Journal of Global Optimization, **29:** 249–264, 2004.
© 2004 *Kluwer Academic Publishers. Printed in the Netherlands.*

Computational Experience with a New Class of Convex Underestimators: Box-constrained NLP Problems

IOANNIS G. AKROTIRIANAKIS and CHRISTODOULOS A. FLOUDAS
Department of Chemical Engineering, Princeton University, Princeton, NJ 08544, USA
e-mail: floudas@titan.princeton.edu

(Received 31 July 2003; accepted 7 August 2003)

Abstract. In Akrotirianakis and Floudas (2004) we presented the theoretical foundations of a new class of convex underestimators for C^2 nonconvex functions. In this paper, we present computational experience with those underestimators incorporated within a Branch-and-Bound algorithm for box-conatrained problems. The algorithm can be used to solve global optimization problems that involve C^2 functions. We discuss several ways of incorporating the convex underestimators within a Branch-and-Bound framework. The resulting Branch-and-Bound algorithm is then used to solve a number of difficult box-constrained global optimization problems. A hybrid algorithm is also introduced, which incorporates a stochastic algorithm, the Random-Linkage method, for the solution of the nonconvex underestimating subproblems, arising within a Branch-and-Bound framework. The resulting algorithm also solves efficiently the same set of test problems.

Key words. Branch-and-bound, convex underestimators, global optimization

1. Introduction

In this paper we present our computational experience with a new global optimization algorithm that can solve within ϵ-global optimality, problems involving twice continuously differentiable functions. The general mathematical definition of those problems is as follows

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & h_j(x)=0, \quad j=1,2,\dots,m_1 \\ & g_k(x)\leqslant 0, \quad k=1,2,\dots,m_2 \\ & x\in X=[x^L,x^U], \end{aligned} \tag{1}$$

where $x\in\Re^n$ is the vector of variables, $x^L,x^U\in\Re^n$ are the vectors of the lower and upper bounds of the hyper-rectangular domain $X\subseteq\Re^n$, f is the objective function and $h_j(x),g_k(x)$ are the constraints of the problem. Problem (1) generally possesses many local minima, since the functions involved in it may be nonconvex. There exist two broad categories of Global Optimization algorithms: (i) deterministic and (ii) stochastic. Deterministic Global Optimization algorithms (see for example, Al-Khayyal and Falk, 1983; Horst and Tuy, 1987; Tuy, 1987; Adjiman et al., 1998b; Sherali and Alameddine, 1992; Ryoo and Sahinidis, 1996) guarantee to reach an ϵ-neighbourhood of the global minimum of problem (1) within a finite number of iterations. On the other hand, in Stochastic methods (e.g., Gelatt et al., 1983; Goldberg, 1987; Rinnoy-Kan and Timmer, 1987a; Schoen, 1991) the probability of finding the global optimum of problem (1) goes to one as the number of steps goes to infinity.

The main objective of this paper is to investigate the computational performance of a Branch-and-Bound approach that uses the underestimators developed in Akrotirianakis and Floudas (2004). The general structure of the algorithm is similar to that of the αBB Global Optimization algorithm (Maranas and Floudas, 1994; Adjiman et al., 1998a; Floudas, 2000). The algorithm is designed in such a way that it can accommodate both convex and nonconvex underestimators. If the underestimators are convex then the lower bound is found by a local minimization algorithm. When nonconvex underestimators are used then the global minimum of the lower bounding problem is determined by a stochastic optimization algorithm.

The paper is structured as follows. Section 2 briefly discusses the fundamentals of the new underestimators. Section 3 presents the Branch-and-Bound algorithm, named GαBB, that uses the new underestimators to solve global optimization problems. Section 4 discusses two versions of the GαBB algorithm, one with convex underestimators and another one with nonconvex underestimators. Section 5 presents our numerical experience with both versions of the GαBB algorithm. Finally, Section 6 concludes the paper and presents directions of future research.

2. The New Class of Convex Underestimators

The new underestimating function, $L_1(x;\gamma)$, of an arbitrary nonconvex function, $f(x)$, is defined as follows

$$L_1(x;\gamma)=f(x)+\Phi(x;\gamma) \tag{2}$$

where

$$\Phi(x;\gamma)=-\sum_{i=1}^{n}(1-e^{\gamma_i(x_i-x_i^L)})(1-e^{\gamma_i(x_i^U-x_i)})$$

and $\gamma=(\gamma_1,\gamma_2,\ldots,\gamma_n)^T$ is a vector of non-negative parameters. In Akrotirianakis and Floudas (2004), the following properties of the function $L_1(x;\gamma)$ were proved.

PROPERTY 1. $L_1(x;\gamma)\leqslant f(x)$, for all $x\in[x^L,x^U]$, because $\Phi(x;\gamma)\leqslant 0$ for all $x\in[x^L,x^U]$ and $\gamma\geqslant 0$.

PROPERTY 2. $L_1(x^C;\gamma)=f(x^C)$, for every corner point x^C of X, because $\Phi(x^C;\gamma)=0$ for all $x^C\in X$.

PROPERTY 3. There exist certain values of the parameters γ_i so that $L_1(x;\gamma)$ is a convex function. This is due to the fact that the relaxation function $\Phi(x;\gamma)$ is convex for every $x \in X$ and $\gamma_i \geqslant 0$, $i=1,2,\dots,n$. Hence if the parameters γ_i have large enough values then all the non-convexities in the original function $f(x)$ can be eliminated, thereby producing a convex function $L_1(x;\gamma)$.

