Journal of Global Optimization **8:** 107–138, 1996.

A Branch-and-Reduce Approach to Global Optimization

HONG S. RYOO and NIKOLAOS V. SAHINIDIS*
Department of Mechanical & Industrial Engineering, The University of Illinois at Urbana-Champaign, 1206 West Green Street, Urbana, Illinois 61801, U.S.A.

(Received 15 October 1994; accepted: 4 September 1995)

Abstract. This paper presents valid inequalities and range contraction techniques that can be used to reduce the size of the search space of global optimization problems. To demonstrate the algorithmic usefulness of these techniques, we incorporate them within the branch-and-bound framework. This results in a *branch-and-reduce* global optimization algorithm. A detailed discussion of the algorithm components and theoretical properties are provided. Specialized algorithms for polynomial and multiplicative programs are developed. Extensive computational results are presented for engineering design problems, standard global optimization test problems, univariate polynomial programs, linear multiplicative programs, mixed-integer nonlinear programs and concave quadratic programs. For the problems solved, the computer implementation of the proposed algorithm provides very accurate solutions in modest computational time.

Key words: Global optimization, range reduction, branch-and-bound, polynomial programming, multiplicative programming, mixed-integer nonlinear programming, quadratic programming.

1. Introduction

The problem considered in this paper is:

$$
(P): \qquad \begin{array}{ll} \text{glob min} & f(x) \\ \text{s.t.} & g(x) \leq 0 \\ & x \in X \end{array}
$$

where $f : X \rightarrow \Re$, $g : X \rightarrow \Re^{m_P}$, and $X \subset \Re^{n_P}$.

It will be assumed that a solution exists. We are interested in finding the solution and in proving its global optimality. Global optimization problems are *NP*-hard problems. Even proving that a solution is not a local minimizer for certain problems is *NP*-complete (e.g., Murty and Kabadi [44], Pardalos and Schnitger [49]). In the absence of convexity, traditional (local) nonlinear programming methods may fail to locate the global optimum of P. Recently, however, there has been a growing interest in global optimization problems as they have numerous applications in various fields: structural and shape optimization (Rozvany [52], Haftka and Gurdal [21]), mechanical equipment and parts design (Wilde [73], Papalambros and Wilde

* Address all correspondence to this author (e-mail: `nikos@uiuc.edu`).

[46], Anagnostou *et al.* [3]), analysis and design of control systems (Balakrishnan [4]), integrated circuit design (Brayton *et al.* [6]), prediction of molecular structures (Pardalos *et al.* [50]) and complex process design (Floudas and Pardalos [18]).

Motivated by the large number of applications, a number of global optimization algorithms have been developed. These algorithms can be classified as either stochastic or deterministic depending upon whether they involve stochastic elements or not. Stochastic methods converge to the global optimum with a probability approaching one as their running time goes to infinity (Törn and Zilinskas [65], Schoen [56]). Deterministic approaches, on the other hand, take advantage of the mathematical structure of the problem and often guarantee finite convergence within a prespecified level of accuracy. Deterministic approaches include branch-and-bound (Falk and Soland [16], Tuy and Horst [69]), cutting plane algorithms (Tuy [66], Hillestad and Jacobsen [26], Tuy [67]) and hybrid schemes that involve cutting planes, decomposition and branch-and-bound. Recent surveys of deterministic methods can be found in Horst and Tuy [28] and Pardalos and Horst [48].

Shortly after the development of branch-and-bound methods for integer programs, the application of the same principles was suggested for continuous global optimization problems (Falk and Soland [16], Soland [59]). Branch-and-bound methods develop lower and upper bounds of the optimal objective function value over subregions of the search space. Optimality and feasibility criteria are employed in order to exclude certain subregions from further consideration while other subregions are dynamically refined. Although branch-and-bound techniques have led to the successful solution of large-scale integer problems, a great deal of difficulty seems to arise in the context of continuous problems. Continuous problems solved to proven global optimality so far have typically involved only a few variables and constraints.

To improve the performance of branch-and-bound-based algorithms, several approaches have been suggested. First, the observation that globally optimal solutions are often found early in the search has motivated the development of sufficient conditions under which a local minimizer is also a global minimizer (e.g., Falk [15], McCormick [40], Hiriart-Urruty [27], Phillips and Rosen [51], Danninger [9], Neumaier [45]). Another attempt for improving the performance of branch-and-bound methods has been through the development of tight relaxations (e.g., McCormick [39], Sherali and Alameddine [57]). In other approaches, range reduction techniques have been developed to confine the search to a smaller space. By construction, relaxations developed over the resulting, smaller feasible spaces are tighter, and the convergence of the algorithm is accelerated (Thakur [63], Hansen *et al.* [23], Hamed and McCormick [22], Lamar [34], Ryoo and Sahinidis [54]). Finally, realizing the importance of the subdivision strategy employed in branch-and-bound algorithms, Tuy [68] presented subdivision strategies that are less restrictive from the theoretical point of view and improve the practical performance of the algorithms substantially.

The approach taken in this paper aims at improving the performance of global optimization algorithms by means of effective range reduction techniques. Although these techniques can be used in conjunction with any global optimization algorithm, we demonstrate their effectiveness within the branch-and-bound framework. We further develop the algorithm of Ryoo and Sahinidis [54], provide convergence proofs, develop specialized algorithms for univariate polynomial programs and linear multiplicative programs, describe a highly efficient implementation and present extensive computational results. Section 2 provides necessary background material on branch-and-bound. Section 3 presents optimality-based valid inequalities for problem P. These inequalities form the basis for the development of range reduction mechanisms. Applied at each node of the search tree, range reduction converts a standard branch-and-bound algorithm into a *branch-and-reduce* global optimization algorithm. The details of the resulting global optimization algorithm are discussed in Section 4 and a convergence analysis is provided. An example is presented in Section 5. Specialized branch-and-reduce algorithms for univariate polynomial problems and linear multiplicative programming problems are developed in Sections 6 and 7, respectively. Section 8 describes BARON, the computer implementation of the proposed algorithm, and presents extensive computational experiments with various classes of problems. The problems tested include engineering design problems, standard global optimization test problems, polynomial programs, concave quadratic minimization programs, linear multiplicative programs and mixed-integer nonlinear programs. Finally, conclusions are provided in Section 9.

2. Preliminaries

Branch-and-bound (BB) is one of the most commonly used techniques in global optimization. Here, branching refers to successive partitioning (or subdivision) of the feasible domain, and bounding refers to the computation of lower and upper bounds, L and U, respectively, for the global optimum. The main feature of BB is its ability to delete inferior subsets of the original search space during the iteration process. At any iteration, subregions whose lower bounds, L_i, are no better than the current upper bound ($L_i \geq U$) can be deleted from the search. A typical BB for solving P is as follows:

ALGORITHM 1. *Branch-and-Bound* (at iteration k):

Step 1. Partitioning:

Partition the search region into finitely many subregions, M_l, $l = 1, \ldots, s$.

Step 2. Bounding:

Select a subregion, M_i, and determine lower and upper bounds, L_i and U_i, such that $L_i \leq f(x) \leq U_i$, $\forall x \in M_i$.

