Optimization
Vol. 52, Nos. 4–5, August–October 2003, pp. 379–394

Taylor & Francis
Taylor & Francis Group

GEOMETRY AND COMBINATORICS OF THE CUTTING ANGLE METHOD

GLEB BELIAKOV*

School of Information Technology, Deakin University, 221 Burwood Hwy, Burwood, 3125, Australia

(Received 11 December 2002; In final form 23 July 2003)

Lower approximation of Lipschitz functions plays an important role in deterministic global optimization. This article examines in detail the lower piecewise linear approximation which arises in the cutting angle method. All its local minima can be explicitly enumerated, and a special data structure was designed to process them very efficiently, improving previous results by several orders of magnitude. Further, some geometrical properties of the lower approximation have been studied, and regions on which this function is linear have been identified explicitly. Connection to a special distance function and Voronoi diagrams was established. An application of these results is a black-box multivariate random number generator, based on acceptance–rejection approach.

Keywords: Lipschitz optimization; Global optimization; Cutting angle method; Random number generator; Saw tooth cover

Mathematics Subject Classifications 2000: 65K05; 90C59

1 INTRODUCTION

Several methods of deterministic global optimization rely on construction of the lower approximation of the objective function. This lower approximation is used to locate an approximation to the global minimum of the original objective function. Since the lower approximation has a simpler structure, its minimization is also a much simpler problem (called a relaxed problem). The point of this exercise is to construct a sequence of lower approximations converging to the objective function. The sequence of global minima of the relaxed problems will converge to the global minimum of the original function.

The best illustration of this technique is the classical cutting plane method of Kelley [13]. The convex objective function is approximated with a sequence of minorants, consisting of tangent (or supporting) hyperplanes. The relaxed problem of minimization of the lower approximation is a linear programming problem. Hence

*E-mail: gleb@deakin.edu.au

ISSN 0233-1934 print: ISSN 1029-4945 online © 2003 Taylor & Francis Ltd
DOI: 10.1080/0233193031000161556

it is easily solved, and the point of the global minimum is where the new tangent plane is built. It is subsequently added to the lower approximation, and the process repeats.

The theory of abstract convexity [19] generalizes some of the properties of convex functions. In particular, many nonconvex functions (so-called abstract convex) can be represented as lower envelopes of some basic simple functions, other than linear. The generalizations of the cutting plane method replace the supporting hyperplanes with these simple functions, but otherwise leave the method unchanged. The relaxed problem is no longer linear, but with the right choice of the basic support functions it will have a special structure, which would allow one to find its global minimum algorithmically.

In the one-dimensional case, Pijavski–Shubert method is a good example [9,17,20]. The basic support functions are minima of two straight lines passing through a given point whose slope is related to the Lipschitz constant of the objective function (see Fig. 1). It is not difficult to list all local minima of the lower approximation, and hence to find the global minimum by comparing them, because there is exactly one local minimum between two tips of the support functions. Multivariate extensions of this method have been proposed (e.g., Mladineo's method), in which the lower approximation is a collection of hypercones. However, the relaxed problem requires computing all intersections of these hypercones, and this is a significant computational challenge [9,15]. Note that in higher dimension many support functions are required in order to have an acceptable quality of lower approximation and to achieve convergence of the optimization algorithm. Detailed studies of Lipschitz optimizations can be found in [11,12,18].

The cutting angle method [1,2,19] falls into the same class of optimization techniques, and is a generalization of the cutting plane method. Similar to Pijavski–Shubert method, the lower approximation is a piecewise linear function (see Fig. 2), and the multivariate case involves the hypercone intersection problem. However, the structure of the support functions is quite special, and it was possible to find a very efficient combinatorial algorithm to enumerate all local minima of the relaxed problem [3,4,19]. This article extends these results and provides an even faster method. It studies the relaxed problem in detail, develops an efficient data structure for it, analyses its complexity, and makes

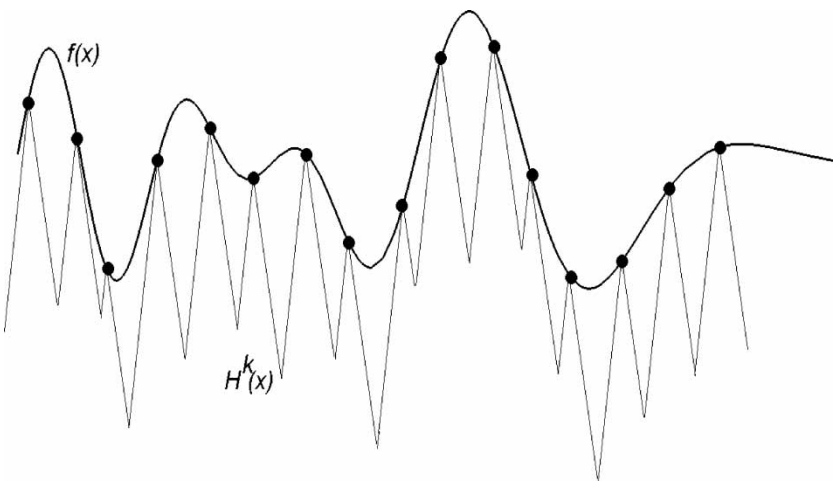

FIGURE 1 The lower approximation of $f(x)$ in the Pijavski–Shubert method (the saw-tooth underestimate).