PROPERTY 4. The maximum separation distance of between the nonconvex function $f(x)$ and its underestimator $L_{G\alpha BB}(x;\gamma)$ is

$$\max_{x^L \leqslant x \leqslant x^U} \{f(x) - L_1(x;\gamma)\} = \sum_{i=1}^{n} (1 - e^{\frac{1}{2}\gamma_i(x_i^U - x_i^L)})^2 \tag{3}$$

PROPERTY 5. The underestimators constructed over supersets of the current set are always less tight than the underestimator constructed over the current box constraints.

The values of the parameters γ_i, $i=1,2,\dots,n$ are determined by an iterative procedure that not only guarantees the convexity of the underestimator $L_1(x;\gamma)$ but also ensures that $L_1(x;\gamma)$ is tighter than the αBB underestimator

$$L_{\alpha BB}(x;\alpha) = f(x) - \sum_{i=1}^{n} \alpha_i (x_i - x_i^L)(x_i^U - x_i)$$

Recall that the value for each parameter α_i is determined by the equation

$$\alpha_i = \max\left\{0, -\frac{1}{2}\underline{f}_{ii} - \sum_{j \neq i} \max\{|\underline{f}_{ij}|, |\overline{f}_{ij}|\} \frac{d_j}{d_i}\right\} \tag{4}$$

where $\underline{f}_{ij}$ and $\overline{f}_{ij}$ are the lower and upper bounds of $\partial^2 f/\partial x_i x_j$ as calculated by interval analysis, and $d_i = x_i^U - x_i^L, i=1,2,\dots,n$ are positive parameters.

The initial values of the γ_i parameters are selected by solving the system of non-linear equations

$$\ell_i + \gamma_i^2 + \gamma_i^2 e^{\gamma(x_i^U - x_i^L)} = 0, \quad i=1,2,\dots,n \tag{5}$$

where $\ell_i \leqslant 0$, $i=1,2,\dots,n$. The parameters ℓ_i convey second order characteristics of the original nonconvex function into the construction process of the underestimator. Candidate values for these parameters can be provided by the scaled Gerschgorin method (Adjiman et al., 1998b), that is

$$\ell_i = \max\left\{0, -\underline{f}_{ii} - \sum_{j \neq i} \max\{|\underline{f}_{ij}|, |\overline{f}_{ij}|\} \frac{d_j}{d_i}\right\}, \quad i=1,2\dots,n \tag{6}$$

Note that,

$$\alpha_i = \frac{1}{2}\ell_i, \quad i = 1, 2\ldots, n \tag{7}$$

In Akrotirianakis and Floudas (2004), we proved two important results regarding the relationship between the maximum separation distances between $f(x)$ and the two underestimators $L_1(x;\gamma)$ and $L_{\alpha BB}(x;\alpha)$. Here we present the theorems without their proofs.

THEOREM 1. *Let $\underline{\gamma} = (\underline{\gamma}_1, \underline{\gamma}_2, \ldots, \underline{\gamma}_n)^T$ be the solution of system* (5), *with ℓ_i defined by* (6). *Then, the two underestimators $L_1(x;\underline{\gamma})$ and $L_{\alpha BB}(x;\underline{\alpha})$, where*

$$\underline{\alpha} = \left(\frac{4(1 - \mathrm{e}^{0.5\underline{\gamma}_1 (x_1^U - x_1^L)})^2}{(x_1^U - x_1^L)^2}, \ldots, \frac{4(1 - \mathrm{e}^{0.5\underline{\gamma}_n (x_n^U - x_n^L)})^2}{(x_n^U - x_n^L)^2} \right)^T \tag{8}$$

have the same maximum separation distance from $f(x)$.

THEOREM 2. *Let $\overline{\alpha} = (\overline{\alpha}_1, \overline{\alpha}_2, \ldots, \overline{\alpha}_n)^T$ be the values of the α parameters as computed by* (4). *Then, the two underestimators $L_1(x;\overline{\gamma})$ and $L_{\alpha BB}(x;\overline{\alpha})$, where*

$$\overline{\gamma} = \left(\frac{2\log(1 + \sqrt{\overline{\alpha}_1}(x_1^U - x_1^L)/2)}{x_1^U - x_1^L}, \ldots, \frac{2\log(1 + \sqrt{\overline{\alpha}_n}(x_n^U - x_n^L)/2)}{x_n^U - x_n^L} \right)^T \tag{9}$$

have the same maximum separation distance from $f(x)$.

The above two theorems reveal that for any $\gamma \in [\underline{\gamma}, \overline{\gamma}]$ there exists an $\alpha \in [\underline{\alpha}, \overline{\alpha}]$, such that the underestimators $L_1(x;\gamma)$ and $L_{\alpha BB}(x;\alpha)$ have the same maximum separation distance from the nonconvex function $f(x)$. From all these pairs of underestimators, the only one that is known to be convex *a priori* is $L_{\alpha BB}(x;\overline{\alpha})$, since this is the one resulting from the classical αBB method. However, for most arbitrarily nonconvex functions the underestimators $L_{\alpha BB}(x;\alpha)$ and $L_1(x;\gamma)$ are convex within a large portion of the intervals $[\underline{\alpha}, \overline{\alpha}]$ and $[\underline{\gamma}, \overline{\gamma}]$ respectively. Based on the above observations, it is natural to search for a vector γ in the interval $[\underline{\gamma}, \overline{\gamma}]$ or for a vector α in the interval $[\underline{\alpha}, \overline{\alpha}]$, so that at least one of the underestimators $L_1(x;\gamma)$, $L_{\alpha BB}(x;\alpha)$ is convex.