Step 3. Global Bounding:

Set $L^{(k)} = \min\{L_l : l = 1, \ldots, s\}$ and $U^{(k)} = \min\{U_l : l = 1, \ldots, s\}$.

Step 4. Termination and Subproblem Selection:

If $L^{(k)} = U^{(k)}$,

Stop. An optimal solution has been found.

Otherwise,

Select a subregion.

Set $k \leftarrow k + 1$.

Repeat the process from Step 1.

DEFINITION 1. A BB algorithm is called *finite* if $L^{(k)} = U^{(k)}$ for some $k < \infty$.

If a BB algorithm is finite, a globally optimal solution is obtained at termination of the algorithm. On the other hand, if BB does not terminate in a finite number of steps, one needs to address the limit behavior.

DEFINITION 2. A BB algorithm is called *convergent* if $\lim_{k\to\infty} |U^{(k)} - L^{(k)}| = 0$.

There are three crucial operations in BB which determine the convergence properties of the algorithm. These are partitioning (branching), bounding and selection and correspond to steps 1, 2 and 4, respectively, in the above algorithm.

DEFINITION 3. (Definition IV.4 of Horst and Tuy [28]). A bounding operation is called *consistent* if at every step any unfathomed partition element is capable of further refinement and any infinitely decreasing sequence M_{i_q} of successively refined partition elements satisfies $\lim_{q\to\infty} |U_i - L_{i_q}| = 0$.

The last relationship requires that, whenever an infinitely decreasing sequence of partition sets emanating from a parent set converges to a certain limit set, the lower bound over this limit set also converges to the upper bound of the objective over the parent set. This condition is implied if the lower bound over the limit set converges to the upper bound over this limit set: $\lim_{q\to\infty} |U_{i_q} - L_{i_q}| = 0$.

DEFINITION 4. (Definition IV.6 of [28]). The selection operation is said to be *bound improving* if at least one partition element where the actual lower bound is attained is selected for further partition.

THEOREM 1. *(Theorem IV.3 of [28]). In the infinite BB procedure, suppose that the bounding operation is consistent and the selection operation is bound improving. Then, the procedure is convergent.*

In this paper, it will be assumed that a convergent (or finite) branch-and-bound algorithm is readily available for problem P. Although several such algorithms exist, often times the performance of a BB algorithm is not satisfactory, especially when the gap between the initial upper and lower bounds is large. In such a case, BB exhibits a slow, asymptotic convergence behavior. This behavior necessitates the development of methods to expedite convergence. To that effect, the following section develops range reduction mechanisms which can be used to detect and delete from further consideration inferior parts of the search space.

3. Valid Inequalities and Range Reduction

Consider the following relaxation of P:

$$\begin{aligned}(R):\qquad & \min\ \bar{f}(x)\\ & \text{s.t. } \bar{g}(x) \leq 0\\ & x \in \bar{X}\end{aligned}$$

where $\bar{f} : \bar{X} \to \Re$, $\bar{g} : \bar{X} \to \Re^m$, $X \subseteq \bar{X} \subset \Re^n$, such that, for any x feasible to P, $\bar{f}(x) \leq f(x)$, and where $\{x :\ g(x) \leq 0,\ x \in X\} \subseteq \{x : \bar{g}(x) \leq 0,\ x \in \bar{X}\}$.

ASSUMPTION 1. R is a convex programming problem.

ASSUMPTION 2. A dual-adequate algorithm is available for solving R (An algorithm is called *dual-adequate* if it provides the dual solution in addition to the primal solution).

Now, consider the perturbation of problem R:

$$\begin{aligned}(R_y):\qquad \varphi(y) = & \min\ \bar{f}(x)\\ & \text{s.t. } \bar{g}(x) \leq y\\ & x \in \bar{X}\end{aligned}$$

As R is convex, so is R_y for any y. Therefore, traditional nonlinear optimization techniques can be used to solve R_y to global optimality since a local minimizer of R_y is also a global optimizer. The following lemma is trivial to prove by contradiction.

LEMMA 1. *Let constraint $\bar{g}_i(x) \leq 0$ be active at a solution of problem R. Then, $\bar{g}_i(x) \leq y_i$ for $y_i \leq 0$ is also active at the solution of problem R_y.*

The next lemma summarizes well-known properties of the perturbation problem.

LEMMA 2. *(e.g., Theorem 5.4 of Minoux [41]). Assuming that R has an optimum of finite value, $\mu \in \Re^m$ is a saddle-point multiplier if and only if the hyperplane with equation $z = \varphi(0) - \mu \cdot y$ is a supporting hyperplane at $y = 0$ of the graph* *of the perturbation function $\varphi(y)$. In other words, μ is a saddle-point multiplier if and only if*

$$\forall y \in \Re^m : \varphi(y) \geq \varphi(0) - \mu \cdot y$$

This lemma ensures that a saddle point exists and the perturbation function is convex if R is a convex program satisfying standard constraint qualifications (e.g., [41]). On the basis of Lemmas 1 and 2, valid inequalities for P can be derived.

DEFINITION 5. Let U be a known upper bound for P. An inequality is called *valid* for P if it does not exclude any solutions of P with objective function values better than U.

THEOREM 2. *Let R be a convex optimization problem with an optimal objective function value of L and consider a constraint $\bar{g}_i(x) \leq 0$ that is active at the solution of problem R with a dual multiplier value of $\mu_i > 0$. Let U be a known upper bound for problem P. Then, the following constraint is valid for P:*

$$\bar{g}_i(x) \geq -(U - L)/\mu_i \tag{1}$$

Proof. Consider the following perturbation of problem P:

$$(P_y): \qquad \Phi(y) = \text{glob min } f(x) \quad \text{s.t. } g(x) \leq y, \quad x \in X$$

Obviously, R_y is a convex relaxation of P_y for any y. Therefore, $\varphi(y) \leq \Phi(y)$, and a valid underestimator of $\varphi(y)$ is also a valid underestimator of $\Phi(y)$. Now consider the perturbation problem R_y where only the right hand side of constraint $\bar{g}_i(x) \leq y_i$ is perturbed. From Lemma 2, we know that an underestimator for $\varphi(y_i)$ is provided by its linear support $L - \mu_i y_i$. Therefore, for any y_i, we have $L - \mu_i y_i \leq \varphi(y_i) \leq \Phi(y_i)$. Now, requiring the value of $\Phi(y_i)$ to be no higher than the already known upper bound U yields $L - \mu_i y_i (\leq \Phi(y_i)) \leq U$. Finally, since $\bar{g}_i(x) \leq y_i$ is active for $y_i = 0$, the constraint will also be active in the solution of R_y for any $y_i \leq 0$ (Lemma 1). Therefore, $L - \mu_i \bar{g}_i(x) = L - \mu_i y_i \leq U$. This constraint is valid for all feasible values of x, as for any feasible x there exists a corresponding y_i for which $\bar{g}_i(x) \leq y_i$ is active. □

In the above proof, (1) – which might be nonconvex – was derived as a relaxation of the objective function cut $\bar{f}(x) \leq U$ – which is a convex constraint. However, (1) is easier to work with computationally as it is often linear as shown next.

