Journal of Global Optimization (2006) 35: 367–385
DOI 10.1007/s10898-005-3843-3

A Branch-and-Bound Based Method for Solving Monotone Optimization Problems*

X. L. SUN[1] and J. L. LI[1,2]
[1]*Department of Mathematics, Shanghai University, 99 Shangda Road, Baoshan, Shanghai 200444, P. R. China (e-mail: xlsun@staff.shu.edu.cn)*
[2]*College of Mathematics and Information Science, Guangxi University, Nanning, Guangxi 530004, P. R. China (e-mail: jianlingli@graduate.shu.edu.cn)*

(Accepted 6 October 2005)

Abstract. Monotone optimization problems are an important class of global optimization problems with various applications. In this paper, we propose a new exact method for monotone optimization problems. The method is of branch-and-bound framework that combines three basic strategies: partition, convexification and local search. The partition scheme is used to construct a union of subboxes that covers the boundary of the feasible region. The convexification outer approximation is then applied to each subbox to obtain an upper bound of the objective function on the subbox. The performance of the method can be further improved by incorporating the method with local search procedure. Illustrative examples describe how the method works. Computational results for small randomly generated problems are reported.

Key words: global optimization, monotone optimization, branch-and-bound method, partition, convexification, local search

1. Introduction

Consider the following monotone global optimization problem:

$$\begin{array}{lll} (P) & \max & f(x) \\ & \text{s.t.} & g_i(x) \leqslant b_i, \quad i = 1, \ldots, m, \\ & & x \in X = \{x \mid l_j \leqslant x_j \leqslant u_j, \ \ j = 1, \ldots, n\}, \end{array}$$

where f and all g_i are increasing functions of x_j on $[l_j, u_j]$ for $j = 1, \ldots, n$, $i = 1, \ldots, m$, functions f and g_i are not necessarily convex or separable.

Monotone optimization problems are an important class of global optimization problems. The idea of monotone optimization problems was

*Dedicated to Professor Alex Rubinov on the occasion of his 65th birthday. The authors appreciate very much the discussions with Professor Alex Rubinov and his suggestion of using local search. Research supported by the National Natural Science Foundation of China under Grants 10571116 and 10261001, and Guangxi University Scientific Research Foundation (No. X051022).

discussed in [5]. Many real-world applications can be modeled as a monotone optimization problem or its equivalent forms (see [17]). One of the prominent applications of the monotone optimization arises from the complex reliability network systems (see [16, 19]). The problem is to determine the reliability levels of each subsystem in order to maximize the overall reliability:

$$\begin{aligned} \max \quad & f(r_1, \ldots, r_n) \\ \text{s.t.} \quad & g_i(r_1, \ldots, r_n) \leqslant C_i, \quad i = 1, \ldots, m, \\ & l_j \leqslant r_j \leqslant u_j, \quad j = 1, \ldots, n, \end{aligned}$$

where f represents the overall reliability of the system, g_i the ith resource (cost, weight and volume, etc) consumed, C_i the total amount of ith resource, l_j and u_j the lower and upper bounds of the reliability level in j-th subsystem, $0 < l_j < u_j < 1$. By the very nature of the problem, f and g_i are strictly increasing functions of each r_j (see [1, 4]). A typical 5-link bridge system is shown in Figure 1. The overall reliability function has the following form: $f(r) = r_1 r_2 + q_2 q_3 r_4 + q_1 r_2 r_3 r_4 + r_1 q_2 q_3 r_4 r_5 + q_1 r_2 r_3 q_4 r_5$, where $q_j = 1 - r_j$, $j = 1, \ldots, 5$.

We point out that continuous resource allocation problems that have been extensively studied in the literature (see [6, 9, 10]) can be regarded as special cases of (P) with a single separable convex constraint and f being separable concave.

Although the global optimal solutions to (P) are located on the boundary of the feasible region, there may exist multiple local solutions. Therefore, problem (P) is of a global optimization problem. The monotone optimization method proposed by Rubinov et al. [13] was the first specialized algorithm for (P) (see also [17, 18]). Recognizing the fact that the objective function f always achieves the global maximum over a polyblock at one of

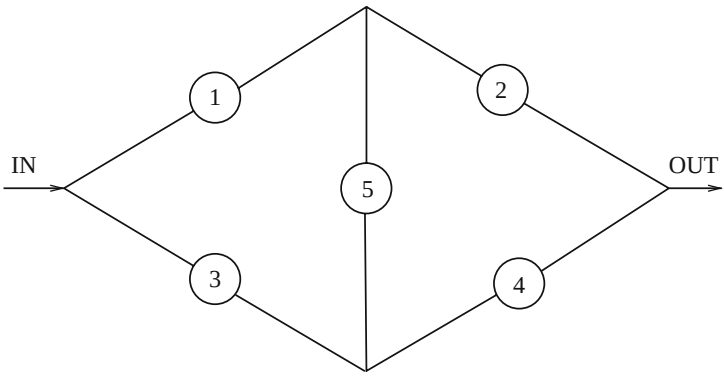

Figure 1. A 5-link Bridge network.

its extreme points, Rubinov et al. [13] introduced a polyblock approximation to the feasible region. The extreme point of a polyblock with the maximum function value can be regarded as an approximate optimal solution. The polyblock method can find an approximate global optimal solution in a finite number of iterations. We observe, however, that each extreme point of a polyblock is formed by n hyper-planes paralleling to n axes. Therefore, the extreme points of the polyblocks approach the global optimal solution in a zigzagging way and the polyblock method requires many iterations to reach an approximate optimal solution. Example 1 in Section 4 of this paper illustrates this phenomenon.

The convexification method proposed in [12] (see also [11, 15]) is essentially of a polyhedral approximation method. The monotone optimization (P) is first converted into a convex maximization problem (or a concave minimization problem) via a variable transformation. Outer polyhedra are then constructed to approximate the convexified feasible region. Cutting planes are added successively to form refined outer polyhedra. It is worth pointing out that the outer polyhedra are formed by hyper-planes tangential to the convexified feasible region. Thus, it can approximate the feasible region with higher accuracy than the outer polyblock, which is formed by hyper-planes paralleling to the axes. Comparison numerical results with polyblock approximation and outer approximation can be found in [20].

In order to implement the convexification method for high-dimensional problems, two computational issues have to be considered: (a) the determination of a suitable convexification parameter that controls the convexity; (b) the rapid increase of the number of vertices of the outer approximation polyhedron.