In Akrotirianakis and Floudas (2004), we proposed an approach that iteratively determines, using interval analysis, the minimum values of the γ or α parameters that result in an underestimator that is convex and tighter than the classical αBB method. The details of that algorithm are presented in Section 4.

3. The GαBB Global Optimization Algorithm

In this section the overall description of the Branch-and-Bound algorithm, named GαBB, that uses the new underestimators is presented. As in the αBB algorithm, the GαBB algorithm locates the global optimum of the nonconvex problem (1) by using an enumeration tree. Two converging sequences of lower and upper bounds on the global optimum of problem (1) are generated. Each lower bound is determined by solving a convex optimization problem that derives from the original nonconvex problem where the nonconvex functions have been replaced by the convex underestimating functions described in the previous section. Upper bounds are generated by solving the original nonconvex problem locally using as starting point the optimum solution of the corresponding lower bounding problem.

The sequence of lower bounds of the global optimum is monotonically non-decreasing as the Branch-and-Bound tree expands. This is achieved by keeping a list of all the lower bounds that have been generated so far in increasing order and partitioning the subdomain of the problem that corresponds to the first element in the list, that is the problem with the minimum lower bound. The partitioning of the subdomain is done by bisecting its longest subinterval, say $[x_k^L, x_k^U]$, provided that the variable x_k participates in at least one nonconvex term in the original problem (1). As far as the sequence of the updated upper bounds is concerned, it is monotonically non-increasing. This is achieved by storing only the minimum of all previously found upper bounds.

Furthermore, from Property 5 of the underestimating function L_1 and Eq. (3), we can conclude that its maximum separation distance from the nonconvex function that underestimates goes to zero, as the size of the subinterval $[x^L, x^U]$ goes to zero. This implies that, as the current subinterval $[x^L, x^U]$ reduces to a point, the maximum separation distance of every nonconvex function in the nonconvex problem (1) from its convex underestimatior becomes zero.

The basic steps of the GαBB method are described in Algorithm 1. Throughout the algorithm, $P(x^{L,\text{Iter}}, x^{U,\text{Iter}})$ and $P_{\text{LOW}}(x^{L,\text{Iter}}, x^{U,\text{Iter}})$ represent the original nonconvex optimization problem (1) and its lower bounding problem at iteration *Iter* respectively, with $x \in [x^{L,\text{Iter}}, x^{U,\text{Iter}}]$.

Algorithm 1. *The GαBB Algorithm*

STEP 1: *Initialization*
The following quantities are initialized: convergence tolerance: $\epsilon_c > 0$; feasibility tolerance: $\epsilon_f > 0$; iteration counter: Iter $= 1$; current subdomain at Iter $= 1$: $[x^{L,\text{Iter}}, x^{U,\text{Iter}}] = [x^L, x^U]$; list of unsolved nodes: $\Lambda = \{[x^{L,\text{Iter}}, x^{U,\text{Iter}}]\}$; $LBD = -\infty$; $UBD = \infty$; the initial starting point, $x^{c,\text{Iter}}$, is randomly generated.

STEP 2: *Update of the upper bound UBD*
Remove the first element of the list of unexplored nodes Λ.
Solve the nonconvex optimization problem $P(x^{L,\text{Iter}}, x^{U,\text{Iter}})$ locally, within the current subdomain. Let x_{UP}^{Iter} be its optimum solution.

If x_{UP}^{Iter} is ϵ_f feasible and $f(x_{UP}^{\text{Iter}}) < UBD$ then update the upper bound on the global optimum of the original problem, that is, $UBD = f(x_{UP}^{\text{Iter}})$ and $x* = x_{UP}^{\text{Iter}}$.

STEP 3: *Solution of the Lower Bounding Problem and Update of LBD*
Construct the lower bounding problem, $P_{\text{LOW}}(x^{L,\text{Iter}}, x^{U,\text{Iter}})$, by replacing all arbitrarily nonconvex terms in the original problem, $P(x^{L,\text{Iter}}, x^{U,\text{Iter}})$, by underestimating functions defined by (2).
Solve the lower bounding problem $P_{\text{LOW}}(x^{L,\text{Iter}}, x^{U,\text{Iter}})$. Let $x_{\text{LOW}}^{\text{Iter}}$ be its optimum solution. Also let $L_1^f(x;\gamma^f)$ be the underestimator of the objective function of the problem $P_{\text{LOW}}(x^{L,\text{Iter}}, x^{U,\text{Iter}})$. Update LBD, if $L_1^f(x_{\text{LOW}}^{\text{Iter}};\gamma^f) \geqslant LBD$, by setting $LBD = L_1^f(x_{\text{LOW}}^{\text{Iter}};\gamma^f)$.
The current starting point is updated so that $x^{c,\text{Iter}+1} = x_{\text{LOW}}^{\text{Iter}}$.

