Journal of Global Optimization **12:** 383–404, 1998.

A Multivariate Global Optimization Using Linear Bounding Functions

XIAOJUN WANG[1] and TSU-SHUAN CHANG[2]
[1]*Applied Mathematics Group, Department of Mathematics, University of California, Davis, CA 95616, USA;* [2]*Department of Electrical and Computer Engineering, University of California, Davis, CA 95616, USA*

(Received 27 June 1995; accepted 2 September 1997)

Abstract. Recently linear bounding functions (LBFs) were proposed and used to find ϵ-global minima. This paper presents an LBF-based algorithm for multivariate global optimization problems. The algorithm consists of three phases. In the global phase, big subregions not containing a solution are quickly eliminated and those which possibly contain the solution are detected. An efficient scheme for the local phase is developed using our previous local minimization algorithm, which is globally convergent with superlinear/quadratic rate and does not require evaluation of gradients and Hessian matrices. To ensure that the found minimizers are indeed the global solutions or save computation effort, a third phase called the verification phase is often needed. Under adequate conditions the algorithm finds the ϵ-global solution(s) within finite steps. Numerical testing results illustrate how the algorithm works, and demonstrate its potential and feasibility.

Key words: Factorable functions, Global optimization, Linear bounding functions.

1. Introduction

In recent years, global optimization has become one of the most interesting research areas in optimization, since many real world problems are global rather than local [5, 11, 17, 22]. Unlike local optimization, which has a rich theory and for which many excellent numerical methods are available, global optimization has only been partially researched. The enormous difficulties are due to the intrinsic multiextremality of the formulation. Although properties such as smoothness or continuity are often presented in global optimization problems, standard nonlinear programming techniques have not been successful for solving these problems since they can only be used to find a local minimizer instead of a global one. There is no local criterion for deciding whether a local solution is global or not. On the other hand, most criteria proposed for a global minimizer [11, 21] are not very practical for solving general global optimization problems. For these reasons global optimization methods are significantly different from standard nonlinear programming techniques, and they are, in general, much more expensive computationally.

One class of deterministic approaches to global optimization is called covering methods [11, 22]. It throws away subregions not containing global minima until the remaining set is small enough and is known to contain global minimizers.

Many methods have been developed for Lipschitz functions (e.g. [8, 12, 15, 25]). A natural approach proposed by Pijavskij [16] and Shubert [20] is a sequential algorithm which iteratively constructs piecewise linear underestimating functions for an objective function with a known Liptschitz constant, and evaluates the objective function at a point corresponding to a minimum of this bounding function. From this point, the bounding function is split into two higher piecewise functions.

A widely used method is called branch and bound, and many algorithms can be put into its framework [10, 11]. The basic idea of branch and bound methods is to generate sequences of nonincreasing upper bounds and nondecreasing lower bounds for an objective function by successfully refining partitions of feasible regions until they approach each other closely enough. Several methods can be applied to obtain upper and lower bounds of objective functions. For instance, interval analysis can be used [6–9, 18]. The convex underestimating functions over convex sets can be constructed iteratively for factorable function [14].

Recently the concept of linear bounding functions (LBFs) was introduced to develop global optimization algorithms [1, 2, 4, 23].* For univariate problems, a linear lower (upper) bounding function (LLBF/LUBF) of a given objective function $f(x)$ on an interval $[a, b]$ is a linear underestimating (overestimating) function of $f(x)$ whose function value at one end point matches that of the given function. Not only can LBFs be constructed for a large class of functions such as factorable functions, but also they can be easily used to throw away subregions not containing ϵ-global minima. These advantages make the LBF method a promising approach for global optimization problems.

The LBF-based algorithm for univariate global optimization in [23] consists of two phases. In the global phase, it has the capability to quickly eliminate bigger regions which do not contain a solution. In the local phase, it performs a finer search over the remaining smaller regions which possibly contain a solution. To find an efficient scheme for n-dimensional cases, we went back to re-examine local optimization problems and discovered that the use of LBFs enables us to develop a new framework of (local) optimization. From this framework we can develop an optimization algorithm which is not only globally convergent with superlinear/quadratic rate, but also does not require function and Hessian evaluations [24]. Usually, two-phase global optimization algorithms work well for univariate problems. For multivariate problems, however, two phases are probably not enough. To ensure that the found minimizers are indeed the global solutions or save computation effort, we often need a third phase, called the verification phase. Unavoidably, more information about the problem is needed in this phase.

The paper is organized as follows. In Section 2, the definition and properties of LBFs are presented. In Section 3, the algorithm for the global phase is developed. In Section 4, we review the new framework of local optimization and the algorithm for the local phase is developed. In Section 5, the algorithm of the verification phase

* An LBF was called a linear bound in [1, 2, 4]. Since an LBF is actually a function, its new name is more adequate.

is developed. The global optimization algorithm and its convergence are given in Section 6. Numerical testing results are presented in Section 7. In Section 8, a discussion is given.

2. Linear bounding functions

2.1. LINEAR BOUNDING FUNCTIONS (LBFS)

Let $f(x) : R^n \to R$ be a continuous function, and D an n-dimensional box in R^n, i.e.,

$$D = \{x \in R^n \mid A(i) \leqslant x(i) \leqslant B(i), A(i) < B(i), i = 1, \ldots, n\}, \tag{2.1}$$

where $A = [A(1), \cdots, A(n)]^T$ is the 'lower left' vertex of D, $B = [B(1), \ldots, B(n)]^T$ the 'upper right' vertex of D.

Let x_0 be an vertex of D. A linear function

$$l(x) = m_f^T(x - x_0) + f(x_0) \tag{2.2}$$

is a linear lower bounding function (LLBF) of $f(x)$ over the box D with the matching point x_0 [23] if

$$f(x) \geqslant l(x), \quad \forall x \in D. \tag{2.3}$$

A linear upper bounding function (LUBF) is similarly defined by replacing $\geqslant$ in (2.3) by $\leqslant$. We will use linear bounding functions (LBFs) to refer either or both of LLBFs and LUBFs.

For a function $F : R^n \to R^p$, $F(x) = [f_1(x), \ldots, f_p(x)]^T$, the function

$$\begin{aligned} l(x) &= [l_1(x), \ldots, l_p(x)]^T \\ &= F(x_0) + m_F^T(x - x_0), \end{aligned} \tag{2.4}$$

where x_0 is an extreme of D and m_F is an $n \times p$ matrix, is called an LLBF of $F(x)$ over the box D if each $l_i(x), i = 1, \ldots, p$ is an LLBF of $f_i(x)$ on D. An LUBF is defined similarly.

Although it seems very difficult to obtain LBFs for any given function, we have presented in [23] a procedure to construct LBFs for factorable functions over a specified box. The procedure provides an LLBF, an LUBF, estimations of min and max over the box and its function value at x_0.