In this paper, we will present a new branch-and-bound method for (P). Three basic strategies: partition, convexification and local search will be incorporated into the branch-and-bound framework in order to design a more efficient algorithm for monotone optimization problems. The partition scheme is used to decompose the domain X into a series of subboxes. The union of these subboxes forms a *generalized* polyblock that covers the boundary of the feasible region. To obtain a better upper bound on each subbox, convexification method is used to construct polyhedral outer approximation, thus enabling more efficient node fathoming and speeding up the convergence of the branch-and-bound process. A local search procedure is employed to improve the lower bound of the optimal value. Since only an approximate solution is needed in the upper bounding procedure, the number of polyhedral vertices can be limited and controlled. Moreover, as the domain shrinks during the branch-and-bound process, the convexity can be achieved with a smaller parameter, thus avoiding the ill-conditional effect of the convexified problems.

In Section 2, we introduce some basic concepts and preliminary results for monotone optimization. A domain partition scheme and a modified variable convexification transformation are presented in this section. In Section 3, the main algorithm is described and its convergence is analyzed. In Section 4, illustrative examples are presented to show the feasibility and efficiency of the proposed branch-and-bound method. Computational results are reported in Section 5. Finally, a few concluding remarks are given in Section 6.

2. Preliminary Results

2.1. PARTITION SCHEME

Let $J=\{1,2,\ldots,n\}$, $l=(l_1,\ldots,l_n)^T$ and $u=(u_1,\ldots,u_n)^T$. Let $\alpha,\beta\in\mathbb{R}^n$ be such that $l\leqslant\alpha\leqslant\beta\leqslant u$. The rectangular determined by α and β is called a *box* in $\mathbb{R}^n$.

$$[\alpha,\beta]=\{x\mid\alpha_j\leqslant x_j\leqslant\beta_j,\quad j\in J\}=\prod_{j\in J}[\alpha_j,\beta_j].$$

The semi-closed box $[\alpha,\beta)$ and $(\alpha,\beta]$ can be defined similarly. The point α and β are called the *lower bound point* and the *upper bound point* of $[\alpha,\beta]$, respectively. For convenience, we define $[\alpha,\beta]=\emptyset$ if $\alpha\not\leqslant\beta$.

Consider a subproblem of (P) by replacing $X=[l,u]$ with a subbox $[\alpha,\beta]\subseteq X$:

$$\begin{array}{lll}(SP) & \max & f(x)\\ & \text{s.t.} & g_i(x)\leqslant b_i,\quad i=1,\ldots,m,\\ & & x\in[\alpha,\beta].\end{array}$$

Define

$$G(x)=\max_{i=1,\ldots,m}[g_i(x)-b_i]. \tag{1}$$

The boundary of the constraints can then be expressed as $\Gamma=\{x\in X\mid G(x)=0\}$. Let $S=\{x\in X\mid g_i(x)\leqslant b_i,\ i=1,\ldots,m\}$. Let x_b be a point on the boundary Γ. By the monotonicity of f and g_i's, we always have

$$f(x_b)\leqslant f_S\leqslant f(\beta),$$

where f_S is the global optimal value of (SP), $f_S=\max\{f(x)\mid x\in S\cap[\alpha,\beta]\}$.

By the monotonicity of f and g_i's, there are no better feasible points than x_b in $[\alpha,x_b)$ and there are no feasible points in $(x_b,\beta]$. Therefore, the

two boxes $[\alpha, x_b)$ and $(x_b, \beta]$ can be removed from $[\alpha, \beta]$ without missing any optimal solution of (SP). The following lemma shows that the set of the points left in $[\alpha, \beta]$ after removing $[\alpha, x_b)$ and $(x_b, \beta]$ can be partitioned into at most $2n-2$ subboxes.

LEMMA 1. *Let* $\alpha, \beta, \gamma \in \mathbb{R}^n$ *and* $\alpha \leqslant \gamma \leqslant \beta$. *Denote* $A = [\alpha, \beta]$, $B = [\alpha, \gamma)$ *and* $C = (\gamma, \beta]$. *Then* $A \backslash (B \cup C)$ *can be partitioned into* $2n-2$ *subboxes.*

$$A\backslash(B\cup C)=\left\{\bigcup_{i=2}^{n}\left(\prod_{k=1}^{i-1}[\alpha_k,\gamma_k]\times[\gamma_i,\beta_i]\times\prod_{k=i+1}^{n}[\alpha_k,\beta_k]\right)\right\}$$
$$\bigcup\left\{\bigcup_{i=2}^{n}\left(\prod_{k=1}^{i-1}[\gamma_k,\beta_k]\times[\alpha_i,\gamma_i]\times\prod_{k=i+1}^{n}[\alpha_k,\beta_k]\right)\right\}. \tag{2}$$

Proof. We first show that

$$A\backslash B=\bigcup_{i=1}^{n}\left(\prod_{k=1}^{i-1}[\alpha_k,\gamma_k]\times[\gamma_i,\beta_i]\times\prod_{k=i+1}^{n}[\alpha_k,\beta_k]\right). \tag{3}$$

Let $A_j = \prod_{i=j+1}^{n}[\alpha_i, \beta_i]$ and $B_j = \prod_{i=j+1}^{n}[\alpha_i, \gamma_i)$. As illustrated in Figure 2, we have

$$\begin{aligned}
&A_{j-1}\backslash B_{j-1}\\
&\quad=\prod_{i=j}^{n}[\alpha_i,\beta_i]\backslash\prod_{i=j}^{n}[\alpha_i,\gamma_i)\\
&\quad=\left\{([\alpha_j,\gamma_j]\times\prod_{i=j+1}^{n}[\alpha_i,\beta_i])\cup([\gamma_j,\beta_j]\times\prod_{i=j+1}^{n}[\alpha_i,\beta_i))\right\}\backslash\prod_{i=j}^{n}[\alpha_i,\gamma_i)\\
&\quad=([\gamma_j,\beta_j]\times\prod_{i=j+1}^{n}[\alpha_i,\beta_i])\cup\left\{([\alpha_j,\gamma_j]\times\prod_{i=j+1}^{n}[\alpha_i,\beta_i])\backslash\prod_{i=j}^{n}[\alpha_i,\gamma_i)\right\}\\
&\quad=([\gamma_j,\beta_j]\times\prod_{i=j+1}^{n}[\alpha_i,\beta_i])\cup\left\{[\alpha_j,\gamma_j]\times(\prod_{i=j+1}^{n}[\alpha_i,\beta_i]\backslash\prod_{i=j+1}^{n}[\alpha_i,\gamma_i))\right\}\\
&\quad=([\gamma_j,\beta_j]\times\prod_{i=j+1}^{n}[\alpha_i,\beta_i])\cup\left\{[\alpha_j,\gamma_j]\times(A_j\backslash B_j)\right\}.
\end{aligned} \tag{4}$$

Notice that $A = A_0$, $B = B_0$, $A_{n-1} \setminus B_{n-1} = [\alpha_n, \beta_n] \backslash [\alpha_n, \gamma_n) = [\gamma_n, \beta_n]$. Using the decomposition (4) recursively for $j = 1, \ldots, n-1$, we obtain (3).