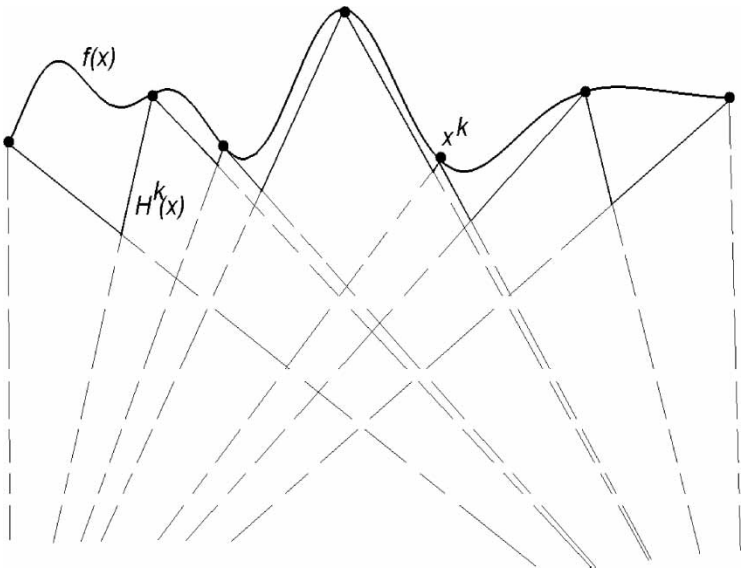

FIGURE 2 The lower approximation of $f(x)$ in the CAM.

important geometrical connections (especially to Voronoi diagrams). Understanding its geometry allowed us to find yet another important application of the lower approximation, other than optimization.

2 CUTTING ANGLE METHOD

The Cutting Angle Method (CAM) was presented in a series of papers [1–3] and the book [19], as an application of the theory of abstract convexity. This is a deterministic global optimization technique, in which a sequence of lower approximations to a non-convex objective function is made. The original optimization problem is replaced with a sequence of relaxed auxiliary problems of enumerating local minima of the lower approximation. It is shown that the sequence of global minima of the relaxed problems converges to the global minimum of the objective function [19].

This method is applicable to increasing positive homogeneous functions of degree one (IPH), with the unit simplex as their domain (i.e., the search for the global minimum is performed within the unit simplex). However, it was extended to restrictions of Lipschitz functions to the unit simplex, with the help of an additive constant [19]. The overall structure of the algorithm is the same as that of the cutting plane or of the Pijavski–Shubert methods, described in the Introduction. The key difference is the form of the lower approximation and solution to the relaxed problem.

In the remainder of this article we assume that the objective function $f > 0$ is IPH, and its domain is n-dimensional unit simplex $S_1 = \{x \in R^n: x_i \geq 0, \sum_{i=1}^n x_i = 1\}$. Assuming that there are K support functions, based at the points $(x^k, f(x^k))$, $k = 1, \ldots, K$, the lower approximation to the objective function f is given by

$$H^K(x) = \max_{k \leq K} \min_{i=1,\ldots,n} \frac{x_i}{l_i^k}, \tag{1}$$

where the kth support function is given by

$$h^k(x) = \min_{i=1,\ldots,n} \frac{x_i}{l_i^k} = f(x^k) \min\left\{\frac{x_1}{x_1^k}, \ldots, \frac{x_n}{x_n^k}\right\}. \tag{2}$$

The vectors

$$l^k = \left(\frac{x_1^k}{f(x^k)}, \frac{x_2^k}{f(x^k)}, \ldots, \frac{x_n^k}{f(x^k)}\right) \tag{3}$$

are called the support vectors, and they include those based at the vertices of the unit simplex (the first n support vectors). The case of two variables on the unit simplex (or one variable on the unit interval) is illustrated on Fig. 2. The lower approximation is frequently called the saw-tooth underestimate (or saw-tooth cover) of function f.

The method proceeds by building increasingly accurate lower approximations to f, using more and more support vectors, which are taken at the global minima of the current lower approximation (i.e., at the teeth of the saw-tooth cover). The key part of the algorithm is the solution of the auxiliary problem, described in the next section. It is important to solve this problem very efficiently, because in the case of several variables, the number of support functions required for the convergence of the algorithm is very large. The original algorithm in [1,19] solved this problem using standard integer programming approach, and was able to deal with about 100 support vectors. Current algorithm processes hundreds of thousands of support vectors.

3 SOLUTION TO THE AUXILIARY PROBLEM

The lower approximation to the objective function f using K support functions is given as Eq. (1), which can be also written as

$$H^K(x) = \max\{H^{K-1}(x), h^K(x)\}, \tag{4}$$

and the goal is to enumerate all its local minima. Let I denote $\{1, 2, \ldots, n\}$. In [2] the authors prove that this can be done by solving a combinatorial problem of enumerating all combinations of n support vectors that satisfy certain properties. Namely, given a set of K support vectors $\Lambda^K = \{l^k\}_{k=1}^K$, we need to find ordered combinations $L = \{l^{k_1}, l^{k_2}, \ldots, l^{k_n}\}$ of n support vectors, such that the following two conditions hold:

(I) $\forall i, j \in I, i \neq j$: $l_i^{k_i} > l_i^{k_j}$
(II) $\forall v \in \Lambda^K \setminus L\ , \exists i \in I$: $l_i^{k_i} \leq v_i$.

The combinations of support vectors L satisfying conditions (I), (II) are precisely those intersections of hypercones where functions $H^K(x)$ have local minima. An illustration of conditions (I), (II) was given in [2,4]. Think of combinations L as $n \times n$ matrices, whose rows are support vectors $l^{k_1}, l^{k_2}, \ldots, l^{k_n}$:

$$L = \begin{pmatrix} l_1^{k_1} & l_2^{k_1} & \ldots & l_n^{k_1} \\ l_1^{k_2} & l_2^{k_2} & \ldots & l_n^{k_2} \\ \vdots & \vdots & \ddots & \vdots \\ l_1^{k_n} & l_2^{k_n} & \ldots & l_n^{k_n} \end{pmatrix} \tag{5}$$

Condition (I) implies that the diagonal elements strictly dominate their columns, and condition (II) implies that the diagonal of L does not dominate any other support vector v, not already in L. The location of the local minimum $x_{\min}$ and its value $d = H^K(x_{\min})$ can be found from the diagonal of L:

$$\begin{aligned} x_{\min}(L) &= \operatorname{diag}(L)/\operatorname{Trace}(L) \\ d(L) &= H^K(x_{\min}) = \operatorname{Trace}(L)^{-1} \end{aligned} \tag{6}$$

The algorithm of [2] tested all possible combinations of n support vectors out of K against the above two conditions. The computational complexity of that was $O\left(n\binom{K-1}{n-1}\right)$.