COROLLARY 1. *Let R be a convex programming problem with an optimal objective function value of L and consider a linear constraint $a_i^t x - b_i \leq 0$ that is active at the solution of problem R with a dual multiplier value of $\mu_i > 0$. Let U be a known upper bound for problem P. Then, the following constraint is valid for P:*

$$a_i^t x \geq b_i - (U - L)/\mu_i. \tag{2}$$

COROLLARY 2. *Let R be a convex programming problem with an optimal objective function value of L and consider a range constraint* $x_j - x_j^U \leq 0$ *that is active at the solution of problem R with a dual multiplier value of* $\lambda_j > 0$. *Let U be a known upper bound for problem P. Then, the following constraint is valid for P:*

$$x_j \geq x_j^U - (U - L)/\lambda_j. \tag{3}$$

COROLLARY 3. *Let R be a convex programming problem with an optimal objective function value of L and consider a range constraint* $x_j^L - x_j \leq 0$ *that is active at the solution of problem R with a dual multiplier value of* $\lambda_j > 0$. *Let U be a known upper bound for problem P. Then, the following constraint is valid for P:*

$$x_j \leq x_j^L + (U - L)/\lambda_j. \tag{4}$$

The development of valid inequalities (1)-(4) is based on the set of constraints that are active at the solution of the relaxed problem R. Valid inequalities, however, can also be derived from constraints that are not active in the solution of R by *probing* at certain parts of the feasible region where constraints might become active. In particular, one can temporarily fix the right-hand-side of an inactive constraint at some point, solve the partially restricted relaxed problem and obtain a linear support of the perturbation function at the solution point. This support can then be used to derive a valid inequality:

THEOREM 3. *Let R be a convex programming problem and consider a linear constraint* $a_i^t x - b_i \leq 0$ *that is not active at the solution of R. Let U be a known upper bound for problem P. Solve R after fixing* $a_i^t x$ *at* b_i, *i.e., after adding the constraint* $b_i \leq a_i^t x$ *in the formulation. Let Z be the optimal objective function value of this partially restricted relaxed problem. If a positive dual multiplier* μ_i *is obtained for constraint* $b_i \leq a_i^t x$ *in the solution of the new problem, then the following constraint is valid for P:*

$$a_i^t x \leq b_i + (U - Z)/\mu_i. \tag{5}$$

The proof of this theorem is omitted as it is similar to that of Theorem 2. Note only that the optimal value Z of the partially restricted relaxed problem may not provide a valid lower bound for P. In fact, (5) provides a useful constraint only whenever $Z > U$.

COROLLARY 4. *Let R be a convex programming problem and consider a range constraint* $x_j - x_j^U \leq 0$ *that is not active at the solution of R. Let U be a known upper bound for problem P. Solve R after fixing* x_j *at* x_j^U, *i.e., after adding* $x_j^U \leq x_j$ *in the formulation. Let Z be the optimal objective function value of this partially restricted relaxed problem. If a positive dual multiplier* λ_j *is obtained for constraint* $x_j^U \leq x_j$ *in the solution of the new problem, then the following constraint is valid for P:*

$$x_j \leq x_j^U + (U - Z)/\lambda_j. \tag{6}$$

COROLLARY 5. *Let R be a convex programming problem and consider a range constraint $x_j - x_j^L \geq 0$ that is not active at the solution of R. Let U be a known upper bound for problem P. Solve R after fixing x_j at x_j^L, i.e., after adding $x_j \leq x_j^L$ in the formulation. Let Z be the optimal objective function value of this partially restricted relaxed problem. If a positive dual multiplier λ_j is obtained for constraint $x_j \leq x_j^L$ in the solution of the new problem, then the following constraint is valid for P:*

$$x_j \geq x_j^L - (U - Z)/\lambda_j. \tag{7}$$

The valid inequalities derived in this section and the range reduction mechanisms based on them are summarized in Tables I and II. We use μ to denote the optimal dual multipliers of linear/nonlinear constraints and λ to denote the optimal multipliers (reduced costs) of simple variable bounds (range constraints). These valid inequalities were derived based on the optimal solution of the relaxed problem and by using an optimality argument. For this reason, they will be referred to as *optimality-based* valid inequalities. Although they may exclude solutions that are feasible to P, they do not exclude any solutions of P with objective function values better than U. Also, as (1)-(7) reduce the range of constraints and variables, they will be referred to as optimality-based *range reduction mechanisms*.

TABLE I. Valid inequalities derived from active constraints.

Active Constraint	Requirement	Valid Inequality
$\bar{g}_i(x) \leq 0$	$\mu_i > 0$	$\bar{g}_i(x) \geq -(U - L)/\mu_i$
$a_i^t x - b_i \leq 0$	$\mu_i > 0$	$a_i^t x \geq b_i - (U - L)/\mu_i$
$x_j \leq x_j^U$	$\lambda_j > 0$	$x_j \geq x_j^U - (U - L)/\lambda_j$
$x_j^L \leq x_j$	$\lambda_j > 0$	$x_j \leq x_j^L + (U - L)/\lambda_j$

TABLE II. Valid inequalities derived from inactive constraints after probing.

Inactive Constraint	Requirement	Valid Inequality
$a_i^t x - b_i \leq 0$	Add $b_i \leq a_i^t x$ to R. Solve R and obtain Z. $\mu_i > 0$	$a_i^t x \leq b_i + (U - Z)/\mu_i$.
$x_j \leq x_j^U$	Add $x_j^U \leq x_j$ to R. Solve R and obtain Z. $\lambda_j > 0$	$x_j \leq x_j^U + (U - Z)/\lambda_j$
$x_j^L \leq x_j$	Add $x_j \leq x_j^L$ to R. Solve R and obtain Z. $\lambda_j > 0$	$x_j \geq x_j^L - (U - Z)/\lambda_j$

Remark 1. Since R was assumed to be a convex problem, it follows that (1) is a nonconvex (reverse convex) constraint if the corresponding inequality of R is nonlinear. Therefore, adding (1) to the relaxation destroys convexity. Still, (1) – as well as $\bar{f}(x) \leq U$ and $f(x) \leq U$ – can be used for tightening variable bounds. In general, it is possible to use constraints along with feasibility arguments to reduce variable ranges (Hansen *et al.* [23], Hamed and McCormick [22], Ryoo [53], Ryoo and Sahinidis [54]). This process will be referred to as *feasibility-based* range reduction.

Remark 2. Theorem 3 applies equally well when probing is done to any point b_i', not necessarily equal to the right-hand-side of the constraint. In particular, probing can be applied by adding, for example, the constraint $b_i' \leq a_i^t x$ into the formulation of R for any $b_i' \leq b_i$. Following the solution of problem R, one possibility is to use parametric optimization techniques to calculate the optimal value of the objective function as the value of a certain constraint is changed. This process can be stopped at the point where the objective becomes equal to the known upper bound U. At that point, the support of the perturbation function can be used to derive a new valid inequality.

4. A Branch-and-Reduce Global Optimization Algorithm

It should be obvious that the range reduction techniques of the previous section can be used to preprocess a global optimization problem before the use of any global optimization algorithm. In the context of a branch-and-bound algorithm, range reduction can be used to improve the performance of the bounding procedure at *every* node of the search tree. The following are the steps of the proposed algorithm:

ALGORITHM 2. *Branch-and-Reduce*:

Initialization Step

Set $k = 0$.