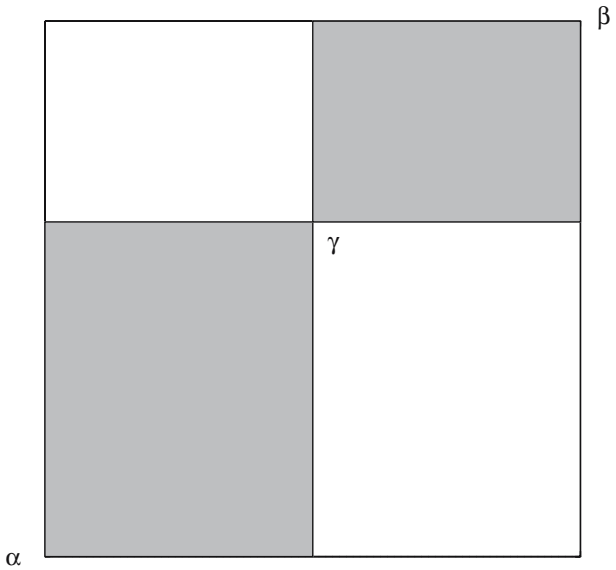

Figure 2. Partition of set $A \setminus (B \cup C)$.

Notice from (3) that $C = (\gamma, \beta]$ is included in the first subbox of the partition of $A \setminus B$, namely,

$$C = (\gamma, \beta] \subseteq \widetilde{C} = [\gamma_1, \beta_1] \times \prod_{k=2}^{n} [\alpha_k, \beta_k].$$

Using the similar arguments as in the proof of (3), we obtain

$$\widetilde{C} \setminus C = \bigcup_{i=2}^{n} \left(\prod_{k=1}^{i-1} [\gamma_k, \beta_k] \times [\alpha_i, \gamma_i] \times \prod_{k=i+1}^{n} [\alpha_k, \beta_k] \right), \tag{5}$$

which, combined with (3), gives rise to (2).

Remark 1. In the implementation of the above box partition process, it may occur $\beta_j - \alpha_j < \eta$ for some $j \in J$, where $\eta > 0$ is a small constant. Let $I = \{j \in J \mid \beta_j - \alpha_j \leqslant \eta\}$. From computational point of view, it is reasonable not to further partition $[\alpha, \beta]$ along jth axis for $j \in I$. In such case, we can partition the set $A \setminus (B \cup C)$ in terms of the indexes in $Q = J \setminus I$. In other words, for $j \in Q$, the jth lower bound and jth upper bound of each subbox in (2) are fixed to be α_j and β_j, respectively. The number of subboxes in (2) is then $2|Q| - 2$.

2.2. CONVEXIFICATION AND OUTER APPROXIMATION

Let $\alpha,\ \beta \in \mathbb{R}^n$ with $0 < \alpha < \beta$. Let $h(x)$ be a twice differentiable function defined on $[\alpha, \beta]$. Suppose that $h(x)$ is a strictly increasing function on $[\alpha, \beta]$, i.e., $\partial h/\partial x_j > 0$ for $x \in [\alpha, \beta],\ j \in J$.

Let $t(y) = (t_1(y_1), \ldots, t_n(y_n)) : \mathbb{R}^n \mapsto \mathbb{R}^n$ be a one-to-one mapping. Let $x = t(y)$. Define

$$h_t(y) = h(t(y)). \tag{6}$$

The domain of h_t is:

$$Y = \prod_{j=1}^{n} Y_j = \prod_{j=1}^{n} t_j^{-1}([\alpha_j, \beta_j]). \tag{7}$$

Define

$$\sigma = \min \left\{ d^T \nabla^2 h(x) d \mid x \in [\alpha, \beta],\ \ \|d\|_2 = 1 \right\}, \tag{8}$$

$$\mu = \min \left\{ \frac{\partial h}{\partial x_j} \mid x \in [\alpha, \beta],\ \ j \in J \right\}. \tag{9}$$

We have the following theorem (see [15]).

THEOREM 1. *Assume that h is a twice differentiable and strictly increasing function on $[\alpha, \beta]$. Let t be a twice differentiable and strictly monotonic mapping. If t satisfies the following condition:*

$$\frac{t_j''(y_j)}{[t_j'(y_j)]^2} \geqslant -\frac{\sigma}{\mu} \quad \text{for } y_j \in Y_j = t^{-1}([\alpha_j, \beta_j]),\ \ j \in J, \tag{10}$$

then $h_t(y)$ is a convex function on Y.

Remark 2. There are many mappings which satisfy condition (10). For example, let

$$t_j(y_j) = (1/p) \ln(1 - q/y_j), \tag{11}$$

where $p > 0$ is the parameter that controls the convexity and q is the parameter that controls length of the interval Y_j. It is easy to verify that the above t_j satisfies (10) for sufficient large $p > 0$.

Remark 3. One of the interesting questions arising from Theorem 1 is whether or not condition (10) still guarantees the convexity of h_t for non-smooth function h. Recently, an affirmative answer was given in [14] for semismooth functions.

We are now able to convert the subproblem (SP) into an equivalent convex maximization (or concave minimization) problem via the variable transformation:

$$\begin{aligned} (SP_t) \qquad \max \quad & f(t(y)) \\ \text{s.t.} \quad & g_i(t(y)) \leqslant b_i, \quad i = 1, \ldots, m, \\ & y \in t^{-1}([\alpha, \beta]). \end{aligned}$$

Let $S_t = \{y \in t^{-1}([\alpha, \beta]) \mid g_i(t(y)) \leqslant b_i,\ i = 1, \ldots, m\}$. By Theorem 1, if f and g_i's are strictly increasing function and t satisfies (10), then S_t is a convex set and problem (SP_t) is a convex maximization problem which can be solved by the outer approximation method (see [7] for details). It is well-known that the number of vertices of the polyhedral outer approximation method increases exponentially as the number of cutting planes increases (see [3, 18]). This makes the computation of the vertices for the convexified subproblem (SP_t) time-consuming as more and more cutting planes are added to the outer approximate polyhedron. In order to avoid such computational problem, we will calculate an upper bound of (SP_t) instead of searching for a high-accuracy solution. This can be done by stopping the outer approximation after adding a given number of cutting planes.

Another computational issue is the estimation of a suitable parameter in the variable transformation, for example, parameter p in (11). Let S_p denote the convexified feasible region. In practical cases, the parameter is determined empirically or by a self-adopted mechanism: choose an initial value of $p > 0$, compute the vertices of the outer polyhedron of S_p, where S_p is the transformed feasible region by using (11), if any vertex of the outer polyhedron is inside S_p, then increase p and restart the outer approximation process.