STEP 4: *Fathoming or Branching*
If $L_1^f(x_{\text{LOW}}^{\text{Iter}};\gamma^f) - UBD \geqslant \epsilon_c$ then the current node can be safely *fathomed.*
Otherwise, the current subdomain is partitioned into two subdomains, by bisecting the largest component interval. That is, the current subdomain $[x^{L,\text{Iter}}, x^{U,\text{Iter}}]$ is partitioned into the following two subdomains

$$\left([x_1^{L,\text{Iter}}, x_1^{U,\text{Iter}}], \ldots, \left[x_k^{L,\text{Iter}}, \frac{x_k^{U,\text{Iter}} + x_k^{L,\text{Iter}}}{2}\right], \ldots, [x_n^{L,\text{Iter}}, x_n^{U,\text{Iter}}]\right)^T$$

and

$$\left([x_1^{L,\text{Iter}}, x_1^{U,\text{Iter}}], \ldots, \left[\frac{x_k^{U,\text{Iter}} + x_k^{L,\text{Iter}}}{2}, x_k^{U,\text{Iter}}\right], \ldots, [x_n^{L,\text{Iter}}, x_n^{U,\text{Iter}}]\right)^T$$

where $[x_k^{L,\text{Iter}}, x_k^{U,\text{Iter}}]$ is assumed to be the largest component interval of the current subdomain.
The two new subdomains are then stored in the list Λ at the appropriate places so that the order of the list is kept in increasing order with respect to the lower bounds.

STEP 5: *Convergence check*
If $\Lambda = \emptyset$ then STOP. The global optimum solution of the original problem is x_* and its value is $f(x*)$.
Otherwise, increase the counter, Iter = Iter + 1 and GoTo Step 2.

4. Computational Studies

In this section we present a computational study regarding the efficiency of the GαBB algorithm and the new class of underestimators. We have implemented two versions of the GαBB algorithm. In the first version the underestimators used throughout the enumeration tree are convex. That is achieved by calculating proper values for the γ parameters that give rise to convex underestimating functions. We denote this version as the *deterministic* GαBB method.

The second version of the GαBB algorithm uses underestimators that may be nonconvex. The values of the γ parameters used in the definition of the underestimator $L_1(x;\gamma)$ are given by the solution of the system (5). That is, $\gamma_i = \underline{\gamma}_i$, for all $i = 1, 2, \ldots, n$. Since there is no guarantee that the lower bounding problem is convex, we have developed a stochastic approach to solve it. We denote the second version as the *hybrid* GαBB method, since it combines a stochastic method within a deterministic Branch-and-Bound framework.

In the remaining of this section we provide a more detailed description of the two versions of the GαBB algorithm.

4.1. THE DETERMINISTIC GαBB METHOD

The deterministic version of the GαBB method employs an iterative procedure that determines the appropriate values for the γ parameters that yield a convex underestimator. The procedure searches for a vector $\gamma \in [\underline{\gamma}, \overline{\gamma}]$ so that the corresponding $\alpha \in [\underline{\alpha}, \overline{\alpha}]$, produces an underestimating function $L_{\alpha BB}(x;\alpha)$ that is convex. The search starts by setting $\gamma = \underline{\gamma}$ and $\alpha = \underline{\alpha}$ and then checking whether $L_{\alpha BB}(x;\underline{\alpha})$ is convex. This is done by using the scaled Gerschgorin method to determine lower bounds on the eigenvalues of the Hessian matrix $\nabla^2 L_{\alpha BB}(x;\underline{\alpha})$. For those lower bounds that are negative, we bisect the intervals of the corresponding variables, thereby generating a number of sub-domains that are stored in a list, denoted by Λ_2. Then the algorithm checks whether $\nabla^2 L_{\alpha BB}(x;\underline{\alpha})$ is positive semi-definite in each of those sub-domains using again the scaled Gerschgorin method. If the size of the list, Λ_2, exceeds a certain number of nodes then $\nabla^2 L_{\alpha BB}(x;\underline{\alpha})$ is most likely *not* positive semi-definite, and the values of all γ_i's are increased by a prespecified positive quantity, $\eta > 0$, and the corresponding values of the new α_i's are calculated. The algorithm tries to verify whether $\nabla^2 L_{\alpha BB}(x;\alpha)$, with the new increased α, is positive semi-definite. It continues in this manner until the list Λ_2 becomes empty. In that case the corresponding α makes the Hessian matrix, $\nabla^2 L_{\alpha BB}(x;\alpha)$, positive semi-definite for all $x \in X$ and consequently $L_{\alpha BB}(x;\alpha)$ a convex underestimator. The main reason for using the underestimator $L_{\alpha BB}(x;\alpha)$ instead of the underestimator $L_1(x;\gamma)$ is that, it is easier to verify the positive definiteness of the matrix $\nabla^2 L_{\alpha BB}(x;\alpha)$ than that of the matrix $\nabla^2 L_1(x;\gamma)$.