In [4] the authors have proved that the required combinations of support vectors lie on a directed acyclic graph. It is possible to find all these combinations by examining only the nodes of this graph, rather than all possible combinations. The computational complexity was reduced to $O(n|V^{K-1}|)$, where $|V^{K-1}|$ is the number of local minima of $H^{K-1}(x)$.

Let us briefly describe the method from [4]. Suppose we have the set of $K-1$ support vectors Λ^{K-1} to which we add l^K. Also suppose we already know the set V^{K-1} of combinations of $K-1$ vectors satisfying (I), (II) (i.e., all local minima of $H^{K-1}(x)$). We need to update V^{K-1} to V^K. At this stage two events can take place: (a) some of the elements of V^{K-1} may be deleted because they fail test (II) (with l^K playing the role of v), and (b) new combinations containing l^K may be added to V^K.

The authors of [4] prove that these new combinations containing l^K can only be found from those that just have been deleted from V^{K-1}. The way to do it is to repeatedly replace each support vector in these deleted combinations with l^K and check condition (I). The main steps of the algorithm are as follows.

Algorithm 1 (update of the set V^{K-1})

Input The set V^{K-1} of local minima of $H^{K-1}(x)$; the new support vector l^K.
Output The set V^K.

Step 1 Let $V^K = \emptyset$.
Step 2 Test all elements L of V^{K-1} against condition (II), with $v = l^K$. Put those L that fail the test into Temp and those that pass into V^K.
Step 3 For every L in Temp, form n copies of it, and replace row i in the ith copy with l^K. Test condition (I) (diagonal dominance). If test passed, add this modified copy to V^K, otherwise discard it.

The Cutting Angle global optimization algorithm repeatedly calls Algorithm 1 to solve the relaxed problem; it sorts the set V^K for the lowest local minimum of $H^K(x)$, evaluates f at this point and adds the newly formed support vector to Λ^K.

Cutting Angle Algorithm

Assume V^K is sorted with respect to $d(L)$ in ascending order.

Step 1 Evaluate f at the vertices of the unit simplex and form $\Lambda^n = \{l^1, l^2, \ldots, l^n\}$.
Set $K = n$.
$L = \{l^1, l^2, \ldots, l^n\}$. $V^K = \{L\}$.

Step 2 Select $L = \text{Head}(V^K)$ (the global minimum of $H^K(x)$)
Evaluate $f = f(x_{\min}(L))$.
Set $K = K + 1$.
Form l^K using (3).
$\Lambda^K = \Lambda^{K-1} \cup l^K$.

Step 3 Call Algorithm 1 (V^{K-1}, l^K, V^K).
Go to Step 2.

The algorithm starts with just one combination of the first n support vectors built on the vertices of the unit simplex. By repeatedly generating new combinations from the old ones, it builds a tree, whose vertices are local minima of functions $H^n, H^{n+1}, \ldots$. After K support vectors have been created, the leaves of this tree are the local minima of the function $H^K(x)$. The stopping criterion is as in [2,19]

Figure 3 illustrates the beginning of the algorithm. Only the leaves of the tree are kept by the Algorithm 1, as they correspond to the set V^K.

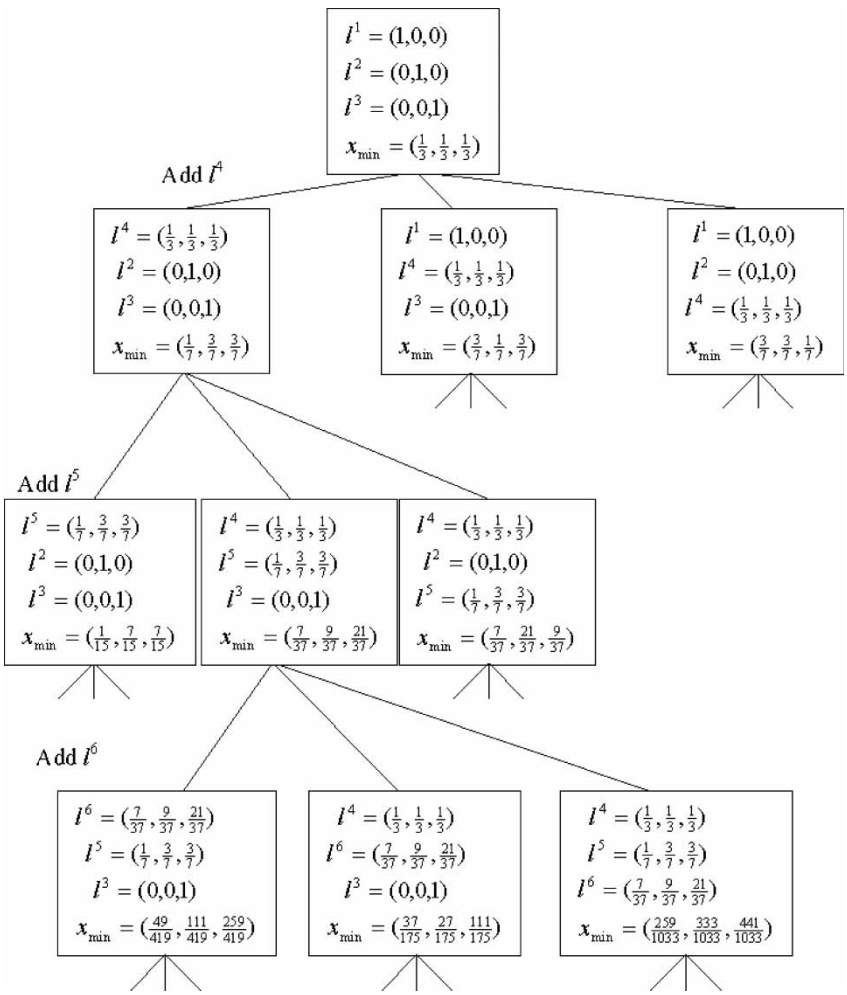

FIGURE 3 The tree of local minima of functions $H^n, H^{n+1}, \ldots$ for $f(x) = 1$.