3. The Algorithm

The proposed method is a branch-and-bound method (see [8] for basic concepts of branch-and-bound methods in global optimization). A subproblem of (P) is formed by replacing X with certain subbox of X. Each subproblem corresponds to a *node* in the search tree. Upper bounds of the subproblems are estimated by using the convexification and outer approximation method discussed in Section 2.2. The partition formula (2) is used to branch new subproblems. The algorithm starts by calculating an upper bound on the initial box $X = [l, u]$. At the kth iteration, the algorithm maintains a list of subboxes. The subbox with the maximum upper bound is chosen from the list. Let $[\alpha^k, \beta^k]$ be such a subbox. The boundary point x^k on the line linking α^k and β^k is computed. A local search procedure is applied to the subproblem on $[\alpha^k, \beta^k]$ and finds a local solution x^k_{loc} in

domain $[\alpha^k, \beta^k]$ starting from initial point x^k. If x^k_{loc} is better than the current best feasible solution, then x^k_{loc} is set to be the incumbent and $f(x^k_{\text{loc}})$ is the new lower bound. The box $[\alpha^k, \beta^k]$ is then partitioned into at most $2|Q| - 2$ subboxes according to Lemma 1 and Remark 1. For each new subbox, an upper bound is computed by the convexification and outer approximation method. All the subboxes with upper bound equal to or less than the lower bound are fathomed. The subboxes left are added to the list. The process repeats until the list is empty.

Let d_j be the solution of $G(l_1, \ldots, l_{j-1}, x_j, l_{j+1}, \ldots, l_n) = 0$. By the monotonicity of the problem, we can reset $\beta_j := \min(d_j, \beta_j)$ to get a tighter initial box.

A feasible solution x is said to be an η-optimal solution if $x \in [\alpha, \beta]$ with $\|\beta - \alpha\|_\infty \leqslant \eta$ and $f(\alpha) \leqslant f^* \leqslant f(\beta)$, where f^* is the global optimal value of (P). A feasible solution x is said to be an ϵ-optimal solution if $f(x) \geqslant f^* - \epsilon$.

The algorithm can be formally described as follows.

ALGORITHM 1

Step 0. (Initialization). Choose tolerance parameters $\epsilon > 0$ and $\eta > 0$. If l is infeasible then problem (P) has no feasible solution. If u is feasible then u is the optimal solution to (P), stop. Otherwise, set $x_{\text{best}} = l$, $f_{\text{best}} = f(x_{\text{best}})$, $f^1_u = f(u), \alpha^1 = l, \beta^1 = u, X^1 = \{[\alpha^1, \beta^1]\}$. Set $k = 1$.

Step 1. (Box Selection). Select the subbox $[\alpha^k, \beta^k] \in X^k$ with maximum upper bound f^k_u. Let $I^k = \{j \in J \mid \beta^k_j - \alpha^k_j \leqslant \eta\}$ and $Q^k = J \setminus I^k$. If $Q^k = \emptyset$, stop, $x = \alpha^k$ is an η-optimal solution.

Step 2. (Boundary Point). Set $X^k := X^k \setminus [\alpha^k, \beta^k]$. Find the root λ^k of the following equation:

$$G(\lambda\alpha^k + (1-\lambda)\beta^k) = 0, \quad \lambda \in [0, 1], \tag{12}$$

where G is defined in (1). Set the boundary point $x^k = \lambda^k\alpha^k + (1-\lambda^k)\beta^k$.

Step 3. (Local Search). Starting from x^k, apply a local search procedure to find a local solution x^k_{loc} of the subproblem on $[\alpha^k, \beta^k]$. If $f(x^k_{\text{loc}}) > f_{\text{best}}$, set $x_{\text{best}} = x^k_{\text{loc}}$, $f_{\text{best}} = f(x^k_{\text{loc}})$.

Step 4. (Partition). Partition the set $\Omega^k = [\alpha^k, \beta^k] \setminus ([\alpha^k, x^k) \cup (x^k, \beta^k])$ into $2|Q^k| - 2$ new subboxes using formulation (2). Let X^{k+1} be the set of subboxes after adding the new subboxes to X^k. Removing all subboxes $[\gamma, \delta]$ in X^{k+1} with $f(\delta) \leqslant f_{\text{best}}$.

Step 5. (Upper Bounding). For each subbox $[\alpha, \beta]$ in X^{k+1}, apply a convexification transformation $x = t(y)$ to the subproblem of (P) on $[\alpha, \beta]$, where t satisfies the conditions in Theorem 1. Solve the

resulting problem (SP_t) by the outer approximation method and obtain an approximate solution $\tilde{y}$. Set $\tilde{x} = t(\tilde{y})$. $UB_{[\alpha,\beta]} = f(\tilde{x})$ is an upper bound of f on $[\alpha, \beta]$.

Step 6. (Fathoming). Remove all subboxes $[\alpha, \beta]$ in X^{k+1} with $UB_{[\alpha,\beta]} \leqslant f_{\text{best}}$. Let f_u^{k+1} be the maximum upper bound of all the subboxes. If $f_u^{k+1} - f_{\text{best}} < \epsilon$, then stop, x_{best} is an ϵ-optimal solution to (P). Otherwise, set $k := k+1$, goto Step 1.

Remark 4. In Step 2, bisection method or Newton's method can be used to search for the root of equation (12). The numerical experiment in [13] showed that bisection method performs better than Newton's method.

Remark 5. Since additional computational efforts are needed to search for a local solution of the subproblems, the local search procedure is not necessarily applied to each subproblem during the branch-and-bound process. A practical way is to use the boundary point x^k and the local solution x_{loc}^k alternatively as the new feasible solution. In Example 2 of Section 4 and the computational experiment for linearly constrained problems in Section 4, we will use the Reduced Gradient Method of Wolfe (see, for example [2]) as the local search procedure in Step 3.

Remark 6. As we mentioned in Section 2.2, there is some trade-off between the quality of the upper bound $UB_{[\alpha,\beta]}$ in Step 5 and the amount of computation and storage of the polyhedral vertices in the convexification and outer approximation procedure. In our implementation of the algorithm, at most $4n + k$ cutting planes are generated at k-th iteration to approximate the convexified feasible region S_t in problem (SP_t) and the maximum function value of the vertices is set to be $UB_{[\alpha,\beta]}$.