The detailed algorithmic steps are as follows:

Algorithm 2. Generation of convex underestimators

STEP 1: (*Initialization*) Set $K = 1, J = 1, J_{\max} = 2^n + 1$, $\eta = 1.1$ $X_J = X$, $\Lambda_2 = \{X_J\}$ and $\gamma_{i,K} = \underline{\gamma}_i$

STEP 2: Use (8) to calculate the $\alpha_{i,K}$, $i = 1, 2, \ldots, n$ that correspond to the $\gamma_{i,K}$, $i = 1, 2, \ldots, n$, and form the underestimator $L_{\alpha BB}(x;\alpha_K)$.

```
STEP 3: If the maximum separation distance of $L_{\alpha BB}(x;\alpha_K)$ from $f(x)$ is less
        than the maximum separation distance of $L_{\alpha BB}(x;\overline{\alpha})$ from $f(x)$ then
        GoTo Step 4.
        Otherwise, adopt as underestimator the classical $\alpha$BB underestimator,
        $L_{\alpha BB}(x;\overline{\alpha})$, and STOP.
STEP 4: Check whether $L_{\alpha BB}(x;\alpha_K)$ is convex:
        Repeat
            Step 4.1: Remove the last element from the list $\Lambda_2$ of unexplored sub-
                      domains. Let us name that sub-domain $X_{\text{last}}$
            Step 4.2: Form the interval Hessian $[\nabla^2 L_{\alpha BB}(x;\alpha_K)]$ with $x \in X_{\text{last}}$
            Step 4.3: Use (6) to find lower bounds on each eigenvalue of the interval
                      Hessian $[\nabla^2 L_{\alpha BB}(x;\alpha_K)]$ in $X_{\text{last}}$.
            Step 4.4: Form the set $I_- = \{i : \ell_i < 0\}$.
            Step 4.5: If $I_- \neq \emptyset$, bisect all intervals $[x^L_{i,\text{last}}, x^U_{i,\text{last}}]$ with $i \in I_-$, and add
                      them at the end of the list $\Lambda_2$.
            Step 4.6: Set $J = J + 2^{|I_-|} - 1$, where $|I_-|$ represents the cardinality of
                      the set $I_-$ (i.e., a total of $2^{|I_-|}$ new sub-domains have been gen-
                      erated and added to the list and one node have been removed).
        Until ($\Lambda_2 = \emptyset$ or $J = J_{\max}$)
STEP 5: If $\Lambda_2 = \emptyset$ then STOP. The Hessian $\nabla^2 L_{\alpha BB}(x;\alpha_K)$ is positive semi-
        definite for all $x \in X$ and $L_{\alpha BB}(x;\alpha_K)$ is a convex underestimator.
        Also the underestimator $L_{\alpha BB}(x;\alpha_K)$ is tighter than the underestimator
        $L_{\alpha BB}(x;\overline{\alpha})$ obtained by the classical $\alpha$BB method.
        Otherwise, increase the values of all $\gamma_{i,K}, i = 1,2,\ldots,n$ by setting
        $\gamma_{i,K+1} = \eta\gamma_{i,K}$. Set $K = K+1$ and GoTo Step 2.
```

Termination of Algorithm 2 with a convex underestimator of $f(x)$ is guaranteed by the fact that $L_{\alpha BB}(x;\overline{\alpha})$ is known *a priori* to be convex underestimator.

4.2. THE HYBRID GαBB METHOD

The underestimators used in the *hybrid* GαBB method are formed by setting $\gamma_i = \underline{\gamma}_i$ for all $i = 1,2,\ldots,n$. Although, the underestimator $L_1(x;\underline{\gamma})$ may be nonconvex function, it is 'less' nonconvex than the function it underestimates. Hence, the regions of attraction of the local minima of every underestimator are much larger than those of the corresponding nonconvex function. This is a direct consequence of the fact that the Hessian of the new underestimating function is the sum of the Hessian of the original nonconvex function and the positive definite Hessian of the relaxation term. Hence, the resulting underestimator, if it is nonconvex, it will posses fewer local minima than the original function. A stochastic global optimization technique would, therefore, locate the global minimum of the underestimating function faster than the global minimum of the original nonconvex function.

Based on the above observation we have developed a stochastic method to solve the lower bounding problem. At every node of the tree we use the Random-Linkage stochastic optimization algorithm, developed by Locatelli and Schoen in (Locatelli and Schoen (1999)). The general definition of the Random-Linkage algorithm is described in Algorithm 3.

Algorithm 3. *The Random-Linkage Algorithm*

STEP 1: Set $k=0$;

STEP 2: Sample a single point x_{k+1} from the uniform distribution over the set X;

STEP 3: Start a local search from X_{k+1} with probability:

$$p_k(\delta_k(x_{k+1})) \tag{10}$$

with

$$\delta_k(x)=\min\{\| x-x_j \|:\ j=1,2,\ldots,k \text{ and } f(x_j)>f(x)\} \tag{11}$$

It is understood that $\delta_k(x)=\infty$ if there is no j such that $f(x_j)>f(x)$;

STEP 4: If $k>k_{\max}$ then STOP. Otherwise set $k=k+1$ and go to Step 2.

The function p_k represents a probabilistic threshold that allows the start of the local optimization solver from the current sample point. It is defined as follows:

$$p_k(\delta)=\begin{cases} 1, & \text{if } \delta>r_{k+1;1;\sigma} \\ 0, & \text{otherwise} \end{cases}$$

where

$$r_{k+1;1;\sigma}=\pi^{-1/2}\left(\Gamma(1+d/2)\mu(X)\sigma\frac{\log k}{k}\right)^{1/d}$$

and $\Gamma(\cdot)$ is the gamma distribution function, $\mu(\cdot)$ is the Lebesque measure for the set X, and σ is a user defined parameter. The parameter $k_{\max}$ represents the maximum number of iterations that Algorithm 3 is allowed to sample the feasible region of the current node. In our implementation we used $k_{\max}=20$.