Set the upper bound $U^{(k)} = +\infty$.

Put $R_1 = R$ in the list $ACTIVE$ of active subproblems with a corresponding lower bound of $L_1 = -\infty$.

Go to the main step.

Main Step (at iteration k)

Step 1. Termination:

Set the lower bound $L^{(k)} = \min_{i:R_i \in ACTIVE} \{L_i\}$.

Set $ACTIVE \leftarrow ACTIVE \backslash \{R_j\}$ for all R_j with $L_j \geq U^{(k)}$.

If $ACTIVE = \emptyset$,

Stop. The current best solution is optimal.

Otherwise,

Set $k \leftarrow k+1$, $U^{(k)} \leftarrow U^{(k-1)}$ and $L^{(k)} \leftarrow L^{(k-1)}$.

Go to Step 2.

Step 2. Subproblem Selection:

Select R_i from $ACTIVE$ according to a node selection rule.

Set $ACTIVE \leftarrow ACTIVE \backslash \{R_i\}$.

Go to Step 3.

Step 3. Pre-processing:

Tighten variable bounds for R_i using feasibility-based range reduction.

Go to Step 4.

Step 4. Bounding:

Solve R_i, or bound its solution from below. Let L_i be this lower bound ($L_i = +\infty$ if R_i is infeasible.)

If the solution, x^i, found for R_i is feasible to P and $f(x^i) < U^{(k)}$,

Update $U^{(k)} \leftarrow f(x^i)$.

Make x^i the current best solution: $x^* \leftarrow x^i$.

If $L_i \geq U^{(k)}$,

Go to Step 1.

Otherwise,

Go to Step 5.

Step 5. Optional Upper Bounding:

Apply local search heuristics to find a better feasible solution, x^h, for P. If successful,

Update $U^{(k)} \leftarrow f(x^h)$.

Make x^h the current best solution: $x^* \leftarrow x^h$.

Go to Step 6.

Step 6. Post-processing:

Strengthen the bounds of variables using optimality-based and feasibility-based range reduction.

If the range reduction was successful in reducing the range of at least one variable of R_i by at least a prespecified amount $\delta > 0$, then:

Reconstruct R_i, using the new variable bounds.

Go to Step 4.

Otherwise,

Go to Step 7.

Step 7. Partitioning:

Apply a branching rule to R_i: obtain a set of new subproblems R_{i_1}, R_{i_2}, ..., R_{i_q} and place them on $ACTIVE$.

Go to Step 1.

4.1. COMPONENTS OF THE BRANCH-AND-REDUCE ALGORITHM

4.1.1. *Selection Operation*

Selection of a subproblem is accomplished by means of the *best-bound-first* rule:

Operation *Node Selection:*

Select a subproblem R_i with $i \in arg\min_{j:R_j \in ACTIVE}\{L_j\}$.

Therefore, a subproblem where the actual lower bound of the previous iteration is attained is always selected for further refinement. This node selection rule is bound improving by definition. Bound improvingness will, in general, be required for convergence and is not shared by other branching rules such as depth-first.

4.1.2. *Partitioning Operation*

The partitioning operation is required to satisfy $\bar{X}_{i_q} \subset \bar{X}_i$ for all q. This requirement is met by several partitioning schemes: conical, simplicial or rectangular (Tuy *et al.* [70], Tuy [68]). Any of them can be used in the context of the above algorithm. We prefer rectangular subdivisions for their simplicity. Let x^k be an optimal solution of the relaxed subproblem R_i selected in Step 2 at iteration k of the algorithm. Also, let x^* be the incumbent solution and K be a pre-specified positive integer. The following is the rectangular partitioning operation of the algorithm:

Operation *Partitioning:*

Select a variable x_j which is "mostly responsible" for the difference $U_i - L_i$.

if $k = NK$ $(N = 1, 2, ...)$, **then**

set $\omega = (x_j^{L,k} + x_j^{U,k})/2$

else

if $x_j^{L,k} < x_j^* < x_j^{U,k}$ **then** set $\omega = x_j^*$ **else** set $\omega = x_j^k$

endif.

Create two subproblems by subdividing $[x_j^{L,k}, x_j^{U,k}]$ into $[x_j^{L,k}, \omega]$ and $[\omega, x_j^{U,k}]$.

The first step in the above operation is to select the branching variable. This should be done in a way that will lead to the largest possible reduction of the relaxation gap. The exact variable selection rule will therefore depend on the bounding procedure used. If, for example, separable relaxations are used (e.g., [38]), one can select a variable corresponding to a nonconvex term in P whose underestimator in R_i has the largest distance from the nonconvex term at the solution of R_i. A simpler variable selection rule is to select a variable corresponding to the largest range: $j \in arg\max_{j'}(x_{j'}^{U,k} - x_{j'}^{L,k})$. Once the branching variable is selected by standard means, *Partitioning* uses a combined maximum-deviation/bisection/incumbent-branching rule for branching point selection. In a typical iteration, the solution of R_i is used as the branching point. Bisection ensures a reasonable reduction in the sizes of the descendant subrectangles every K iterations. Finally, whenever possible, the branching point is positioned in a way that eliminates the gap at the incumbent solution. For any given x^*, this last modification of the standard maximum deviation branching point will occur no more than n times in any nested sequence of subdivisions. The following is immediate.

PROPOSITION 1. *Operation Partitioning guarantees* $\bar{X}_{i_q} \subset \bar{X}_i$ *for all* q.

4.1.3. *Bounding Procedure*

The bounding procedure is comprised of Steps 3, 4, 5 and 6 of the algorithm. The underestimating functions and relaxed problems of Step 4 are required to possess the following properties:

REQUIREMENT 1. Let $G_i := \{x \in \Re^n : \bar{g}_l(x) \leq 0,\ l = 1, \ldots, m\}$ for some subproblem R_i. Then, we have $G_{i_q} \subseteq G_i$ for all q descendants of R_i.

REQUIREMENT 2. $L_i = U_i$ if $x_j^U = x_j^L$ for all nonconvex variables of P in the relaxed problem R_i. (Here, the term *nonconvex* is used to denote those variables that appear in nonconvex terms in P.)

From Requirement 1, Assumption 1 and Proposition 1, it follows that:

PROPOSITION 2. $M_{i_q} = G_{i_q} \cap \bar{X}_{i_q} \subset G_i \cap \bar{X}_i = M_i$ *and* $L_{i_q} \geq L_i$ *for all* q.

The construction of relaxed problems that satisfy the above requirements can be done in more than one way (e.g., Falk and Soland [16], McCormick [38, 39], Sherali and Alameddine [57]). We prefer to use factorable programming techniques [38, 39] for their simplicity. In all of the above lower bounding approaches, the tightness of the lower bound directly depends on the tightness of the variable bounds. For this reason, Step 3 applies feasibility-based range reduction techniques to obtain tighter variable bounds. Similarly, optimality-based and feasibility-based range reduction mechanisms are used in Step 6 after the solution of the current relaxed problem.