THEOREM 2. *Algorithm* 1 *either stops at an* η*-optimal solution in Step* 1 *or stops at an* ϵ*-optimal solution in Step* 6 *within a finite number of iterations.*

Proof. By the monotonicity of the problem and the box selection rule in Step 1, it is clear that the algorithm either finds an η-optimal solution $x = \alpha^k$ when it stops in Step 1 or an ϵ-optimal solution x_{best} when it stops in Step 6. Next, we show that if the algorithm does not stops in Step 6, then it will terminate in Step 1 within a finite number of iterations with $\|\beta^k - \alpha^k\|_\infty \leqslant \eta$ satisfied. Suppose that the algorithm generates an infinite subbox sequence $\{[\alpha^k, \beta^k]\}$. Then, by the algorithm and the partition formulation (2), there exists a subsequence $\{p_k\} \subset \{1, 2, \dots\}$ such that $[\alpha^{p_{k+1}}, \beta^{p_{k+1}}] \subset [\alpha^{p_k}, \beta^{p_k}]$ for $k = 1, 2, \dots$ and

$$\beta^{p_{k+1}} = \beta^{p_k} - (\beta_{s_k}^{p_k} - x_{s_k}^{p_k})e^{s_k}, \tag{13}$$
$$\alpha^{p_{k+1}} = \alpha^{p_k} + (x_{t_k}^{p_k} - \alpha_{t_k}^{p_k})e^{t_k}, \tag{14}$$

where s_k, $t_k \in Q^k \subseteq J$ and e^j is the jth unit vector of $\mathbb{R}^n$. Equations (13) and (14) yield

$$\beta_{s_k}^{p_k} - x_{s_k}^{p_k} \leqslant \|\beta^{p_{k+1}} - \beta^{p_k}\| \to 0, \quad k \to \infty, \tag{15}$$
$$x_{t_k}^{p_k} - \alpha_{t_k}^{p_k} \leqslant \|\alpha^{p_{k+1}} - \alpha^{p_k}\| \to 0, \quad k \to \infty. \tag{16}$$

Notice that $x^k = \lambda^k \alpha^k + (1-\lambda^k)\beta^k$ for any k. It then follows from (15) and (16) that

$$\lambda^{p_k}(\beta_{s_k}^{p_k} - \alpha_{s_k}^{p_k}) = \beta_{s_k}^{p_k} - x_{s_k}^{p_k} \to 0, \quad k \to \infty,$$
$$(1-\lambda^{p_k})(\beta_{t_k}^{p_k} - \alpha_{t_k}^{p_k}) = x_{t_k}^{p_k} - \alpha_{t_k}^{p_k} \to 0, \quad k \to \infty.$$

Thus

$$\lambda^{p_k}(\beta_{s_k}^{p_k} - \alpha_{s_k}^{p_k}) + (1-\lambda^{p_k})(\beta_{t_k}^{p_k} - \alpha_{t_k}^{p_k}) \to 0, \quad k \to \infty,$$

which implies either $\beta_{s_k}^{p_k} - \alpha_{s_k}^{p_k} \to 0$ or $\beta_{t_k}^{p_k} - \alpha_{t_k}^{p_k} \to 0$. Without loss of generality, suppose that

$$\beta_{s_k}^{p_k} - \alpha_{s_k}^{p_k} \to 0, \quad k \to \infty. \tag{17}$$

Since s_k is taken from the finite set J, (17) implies that there exist a fixed $s \in J$ and a subsequence $\{q_k\} \subset \{p_k\}$ such that $\beta_s^{q_k} - \alpha_s^{q_k} \to 0$, $k \to \infty$. Then, $\beta_s^k - \alpha_s^k \to 0$, $k \to \infty$. Hence there exists k_0 such that $\beta_s^k - \alpha_s^k \leqslant \eta$ when $k \geqslant k_0$. Since $s \notin Q^k$ for $k \geqslant k_0$, using the same arguments and the finiteness of J, we can prove $\beta_i^k - \alpha_i^k \leqslant \eta$ for all $i \in J$. Therefore, $\|\beta^k - \alpha^k\|_\infty \leqslant \eta$ will be satisfied when k is sufficiently large.

4. Illustrative Examples

In this section, we will demonstrate Algorithm 1 by two small examples.

EXAMPLE 1.

$$\begin{aligned} \max \quad & f(x) = 4.5(1-0.40^{x_1-1})(1-0.40^{x_2-1}) + 0.2\exp(x_1+x_2-7) \\ \text{s.t.} \quad & g_1(x) = 5x_1x_2 - 4x_1 - 4.5x_2 \leqslant 32, \\ & x \in X = \{x \mid 2 \leqslant x_1 \leqslant 6.2,\ 2 \leqslant x_2 \leqslant 6\}. \end{aligned}$$

It is clear that f and g_1 are strictly increasing functions on X. The problem has three local optimal solutions: $x^1_{\text{loc}} = (2.2692, 6)^T$ with $f(x^1_{\text{loc}}) = 3.7735$, $x^2_{\text{loc}} = (3.4528, 3.5890)^T$ with $f(x^2_{\text{loc}}) = 3.857736$ and $x^3_{\text{loc}} = (6.2, 2.1434)^T$ with $f(x^3_{\text{loc}}) = 3.6631$. Figure 3 shows the feasible region of the example.

Take t to be the convexification transformation (11) with $p = 2$. Set $\eta = \epsilon = 10^{-4}$. The algorithm starts by computing an upper bound of f on X via convexification and outer approximation. Figures 4 and 5 depict the convexified feasible region and the outer approximation for the first iteration. Figure 6 shows the partition process in the first three iterations of the algorithm. The algorithm stops at an approximation optimal solution $x_{\text{best}} = (3.4476, 3.5948)^T$ with $f(x_{\text{best}}) = 3.857732$ after six iterations and generating 95 vertices.

To see the effect of using convexification and partition method on this example, let's compare Algorithm 1 with the monotone optimization method using polyblock approximation ([13]). Figure 7 illustrates the first three iterations of the polyblock approximation for Example 1. Using the same accuracy $\epsilon = 10^{-4}$, the method finds an ϵ-approximate global solution $x_{\text{best}} = (3.4526, 3.5890)^T$ with $f(x_{\text{best}}) = 3.857736$ after 359 iterations and generating 718 vertices.