A generic factorable function, $f : R^n \to R$, has the form

$$f(x) = T(t(x)) + U(u(x)) \cdot V(v(x)), \tag{2.5}$$

where $t(\cdot), u(\cdot)$ and $v(\cdot)$ are continuous functions of n variables and $T(\cdot), U(\cdot)$ and $V(\cdot)$ are continuous functions of a single variable. A general factorable function $f_N(x)$ can be constructed recursively by using the generic functions and is expressed as

$$f_N(x) \equiv \sum_{p=1}^{N-1} T_{N,p}(f_p(x)) + \sum_{p=1}^{N-1} \sum_{q=1}^{p} U_{N,p,q}(f_q(x)) \cdot V_{N,q,p}(f_p(x)), \tag{2.6}$$

where

$$f_j(x) \equiv x_j,\ j = 1, \ldots, n, \tag{2.7a}$$

$$f_j(x) \equiv \sum_{p=1}^{j-1} T_{j,p}(f_p(x)) + \sum_{p=1}^{j-1} U_{j,p,q}(f_q(x)) \cdot V_{j,q,p}(f_p(x)), \tag{2.7b}$$

$$j = n+1, \cdots, N-1, \tag{2.7c}$$

and the T's, U's and V's are scalar functions of one variable. As mentioned in [13], many functions belong to this large class of factorable functions.

In [23], the construction of an LBF for a general factorable function has two parts. The first part is to construct LBFs for commonly used 1-D functions. The second part generates LBFs for those functions with function forms such as the minimum or maximum of several functions, summation, composition or production. From these two parts, the LBFs for factorable functions can be recursively constructed just like the function itself.

In this paper, the procedure to construct LBFs for factorable functions is used in our numerical examples. However, the results hold for any functions, as long as their LBFs can be obtained and have the same important properties as those presented in [24] for factorable functions. Some of these important properties are summarized in the next two subsections.

2.2. PROPERTIES OF LBFS

Assume that $f : R^n \to R$ is twice continuously differentiable. An LBF $l(x)$ (either an LLBF or an LUBF) of $f(x)$ on D is given by (2.4). Note that the $n \times p$ matrix $m_F = m_F(D, x_0)$ is a function of D and x_0.

Define

$$\begin{aligned} D + \Delta D \ \stackrel{\Delta}{=}\ & \{x \in R^n \mid A(i) + \Delta A(i) \leqslant x(i) \\ & \leqslant\ B(i) + \Delta B(j), j = 1, \ldots, n\}. \end{aligned} \tag{2.8}$$

PROPOSITION 2.1. *$\|m_F(D, x_0)\|$ is finite if width$(D) \stackrel{\Delta}{=} \max_i\{B(i) - A(i)\}$ is finite.*

PROPOSITION 2.2. *$m_F = m_F(D, x_0)$ is a continuous function of D and x_0. In other words, it satisfies*

(i) for any given $\epsilon > 0$, there exists a $\delta > 0$ such that

$$\|m_F(D + \Delta D, x_0') - m_F(D, x_0)\| < \epsilon, \tag{2.9}$$

whenever

$$\left\{\sum_i \left(|\Delta A(i)| + |\Delta B(i)|\right)^2\right\}^{1/2} < \delta \ \text{ and } \ \|x_0' - x_0\| < \delta; \tag{2.10}$$

(ii) if D contains only one point, i.e., $D = \{x_0\}$, then

$$m_F(D, x_0) = \nabla F(x_0). \tag{2.11}$$

For any $x^* \in R^n$, box$_i \subset R^n$, let $(m_F)_i$ be associated with an LBF in the form of (2.4) on box$_i$ with a vertex x_i. If $x_i \to x^*$ and width(box$_i$) $\to 0$ as $i \to \infty$, then according to the above propositions, $\|(m_F)_i - \nabla F(x^*)\| \leqslant \|(m_F)_i - \nabla F(x_i)\| + \|\nabla F(x_i) - \nabla F(x^*)\| \to 0$ as $i \to \infty$.

2.3. QUADRATIC BOUNDING FUNCTIONS OF $f(x)$

In the LBF method we also need the concept of quadratic bounding functions (QBFs). The quadratic functions $Q(x)$ and $q(x)$ are called a quadratic upper (lower) bounding function (QUBF/QLBF) if they satisfy

$$\begin{aligned} f(x) \leqslant Q(x) &\triangleq f(x_0) + g(x_0)^T(x - x_0) \\ &+ \frac{1}{2}(x - x_0)^T M(x - x_0), \quad \forall x \in D, \end{aligned} \tag{2.12}$$

$$\begin{aligned} f(x) \geqslant q(x) &\triangleq f(x_0) + g(x_0)^T(x - x_0) \\ &+ \frac{1}{2}(x - x_0)^T m(x - x_0), \quad \forall x \in D, \end{aligned} \tag{2.13}$$

where $g(x) \triangleq \nabla f(x)$ and $x_0 \in D$. By using the LBFs of $g(x)$ we can generate two sets of symmetric matrices, $\mathcal{M}_{\mathrm{M}}$ and $\mathcal{M}_{\mathrm{m}}$, each $M \in \mathcal{M}_{\mathrm{M}}$ or $m \in \mathcal{M}_{\mathrm{m}}$ gives us a QUBF or a QLBF.

THEOREM 2.1 (the property of QBFs). *For each $M \in \mathcal{M}_M$ or $m \in \mathcal{M}_m$,*

(i) we have quadratic upper and lower bounding functions $Q(x)$ and $q(x)$, i.e.,

$$q(x) \leqslant f(x) \leqslant Q(x), \quad \forall x \in D, \tag{2.14}$$

where $Q(x)$ and $q(x)$ are determined by (2.12) and (2.13);

(ii) $M = M(D, x_0) \to H(x_0)$, $m = m(D, x_0) \to H(x_0)$ as width$(D) \to 0$, where $H(x)$ is the Hessian of $f(x)$;

(iii)

$$0 \leqslant f(x) - q(x) = O(\|x - x_0\|^3), \tag{2.15}$$

$$0 \leqslant Q(x) - f(x) = O(\|x - x_0\|^3). \tag{2.16}$$

Theorem 2.1 enables us to develop a globally convergent optimization algorithm with superlinear/quadratic rate.

3. Phase I: detecting subregions which possibly contain the solution(s)

Consider an n-dimensional global optimization problem

$$\min_{x \in S} f(x), \tag{3.1}$$

where $f : R^n \to R$ is continuously differentiable and

$$S = \{x \in R^n \mid A_s(i) \leqslant x(i) \leqslant B_s(i),\ i = 1, \ldots, n\} \tag{3.2}$$

is a box in R^n. As mentioned in Section 1, an LBF-based global optimization algorithm consists of three phases. In this and the following sections we will discuss each phase and develop an LBF-based global optimization algorithm.

The purpose of Phase I is to detect those subregions which possibly contain solutions. In the meantime, we hope that in this phase large subregions which do not contain the solutions can be quickly removed.

The idea for Phase I is rather simple. Assume that we have a set of subregions and the LBFs of f or ∇f over each subregion are available. Thanks to these LBFs, those subregions assured not to contain a solution will be deleted. To improve the estimation of the solution(s) and get rid of more subregions which do not contain a solution, we should refine the remaining region and obtain LBFs over the refined subregions. A subregion is saved for the next phase if it is considered to possibly contain a solution. The procedure is repeated until all subregions are either removed or saved for the local phase. Thus the basic operations of Phase I are decomposition, LBF generating, shrinking and region storing (for Phase II). To get more detail, let us assume that $\mathcal{P}$ is a collection of subboxes of S and $D \in \mathcal{P}$.

(1) *Decomposition.* Decomposition strategy used in the global phase of the algorithm for univariate global optimization [24] cannot be simply extended to n-dimensional cases, since it is not easy to determine a decomposition by the intersection of two n-dimensional hyperplanes. Instead we simply use bisection along the longest edges of D.