At this stage we observe the following.

PROPOSITION 1 *Let* T^K *be the tree whose nodes are elements of*

$$V = \bigcup_{k=n,\ldots,K} V^K.$$

If condition (*II*) *fails for any node* L, *it also fails for its immediate predecessor node* (*and hence for all predecessor nodes*, *including the root*).

Proof By construction, the parent and child nodes of the tree differ only by one support vector, say $\hat{l}$ (see Step 3 of Algorithm 1). Let L_p denote the parent node and L_c denote the child node. The parent node L_p must have failed condition (II), with $\hat{l}$ playing role of v, i.e. $\text{diag}(L_p) > \hat{l}$. Therefore $\text{diag}(L_p) \geq \text{diag}(L_c)$. If now condition (II) fails for L_c (with any v) (i.e., $\text{diag}(L_c) > v$ then it also fails for L_p, since $\text{diag}(L_p) \geq \text{diag}(L_c) > v$. ■

Proposition 1 allows one to improve the complexity of the Algorithm 1 by reducing the number of tests at Step 2. Namely, if we keep not only the leaves of the tree, but the tree itself, we can test condition (II) with the newly added vector l^K starting from the root, and then continuing with the descendent nodes. We need to identify all the leaves for which condition (II) fails. By Proposition 1, condition (II) must also fail for all predecessors of such leaves. If it does not fail for any intermediate node, its descendants need not be tested. So the majority of nodes will hopefully not be tested because the corresponding branches of the tree will be cut off early in the process. If the tree is balanced, this would reduce the number of tested nodes from $|V^{K-1}|$ to $\log_n |V^{K-1}|$.

In addition to that, the cost of each test is reduced from $O(n)$ to $O(1)$: if the parent node has failed test (II) we only need to compare elements v_i and $l_i^{k_i}$, because the parent and the child nodes differ only by one support vector at position i. The price for the reduction of computational complexity is the need to keep the whole tree in memory (i.e., increase in space complexity). Notice, however, that given the relationship between parent and children nodes, one does not have to keep all support vectors that constitute a node, but only the reference to one vector by which the node differs from its parent, and its position. Hence the space complexity increases only marginally.

The improved (recursive) algorithm looks as follows.

Algorithm 2 (update of the tree T^{K-1})

Input: The tree T^{K-1} of local minima of $H^n, H^{n+1}, \ldots, H^{K-1}(x)$; the new support vector l^K; tested node L.

Output: The tree T^K; set V^K.

Step 1 Test L against condition (II), with $v = l^K$.

Step 2 If test succeeds, go to Step 5 (cut off this branch).

Step 3 If test fails, and L is not a leaf, then call Algorithm 2 (T^{K-1}, l^K, child(L), T^K, V^K) for all children of L.
Go to Step 5

Step 4 Otherwise (test failed, and L is a leaf) add n children to L.
Each child node is a copy of L, with l^{k_i} replaced with l^K in the ith child.
Test condition (I) for each child. If test fails, delete this child node.

Step 5 If L is V^n (root), then $T^K = T^{K-1}$; $V^K = \text{leaves}(T^K)$ (we need this only once, at the first level of recursion). Return.

The Cutting Angle algorithm is modified accordingly: At Step 1 add: $T^K = V^K$, and at Step 3 replace call Algorithm 1 with call Algorithm 2 (T^{K-1}, l^K, V^n, T^K, V^K). Notice that V^n is the root of T^{K-1}.

It can be seen from the algorithm that we require a data structure that maintains the tree of local minima of H, and efficiently locates the global minimum of $H^K(x)$ on that tree (which is one of the leaves). We implemented this data structure as the usual n-ary tree, together with a priority queue (binary heap), which contains two-way references to all leaves of the tree. The priority queue locates the global minimum of H^K in $O(1)$ operations. Then the value of f at the global minimum of H^K is evaluated and the new support vector l^K is built. The Algorithm 2 runs test (II) for nodes of the tree, starting with the root, and if this test fails for a leaf, the leaf is deleted from the priority queue. For all deleted leaves, the new support vector l^K repeatedly substitutes each of the n support vectors forming this node, and test (I) is performed. The leaf becomes an intermediary node, it branches into at most n new leaves, and the references to them are inserted into the heap. The maintenance of the binary heap takes $O(\log|V^K|)$ time.

Among various heaps, we investigated (empirically) the performance of the binary, Fibonacci, and trinomial heaps, and concluded that the binary heap is the most efficient structure in this case.

The performance of the Algorithm 2 depends on how quickly the number of local minima of H^K grows with K. Indicative values are presented in Table I. Each step of the cutting angle algorithm now takes $O(\log|V^K|)$ time.

The Algorithm 2 has been applied to a few test problems, such as Griewank's function

$$f(x) = \frac{1}{d}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1, \qquad d = 4000, \quad -50 \le x_i \le 50.$$

for $n = 2, 3, \ldots, 8$, and function

$$f(x) = -\sum_{i=1}^{10} \frac{1}{\|x - a^i\|^2 + c^i} \qquad 0 \le x_i \le 10, \quad i = 1, 2.$$

TABLE I The number of local minima of $H^K(x)$ as a function of n and K in the case $f(x) = 1$

K	$n=1$	$n=3$	$n=5$	$n=7$	$n=9$
100	99	334	686	966	1206
1000	999	4699	13 495	24 810	31 217
2000	1999	96 31	28 210	50 526	74 132
4000	3999	20 435	104 117	177 358	187 973
8000	7999	42 031	270 328	527 995	886 249
15 000	14 999	81 301	532 387	1 093 040	1 956 075
20 000	19 999	109 587	738 888	1 605 995	2 661 807
25 000	24 999	137 770	993 812	3 861 070	6 175 083
30 000	29 999	167 251	1 234 810	6 340 898	10 521 070

from [11], p. 261 (where the parameters a^i and c^i are specified), as well as to some benchmark problems from computational chemistry, such as protein folding problem, [7]. Some numerical results are described in [4,14]. This method has successfully solved all the proposed problems competitively to other methods (in most cases faster). For example, the difficult problem of minimizing the potential energy of Met-enkaephalin molecule (with respect to 24 dihedral angles, 11 of which were treated as global variables), which is a benchmark global optimization problem in computational chemistry [7,16], was solved using 120 000 function evaluations in 79 min (on a cluster of 36 DEC Alpha workstations) [14].