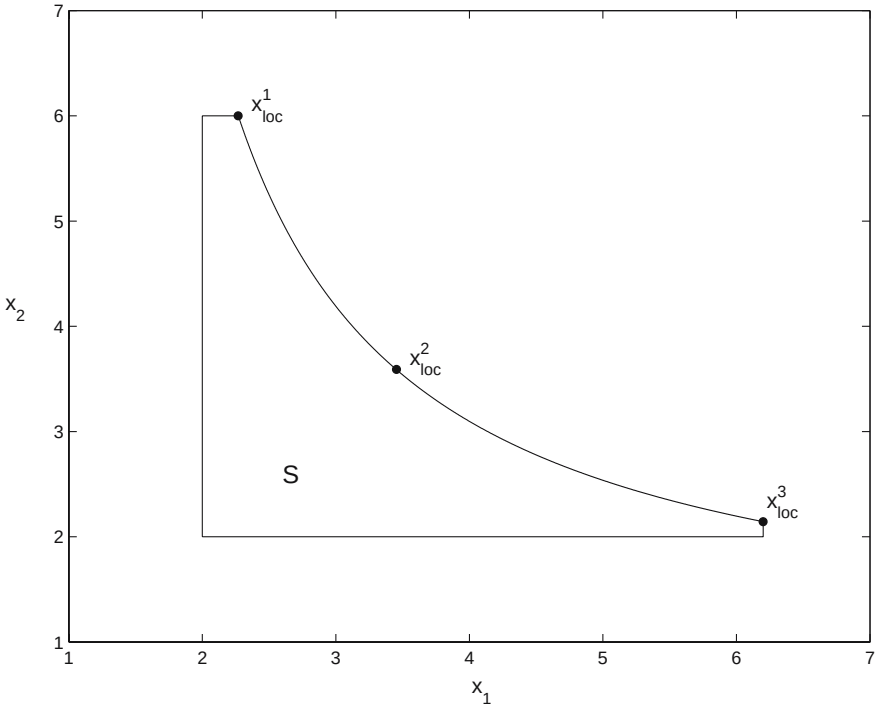

Figure 3. Multiple local solutions of Example 1.

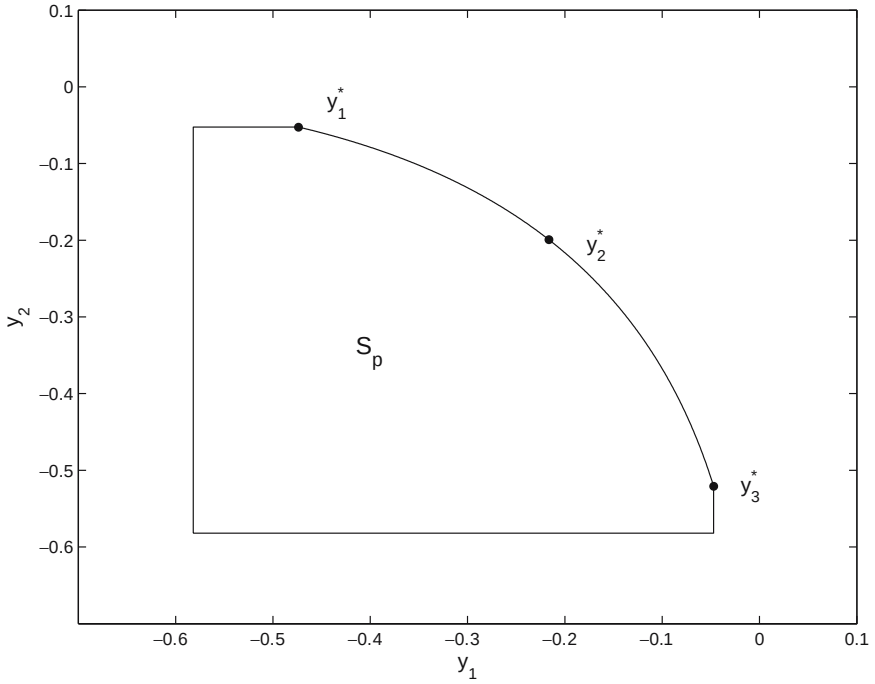

Figure 4. Convexified feasible region with $p=1$.

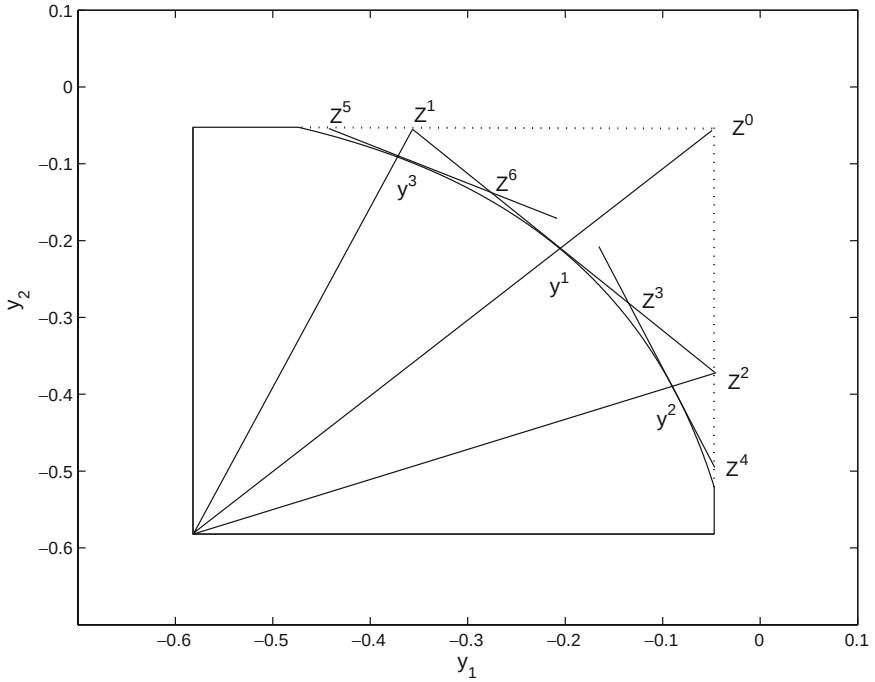

Figure 5. Convexification and outer approximation.

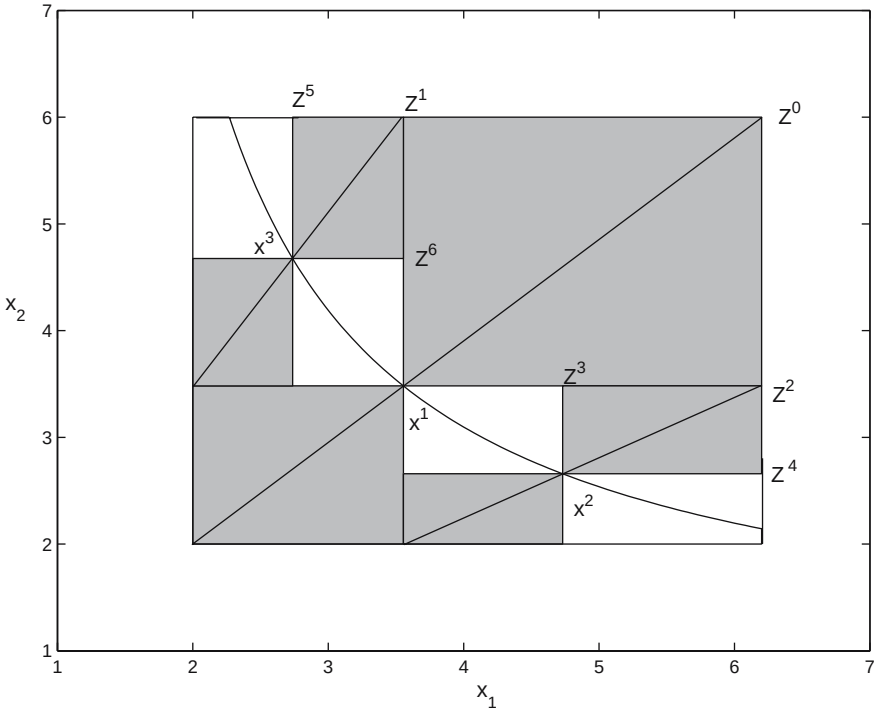

Figure 6. Partition process for Example 1.