(2) *LBF generating.* For the current considered subbox we improve the estimations of f or/and ∇f by generating their LBFs. In the LBF methods we can choose different types of LBFs to use. We can choose LBFs of f (e.g. in the global phase of Algorithm 3.2 in [23]), LBFs of ∇f (e.g. in local optimization Algorithm 4.1 in [24]), LBFs of f and ∇f (e.g. in the local phase of Algorithm 3.2 in [23]), or even LBFs of the Hessian as long as they can be constructed. Our numerical experience indicates that it is better to use LBFs of f and ∇f alternatively in Phase I for n-dimensional problems, in contrast with Phase I in the univariate case, where only the LBFs of f are needed. Besides, LBFs of ∇f will be needed in the criterion when a subregion has to be stored. As mentioned before, our procedure for constructing LBFs of a factorable function provides LBFs, bounds (estimations of min and max) and the function value at one vertex of the specific box.

(3) *Shrinking.* Shrinking the currently considered box by LBFs is crucial for quickly detecting the subregions which possibly contain the solutions. Assume that $\bar{f}$ is the

best function value we have so far and that LBFs of f and $g = \nabla f$ over the box D are available:

$$l_f(x) \leqslant f(x) \leqslant L_f(x), \tag{3.3}$$

$$\alpha_f \leqslant f(x) \leqslant \beta_f, \tag{3.4}$$

where

$$l_f(x) = f(x_0) + m_f^T(x - x_0), \tag{3.5}$$

$$L_f(x) = f(x_0) + M_f^T(x - x_0), \tag{3.6}$$

and

$$l_g(x) \leqslant g(x) \leqslant L_g(x), \tag{3.7}$$

$$\alpha_g \leqslant g(x) \leqslant \beta_g, \tag{3.8}$$

where

$$l_g(x) = g(x_0) + m_g(x - x_0), \tag{3.9}$$

$$L_g(x) = g(x_0) + M_g(x - x_0). \tag{3.10}$$

Then the box D can be shrunk either by (3.3) to (3.6) or (3.7) to (3.10).

(i) Shrinking by LBFs of f. Obviously we can remove the whole D if $\alpha_f > \bar{f}$. If $\alpha_f \leqslant \bar{f}$, D cannot be removed but it can possibly be shrunk by the LLBF of f. Although $\{x \in D \mid l_f(x) \geqslant \bar{f}\}$ is the largest subregion we can remove from D by $l_f(x)$ and $\bar{f}$, we will not delete the whole subregion, since typically such a subregion is not a box and its deletion will necessitate additional box decomposition and storage in order to continue the procedure. To avoid additional box decomposition, we instead remove only a subbox from the set such that the remaining subregion of D is still a box. For each $1 \leqslant i \leqslant n$, we have

$$\begin{aligned} l_f(x) &= m_f^T x + r \\ &\geqslant m_f(i)x(i) + s + r, \end{aligned} \tag{3.11}$$

where

$$r = f(x_0) - m_f^T x_0, \tag{3.12}$$

$$s = \min_{x \in D} \sum_{j \neq i} m_f(j)x(j). \tag{3.13}$$

If $m_f(i) \neq 0$, let

$$z_i = \frac{\bar{f} - s - r}{m_f(i)}. \tag{3.14}$$

If $m_f(i) > 0$ and $z_i < B(i)$, then for any point of

$$\bar{D}_i = \{x \mid A(i) \leqslant x(j) \leqslant B(j), j = 1, \ldots, n, j \neq i, z_i \leqslant x(i) \leqslant B(i)\}, \tag{3.15}$$

$$f(x) \geqslant l_f(x) \geqslant m_f(i)x(i) + s + r \geqslant \bar{f}, \tag{3.16}$$

and $\bar{D}_i$ can be discarded. Similarly, if $m_f(i) < 0$ and $z_i > A(i)$, then

$$\hat{D}_i = \{x \mid A(i) \leqslant x(j) \leqslant B(j), j = 1, \ldots, n, j \neq i, A(i) \leqslant x(i) \leqslant z_i\} \tag{3.17}$$

can be removed.

(ii) Shrinking by LBFs of $g(x) = \nabla f$. In a way similar to the use of interval analysis for global optimization [7, 18], a box can be shrunk by LBFs of ∇f. If $\alpha_g(i)\beta_g(i) > 0$ for some $1 \leqslant i \leqslant n$, then $\partial f/\partial x_i \neq 0$ for any $x \in D$, the whole D can be deleted. Assume $\alpha_g(i)\beta_g(i) \leqslant 0$ for all $i = 1, \ldots, n$. The box can be shrunk by the LBFs of ∇f. For each $1 \leqslant i, j \leqslant n$, the ith term of $l_g(x)$

$$\begin{aligned} l_g^i(x) &= (m_g^i)^T x + r^i \\ &= m_g^i(j)x(j) + s^i + r^i, \end{aligned} \tag{3.18}$$

where m_g^i is the ith row of m_g and

$$r^i = g^i(x_0) - m_g^i x_0, \tag{3.19}$$

$$s^i = \min_{x \in D} \sum_{k \neq j} m_g^i(k)x(k). \tag{3.20}$$

If $m_g^i(j) \neq 0$, let

$$z_{ij} = -\frac{s^i + r^i}{m_g^i(j)}. \tag{3.21}$$

If $m_g^i(j) > 0$ and $z_{ij} < B(j)$, then we can discard a subbox

$$\begin{aligned} \bar{D}_{ij} &= \{x \mid A(k) \leqslant x(k) \leqslant B(k), k = 1, \ldots, n, k \neq j, \\ z_{ij} &\leqslant x(j) \leqslant B(j)\} \end{aligned} \tag{3.22}$$

since for any $x \in \bar{D}_{ij}$

$$g^i(x) \geqslant l_g^i(x) \geqslant 0. \tag{3.23}$$

If $m_g^i(j) < 0$ and $z_{ij} > A(j)$, then the subbox

$$\begin{aligned} \hat{D}_{ij} \;&=\; \{x \mid A(k) \leqslant x(k) \leqslant B(k), k = 1, \ldots, n, k \neq j, \\ A(j) \;&\leqslant\; x(j) \leqslant z_{ij}\} \end{aligned} \tag{3.24}$$

can be discarded. Similarly, we can use $L_g(x)$ to delete subboxes in which

$$g^i(x) \leqslant L_g^i(x) \leqslant 0, \quad 1 \leqslant i \leqslant n. \tag{3.25}$$

(4) Box storing (for Phase II). Phase I stops when all the remaining subboxes possibly contain solutions. What do we mean by 'possibly containing a solution'? Recall that LBFs of ∇f provide us a quadratic lower bounding function (QLBF) as expressed in (2.13)

$$f(x) \geqslant q(x), \quad \forall x \in D, \tag{3.26}$$

where

$$q(x) = f(x_0) + g(x_0)^T(x - x_0) + \tfrac{1}{2}(x - x_0)^T m(x - x_0). \tag{3.27}$$

Since $f(x) = f(x_0) + g(x_0)^T(x - x_0) + \frac{1}{2}(x - x_0)^T H(\xi)(x - x_0)$, we have

$$(x - x_0)^T(H(\xi) - m)(x - x_0) \geqslant 0, \quad \forall x \in D. \tag{3.28}$$

According to the continuity of LBFs, $\|m - H(x)\| \to 0$ as width$(D) \to 0$. Thus $m > 0$ (positive definite) implies $H(x) > 0$ for all $x \in D$ if width(D) is small enough. We do not know how small width(D) should be, i.e., we cannot guarantee that the function is convex if $m > 0$, but it is reasonable to think that the box 'possibly contains a solution (more precisely, a local minimizer)' if $m > 0$.