5. Numerical Results

In this section we present the computational performance of the two versions of the GαBB algorithm. We have used both versions to solve several box constrained NLP problems. The problems we considered are difficult since they possess hundreds or thousands of local minima. In all cases the GαBB algorithm found the global minimum efficiently. All the computational results were obtained using an HP9000/730 workstation.

This section is organized as follows. In Section 5.1 we provide the mathematical definition of each test problem as well as the number of local minima it possesses. In Section 5.2 we discuss the performance of the GαBB algorithm.

5.1. TEST PROBLEMS

PROBLEM 1. The Levy function (Ali and Torn (1999)) is defined as follows

$$f(x) = \sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i - 1)^2(1 + \sin^2(3\pi x_{i+1})) + \\ + (x_n - 1)(1 + \sin^2(2\pi x_n))$$

where $n=4$, $x \in X = [-10, 10]^n$. The function possess 7100 local minima and its global minimum is $X^* = (1, 1, 1, -9.752356)^T$ with objective function value $f^* = -11.504403$.

PROBLEM 2. The Shubert function (Shubert (1972)) is defined as follows

$$f(x) = -\sum_{i=1}^{n}\sum_{j=1}^{5} j \sin((j+1)x_i + j)$$

where $n=2$, $x \in X = [-10, 10]^n$. The function possess 400 local minima and its global minimum is achieved in the following 9 different points

$$\begin{pmatrix} X_1^* \\ X_2^* \\ X_3^* \\ X_4^* \\ X_5^* \\ X_6^* \\ X_7^* \\ X_8^* \\ X_9^* \end{pmatrix} = \begin{pmatrix} (-6.774576, -6.774576)^T \\ (-6.774576, -0.491391)^T \\ (-6.774576, 5.791794)^T \\ (-0.491391, -6.774576)^T \\ (-0.491391, -0.491391)^T \\ (-0.491391, 5.791794)^T \\ (5.791794, -6.774576)^T \\ (5.791794, -0.491391)^T \\ (5.791794, 5.791794)^T \end{pmatrix}$$

with objective function value $f^* = -24.062499$.

PROBLEM 3. The following function is taken from Zhu (2002)

$$f(x) = -\frac{1}{2}\sum_{i=1}^{n}(x_i^4 - 16 * x_i^2 + 5x_i)$$

where $n=7$, and $x \in X = [-5, 2]^n$ has 2^n local minima. Its global minimum is achieved at $X^* = (-2.90354, -2.90354, \ldots, -2.90354)^T$.

PROBLEM 4. The Hansen function (Hansen (1980)) is defined as follows

$$f(x) = \sum_{i=1}^{5}(i\cos((i-1)x_1 + i))\sum_{j=1}^{5}(j\cos((j+1)x_2 + j))$$

where $n=2$, and $x \in X=[-10,10]^n$. The function has 760 local minima. Its global minimum is achieved at the following nine different points

$$\begin{pmatrix} X_1^* \\ X_2^* \\ X_3^* \\ X_4^* \\ X_5^* \\ X_6^* \\ X_7^* \\ X_8^* \\ X_9^* \end{pmatrix} = \begin{pmatrix} (-7.589893,-7.708314)^T \\ (-7.589893,-1.425128)^T \\ (-7.589893,4.858057)^T \\ (1.306708,-7.708314)^T \\ (1.306708,-1.425128)^T \\ (1.306708,4.858057)^T \\ (4.976478,-7.708314)^T \\ (4.976478,-1.425128)^T \\ (4.976478,4.858057)^T \end{pmatrix}$$

with objective function value $f^*=-176.541793$.

PROBLEM 5. The following function was proposed by Trefethen (2002)

$$f(x)=\mathrm{e}^{\sin(50x)}+\sin(60\,\mathrm{e}^y)+\sin(70\sin(x))+\sin(\sin(80y))- \\ -\sin(10(x+y))+(x^2+y^2)/4$$

where $x \in X=[-1,1]$ and $y \in Y=[-1,1]$. The function has 2400 local minima. It global minimum is $X^*=(-0.02440307955988, 0.210612427679)$ and the corresponding objective function value is $f(X^*)=-3.306868647475$.

PROBLEM 6. The Hartman function (Jansson and Knuppel (1994)) is defined as follows

$$f(x)=-\sum_{i=1}^{4} c_i \mathrm{e}^{-\sum_{j=1}^{n} A_{ij}(x_j-P_{ij})^2}$$

where $x \in X=[0,1]^n$. If $n=3$ the data of the problem is as follows

$$A=\begin{pmatrix} 3 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3 & 10 & 30 \\ 0.1 & 10 & 35 \end{pmatrix}, \quad c=\begin{pmatrix} 0.1 \\ 0.2 \\ 0.2 \\ 0.4 \end{pmatrix}, \quad P=\begin{pmatrix} 0.3689 & 0.1170 & 0.2673 \\ 0.4699 & 0.4378 & 0.7470 \\ 0.1091 & 0.8732 & 0.5547 \\ 0.03815 & 0.5743 & 0.8828 \end{pmatrix}.$$

The function achieves its minimum at $X^*=(1.14525, 5.555231, 8.526)$ and the corresponding objective function value is $f(X^*)=-3.861305797098$.