4 GEOMETRICAL PROPERTIES

Let us now have a closer look at the geometry of the functions $H^K(x)$. First of all, $H^K(x)$ is a piecewise linear function. It consists of nK hyperplanes, each hyperplane passes through one of the K points which constitute support vectors and $n-1$ vertices of the unit simplex (Fig. 4).

Local minima of $H^K(x)$ are intersections of n cones. Tips of these cones have different heights (the values $f(x^k)$). Let us establish the set of points on the unit simplex where these local minima can be located (given an arbitrary IPH f).

Let us denote by $\hat{x}^r$ the rth local minimum of $H^K(x)$ that corresponds to some combination of support vectors L^r satisfying (I) and (II). $H^K(\hat{x}^r) = 1/\mathrm{Trace}(L^r)$ by Eq. (6), and the ith component of $\hat{x}^r$ is given as

$$\hat{x}_i^r = \frac{l_i^{k_i}}{\mathrm{Trace}(L^r)} = \frac{x_i^{k_i}}{f(x^{k_i})\mathrm{Trace}(L^r)}.$$

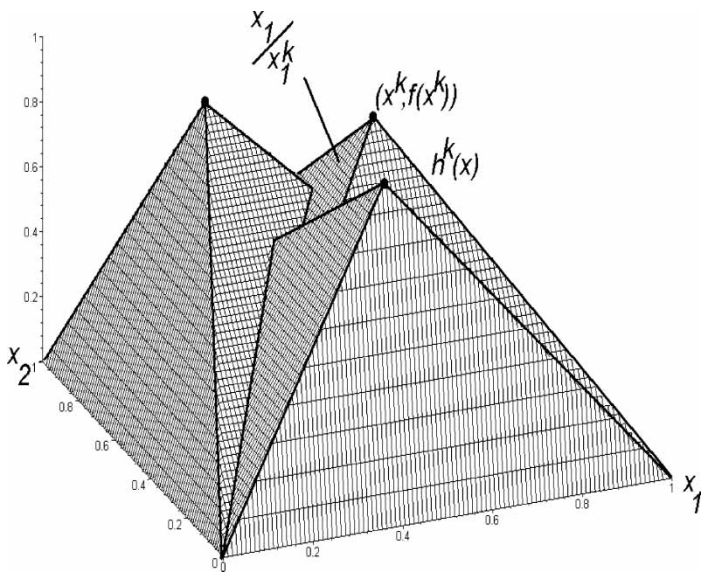

FIGURE 4 The lower approximation $H^K(x)$ in the case of three variables.

The diagonal elements of L^r are dominant in their columns by condition (I): $l_i^{k_i} > l_i^{k_j}$, or, using (3),

$$\frac{x_i^{k_i}}{f(x^{k_i})} > \frac{x_i^{k_j}}{f(x^{k_j})}.$$

Multiply by $x_j^{k_j}$ and divide by $\mathrm{Trace}(L^r)$:

$$\frac{1}{\mathrm{Trace}(L^r)} x_j^{k_j} \frac{x_i^{k_i}}{f(x^{k_i})} > \frac{1}{\mathrm{Trace}(L^r)} \frac{x_j^{k_j}}{f(x^{k_j})} x_i^{k_j}.$$

This results in the inequality

$$x_j^{k_j} \hat{x}_i^r > x_i^{k_j} \hat{x}_j^r.$$

The same analysis is valid for all $i, j \in I$, and therefore we arrive to $n(n-1)$ inequalities

$$x_j^{k_j} \hat{x}_i^r > x_i^{k_j} \hat{x}_j^r, \quad i, j \in I, \quad i \neq j. \tag{7}$$

Let us denote the set satisfying the system of inequalities (7) by S^r. Notice that the function f does not form part of the characterization of S^r. The following results are helpful in the analysis of $H^K(x)$.

PROPOSITION 2 *Let S^r denote the solution of* (7) *for a given combination of support vectors L^r. Then on S^r*

$$H^K(x) = \max\left\{\frac{x_1}{l_1^{k_1}}, \frac{x_2}{l_2^{k_2}}, \ldots, \frac{x_n}{l_n^{k_n}}\right\}.$$

Proof First we prove that on S^r $\min_{i \in I} x_i / l_i^{k_j} = x_j / l_j^{k_j}$. For a fixed j the inequalities from (7), $x_j^{k_j} x_i > x_i^{k_j} x_j$, $i \in I, i \neq j$, can be written as

$$\frac{x_i}{x_i^{k_j}} > \frac{x_j}{x_j^{k_j}},$$

and multiplying by $f(x^{k_j})$,

$$\frac{x_i}{l_i^{k_j}} = \frac{x_i}{x_i^{k_j}} f\left(x^{k_j}\right) > \frac{x_j}{x_j^{k_j}} f\left(x^{k_j}\right) = \frac{x_j}{l_j^{k_j}}.$$

Next, suppose that

$$H^K(x) = \max_{k \leq K} \min_{i \in I} \frac{x_i}{l_i^k} > \max\left\{\frac{x_1}{l_1^{k_1}}, \frac{x_2}{l_2^{k_2}}, \ldots, \frac{x_n}{l_n^{k_n}}\right\},$$

which means that there is another support vector $v \notin L^r = \{l^{k_1}, l^{k_2}, \ldots, l^{k_n}\}$:

$$\min_{i \in I} \frac{x_i}{v_i} > \max_{i \in I} \frac{x_i}{l_i^{k_i}}.$$

This would mean that $\forall i \in I$: $1/v_i > 1/l_i^{k_i}$, and hence $\forall i \in I$: $v_i < l_i^{k_i}$, which is impossible because L^r satisfies condition (II) (i.e., its diagonal does not dominate any other support vector). This completes the proof. ■

Geometrically this means that on S^r, the graph of the function $H^K(x)$ consists of only n hyperplanes, each hyperplane is defined by its corresponding support point $p^{k_i} = (x_1^{k_i}, x_2^{k_i}, \ldots, x_n^{k_i}, f(x^{k_i}))$, $i \in I$ and the n vertices of the unit simplex (we exclude the ith vertex). The local minimum of $H^K(x)$ on S^r is the intersection of these n hyperplanes.