The other condition of storing a subbox for the next phase is that width$(D) \leqslant \delta_0, 1 > \delta_0 \geqslant 0$. This means that when the subregion is small enough, we would like to get into the next phase.

Let $\mathcal{P}$ and $\mathcal{Q}$ be the collections of subboxes of S. Initially, $\mathcal{P}$ is a finite partition of S and $\mathcal{Q}$ is empty. Denote $\alpha_i = est \min_{D_i} f$, where $D_i \in \mathcal{P}$. Let $\bar{f}$ be the best function value we have so far. The algorithm of Phase I is the following.

Phase I (the global phase, detecting subboxes which possibly contain the solution(s))

While $P \neq \emptyset$, do:

Step 1. Choose a box $D \in \mathcal{P}$ with $\alpha = \min\{\alpha_1, \alpha_2, \cdots\}$ and get the stored LBFs of f and g.

Step 2. If $\alpha \geqslant \bar{f} + \epsilon$, go to Step 1; else go to Step 3.

Step 3. Bisect D, for each subbox D_j $(j = 1, 2)$, do:

(1) Use the stored LBFs of f and g to shrink the box (still denote the remaining box by D_j). If $D_j = \emptyset$, go to (5); else go to (2).
(2) Obtain the LBFs of $g(x)$, use them to shrink the box. If $D_j = \emptyset$, go to (5); else go to (3).
(3) If $m > 0$ or width$(D_j) \leqslant \delta_0$, store D_j into $\mathcal{Q}$ and save the corresponding LBFs of f and g; else go to (4).
(4) Obtain the LLBF of f, update $\bar{f}$, use it to shrink the box. If $D_j = \emptyset$, go to (5); else store D_j into $\mathcal{P}$ and save the corresponding LBFs of f and g.
(5) If both D_1 and D_2 are done, go to Step 4; else go to (1).

Step 4. If $\mathcal{P} \neq \emptyset$, go to Step 1; else Phase I is finished.

After Phase I we have a collection $\mathcal{Q}$ of remaining subboxes. Each element of $\mathcal{Q}$ has the possibility of containing a (local) minimizer. Note that the final $\bar{f}$ at the end of Phase I is typically lower than those $\bar{f}$'s at the time such subregions were stored. By using the lower value $\bar{f}$ and the stored LBFs, we expect that the stored subregions can be further shrunk or simply removed. Thus to save local search in the next phase, we should use the final $\bar{f}$ and the stored or new generated LBFs to further shrink or eliminate such small regions. There is another problem we should deal with to save the computation in local optimization. The collection $\mathcal{Q}$ often contains several subboxes which are adjacent to each other and relate to the same minimizer – i.e., they are sort of in the same basin of a local minimizer. Obviously, it will be a waste if we perform local optimization for each of these boxes. Instead we do local search once in the box containing the union of these adjacent boxes.

Therefore, we have a phase after Phase I and before the next local optimization phase; we call it Pre-Phase II.

Pre-Phase II (preparation of Phase II)

Initial data: the collection $\mathcal{Q}$, the collection temp$\mathcal{Q} = \emptyset$.

Step 1. Choose a box $D \in \mathcal{Q}$ with $\alpha = \min\{\alpha_1, \alpha_2, \ldots\}$ and get the corresponding LBFs of f and g.

Step 2. If $\alpha \geqslant \bar{f} + \epsilon$ go to Step 1; else go to Step 3.

Step 3. Obtain LBFs of f on the box, update $\bar{f}$ and α, use the LLBF to shrink D. If $D = \emptyset$, go to Step 1; else go to Step 4.

Step 4. For each $D_i \in \mathcal{Q} - \mathcal{D}$, if $D_i \bigcap D \neq \emptyset$, generate a box $\hat{D} \supset D_i \bigcup D$, store $\hat{D}$ into temp$\mathcal{Q}$. If $\mathcal{Q} \neq \emptyset$ go to Step 1; else go to Step 5.

Step 5. $\mathcal{Q} \leftarrow$ temp$\mathcal{Q}$.

4. Phase II: finding minimizers

Now after Phase I and Pre-Phase II, we are almost assured that each box in $\mathcal{Q}$ contains a global minimizer (still no guarantee of course). In the next phase, we need to perform a finer search over the remaining regions. To develop an efficient scheme for the local phase, let us go back to re-examine local optimization problems.

4.1. A FRAMEWORK FOR LOCAL MINIMIZATION

Consider an unconstrained optimization problem

$$\min_{R^n} f(x), \tag{4.1}$$

where $f(x) : R^n \to R$ is a continuously differentiable function. In unconstrained optimization problems we usually call a point x_1 a better point than x_0 if $f(x_1) < f(x_0)$. One class of methods used most often in optimization is to find a better point in each iteration. That is, given an initial point x_0, a better point x_1 is determined by

$$x_1 = x_0 - c\,h, \tag{4.2}$$

where h is a ascent direction of f, such as the steepest ascent direction $h = g(x_0)$ or the Newton's direction $h = H(x_0)^{-1}g(x_0)$ when $H(x_0)$ is positive definite. Since we do not know how far we can go along the direction, the step size c has to be determined by line search methods such as Armijo rule, golden section, etc. The calculation of c usually involves many function evaluations. Now we introduce a new framework [24] to find a better point without using a line search, and thus it is function evaluation free.

Let D be an n-dimensional box in R^n, i.e.,

$$D = \{x \in R^n \mid A(i) \leqslant x(i) \leqslant B(i), i = 1, \ldots, n\}. \tag{4.3}$$

Assume that

$$\begin{aligned} f(x) \leqslant\ & Q(x) \stackrel{\Delta}{=} f(x_0) + g(x_0)^T(x - x_0) \\ & + \tfrac{1}{2}(x - x_0)^T M(x - x_0), \quad \forall x \in D, \end{aligned} \tag{4.4}$$

where $x_0 \in D$, $g(x) \stackrel{\Delta}{=} \nabla f(x)$ and M is an $n \times n$ symmetric matrix.

THEOREM 4.1 (finding a better point along the gradient direction). *Let* $x_1 = x_0 - c\,g(x_0)$, *where*

$$c = \min\{1,\ c_1,\ c_2\}, \tag{4.5}$$

$$c_1 = \max\{c \geqslant 0 \mid x_0 - c\,g(x_0) \in D\}, \tag{4.6}$$

$$c_2 = \begin{cases} (2-\epsilon)\min_{i\in I^+}\left\{\frac{1}{M(i,i)+\sum_{j\neq i}|M(i,j)|}\right\}, & I^+ \neq \emptyset \\ \infty, & I^+ = \emptyset, \end{cases} \tag{4.7}$$

$$I^+ = \left\{ 1 \leqslant i \leqslant n \;\middle|\; M(i,i) + \sum_{j\neq i} |M(i,j)| > 0 \right\}, \tag{4.8}$$

$0 < \epsilon < 1$ *is a small number. Then* $f(x_1) \leqslant f(x_0)$. *If* $x_1 \neq x_0$, *then* $f(x_1) < f(x_0)$.