EXAMPLE 2.

$$\begin{aligned} \max \quad & f(x) = (x_1^2 - (1/6)x_2^3 + x_1 + 43x_2) \\ \text{s.t.} \quad & g_1(x) = 2x_1 + 8x_2 \leqslant 44, \\ & g_2(x) = 7x_1 + 4x_2 \leqslant 48, \\ & x \in X = \{x \mid 2 \leqslant x_i \leqslant 6, \ \ i = 1, 2\}. \end{aligned}$$

The objective function of this example is nonconvex and increasing. Note that the example is a DC program since $f(x)$ is a difference of two convex functions and g_1 and g_2 are linear. The problem has two local optimal solution: $x^1_{\text{loc}} = (4.3333, 4.4167)^T$ with $f(x^1_{\text{loc}}) = 198.6685$ and $x^2_{\text{loc}} = (2, 5)^T$ with $f(x^2_{\text{loc}}) = 200.1667$.

Take $p = 2$ in the convexification transformation (11) and set $\eta = \epsilon = 10^{-3}$. At the first iteration, a boundary point $x^1 = (4.5413, 4.0526)^T$ is calculated. Starting from x^1, a local solution $x^1_{\text{loc}} = (4.3333, 4.4167)^T$ is obtained by using the local search procedure. Set $x_{\text{best}} = (4.3333, 4.4167)^T$ and $f_{\text{best}} = 198.6685$. Partition the initial domain into two new subboxes:

$$X_1^1 = [(2, 4.0526)^T, (4.5413, 5)^T],$$
$$X_2^1 = [(4.5413, 2)^T, (5.7143, 4.0526)^T].$$

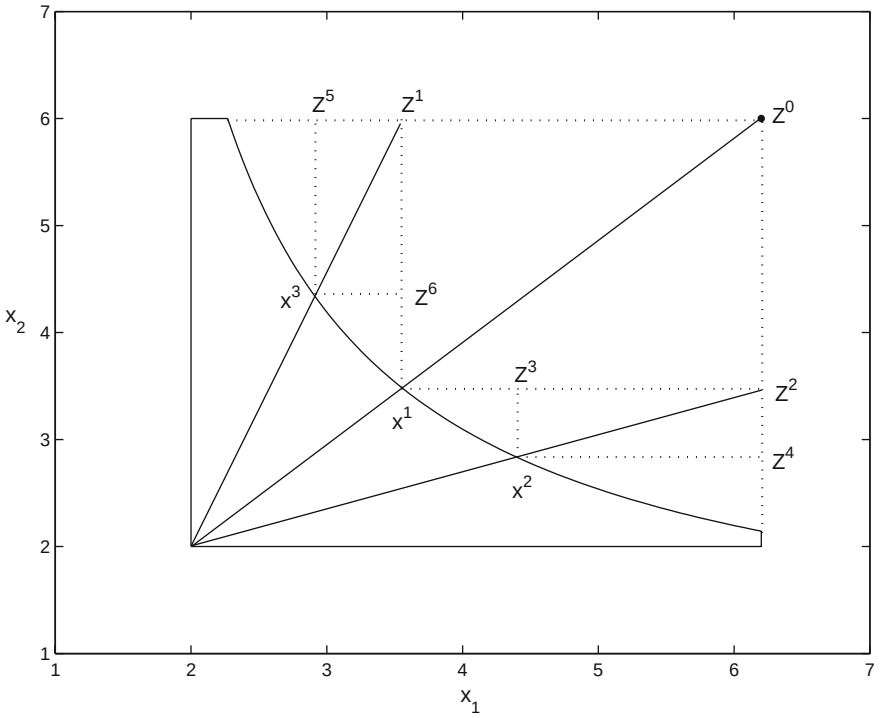

Figure 7. Polyblock approximation for Example 1.

The upper bounds on X_1^1 and X_2^1 are 200.8376 and 193.4519, respectively. The subbox X_2^1 is removed as the upper bound $193.4519 < f_{\text{best}}$. Set $X^2 = \{X_1^1\}$. At the second iteration, a boundary point $x^2 = (3.5212, 4.6197)^T$ is calculated. Starting from x^2, a new local solution $x_{\text{loc}}^2 = (2, 5)^T$ is obtained. Update $x_{\text{best}} = (2, 5)$ and $f_{\text{best}} = 200.1667$. Partition X_1^1 into two new subboxes:

$$X_1^2 = [(2, 4.6197)^T, (3.5212, 5)^T],$$
$$X_2^2 = [(3.5212, 4.0526)^T, (4.5414, 4.6197)^T].$$

The upper bounds on X_1^2 and X_2^2 are 200.1667 and 200.1199, respectively. Both subboxes X_1^2 and X_2^2 are removed because the upper bounds are less than or equal to f_{best}. Therefore, $X^3 = \emptyset$ and the algorithm stops at the global optimal solution $x_{\text{best}} = (2, 5)^T$.

5. Computational Results

Algorithm 1 was programmed by Fortran 90 and run on a Pentium IV PC (2 GHz and 256 Mb RAM) for three types of monotone optimization test problems. The objective functions of the test problems are as follows.

Problem 1. Polynomial functions

$$f(x)=\sum_{i=1}^{q} p_i \prod_{j\in N_i} x_j^{\alpha_{ij}},$$

where q is an integer number, $p_i \in [0,1]$, $N_i \subset \{1,\ldots,n\}$ with $1 \leqslant |N_i| \leqslant 3$, each element of N_i is randomly generated from $\{1,\ldots,n\}$, and α_{ij}'s are randomly generated from $\{1,2,3\}$. In our testing, q is taken to be n.

Problem 2. Quadratic functions $f(x) = (1/2)x^T Ax$, where $A = (a_{ij})_{n\times n}$ with a_{ij} randomly generated from $[0, 10]$.

Problem 3. Reliability functions of complex networks with $n = 5, 7, 8, 12$. Details of the network structures and the expressions of the reliability functions can be found in [1, 12].

Two types of constraint functions are considered in our testing.

- Linear function $g_i(x) = \sum_{j=1}^{n} b_{ij}x_j$, where b_{ij}'s randomly generated from $[0, 20]$.
- Quadratic function $g_i(x) = (1/2)x^T A_i x + c_i^T x$, $i = 1,\ldots,m$, where the entries of A_i are randomly generated from $[0, 5]$ and the components of c_i are from $[0, 20]$.