An important implication of this result is that the local minimum of $H^K(x)$ on S^r is its global minimum there (or alternatively, there are no other local minima on S^r). The collection of sets $S^r, r = 1, \ldots, R$ forms a partition of the unit simplex.

Let us now consider how the functions $H^K(x)$ are related to Voronoi diagrams. We take the constant function $f(x) = 1$ in the sequel. Consider the following distance function on the unit simplex S_1

$$d(t, x) = \max_{i \in I} \left\{ \frac{x_i - t_i}{x_i} \right\} \tag{8}$$

Clearly, $d(t, x) \geq 0$ and $d(t, x) = 0$ iff $x = t$. This function is not a metric (it is obviously not commutative), but it does satisfy the triangular inequality $d(t, x) \leq d(t, y) + d(y, x)$. To prove this consider two cases

(a) Let us fix k at which $(x_k - t_k)/x_k = \max_{i \in I}(x_i - t_i)/x_i$, and let $y_k > x_k$. Then

$$\begin{aligned} \frac{x_k - t_k}{x_k} &= 1 - \frac{t_k}{x_k} \leq 1 - \frac{t_k}{y_k} = \frac{y_k - t_k}{y_k} \\ &\leq \max_{i \in I} \frac{y_i - t_i}{y_i} = d(t, y) \leq d(t, y) + d(y, x) \end{aligned}$$

(b) For k fixed above, let $y_k \leq x_k$. Then

$$\begin{aligned} \frac{x_k - t_k}{x_k} &= \frac{x_k - y_k}{x_k} + \frac{y_k - t_k}{x_k} = \frac{x_k - y_k}{x_k} + \frac{y_k}{x_k} \frac{y_k - t_k}{y_k} \\ &\leq \frac{x_k - y_k}{x_k} + \frac{\max(0, y_k - t_k)}{y_k} \leq \max_{i \in I} \frac{x_i - y_i}{x_i} + \max_{j \in I} \frac{y_j - t_j}{y_j} \\ &= d(y, x) + d(t, y). \end{aligned}$$

It is similar to polyhedral distance functions [5], but the distance between two points depends not only on their relative, but absolute positions (i.e., it is not invariant under translations).

PROPOSITION 3 *For a given point y on the unit simplex, the distance from y to the closest point $\hat{x}$ in the set $\{x^k\}_{k=1}^K$ is $d(y, \hat{x}) = 1 - H^K(y)$.*

Proof Indeed,

$$\begin{aligned} 1 - H^K(y) &= 1 - \max_k \min_i \frac{y_i}{x_i^k} = \min_k \left(1 - \min_i \frac{y_i}{x_i^k} \right) \\ &= \min_k \max_i \left(1 - \frac{y_i}{x_i^k} \right) = \min_k \max_i \left(\frac{x_i^k - y_i}{x_i^k} \right). \end{aligned}$$

■

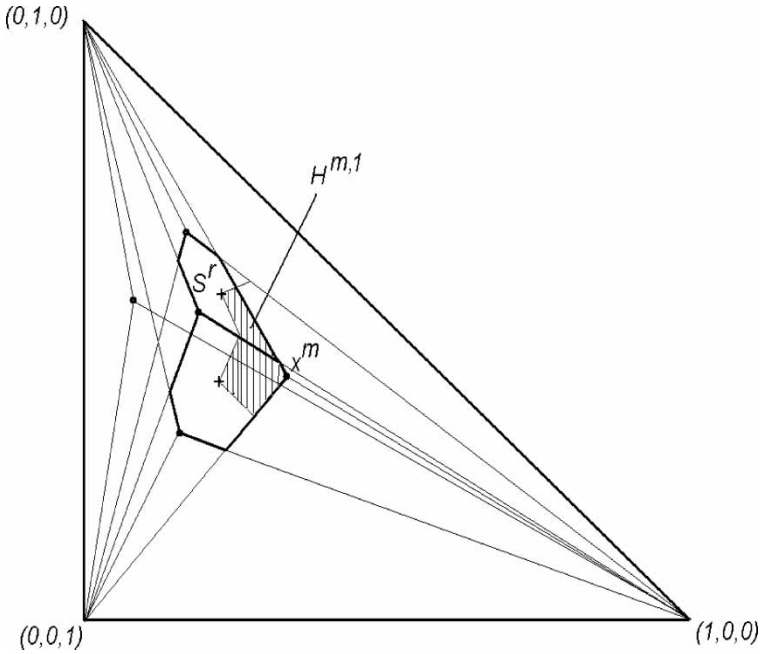

FIGURE 5 Regions S^r, part of the Voronoi region for x^m, and the set $H^{m,1}$ (shaded area), on which $H^K(x)$ is linear. Crosses denote some local minima of $H^K(x)$ and circles denote the support vectors.

Therefore, for $f(x) = 1$, the lower approximation $H^K(x)$ is simply one minus the distance to the closest point $\hat{x}$ in the selected distance function. Consider now sets VR^m on which a given x is closer to x^m than to any other point in $\{x^k\}_{k=1}^K$:

$$VR^m = \{x\colon d(x, x^m) \le d(x, x^k),\ k = 1, \ldots, K\}.$$

These sets constitute Voronoi regions in the selected distance (8). On each Voronoi region VR^m the function $H^K(x)$ depends only on x^m, and is one of the n hyperplanes passing through x^m.