THEOREM 4.2 (finding a better point using a quadratic upper bounding function). *Suppose* $M > 0$. *Let* $x_1 = x_0 - c\,M^{-1}g(x_0)$, *where*

$$c = \max\{2-\epsilon \geqslant c \geqslant 0 \mid x_0 - c\,M^{-1}g(x_0) \in D\}, \tag{4.9}$$

$0 < \epsilon < 1$ *is a small number. Then* $f(x_1) \leqslant f(x_0)$. *If* $x_1 \neq x_0$, *then* $f(x_1) < f(x_0)$.

From these theorems in [24], we can see that it is possible to find a better point without using a line search, and thus without any function evaluation for various situations.

To use the framework, we need to construct the quadratic upper bounding function (QUBF) in (4.4). By the results in Section 2, a QUBF can be obtained by using the LBFs of the gradient function. On the basis of the framework, we have developed a globally convergent local optimization algorithm which has the following properties. First, no line search along the negative gradient direction is needed for global convergence, because the LBFs offer regional information. This enables us to obtain a function evaluation free algorithm. Second, the superlinear/quadratic rate can be achieved without calculating Hessians explicitly. Third, the switching policy from global convergence to superlinear/quadratic rate is derived in a natural manner and has fairly small computation requirement.

4.2. A GLOBALLY CONVERGENT SUPERLINEAR/QUADRATIC RATE ALGORITHM

Denote the jth component of the increment vector dx by

$$\mathrm{d}x(j) = B(j) - A(j), \quad j = 1, \ldots, n, \tag{4.10}$$

where A is the 'lower left' vertex and B the 'upper right' vertex of a box D. In the algorithm, we use x_i to represent the current iterate, which is one of the vertices of the box D_i; we use the notation $g_i = g(x_i) = \nabla f(x_i)$; and all subscript i's indicate the ith iteration.

ALGORITHM 4.1.

Data: Initial x_0, $\mathrm{d}x_0$, g_0 and $h_{-1} \stackrel{\Delta}{=} g_0$.

General steps (the ith iteration, $i = 1, 2, \ldots$):

Step 1. Generate box$_i$ from x_i, dx_i, h_{i-1}.

Step 2. Generate M_i, α_i, β_i on box$_i$ and g_i.

($\alpha_i \leqslant g(x) \leqslant \beta_i$ over box$_i$.)

Step 3. Determine c_i and h_i.

(If $M_i > 0, h_i = M_i^{-1} g_i$; otherwise $h_i = g_i$.)

Step 4. Update x_i to get the next point x_{i+1}.

Step 5. Update dx_i to get the increment vector dx_{i+1}.

In Step 1, box$_i$ is determined by the available increment vector dx_i and the ascent direction of the last iteration h_{i-1}:

$$A_i(j) = \begin{cases} x_i(j) - \mathrm{d}x_i(j), & h_{i-1}(j) > 0 \\ x_i(j), & h_{i-1}(j) \leqslant 0 \end{cases} \tag{4.11}$$

$$B_i(j) = \begin{cases} x_i(j), & h_{i-1}(j) > 0 \\ x_i(j) + \mathrm{d}x_i(j), & h_{i-1}(j) \leqslant 0 \end{cases} \tag{4.12}$$

$j = 1, \ldots, n.$

In Step 3,

$$h_i = \begin{cases} M_i^{-1}, & M_i > 0 \text{ and } \|M_i\| \geqslant \delta_0 > 0 \\ g_i, & \text{otherwise} \end{cases} \tag{4.13}$$

c_i is determined by (4.5) to (4.8).

In Step 5, dx_i is updated as follows:

$$\mathrm{d}x_{i+1}(j) = \begin{cases} \max\{\, K\, \mathrm{d}x_i(j), \gamma_1 \max_k |h_i(k)| \,\}, & |x_{i+1}(j) - x_i(j)| = \mathrm{d}x_i(j) \\ \max\{\gamma_0\, \mathrm{d}x_i(j), \gamma_1 \max_k |h_i(k)| \,\}, & |x_{i+1}(j) - x_i(j)| = 0 \\ \max\{\, |x_{i+1}(j) - x_i(j)|, \gamma_1 \max_k |h_i(k)| \,\}, & 0 < |x_{i+1}(j) - x_i(j)| < \mathrm{d}x_i(j) \end{cases} \tag{4.14}$$

$j = 1, \ldots, n.$

The details of Algorithm 4.1 are given in [24]. According to the algorithm, the direction h_i will be naturally changed from the gradient direction to the Newton's-like direction $h_i = M_i^{-1} g_i$ when x_i is close to a local minimizer ($M_i \approx H(x_i)$ as x_i is close to the minimizer). Its typical behavior can be observed from Figure 4.1. For the Branin function [22] the algorithm starts from an arbitrary chosen initial point $x_0 = [8,\ 12]^T$ with an arbitrary chosen initial increment vector $\mathrm{d}x_0 = [0.5,\ 0.5]^T$ and the initial box is formed along the negative gradient direction. Before the further iteration, the largest step size is taken within the box and the box is enlarged twice along both axes. When reaching x_3, the next box is enlarged in one axis and

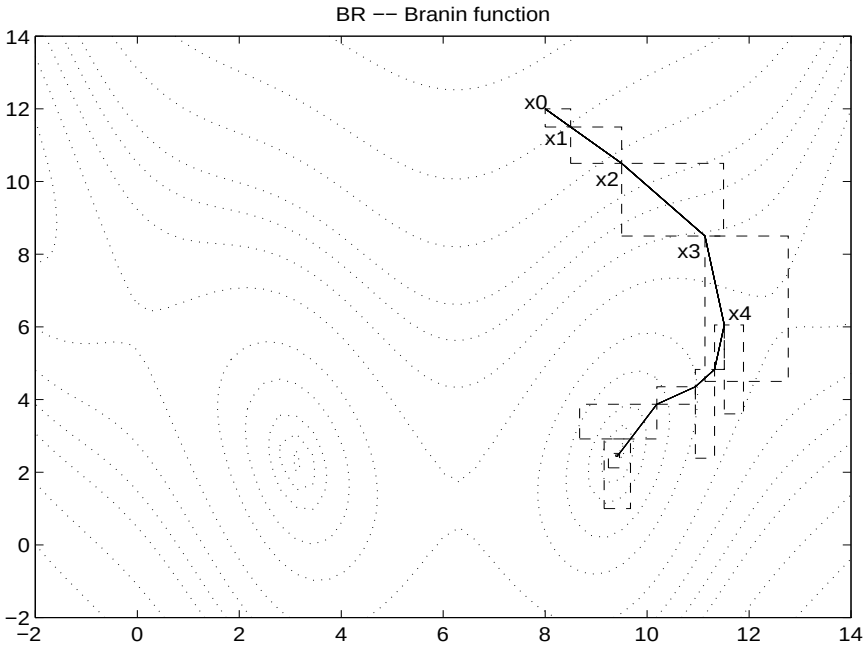

Figure 4.1. A typical trajectory from $x0 = 8[\,12]$.

reduced to the actual step taken in the other axis. At x_4 the box formed along the previous search direction is no longer lined up with the new search direction. A new box is generated for this and its size is shrunk in both axes. The sequence $\{x_i\}$ converges to the minimizer $x^* = [9.4248,\ 2.4750]^T$. The algorithm converges in 14 iterations with the total number of gradient and LBF evaluations equal to 15 (g_0 is needed to start).