PROBLEM 7. The Griewank function (Griewank (1981)) is defined as follows

$$f(x)=\sum_{i=1}^{n}\frac{x_i^2}{4000}-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$$

where $n=2$ and $x \in X=[-500,700]^n$. The function has more than 500 local minima and its global minimum is achieved at the origin $X^*=(0,0)$.

PROBLEM 8. The following problem is taken from (More et al. (1981))

$$f(x)=\sum_{i=1}^{m}[2+2i-(e^{ix_i}+e^{ix_2})]^2$$

where $m=10$. The global minimum of this problem is

$$X^*=(0.25782484, 0.257824896)$$

and the corresponding objective function value is $f(X^*)=124.3621823719$.

PROBLEM 9. This problems is also taken from (More et al. (1981))

$$f(x)=\sum_{i=1}^{11}[y_i-(x_1u_i(u_i+x_2))/(u_i(u_i+x_3)+x_4)]^2$$

where $u_i=1/b_i$, $b=(0.25, 0.5, 1, 2, 4, 6, 8, 10, 12, 14, 16)$ and

$$y=(0.1957, 0.1947, 0.1735, 0.1600, 0.0844, 0.0627, 0.0456, \\ 0.0342, 0.0323, 0.0235, 0.0246)$$

The global minimum of this problem is

$$X^*=(0.19283049, 0.1908834, 0.123118232, 0.13578297)$$

and the corresponding objective function value is $f(X^*)=0.000307486$.

PROBLEM 10. This problems is also taken from (More et al. (1981))

$$f(x)=\sum_{i=1}^{m}\left[(x_1+t_ix_2-e^{t_i})^2+(x_3+x_4\sin(t_i)-\cos(t_i))^2\right]^2$$

where $m=20$ and $t_i=i/5$. The global minimum of this problem is

$$X^*=(-11.5944325, 13.2036530, -0.4034240038, 0.236470548)$$

and the corresponding objective function value is $f(X^*)=85822.20171974$.

PROBLEM 11. This problems is taken from (van Hentenryck et al. (1997))

$$f(x)=\frac{1}{400}\sum_{i=1}^{m}x_i^2-\prod_{i=1}^{m}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$$

where $m=5$ and $x \in (-10,10)^m$. The value of the objective function at the global minimum is $f(X^*)=0.0$.

5.2. COMPUTATIONAL RESULTS AND DISCUSSION

Both versions of the GαBB algorithm solved successfully all the test problems described in the previous section. This was expected for the *Deterministic* version since the underestimators used in the Branch-and-Bound tree are convex. The fact that the *Hybrid* version converged to the global minimum is particularly encouraging, since there is no guarantee that the underestimators, generated throughout the tree, are convex.

The performance of the two versions of GαBB as well as the performance of the classical αBB method is shown in Tables 1 and 2. The first observation, derived from Table 1, is that the classical αBB method always requires to generate more nodes, before it reaches the global optimum, than both versions of the GαBB method. This is expected, because the underestimators used by αBB are looser than those used by GαBB. Another observation is that the *Hybrid* GαBB method performs better than the *deterministic* GαBB method. The difference between the two versions is getting larger as the number of local minima and the number of variables of the problem increases. Also, from Table 2 we can see that the αBB method usually needs less CPU time than the *deterministic* version of GαBB. On the other hand, the *hybrid* versions of the GαBB method usually requires much less CPU time than the αBB method.

The above observations suggest that the convexity verification of the new underestimators, although it provides guarantees that the global optimum will be reached, increases the computational effort required. This is evident in the tests 3–7 where the αBB method although it uses looser underestimators outperforms the *deterministic* GαBB method. One can, therefore, readily realize the trade off between the tightness of the underestimators and the computational work involved in determing those underestimators.

However, the approach taken in the *hybrid* GαBB version seems to be less expensive and robust. The main reason for the efficiency of the *hybrid* version is the fact that the new underestimators are tighter and, if they are nonconvex, they

Table 1. Total number of nodes

Problem	αBB	Deterministic GαBB	Hybrid GαBB
1	15238	9256	5957
2	4282	4250	1008
3 ($n=5$)	1029	1024	814
3 ($n=7$)	8533	3616	2170
4	2820	1165	902
5	5097	3268	2177
6	2719	1736	1273
7	4598	3904	3216
8	118	61	40
9	62831	40873	38238
10	6168	4418	2971
11	4543	3360	2512

Table 2. CPU time (in seconds)

Problem	αBB	Deterministic GαBB	Hybrid GαBB
1	118.81	98.23	68.34
2	38.34	34.78	21.85
3 ($n=5$)	10.30	45.59	10.00
3 ($n=7$)	98.91	196.78	61.45
4	22.69	33.12	16.86
5	39.67	17.22	8.10
6	28.86	69.67	38.41
7	30.82	92.11	49.56
8	1.18	0.49	0.56
9	1525.05	2786.55	1781.02
10	243.45	68.92	26.86
11	108.81	120.97	58.83

have much less local minima than the original nonconvex function. This enables the stochastic optimization algorithm we use, to locate the global minimum of the underestimator efficiently (i.e., it requires a small number of restarts). In all of the above test problems we observed that the number of local searches is kept small throughout the enumeration tree.