Finally, Step 5 performs the optional upper bounding operation. Any problem specific heuristic or any other global optimization method – for example, a stochastic optimization method – can be incorporated in this step. Successful variable bounds tightening in Steps 3 and 6 will facilitate the calculation of good feasible solutions in Step 5. The reverse is also true as a tighter upper bound, U, on the optimal objective implies the possibility of further range reduction through the inequalities of Tables I and II. Additionally, range reduction based on objective function cuts is facilitated by a stronger upper bound U.

PROPOSITION 3. *The branch-and-reduce algorithm does not cycle between Steps 4 and 6.*

Proof. Steps 4 to 6 are repeated only if range reduction is successful in reducing the range of at least one variable of R_i by at least a prespecified amount $\delta > 0$. As the feasible region is bounded ($\bar{X}_i \subseteq \bar{X} \subset \Re^n$), this range reduction can only happen a finite number of times before either R_i is deleted by infeasibility or inferiority (in Steps 4 or 5), or $x_j^U = x_j^L$ for all nonconvex variables, in which case $L_i = U_i$ (from Requirement 2) and subproblem R_i will again be deleted in Step 4. □

Remark 3. Another way to ensure that the algorithm does not cycle between Steps 4 and 6 is to monitor the effect of range reduction on the lower bound of R_i and to return to Step 4 only if the improvement is larger than a certain prespecified positive amount.

4.2. Convergence of the Algorithm

The analysis will be based on Theorem 1 of Section 2 and the above mentioned properties of the partitioning rule and relaxations used. Without loss of generality, we can assume that $\bar{f}$ is continuous over $G \cap \bar{X}$ (follows from convexity).

LEMMA 3. *The bounding operation in the algorithm is consistent.*

Proof. In order to prove the consistency of the bounding operation, we need to prove that for any infinitely nested sequence M_{i_q} generated by the algorithm, we have $\lim_{q\to\infty} |U_{i_q} - L_{i_q}| = 0$. Let x^q be the solution of R_{i_q}. There are two cases to consider:

Case 1. f is (semi) continuous:

Whenever there is a positive underestimation gap, the gap is due to the presence of nonconvex variables with $x_i^{U,q} - x_i^{L,q} > 0$. By the (semi) continuity of f and $\bar{f}$ and by Requirement 2, we have $\lim \bar{f}_q(x^q) = f^* = \lim f(x^q)$ as $(x_i^{U,q} - x_i^{L,q}) \to 0$ for all nonconvex variables meaning that $\lim_{q\to\infty} |U_{i_q} - L_{i_q}| = 0$.

Case 2. f is discontinuous:

(i) First, consider the case in which there is a discontinuity only at x_d, and x_d is the unique globally optimal solution of problem P. Since $(x_i^{U,q} - x_i^{L,q}) \to 0$ as $q \to \infty$ and since f is (semi) continuous at every feasible point other than x_d, we have $\bar{f}_j(x^j) \to f(x^j)$ for all M_{i_j} except M_{i_q} where x_d is contained. Therefore, M_{i_q} will remain as the only unfathomed partition element as $q \to \infty$. Moreover, by the continuity and convexity of $\bar{f}$, we have $\lim_{q\to\infty} \bar{f}_q(x^q) = \bar{f}^* = L_{i_q}$ (Such a limit exists since $\{\bar{f}_q(x^q)\}$ is monotone and bounded above by $f(x^q)$). Now since $x_d = \lim_{q\to\infty} x^q$, as $q \to \infty$, we have (since x_d is feasible to P) $U_{i_q} = f(x_d) = \bar{f}^*(x_d) = L_{i_q}$.

For the case where x_d is one of many globally optimal solutions, since we set U_{i_q} as the lowest upper bound found, $\lim_{q\to\infty} |U_{i_q} - L_{i_q}| = 0$ is ensured as in Case 1. Finally, it should be obvious that for other cases where x_d is not a globally optimal solution, $\lim_{q\to\infty} |U_{i_q} - L_{i_q}| = 0$ is ensured in a similar way as in Case 1.

(ii) The case where there are more than one discontinuities in f eventually reduces to Case 2.(i) as the branch-and-bound procedure is applied, and, therefore, $\lim_{q\to\infty} |U_{i_q} - L_{i_q}| = 0$ will be achieved in the same way as above. □

THEOREM 4. *The branch-and-reduce algorithm converges to the solution of P.*

Proof. The proof follows from Theorem 1 and the following:

(i) The range reduction mechanisms are valid. Hence, even though the range reduction Steps 3 and 6 may eliminate feasible solutions, none of these solutions is better than the current incumbent (Definition 5).
(ii) The bounding operation employed in the algorithm is consistent (Lemma 3).
(iii) The selection operation of the algorithm is bound improving (Definition 4).
(iv) Cycling is not possible in the algorithm (Proposition 3). □

5. Example

The following example is taken from Al-Khayyal and Falk [2]:

$$\text{glob min} \quad -x_1 + x_1x_2 - x_2$$

s.t.

$$-6x_1 + 8x_2 \leq 3 \tag{8}$$

$$3x_1 - x_2 \leq 3 \tag{9}$$

$$(x_1^L,\, x_2^L) = 0 \leq x \leq 5 = (x_1^U,\, x_2^U) \tag{10}$$

This is a jointly constrained bilinear program with a nonextremal boundary optimal solution at $x = (1.16667,\ 0.5)^t$ with $f = -1.08333$. The objective has only one bilinear term (x_1x_2) whose convex envelope is readily available (e.g., McCormick [39], Al-Khayyal and Falk [2]): $\max\{x_2^U x_1 + x_1^U x_2 - x_1^U x_2^U,\ x_2^L x_1 + x_1^L x_2 - x_1^L x_2^L\}$. To avoid non-differentiability in the relaxed problem formulation, we let $x_3 = x_1x_2$ and include two additional constraints:

$$
\begin{aligned}
\min\ & -x_1 - x_2 + x_3 \\
\text{s.t.}\ & \\
& x_2^U x_1 + x_1^U x_2 - x_3 \le x_1^U x_2^U \\
& x_2^L x_1 + x_1^L x_2 - x_3 \le x_1^L x_2^L \\
& \text{Constraints (8)–(10).}
\end{aligned}
$$

A standard branch-and-bound algorithm corresponds to executing Steps 1, 2, 4, 5 and 7 without modifying the maximum deviation branching point as described in Section 4.1.2. This algorithm requires 51 iterations to reduce the difference between the upper and lower bounds to within 10^{-6} despite the fact that the optimal solution was found at the root node by the local minimization step (Step 5). If operation *Partitioning* is used to modify the branching point, then the search requires only 13 iterations using the same termination criterion as above (10^{-6}). We will now illustrate the effect of range reduction mechanisms. In particular, inequalities (3), (4), (6) and (7) will be used for optimality-based range reduction. In addition, feasibility-based range reduction will be done by analyzing the constraints. For example, (8) will be used to generate two valid inequalities: $x_1 \ge (8x_2^L - 3)/6$ and $x_2 \le (6x_1^U + 3)/8$. Finally, the objective function cut $-x_1 + x_1x_2 - x_2 \le U$ yields two inequalities for x_1: $\min\{x_1^L x_2^L - x_2^L - U, x_1^L x_2^U - x_2^U - U\} \le x_1$ and $x_1 \le (U + x_1^U)/x_2^L + 1$. Similar inequalities are obtained for x_2.