THEOREM 4.3 (global convergence). *Assume that $f : R^n \to R$ is twice continuously differentiable, $g \stackrel{\Delta}{=} \nabla f$, and $\{x_i\}$ is an infinite sequence constructed by Algorithm 4.1. Also assume that $\{x_i\}$ is a bounded sequence, $x_{i+1} = x_i - c_i h_i$ and $x_{i+1} \neq x_i$ for all i's. Then there exists an accumulation point $x^* \in R^n$ such that*
(1) $g(x^*) = 0$;
(2) $x_i \to x^*$ as $i \to \infty$;
(3) $dx_i \to 0$ as $i \to \infty$.

THEOREM 4.4 (superlinear convergence with root rate $r = 1.618$). *Assume that $x_i \to x^* \in \arg\min\{f(x)\}$, $H(x^*) > 0$, and $\|H(x^*)\| \geqslant \rho > 0$. Then $x_i \to x^*$ with root rate $r = 1.618$.*

THEOREM 4.5 (quadratic convergence). *Suppose that $x_i \to x^* \in \arg\min\{f(x)\}$, $H(x^*) > 0$ and $\|H(x^*)\| \geqslant \rho > 0$. Then $x_i \to x^*$ quadratically.*

Now Algorithm 4.1 can be used for each subbox in the local phase of solving a global optimization problem. The only difference is that we need to keep the trajectory inside the box (constrained by the box) – i.e., for the current box $D = \{x \mid A(i) \leqslant x(i) \leqslant B(i), i = 1, \ldots, n\}$ the increment vector $\mathrm{d}x_i$ is determined by

$$\mathrm{d}x_i(j) = \min\{\mathrm{d}\tilde{x}_i(j), x_i(j) - A(j), B(j) - x_i(j)\}, \quad j = 1, \ldots, n, \quad (4.15)$$

where $d\tilde{x}_i$ is determined in [24]. In [24] we showed that $\mathrm{d}x_i(j) > 0$ for $j = 1, \ldots, n$ and all i's. However, $\mathrm{d}x_i(j) = 0$ might occur for (4.15) since x_i may reach the boundary. If that happens the algorithm stops at a (constrained) local minimizer on the boundary of the box. (In general, it is not a global one.) It is reasonable to choose the center of the box as the starting point.

Phase II (finding minimizers)

Step 1. For each box $D \in \mathcal{Q}$, set $x_0 =$ the center of D.

Step 2. Use Algorithm 4 to find a minimizer x^* in D.

Step 3. Store x^* and $f(x^*)$ in the sets MINPT and MINVAL.

5. Phase III: verification of the solutions

Usually global solutions are found during Phase II. However, we cannot be assured that they are indeed global ones, since generally $\bar{f} - est \min f > \epsilon$ for some or all boxes in $\mathcal{Q}$. For example, a box may contain one global minimizer and a local one. If the local algorithm found the local one, we are not done yet. If both are global ones, we have not found all of them. If we want to find global solutions for sure, then the third phase, called the verification phase, is needed.

Assume that x^* is a minimizer inside the box D. If the convexity of f over D can be guaranteed, then x^* is also the global minimizer in D and the whole D can be removed. As mentioned before, $m > 0$ implies $H(x) > 0$ on D if width(D) is small enough. The problem is that we do not know how small it should be. In other words, LBFs of ∇f can provide some information about the Hessian, but cannot replace it. To detect the convexity we have to use the bound of the Hessian. Suppose $H(x) \geqslant H$ for all $x \in D$, where H is an $n \times n$ constant matrix. If $H > 0$, then $H(x) > 0$ for all $x \in D$ and D can be removed. If $H > 0$ cannot be guaranteed (remember that the Gerschgorin Circle Theorem gives us a sufficient condition), then we obtain the lower bound $\hat{H}$ of the Hessian on the smaller box $\hat{D}_1 \subset D$ (say, width$(\hat{D}) = \frac{1}{3}$width(D)). If $\hat{H} > 0$ then $\hat{D}_1$ can be deleted; otherwise repeat the procedure on an even smaller box $\hat{D}_2 \subset \hat{D}_1$ until a box $\hat{D} \subset D$ can be removed.

The next step is to decompose $D - \hat{D}$ into $2n$ subboxes (in general). Since the neighborhood $\hat{D}$ of x^* has been removed, these $2n$ subboxes can be quickly discarded, hopefully, by the LBFs of either f or ∇f.

The algorithm of Phase III is the following.

Phase III (verification of the solutions)

Data: the collections $\mathcal{Q}$ and $\mathcal{P} = \emptyset$, the sets MINPT, MINVAL and $0 < \delta_1, \gamma < 1$.

Step 1. Pick a box $D \in \mathcal{Q}$ and $x^* \in D$. If x^* is on the boundary of D go to Step 4; else go to Step 2.

Step 2. Obtain the lower bound H of the Hessian on D. If $H > 0$, go to Step 1; else go to Step 3.

Step 3. $\hat{D} \leftarrow D$.

While $H > 0$ cannot be guaranteed (i.e., when the Gerschgorin Circle Theorem does not hold) or width$(\hat{D}) > \delta_1$, do the following:

Get a smaller $\hat{D}$, find the lower bound H of the Hessian on $\hat{D}$.

If width$(\hat{D}) \leqslant \delta_1$, go to Step 4; else go to Step 5.

Step 4. Generate the set $\mathcal{T}_{\mathrm{D}}$ of boxes by bisecting D; go to Step 6.

Step 5. Generate the collection $\mathcal{T}_{\mathrm{D}}$ of boxes by the decomposing $D - \hat{D}$ into (generally) $2n$ boxes.

Step 6. For each box $D_i \in \mathcal{T}_{\mathrm{D}}$, obtain the LLBF of f. If D_i cannot be removed, obtain the LBFs of ∇f. If D_i cannot be removed, store D_i in P and save the corresponding LBFs.

Step 7. If $\mathcal{P} = \emptyset$, go to Step 8; else go to Step 9.

Step 8. In the set MINPT remove those points whose function values $> \bar{f}$. Global optimization is DONE and the solutions are given by MINPT and $\bar{f}$.

Step 9. $\delta_0 \leftarrow \gamma\delta_0$, then go to Phase I.

The basic idea of Step 5 can be illustrated by Figure 1. Note that Phase III and Phase I use different subdivision methods (bisection along the longest edges is used in Phase I).

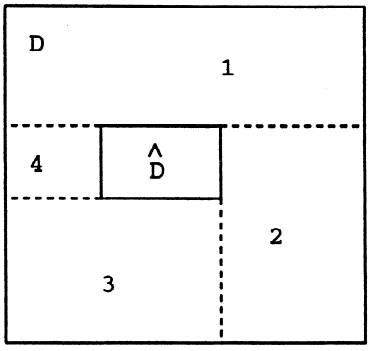

Figure 1. $D - \hat{D}$ is divided into $2n$ boxes.

6. Multivariate global optimization algorithm

Finally, we have the global optimization algorithm for the problem (3.1).