6. Conclusions and Future Work

In this paper we presented our computational experience for box constrained NLPs with a new class of convex underestimators. The underestimators were incorporated within a Branch-and-Bound framework and the resulting algorithm, denoted as GαBB, was used to solve twice continuously differentiable problems to global optimality. We have implemented two versions of the GαBB algorithm. The first version, denoted as *deterministic* GαBB, uses underestimating functions that are always convex. This is guaranteed by calculating, using interval analysis, appropriate values for the γ parameters so that the resulting underestimator $L_1(x;\gamma)$ is convex function. The second version, denoted as *hybrid* GαBB, does not require the underestimators to be convex. The possible nonconvexity of the underestimators is handled by solving the lower bounding problems by a stochastic optimization method.

From the numerical results we conclude that the *hybrid* GαBB variant is more efficient than the *deterministic* GαBB. This result is particularly interesting since it indicates that nonconvex underestimators which possess larger regions of attraction than the corresponding nonconvex functions, together with a stochastic optimization solution technique can enhance the performance of a Branch-and-Bound algorithm for global optimization. In a future publication we shall present a study of the new class of underestimators when they are used to solve constrained nonlinear optimization problems.

Acknowledgments

The authors gratefully acknowledge financial support from the National Science Foundation and the National Institute of Health (R01 GM52032).

References

Adjiman, C.S., Androulakis, I. and Floudas, C. (1998a), A global optimization method, αBB, for general twice-differentiable constrained NLPs II. Implementation and computational results, *Computers and Chemical Engineering* 22, 1159–1179.

Adjiman, C.S., Dallwig, S., Floudas, C.A. and Neumaier, A. (1998b), A global optimization method, αBB, for general twice-differentiable constrained NLPs I: Theoretical aspects, *Computers and Chemical Engineering* 22(9), 1137–1158.

Akrotirianakis, I.G. and Floudas, C.A. (2004), A new class of improved convex underestimators for twice continuously differentiable constrained NLPs, *accepted for publication: Journal of Global Optimization*.

Al-Khayyal, F.H. and Falk, J.E. (1983), Jointly constrained biconvex programming, *Mathematics of Operations Research* 8, 523.

Ali, M.M. and Torn, A. (1999), Population set based global optimization algorithms: Some modifications and numerical studies. Technical report, Department of Computer Science, Abo Academi University, Turku, Finland.

Floudas, C.A. (2000), *Deterministic Global Optimization: Theory, Methods and Applications*. Kluwer Academic Publishers, Dordrecht.

Gelatt, C.D., Kirkpatric, S. and Vecchi, M.P. (1983), Optimization by simulated annealing, *Science* 220, 671.

Goldberg, D.E. (1987), *Genetic Algorithms in Search, Optimization and Machine Learning*. NY. Addison-Welsey, New York.

Griewank, A.O. (1981), Generalized descent for global optimization, *Journal of Optimization Theory and Applications* 34, 11–39.

Hansen, E. (1980), Global optimization using inteval analysis: the multidimensional case, *Numer. Math.* 34, 247–270.

Horst, R. and Tuy, H. (1987), On the convergence of global methods in multiextrimal optimization, *Journal of Optimization Theory and Applications* 54, 283.

Jansson, C. and Knuppel, O. (1994), Numerical results for a self-validating global optimization method. Technical Report 94.1, Forschungsshcwerpunkt Informations und Kommunicationstechnik, Technical University of Harburg, Germany.

Locatelli, M. and Schoen, F. (1999), Random Linkage: A family of acceptance/rejection algorithms for global optimization, *Mathematical Programming* 85, 379–396.

Maranas, C.D. and Floudas, C.A. (1994), Global minimum potential energy conformations for small molecules, *Journal of Global Optimization* 4, 135–170.

More, J., Garbow, B. and Hillstrom, K. (1981), Testing unconstrained optimization software, *ACM Transactions on Mathematical Software* 7, 17–41.

Rinnoy-Kan, A.H.G. and Timmer, G.T. (1987a), Stochastic global optimization methods. Part I: Clustering methods, *Mathematical Programming* 39, 27–56.

Rinnoy-Kan, A.H.G. and Timmer, G.T. (1987b), Stochastic global optimization, Part II: Multi-level Methods, *Mathematical Programming* 39, 57–78.

Ryoo, H.S. and Sahinidis, N.V. (1996), A Branch-and-Reduce approach to global optimization, *Journal of Global Optimization* 8(2), 107–139.

Schoen, F. (1991), Stochastic techniques for global optimization: a survey of recent Advances, *Journal of Global Optimization* 1(3), 207–228.

Sherali, H.D. and Alameddine, A. (1992), A new reformulation linearization technique for bilinear programming problems, *Journal of Global Optimization* 2(4), 379.

Shubert, B. (1972), A sequential method seeking the global minimum of a function, *SIAM Journal of Numerical Analysis* 9, 379–388.

Trefethen, N. (2002), A hundred-dollar, hundred-digit challenge, *SIAM News* 35.

Tuy, H. (1987), Global minimum of the difference of two convex functions, *Mathematical Programming Study* 30, 150.

van Hentenryck, P., Michel, L. and Deville, Y. (1997), *Numerica: A Modeling Language for Global Optimization*. MIT Press, Cambridge, MA.

Zhu, W. (2002), A sequential convexification method (SCM) for continuous global optimization. Technical Report 02.1, Department of Computer Science and Technology, Fuzhou University, China.