When Step 3 of the branch-and-reduce algorithm is entered initially, the bounds on the variables are $0 \le x_1 \le 5$, $0 \le x_2 \le 5$ and $0 \le x_3\ (= x_1x_2) \le 25$. Upon exit from this step, the variable bounds become $0 \le x_1 \le 1.5$, $0 \le x_2 \le 1.5$ and $0 \le x_3 \le 2.25$. (For this example, feasibility-based range reduction has the same effect on bounds as solving $2n$ linear programs to minimize and maximize individual variables.) Using the improved bounds, a relaxed problem is constructed (Step 4). The solution to this relaxed problem is $x^1 = (0.643,\ 0.857,\ 0)^t$ and produces a lower bound of $L = L^1 = -1.5$ and an upper bound $U = -0.949$. Using x^1 as the starting point for local minimization (Step 5) with MINOS [43] yields $U = -1.005$ and $x^* = (0.917,\ 1.062,\ 0.974)^t$. As all range constraints have zero multipliers at x^1, optimality-based range reduction in Step 6 can be applied using only (6) and (7). After selecting x_2 for probing, (6) and (7) improve the bounds on x_2 to $0.004 \le x_2 \le 1.281$. The results hereafter can be seen in Table III. Eventually, after three cycles through Steps 4 to 6, the objective function cut results in $1.125 \le x_1 \le 1.121$. This means that there are no feasible solutions for which

TABLE III. Application of range reduction to the Al-Khayyal and Falk example [2].

Step	x_1^L	x_1^U	x_2^L	x_2^U	x_3^L	x_3^U	L^1	x_1^1	x_2^1	x_3^1	U	x_1^*	x_2^*	x_3^*
0	0	5	0	5	0	25	$-\infty$				$+\infty$			
1	0	5	0	5	0	25	$-\infty$				$+\infty$			
2	0	5	0	5	0	25	$-\infty$				$+\infty$			
3	0	1.5	0	1.5	0	2.25	$-\infty$				$+\infty$			
4	0	1.5	0	1.5	0	2.25	−1.5	0.643	0.857	0	−0.949	0.643	0.857	0
5	0	1.5	0	1.5	0	2.25	−1.5	0.643	0.857	0	−1.005	0.917	1.062	0.974
6	0	1.427	0.004	1.281	0	1.828	−1.5	0.643	0.857	0	−1.005	0.917	1.062	0.974
4	0	1.427	0.004	1.281	0	1.828	−1.392	1.099	0.297	0.004	−1.070	1.099	0.297	0.004
5	0	1.427	0.004	1.281	0	1.828	−1.392	1.099	0.297	0.004	−1.083	1.167	0.5	0.583
6	0.406	1.297	0.004	1.281	0.002	1.662	−1.392	1.099	0.297	0.004	−1.083	1.167	0.5	0.583
4	0.406	1.297	0.004	1.281	0.002	1.662	−1.267	1.097	0.290	0.120	−1.083	1.167	0.5	0.583
5	0.406	1.297	0.004	1.281	0.002	1.662	−1.267	1.097	0.290	0.120	−1.083	1.167	0.5	0.583
6	1.125#	1.121#	0.354	0.363	0.395	0.407	−1.267	1.097	0.290	0.120	−1.083	1.167	0.5	0.583

#: Infeasibility is detected.

the current incumbent can be improved. Hence, the algorithm terminates at the root node with no branching required.

For this example, Table IV compares the algorithm with other approaches. In the table, (m, n), N_{tot}, N_{opt}, N_{mem} and ϵ, respectively, denote the size of the relaxed problem (number of constraints and variables), the total number of iterations, the node where the optimal solution was found, the memory requirements during the search (maximum number of nodes that had to be stored simultaneously), and the termination criterion (difference between upper and lower bounds at termination) used. The entries of the first two rows of Table IV are taken from Sherali and Alameddine [57] and the CPU times of the first three algorithms are all on an IBM 3090 supercomputer whereas the branch-and-reduce time is on an IBM RS/6000 66MHz-Power PC.

TABLE IV. Comparative computational results for the Al-Khayyal and Falk example [2].

Method	(m, n)	N_{tot}	N_{opt}	N_{mem}	CPU sec.	ϵ
Al-Khayyal and Falk [2]	(4,3)	>103	51	20	>55*	0.001
Sherali and Alameddine [57]	(23,5)	11	10	7	14*	0.001
Sherali and Tuncbilek [58]	(23,5)	1	1	1	0.71*	***
Branch-and-Reduce	(4,3)	1	1	1	0.15**	0

*: Computation was done on an IBM 3090 supercomputer.
**: Computation was done on an IBM RS/6000 66MHz-Power PC.
***: A relative criterion of 1% (*i.e.*, $L \geq U - 0.01|U|$) was used for termination, although the lower bound reported was accurate to at least 3 decimal digits.

6. Global Optimization of Univariate Polynomial Functions

In this section we specialize the branch-and-reduce algorithm for the following class of problems:

$$(POLY): \quad \text{glob min } f(x) = \sum_{i=0}^{t} a_i x^i$$
$$\text{s.t.} \quad x^L \leq x \leq x^U$$

where a_i, i=1,...,t, are given real numbers.

The following theorem summarizes important properties of monomials. The proof is immediate.

THEOREM 5. *Let $f_i = a_i x^i$ be a monomial function defined over $x^L \leq x \leq x^U$. f_i is convex over $[x^L, x^U]$ if any of the following conditions holds:*
(i) $x^L = x^U$.
(ii) $x^L \geq 0$ and $a_i \geq 0$.

(iii) $i = 2k \quad (k = 1, 2, 3, \ldots)$ *and* $a_i \geq 0$.
(iv) $x^U \leq 0$, $i = 2k + 1 \quad (k = 1, 2, 3, \ldots)$ *and* $a_i \leq 0$.
(v) $x^L x^U \leq 0$, $i = 2k \quad (k = 1, 2, 3, \ldots)$ *and* $a_i \geq 0$.
In all other cases, f_i *is concave.*

If $0 \notin (x^L, x^U)$, the monomial $a_i x^i$ is either convex or concave for any integer i. In this case, once the convex terms are identified, the terms in the objective of $POLY$ can be rearranged:

$$(POLY): \quad \text{glob min } f(x) = \sum_{i \in cv} f_i(x) + \sum_{i \in cc} f_i(x) := f_{cv}(x) + f_{cc}(x)$$
$$\text{s.t.} \quad x^L \leq x \leq x^U \tag{11}$$

where cv and cc denote the sets of indices of monomials that are convex and concave, respectively. Now construction of the convex envelope for the composite nonconvex function can be easily achieved by underestimating the function f_{cc} in (11) by a linear function:

$$(R - POLY): \quad \min \ \bar{f}(x) = f_{cv}(x) + \alpha + \beta x$$
$$\text{s.t.} \ x^L \leq x \leq x^U$$

where $\beta = (f_{cc}(x^U) - f_{cc}(x^L))/(x^U - x^L)$ and $\alpha = f_{cc}(x^L) - \beta x^L$.