Let us now take the intersections of the sets S^r and VR^m. Let $S^r(x^m, i)$ denote the sets defined by (7), which are based on a combination $L^r(x^m, i)$ that has support vector x^m in the ith position:

$$L^r(x^m, i) = \{l^{k_1}, \ldots, l^{k_{i-1}}, l^m, l^{k_{i+1}}, \ldots, l^{k_n}\}.$$

On each such set, $H^K(x)$ is defined as the maximum of n hyperplanes, one of which passes through x^m (Proposition 2). Taking the union of all such sets and intersecting with VR^m gives the set $H^{m,i}$ on which $H^K(x)$ is linear:

$$H^{m,i} = \left\{x\colon H^K(x) = \frac{x}{x_i^m}\right\} = VR^m \cap \left(\bigcup_r S^r(x^m, i)\right). \tag{9}$$

Figure 5 illustrates the sets discussed above.

5 APPLICATION

Explicit characterization of the sets on which $H^K(x)$ is bounded by one of its local minima permits to construct an explicit piecewise constant approximation of f from below. Namely,

$$G^K(x) = \min_{x \in S^r} H^K(x) \tag{10}$$

The point of this approximation is in the easiness of identifying S^r and $G^K(x)$ given a point x and the value $f(x)$. Indeed, if we build a support vector $v = x/f(x)$, and then run test (II) with this v, the node of the tree L^r which characterizes S^r, will fail the test (perhaps among a few other nodes). Then we can explicitly test these nodes (i.e., by checking conditions (7)), and identify S^r, and therefore the corresponding local minimum of $H^K(x)$ and compute $G^K(x)$. We already mentioned that the cost of test (II) is $O(\log |V^K|)$.

The application of such piecewise constant approximation comes from statistical simulation, where generators of random numbers with arbitrary distributions are frequently needed (e.g., Markov chain Monte Carlo methods (MCMC) [8]). One of the general approaches to generating random numbers with distribution density $\rho(x)$ is called acceptance–rejection method [6]. It consists in building a simple upper approximation to $\rho(x)$, $G(x)$, and generating a random number x with the density $G(x)$, which is a much simpler problem. It is particularly easy if $G(x)$ is piecewise constant, and one efficient method to do it is the alias method [6]. Now, suppose that a random number x from the distribution G was generated. Let us now generate another random number y, uniformly distributed in the interval $[0, G(x)]$. We compare y to the value $\rho(x)$, and decide whether x should be accepted or rejected. If y falls into acceptance region $y \le \rho(x)$ then we accept x, otherwise we discard x and repeat the process. The sequence of random numbers x generated by this algorithm has the desired distribution $\rho(x)$ [6] (Fig. 6).

Acceptance–Rejection Algorithm

Input Distribution $\rho(x)$ and its upper approximation $G(x)$.
Output Random number x from distribution $\rho(x)$.

Step 1 Generate random number x from G using alias method.
Step 2 Generate uniform random number y from $[0, G(x)]$ and evaluate $\rho(x)$.
Step 3 If $y \le \rho(x)$ then return x, else go to Step 1.

The efficiency of the acceptance–rejection approach (i.e., the probability of accepting x) depends on how accurate the upper approximation G is. In principle, one can use the global maximum of $\rho(x)$ as the (constant) upper approximation, but if the range of $\rho(x)$ is large, the method is extremely inefficient because of frequent rejections.

In order to build an accurate upper approximation, log-concave and T-concave distributions have been studied [6,10]. Since $\log(\rho(x))$ (or $T(\rho(x))$) is concave for such distributions, one can build a piecewise linear upper approximation using the classical cutting plane method. In this case, the upper approximation is $G(x) = \exp u(x)$,

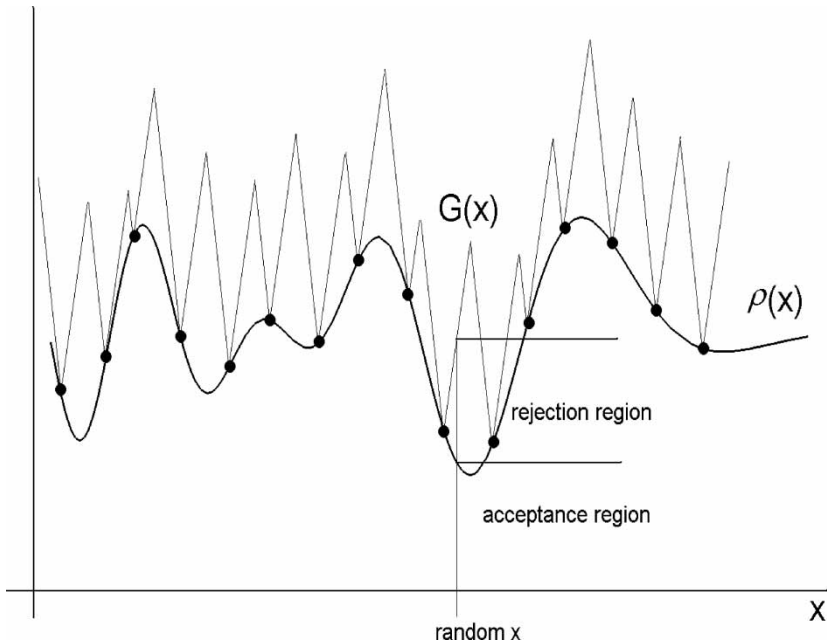

FIGURE 6 An upper approximation of a probability density function ρ (using Pijavski–Shubert method) and the acceptance–rejection method. A random x from distribution G is generated. This x is accepted if another random number y uniform on $[0, G(x)]$ falls into acceptance region; otherwise x is rejected.

where $u(x)$ is the upper piecewise linear approximation to the transformed (concave) probability density $\log(\rho(x))$.