ALGORITHM 6.1 (global optimization for n-dimensional problems).
Initial step: $\mathcal{P}$ — a finite partition of S, $\mathcal{Q} = \emptyset$.
Phase I.
Pre-Phase II.
Phase II.
Phase III.
If $\mathcal{P} \neq \emptyset$, go to Phase I with the new
$\mathcal{P}$; Else STOP.

THEOREM 6.1 (convergence in a finite number of steps). *Assuming that in problem (3.1) $f : R^n \to R$ is continuously differentiable, the normal vectors of the LBFs of f and ∇f are bounded over any subbox of S. Also assume that problem (3.1) has a finite number of solutions. Then Algorithm 6.1 can find the ϵ-global solution(s) within finite steps.*

Proof. First we show that each phase will terminate within a finite number of steps.

The stopping criterion of Phase I is that every subbox in $\mathcal{P}$ is either removed or shrunk to one or more subboxes with its (their) width(s) not longer than δ_0 (and they will be stored in $\mathcal{Q}$). For each subbox in $\mathcal{P}$, if it cannot be removed, it will be bisected along the longest length of the box. Since this decomposition procedure is exhaustive (width of boxes $\to 0$), the width of the subbox will be no larger than δ_0 after a finite number of steps.

Obviously $\mathcal{Q}$ contains a finite number of subboxes, thus Pre-Phase II terminates within a finite number of steps.

The proof for Phase II can be found in [24] (by remembering the fact that we are seeking ϵ-global solutions).

Phase III terminates within a finite number of steps since $\mathcal{Q}$ has a finite number of subboxes and Step 3 stops in a finite number of steps (similar reason to bisection).

If $\mathcal{P} \neq \emptyset$ after Phase III, the algorithm will re-start from Phase I with the new $\mathcal{P}$. Assume that the algorithm keeps re-starting from Phase I to Phase III. Since δ_0 is shrunk for each cycle ($\delta_0 \leftarrow \gamma\delta_0, 0 < \gamma < 1$), all global minimizers will be found in Phase III and the associated neighborhoods (boxes) will be removed. In other words, in the next Phase I, each subbox of $\mathcal{P}$ does not contain a solution. Since its width tends to zero, it will be removed after a finite number of cycles (thus a finite number of steps) because of continuity of LBFs.

Therefore, Algorithm 6.1 terminates within a finite number of steps. □

7. Numerical testing

In this section, we will present numerical examples to illustrate how Algorithm 6.1 works, and demonstrate its potential and feasibility. In [22] there are some standard test problems. The problems which are most often used in [22] are chosen for this purpose.

$$(1) \quad \min f_{BR}(x), \tag{7.1}$$

$$-5 \leqslant x_1 \leqslant 10,\ 0 \leqslant x_2 \leqslant 15, \tag{7.2}$$

where

$$f_{BR}(x) = \left(x_2 - \frac{5.1}{4\pi^2}\, x_1^2 + \frac{5}{\pi}\, x_1 - 6 \right)^2 + 10 \left(1 - \frac{1}{8\pi} \right) \cos x_1 + 10. \tag{7.3}$$

This function has three local minimizers $(-3.142, 12.275)$, $(3.142, 2.275)$ and $(9.425, 2.475)$. They are also global ones, and the global minimum is 0.398.

$$(2) \quad \min f_C(x), \tag{7.4}$$

$$-5 \leqslant x_i \leqslant 5, \quad i = 1, 2, \tag{7.5}$$

where

$$f_C(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4. \tag{7.6}$$

This function is symmetric about the origin and has three conjugate pairs of local minima. The global minimum is equal to -1.0316 and it is attained at $(0.08983, -0.7126)$ and $(-0.08983, 0.7126)$.

$$(3) \quad \min f_{GP}(x), \tag{7.7}$$

$$-2 \leqslant x_i \leqslant 2, \quad i = 1, 2, \tag{7.8}$$

where

$$\begin{aligned} f_G(x) = {} & (1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)) \\ & \times (30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 \\ & + 48x_2 - 36x_1x_2 + 27x_2^2)). \end{aligned} \tag{7.9}$$

The global minimum is equal to 3 and it is reached at $(0, -1)$. There are four local minima in the minimization region.

$$(4) \quad \min f_R(x), \tag{7.10}$$

$$-1 \leqslant x_i \leqslant 1, \quad i = 1, 2, \tag{7.11}$$

where

$$f_R(x) = x_1^2 + x_2^2 - \cos 18x_1 - \cos 18x_2. \tag{7.12}$$

Table 1. Number of function evaluations (1).

	f	∇f	LBF(f)	LBF(∇f)	LBF(H)	total
BR	3	3	13	32	3	54
C	2	2	24	40	2	70
GP	1	1	34	56	6	98
R	1	1	23	23	4	52
H	1	1	73	111	3	189
S	1	1	40	60	1	103

The global minimum is equal to -2 and it is reached at $(0, 0)$. There are about 50 local minima in the region.

$$(5)\quad \min f_H(x), \tag{7.13}$$

$$0 \leqslant x_i \leqslant 1, \quad i = 1, 3, \tag{7.14}$$

where

$$f_H(x) = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{n} \alpha_{ij}(x_j - p_{ij})^2\right], \tag{7.15}$$

and parameters are same as those in [22]. The global minimum is equal to -3.86 and it is reached at the point $(0.114, 0.556, 0.852)$.

$$(6)\quad \min f_S(x), \tag{7.16}$$

$$0 \leqslant x_i \leqslant 10, i = 1, 4, \tag{7.17}$$

where

$$f_S(x) = -\sum_{i=1}^{5} \frac{1}{(x - A(i))(x - A(i))^T + c_i}, \tag{7.18}$$

and parameters are same as those in [22]. The local minima with values approximately equal to $-1/c_i$ are reached at the points close to $A(i), i = 1, m$. The global minimum is equal to -10.1532 and it is reached at $(4.0000, 4.0001, 4.0000, 4.0001)$.

The results of numerical testing are given by Tables 1 and 2. Figure 2 shows the scenario of Problem (1) after Pre-Phase II. The solid boxes (inside the original region) are the remaining subboxes after Pre-Phase II. More figures of other 2-D problems are shown in [24]. The $*$'s indicate the matching points when LBFs were generated. Roughly speaking, the proportion of the computation requirement for each iteration is about $N + 1/2$ vs N vs $N(N + 1)/2$, increasing with each phase.

8. Discussion

In this paper, we develop an LBF-based global optimization algorithm for solving n-dimensional problems. Under certain conditions it can find the ϵ-global solution(s)

Table 2. Number of function evaluations (2).