The Newton–Raphson method will be used for solving $R - POLY$ because of its attractive convergence rate. The method can be further enhanced as follows:

THEOREM 6. *Let* R_i *be the current relaxation problem. The following assertions hold:*
(i) If $\bar{f}'(x^L) \geq 0$, *then* x^L *is an optimal solution to* R_i.
(ii) If $\bar{f}'(x^U) \leq 0$, *then* x^U *is an optimal solution to* R_i.

Proof. Follows from convexity of $\bar{f}$. □

By making use of the two conditions of Theorem 6 in the lower bounding step, one can quickly check whether or not an optimal solution of a relaxed problem is readily available. Even when the conditions do not apply, they provide all the necessary information for range reduction based on probing.

Now, the distinctive steps of the specialized algorithm can be stated.

ALGORITHM 3. *Poly*:

Operation *Initialization:*

if $0 \in (x^L, x^U)$ **then**

Create two subproblems from $R - POLY$: R_1 defined over $[x^L, 0]$ and R_2 over $[0, x^U]$. Put R_1 and R_2 in the list $ACTIVE$.

else

Put $R_1 = R - POLY$ in $ACTIVE$.

endif

Set $L^{(0)} = -\infty$ and $U^{(0)} = +\infty$.

Go to the main step.

Operation *Lower Bounding:*

if $\bar{f}'(x^L) \geq 0$ **then**

Set x^i (the optimal solution of R_i) equal to x^L.

elseif $\bar{f}'(x^U) \leq 0$ **then**

Set x^i equal to x^U.

else

Solve R_i using the Newton–Raphson method. Let x^i be the solution.

endif

Set $L_i = \bar{f}(x^i)$.

The remaining steps of the algorithm for polynomials follow the description of the general algorithm of Section 4.

7. Global Optimization of Linear Multiplicative Programming Problems

This section addresses the development of a specialized branch-and-reduce algorithm for the following class of problems:

$$
\begin{aligned}
(LMP):\quad & \text{globmin } f(x) = \prod_{i=1}^{p} f_i(x) = \prod_{i=1}^{p} \left(c_i^t x + c_{i0}\right) \\
& \text{s.t.} \quad Ax \leq b \\
& \qquad\;\; c_i^t x + c_{i0} \geq 0 \quad (i = 1, \ldots, p)
\end{aligned}
$$

where $x \in \Re^n$, $b \in \Re^m$, $c_i \in \Re^n$ and $c_{i0} \in \Re$ $(i = 1, \ldots, p)$ and $A \in \Re^{m \times n}$.

Linear multiplicative programming problems have applications in microeconomics, VLSI chip design, bond portfolio optimization and multicriteria optimization problems (Kuno and Konno [33]). They are also closely related to other classes of global optimization problems. If $p = 2$, for example, LMP can be transformed into bilinear programming and a class of quadratic programming problems (Pardalos [47]). The problem may possess several local minima (Konno and Kuno [31]) and its complexity is still open even for $p = 2$ (Pardalos [47]), even though more general LMPs are known to be *NP*-hard (Konno *et al.* [32]). Without loss of generality, we will assume that $c_i^t x + c_{i0} > 0$ $(i = 1, \ldots, p)$ over the feasible set (If any one of the linear functions can assume the value of 0, then a globally optimal solution of LMP can be trivially found by individually minimizing the linear functions in the objective subject to the constraint set.)

The following transformation facilitates the development of the relaxation:

$$\begin{aligned}(LMP-T): \quad & \text{globmin } \ln(f(t)) = \sum_{i=1}^{p} \ln(t_i) \\ & \text{s.t.} \quad Ax \le b \\ & \qquad c_i^t x + c_{i0} = t_i \quad (i=1,\dots,p) \\ & \qquad t_i > 0 \quad (i=1,\dots,p)\end{aligned}$$

THEOREM 7. *LMP and $LMP-T$ are equivalent problems.*

Proof. Follows directly from the monotonicity of the logarithmic function. □

As the objective in $LMP-T$ is concave, the computation of lower bounds is achieved through the solution of linear programming subproblems:

$$\begin{aligned}(R-LMP-T): \quad & \text{globmin } \bar{f}(t) = \sum_{i=1}^{p} (\alpha_i + \beta_i t_i) \\ & \text{s.t.} \quad Ax \le b \\ & \qquad c_i^t x + c_{i0} = t_i \quad (i=1,\dots,p) \\ & \qquad t_i > 0 \quad (i=1,\dots,p)\end{aligned}$$

where $\beta_i = (\ln(t_i^U) - \ln(t_i^L))/(t_i^U - t_i^L)$ and $\alpha_i = \ln(t_i^L) - \beta_i t_i^L$ $(i = 1,\dots, p)$.

Note that solving $R-LMP-T$ requires lower and upper bounds for each product variable t_i $(i = 1,\dots, p)$. To obtain these bounds, the problem is preprocessed at the initialization phase of the algorithm. First, each product variable is minimized individually subject to the original problem constraints to obtain its lower bound. This computation also provides upper bounds through function evaluations at the resulting feasible points. Let U denote the best of these bounds. Subsequently, the objective function cut $f(x) = \prod_{i=1}^{p} t_i \le U$ yields the relationships $t_i \le U/\prod_{j=1,j\neq i}^{p} t_j$ $(i = 1,\dots,p)$ which, in turn, provide the required upper bounds:

$$t_i^U \le U/\prod_{j=1,j\neq i}^{p} t_j^L \quad (i=1,\dots,p) \tag{12}$$

Note that (12) can be used in the pre- and post-processing steps of the algorithm at any node of the search tree.

The following are the problem-specific steps of the specialized branch-and-reduce algorithm for LMPs:

ALGORITHM 4. *Linear Multiplicative*:

Operation *Initialization:*

Set $L^{(0)} = -\infty$.

Individually minimize t_i (i = 1, ..., p) subject to the constraint set. Let $\mathbf{x}^i$ (i = 1,..., p) be the solution vectors and t_i^L ($i = 1, \ldots, p$) the corresponding solution values.

Set $U^{(0)} = \min_{i=1,\ldots,p}\{f(\mathbf{x}^i)\}$.

Calculate t_i^U ($i = 1, \ldots, p$) from (12).

Put $R_1 = R - LMP - T$ in the list $ACTIVE$ of active subproblems.

The remaining steps of the algorithm for *LMP*s follow the description of the general algorithm of Section 4.

Remark 4. In order to obtain an ϵ-optimal solution of LMP, *Linear Multiplicative* must use $\exp(L_i) \geq \exp(U^{(k)}) - \epsilon$ as the criterion for deleting inferior nodes.

8. Implementation and Computational Experiments

The computer code BARON (Branch-And-Reduce Optimization Navigator) has been developed to implement the proposed algorithm. BARON is a modular, all-purpose global optimization software that executes the branch-and-reduce global optimization strategy by navigating its way through user-provided subroutines. The user provides only problem-specific subroutines for computing the relaxations and for local minimization. A GAMS [7] and a FORTRAN version of BARON have been developed. Global optimization problems were collected from the literature and others were randomly generated in order to test BARON and demonstrate the wide applicability of range reduction and the branch-and-reduce algorithm.