For all the test problems, $l_j = 1$, $u_j = 5$, $j = 1,\ldots,n$. The right-hand side is set to be $b_i = g_i(l) + r \times (g_i(u) - g_i(l))$, $i = 1,\ldots,m$, where $r = 0.6$ for linear constraints and $r = 0.2$ for quadratic constraints. In our implementation of the algorithm, the variable transformation (11) is used in problem (SP_t). The parameters p and q in (11) are initially taken to be 1. If the outer approximation method identifies that a vertex of the outer polyhedron is inside S_p, then the outer approximation method is restarted by setting $p := p + 1$. The boundary point x_b in Step 1 is calculated by the bisection method. The tolerance parameters are set as $\eta = \epsilon = 0.01$ for Problems 1 and 2 and $\eta = \epsilon = 10^{-5}$ for Problem 3.

Numerical results for Problems 1–3 are summarized in Tables I–VI, where average CPU seconds, average number of iterations and average number of subboxes are obtained by running the algorithm for 20 test problems. Comparing Tables I–VI, we see that the problems with general quadratic constraints are much more difficult than those with linear constraints in terms of the average CPU time of the algorithm. This is mainly because the polyhedral outer approximation to the transformed subproblem (SP_t) produces better upper bounds for problems with linear constraints than those with quadratic constraints.

Table I. Numerical results for Problem 1 with linear constraints

n	m	Average CPU time (seconds)	Average number of iterations	Average number of subboxes
5	1	0.06	2	11
7	1	1.2	3	17
10	1	39.2	3	21
5	5	0.2	3	16
7	5	1.3	2	16
10	5	136.2	4	36

Table II. Numerical results for Problem 1 with quadratic constraints

n	m	Average CPU time (seconds)	Average number of iterations	Average number of subboxes
5	1	0.3	4	28
7	1	2.7	3	28
10	1	183.6	3	32
5	5	0.3	3	28
7	5	5.0	4	35
10	5	224.2	2	23

Table III. Numerical results for Problem 2 with linear constraints

n	m	Average CPU time (seconds)	Average number of iterations	Average number of subboxes
5	1	0.09	1	13
7	1	20.8	9	107
5	5	0.8	5	45
7	5	102.4	22	247

Table IV. Numerical results for Problem 2 with quadratic constraints

n	m	Average CPU time (seconds)	Average number of iterations	Average number of subboxes
5	1	1.3	8	64
7	1	548.5	61	713
5	5	1.9	10	80
7	5	3689.3	180	2128

Table V. Numerical results for Problem 3 with linear constraints

n	m	Average CPU time (seconds)	Average number of iterations	Average number of subboxes
5	1	0.023	1	9
7	1	0.4	3	27
8	1	3.3	5	57
12	1	52.3	5	60
5	5	0.04	2	12
7	5	0.7	2	21
8	5	31.6	42	504
12	5	210.5	3	50

Table VI. Numerical results Problem 3 with quadratic constraints

n	m	Average CPU time (seconds)	Average number of iterations	Average number of subboxes
5	1	0.03	1	9
7	1	0.6	1	19
8	1	4.3	2	36
12	1	249.5	2	51
5	5	0.06	1	13
7	5	0.7	1	24
8	5	6.9	5	66
12	5	294.1	1	43

6. Conclusions

We have presented in this paper a new branch-and-bound algorithm for monotone optimization problems. The domain is partitioned iteratively into a union of subboxes that covers the boundary of the feasible region. On each of the subboxes, an upper bounds of the objective function is computed by a convexification and outer approximation procedure. Local search is used to find an improved lower bound, thus speeding up the convergence of the algorithm. Computational results for randomly generated test problems have been reported to show the feasibility and efficiency of the proposed algorithm.

References

1. Abraham, J.A. (1979), An improved algorithm for network reliability, *IEEE Transactions on Reliability* R-28, 58–61.
2. Bazaraa, M.S., Sherali, H.D. and Shetty, C.M. (1993), *Nonlinear Programming: Theory and Algorithms*, Wiley, New York.
3. Benson, H.P. (1996), Deterministic algorithm for constrained concave minimization: A unified critical survey, *Naval Research Logistics* 43, 765–795.

4. Birolini, A. (1999), *Reliability Engineering: Theory and Practice*, Springer-Verlag, New York.
5. Geoffrion, A.M. (1967), Solving bicriteria mathematical programs, *Operations Research* 15, 38–54.
6. Hochbaum, D. (1995), A nonlinear knapsack problem, *Operations Research Letters* 17, 103–110.
7. Hoffman, K.L.A. (1981), A method for globally minimizing concave functions over convex set, *Mathematical Programming* 20, 22–32.
8. Horst, R. and Tuy, H. (1993), *Global Optimization: Deterministic Approaches*, Springer-Verlag, Heidelberg.
9. Ibaraki, T. and Katoh, N. (1988), *Resource Allocation Problems: Algorithmic Approaches*, MIT Press, Cambridge, MA.
10. Kodialam, M.S. and Luss, H. (1998), Algorithm for separable nonlinear resource allocation problems, *Operations Research* 46, 272–284.
11. Li, D., Sun, X.L., Biswal, M.P. and Gao, F. (2001), Convexification, concavification and monotonization in global optimization, *Annals of Operations Research* 105, 213–226.
12. Li, D., Sun, X.L. and McKinnon, K. (2005), An exact solution method for reliability optimization in complex systems, *Annals of Operations Research* 133, 129–148.
13. Rubinov, A., Tuy, H. and Mays, H. (2001), An algorithm for monotonic global optimization problems, *Optimization* 49, 205–221.
14. Sun, X.L., Luo, H.Z. and Li, D. (2004), Convexification of nonsmooth monotone functions, Technical report, Department of Mathematics, Shanghai University, Shanghai, P. R. China.
15. Sun, X.L., McKinnon, K.I.M. and Li, D. (2001), A convexification method for a class of global optimization problems with applications to reliability optimization, *Journal of Global Optimization* 21, 185–199.
16. Tillman, F.A., Hwuang, C.L. and Kuo, W. (1980), *Optimization of System Reliability*, Marcel Dekker, New York.
17. Tuy, H. (2000), Monotonic optimization: problems and solution approaches, *SIAM Journal on Optimization* 11, 464–494.
18. Tuy, H. and Luc, L.T. (2000), A new approach to optimization under monotonic constraint, *Journal of Global Optimization* 18, 1–15.
19. Tzafestas, S.G. (1980), Optimization of system reliability: A survey of problems and techniques, *International Journal of Systems Science* 11, 455–486.
20. Zhang, J.M. and Sun, X.L. (2003), Convexification approximation method for monotone global optimization, *Communications on Applied Mathematics and Computations* 17, 18–26 (in Chinese).