribute indexing method with good guaranteed performance, *ACM Trans. Database Sys.* **15** (4), 625–658 (1990).
8. J.T. Robinson, The *K-D-B*-tree: A search structure for large multidimensional dynamic indexes, *Proc. ACM SIGMOD Conf. Manage. Data*, pp. 10–18, ACM, New York, (1984).
9. J.L. Bently, Multidimensional binary search trees in database applications, *IEEE Trans. Sofw. Eng.* **SE-5 4**, 333–340 (1979).
10. W. Pugh, Skip lists: A probabilistic alternative to balanced trees, *Comm. of the ACM* **33** (6), 668–676 (1990).
11. R.H. Mladineo, An algorithm for finding the global maximum of a multimodal, multivariate function, *Mathematical Programming* **34**, 188–200 (1986).