	δ_0	Phase I	Pre-Phase II	Phase II	Phase III	Phase I	total
BR	0.5	30	3	18	3	—	54
C	0.5	52	6	10	2	—	70
GP	0.5	67	1	20	10	—	98
R	0.5	36	5	3	8	—	52
H	0.05	163	2	7	15	2	189
S	0.05	96	0	6	1	—	103

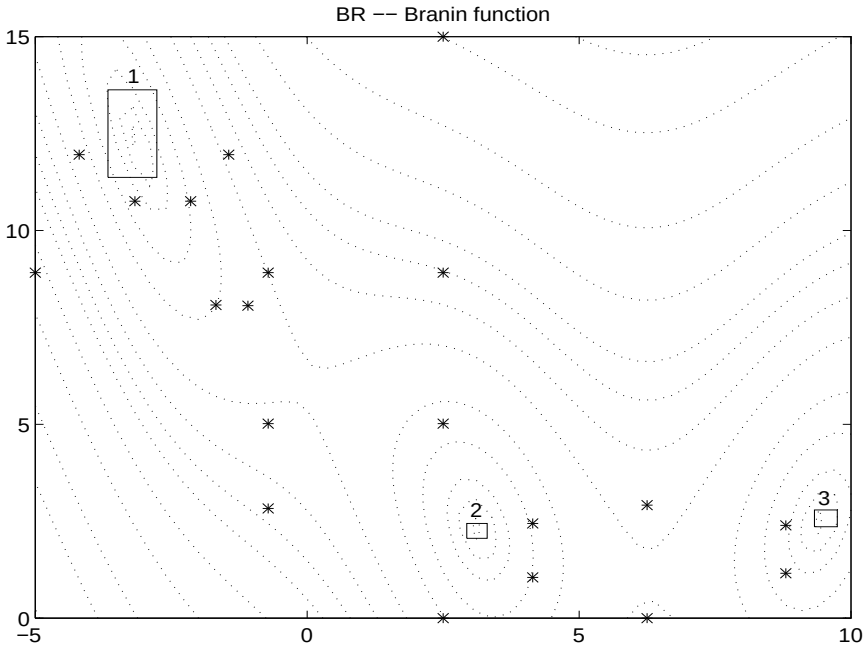

Figure 2. The scenario after Pre-Phase II.

of the given problem within finite steps. We need an extensive numerical testing (especially for high dimensional problems) in the future to validate its effectiveness.

People may notice that we do have choices in the algorithm if the given problem is continuously differentiable and all LBFs can be fairly simply generated. For example, we can choose using the LBFs of f, $\bigtriangledown f$ or both for all phases except Phase II. The different choices will not affect the result of Theorem 6.1. Of course some steps and strategies need to be adjusted. For instance, if we use only LBFs of f instead of those of f and $\bigtriangledown f$, then m_i will not be available and the box storing criterion should be adjusted, say, changed to the width of a box $\leqslant \delta_0$. For different problems or different available information, these alternatives give us flexibility in using the algorithm.

As we mentioned before, the solution of the given problem is usually found after Phase II or even Phase I with no guarantee. To verify the solution we have to use the direct information of the Hessian, since the LBFs of $\bigtriangledown f$ only can provide some indirect information of the Hessian and cannot predict the convexity of the function. Unavoidably the verification costs more computation effort. Depending what is needed in a specific application, we can decide what needs to be done.

Acknowledgments

This research was partly supported by VLSI Technology Inc. and Tyecin Systems Inc. through the UC MICRO program with grant 93-030.

References

1. Bromberg, M. and Chang, T.S. (1992), One Dimensional Global Optimization Using Linear Lower Bounds, in C.A. Floudas and P.M. Pardalos (eds.), *Recent Advances in Global Optimization*, Princeton University Press, pp. 200–220.
2. Bromberg, M. and Chang, T.S. (1993), *Global Optimization Using Linear and Convex Lower Bounds*, Technical Report No. UCD-ECE-SCR-93/5.
3. Cetin, B.C., Barhen, J. and Burdick, J.W. (1993), Terminal Repeller Unconstrained Subenergy Tunneling (TRUST) for Fast Global Optimization, *Journal of Optimization Theory and Applications* 77(1), 97–126.
4. Chang, T.S. and Tseng, C.L. (1994), *A New Linear Lower Bound and One Dimensional Global Optimization*, Technical Report No. UCD-ECE-SCR-94/2.
5. Floudas, C.A. and Pardalos, P.M., eds. (1992), *Recent Advances in Global Optimization*, Princeton University Press.
6. Hansen, E. (1980), Global Optimization Using Interval Analysis – The Multi-Dimensional Case, *Numer. Math.* 34, 247–270.
7. Hansen, P. Jaumard, B. and Lu, S.H. (1991), An Analytical Approach to Global Optimization, *Mathematical Programming* 52, 227–254.
8. Hansen, P., Jaumard, B. and Lu, S.H. (1992), On Using Estimates of Lipschitz Constants in Global Optimization, *Journal of Optimization Theory and Applications* 75(1), 195–200.
9. Hansen, P., Jaumard, B. and Xiong, J. (1994), Cord-Slope Form of Taylor's Expansion in Univariate global Optimization, *Journal of Optimization Theory and Applications* 80(3), 441–464.
10. Horst, R. and Tuy, H. (1987), On the Convergence of Global Methods in Multiextremal Optimization, *Journal of Optimization Theory and Applications* 54(2), 253–271.
11. Horst, R. and Tuy, H. (1990), *Global Optimization: Deterministic Approaches*, Springer Verlag, Berlin.
12. Jones, D.R., Perttunen, C.D. and Stuckman, B.E. (1993), Lipschitzian Optimization Without the Lipschitz Constant, *Journal of Optimization Theory and Applications* 79(1), 157–181.
13. McCormick, G.P. (1972), *Converting General Nonlinear Programming Problems to Separable Nonlinear Programming Problems*, Technical paper serial T-267, Institute for Management Science and Engineering, The George Washington University, Washington, DC.
14. McCormick, G.P. (1976), Computability of Global Solutions to Factorable Nonlinear Programs, Part I: Convex Underestimating Problems, *Mathematical Programming* 10(2), 147–175.
15. Meewella, C.C. and Mayne, D.Q. (1989), Efficient Domain Partitioning Algorithms for Global Optimization of Rational and Lipschitz Continuous Functions, *Journal of Optimization Theory and Applications* 61(2), 247–270.
16. Pijavskij, S.A. (1972), An Algorithm for Finding the Absolute Extremum of a Function, *USSR Computational Mathematics and Physics*, pp. 57–67(2).

17. Ratschek, H. and Rokne, J. (1988), *New Computer Methods for Global Optimization*, Ellis Horwood Limited.
18. Ratschek, H. and Rokne, J. (1991), Interval Tools for Global Optimization, *Computers and Mathematics with Applications* 21(6/7), 41–50.
19. Sergeyev, Y.D. and Grishagin, V.A. (1994), A Parallel Method for Finding the Global Minimum of Univariate Functions, *Journal of Optimization Theory and Applications* 80(3), 513–536.
20. Shubert, B.O. (1972), A Sequential Method Seeking the Global Maximum of a Function, *SIAM Journal on Numerical Analysis* 9, 379–388.
21. Thach, P.T. (1993), Global Optimality Criterion and a Duality with a Zero Gap in Nonconvex Optimization, *SIAM Journal on Mathematical Analysis* 24(6), 1537–1556.
22. Törn, A. and Zilinskas, A. (1989), *Global Optimization*, Springer Verlag, Berlin.
23. Wang, X. and Chang, T.S. (1996), An Improved Univariate Global Optimization Algorithm with Improved Linear Bounding Functions, *Journal of Global Optimization* 8, 393–411.
24. Wang, X. (1995), *Global and Local Optimization Using Linear Bounding Functions*, PhD dissertation, University of California, Davis, CA.
25. Wood, G.R. (1991), Multidimensional Bisection Applied to Global Optimisation, *Computers and Mathematics with Applications* 21(6/7), 161–172.