However, for distributions that are not log-concave (or T-concave [10], where T is a monotone continuous function), building an accurate upper approximation was difficult. Only in one-dimensional case efficient algorithms (e.g., based on Pijavski–Shubert method) have been developed (see [6], p. 348). The multivariate distributions were traditionally generated by repeatedly using univariate generators.

Approximation $G^K(x)$ generated by CAM gives a way to generate random numbers from an IPH (and hence Lipschitz) multidimensional distribution directly. The density function $\rho(x)$ is reflected and modified with an appropriate constant for $G^K(x)$ to be an upper rather than lower approximation. Since regions S^r are given explicitly, generating random numbers with piecewise constant density $G^K(x)$ is rather simple (the algorithms are presented in [6]). Hence one can use the acceptance–rejection approach, which will be quite efficient, since for large K, $G^K(x)$ approximates $\rho(x)$ from above sufficiently tight.

6 CONCLUSION

This article re-examines the problem of locating minima of a lower approximation to the Lipschitz objective function f, called the saw-tooth underestimate. Problems of this type arise in some methods of deterministic global optimization that generalize the classical cutting plane method. The difficulty of locating the minima of the saw-tooth underestimate of multivariate functions stems from its very complicated geometry

and a great number of such minima. Slow convergence of global optimization algorithms translates into the necessity to take a large number of support functions, and consequently to solve this auxiliary problem many times. Even using standard linear solvers is not practically feasible.

In the Cutting Angle method, the saw-tooth underestimate has a peculiar structure, which has allowed us to enumerate all local minima explicitly, and to formulate and efficiently solve a combinatorial problem of hypercone intersections. In this article previous combinatorial approaches have been substantially improved by using a special data structure and by drastically reducing the number of tests needed to locate the local minima. Literally millions of local minima can be efficiently processed at every iteration. A closer examination of the geometry of the saw-tooth underestimate has lead to explicit characterization of sets, on which it is decomposed into simple linear segments. Connection to Voronoi diagrams has been established.

One important application of such explicit characterization comes from statistical simulation, where black-box random number generators are frequently needed. Acceptance–rejection approach has been extended to multimodal multivariate Lipschitz density functions, and computational efficiency of the proposed algorithms makes this method practical for such demanding problem as random number generation.

References

[1] M. Andramonov, A. Rubinov and B. Glover (1999). Cutting angle methods in global optimization. *Applied Mathematics Letters*, **12**(3), 95–100.

[2] A. Bagirov and A. Rubinov (2000). Global minimization of increasing positively homogeneous function over the unit simplex. *Annals of Operations Research*, **98**, 171–187.

[3] A. Bagirov and A. Rubinov (2001). Modified versions of the cutting angle method. In: N. Hadjisavvas and P.M. Pardalos (Ed.), *Convex Analysis and Global Optimization*, *of Nonconvex Optimization and its Applications*, Vol. 54, pp. 245–268. Kluwer, Dordrecht.

[4] L.M. Batten and G. Beliakov (2002). Fast algorithm for the cutting angle method of global optimization. *Journal of Global Optimization*, **24**, 149–161.

[5] J.-D. Boissonnat, M. Sharir, B. Tagansky and M. Yvinec (1998). Voronoi diagrams in higher dimensions under certain polyhedral distance functions. *Discrete and Comput. Geometry*, **19**, 485–519.

[6] L. Devroye (1986). *Non-uniform Random Variate Generation*. Springer Verlag, New York.

[7] C.A. Floudas (2000). *Deterministic Global Optimization*: *Theory*, *Methods and Applications*, *of Nonconvex Optimization and its Applications*, Vol. 37. Kluwer Academic Publishers, Dordrecht, London.

[8] W. Gilks, N.G. Best and K.K.C. Tan (1995). Adaptive rejection metropolis sampling. *Appl. Stat.*, **5**, 455–472.

[9] P. Hansen and B. Jaumard (1995). Lipschitz optimization. In: R. Horst and P. Pardalos (Eds.), *Handbook of Global Optimization*, pp. 407–493. Kluwer, Dordrecht.

[10] W. Hormann (1995). A rejection technique for sampling from t-concave distributions. *ACM Transactions on Mathematical Software*, **21**, 182–193.

[11] R. Horst, P. Pardalos and N. Thoai (2000). *Introduction to Global Optimization*, 2nd Edn. Kluwer Academic Publishers, Dordrecht.

[12] R. Horst and H. Tuy (1993). *Global Optimization*: *Deterministic Approaches*, 2nd Rev. Edn. Springer-Verlag, Berlin, New York.

[13] J.E. Kelley (1960). The cutting-plane method for solving convex programs. *J. of SIAM*, **8**, 703–712.

[14] K.F. Lim, G. Beliakov and L.M. Batten (2003). A new method for locating the global optimum: application of the cutting angle method to molecular structure prediction. In: *Proceedings of the 3rd International Conference on Computational Science*, Volume 4, Springer-Verlag Lecture Notes in Computer Science (LNCS). Springer-Verlag, Heidelberg, 1040–1049.

[15] R. Mladineo (1986). An algorithm for finding the global maximum of a multimodal, multivariate function. *Math. Progr.*, **34**, 188–200.

[16] A. Neumaier (1997). Molecular modeling of proteins and mathematical prediction of protein structure. *SIAM Review*, **39**(3), 407–460.

[17] S.A. Pijavski (1972). An algorithm for finding the absolute extremum of a function. *USSR Comput. Math. and Math. Phys.*, **2**, 57–67.
[18] J. Pintér (1996). *Global Optimization in Action*: *Continuous and Lipschitz Optimization–Algorithms*, *Implementations and Applications*, *of Nonconvex Optimization and its Applications*, Vol. 6. Kluwer Academic Publishers, Dordrecht, Boston.
[19] A.M. Rubinov (2000). *Abstract Convexity and Global Optimization*, *of Nonconvex Optimization and its Applications*, Vol. 44. Kluwer Academic Publishers, Dordrecht, Boston.
[20] B. Shubert (1972). A sequential method seeking the global maximum of a function. *SIAM J. Numer. Anal.*, **9**, 379–388.