8.1. Engineering Design Problems and Standard Global Optimization Test Problems

A set of 27 engineering design problems and global optimization tests problems were solved first as seen in Table V. These problems include engineering design problems (Examples 2-5, 12-18), a pooling problem (Example 7), early global optimization test problems (Examples 1, 22 and 23) and some others. Table V provides for each problem the source and problem size in terms of numbers of constraints (m), continuous (n_c) and integer variables (n_i). Detailed models, local and global solutions of the first 21 problems are reported in Ryoo and Sahinidis [54].

TABLE V. Computational results with engineering design problems and global optimization test problems.

					Branch-and-Bound				Branch-and-Reduce							
		Problem size			BB				BR1 (w/o probing)				BR2 (w/ probing)			
Ex. No.	Source	m	n_c	n_i	N_{tot}	N_{opt}	N_{mem}	CPU sec.	N_{tot}	N_{opt}	N_{mem}	CPU sec.	N_{tot}	N_{opt}	N_{mem}	CPU sec.
1	[55]	1	2		13	12	12	1.10	1	1	1	0.35	1	1	1	0.50
2	[61]	3	3		11	9	3	20.90	1	1	1	0.15	1	1	1	0.13
3	[5]	7	10		*	*	*	*	5	1	3	12.25	1	1	1	13.33
4	[61]	1	3		*	*	*	*	1	1	1	0.21	1	1	1	0.22
5	[36]	3	5		*	*	*	*	49	48	7	4.54	3	1	2	2.22
6	[36]	3	3		1	1	1	0.23	1	1	1	0.20	1	1	1	1.14
7	[25]	7	10		7	2	2	1.94	3	1	2	1.56	1	1	1	0.63
8	[62]	2	2		1	1	1	0.26	1	1	1	0.26	1	1	1	0.38
9	[62]	1	2		7	1	4	0.82	3	1	2	0.53	3	1	2	0.93
10	[59]	1	2		1	1	1	0.16	1	1	1	0.19	1	1	1	0.18
11	[72]	2	3		*	*	*	*	1	1	1	0.65	1	1	1	0.67
12	[60]	3	4		3	1	2	0.31	1	1	1	0.13	1	1	1	0.13
13	[30]	2	1	1	5	2	2	0.48	1	1	1	0.35	1	1	1	0.35
14	[76]	9	3	4	7	7	6	1.65	7	7	6	0.99	7	7	6	1.79
15	[30]	6	2	3	*	*	*	*	1	1	1	0.15	1	1	1	0.17
16	[17]	9	12		*	*	*	*	1	1	1	3.46	1	1	1	3.43
17	[35]	1	2		*	*	*	*	3	2	2	0.54	3	2	2	0.75
18	[71]	4	2		1	1	1	0.24	1	1	1	0.22	1	1	1	0.22
19	[37]	2	2		149	16	15	6.20	3	1	2	0.42	3	1	2	0.45
20	[37]	5	6		*	*	*	*	129	108	66	14.62	129	108	66	34.81
21	[60]	6	6		7	2	4	0.57	3	1	2	0.47	1	1	1	0.44
22	[29]	5	2		9	1	4	0.44	1	1	1	0.18	1	1	1	0.26
23	[2]	2	2		13	1	5	0.78	3	2	2	0.56	1	1	1	0.77
24	[31]	8	4		5	1	3	0.50	1	1	1	0.46	1	1	1	0.49
25	[64]	4	2		1	1	1	0.08	1	1	1	0.09	1	1	1	0.09
26	[16]	4	2		17	9	9	2.89	5	1	3	0.77	5	1	3	0.98
27	[8]	6	5		*	*	*	*	1	1	1	0.44	1	1	1	0.66

The FORTRAN version of BARON was used to solve these problems and the tests were run on a SUN SPARC Station 2. All the relaxation subproblems and NLP problems were solved using MINOS 5.4 [43]. Three different strategies were tested: BB, BR1 and BR2. BB is a standard branch-and-bound strategy and does not involve the use of any range reduction techniques. BR1 features the optimality-based range reduction tools of Table I, and BR2 makes use of all range reduction mechanisms of Tables I and II. An absolute optimality tolerance of $\epsilon = 10^{-6}$ was used throughout the experiments: at any iteration k, all subproblems (R_i) with $L_i \geq U^{(k)} - \epsilon$ were deleted. For the results presented in Table V, N_{tot}, N_{opt} and N_{mem} denote the total number of iterations, the node in which the optimal solution is found, and the maximum number of nodes stored in memory during the search, respectively. A * in this table is used to indicate the examples that did not terminate within 1200 CPU seconds or N_{mem}= 1000 nodes. Finally, n_c and n_i denote the number of continuous and integer variables of the problem, respectively.

As seen in Table V, standard branch-and-bound (BB) did not converge for many of the problems despite their small size. On the other hand, the use of range reduction made possible the solution of all problems within the prespecified time and memory limits. The use of probing (BR2) further reduces the memory requirements of the simpler algorithm (BR1) at the expense of somewhat higher CPU times for some of the problems. Ryoo and Sahinidis [54] solved the first 21 of these test problems using the GAMS implementation of an earlier version of the algorithm. The results presented in Table V for these 21 problems improve those in [54] due to the use of tighter lower bounds, additional range reduction inequalities and an improved implementation.

TABLE VI. Comparative computational results for unconstrained univariate polynomial functions.

			Branch-and-Reduce				Interval Arithmetic Method [24]			
							Nested form		Centered form	
Ex. No.	Source	Order (t)	N_{tot}	N_{opt}	N_{mem}	CPU sec.*	N_{tot}	CPU sec.**	N_{tot}	CPU sec.**
1	[75]	6	27	22	5	0.06	21	0.38	20	1.12
2	[42]	50	17	16	5	0.06	44	6.30	34	72.96
3	[74]	5	27	24	6	0.06	19	0.30	18	0.78
4	[11]	4	11	0	3	0.04	32	0.40	31	1.16
5	[11]	6	7	0	4	0.01	21	0.34	23	1.22
6	[19]	6	31	10	7	0.06	37	0.70	37	2.04
7	[10]	4	9	9	3	0.01	16	0.22	16	0.50

*: IBM RS/6000 66MHz-Power PC.
**: SUN 3/50-12 workstation.

8.2. Polynomial Programs

Comparative computational results for unconstrained univariate polynomial functions are provided in Table VI. The FORTRAN version of BARON was used in this computation and the tests were run on an IBM RS/6000 66MHz-Power PC with an absolute optimality tolerance of $\epsilon = 10^{-7}$. No local minimization was used and upper bounding was based on function evaluations. Hansen *et al.* [24] solved the same set of problems in the same order. The results with two different interval arithmetic methods from [24] are also provided in Table VI for comparison. Although the CPU times of all approaches are small for most problems, [24] reports much larger CPU times for Example 2 than for any of the other problems. Example 2 involves the largest number (50) of monomial functions. The branch-and-reduce algorithm seems to be insensitive to the order of the polynomial and takes less than 0.1 sec to solve any of these problems. This is due to a very efficient implementation of the lower and upper bounding procedures. The performance of the proposed algorithm seems to depend on the difference between the relaxation value at the root node and the optimal solution of the nonconvex problem. The initial lower bounds and global optima for these problems are shown in Table VII. Examples 1 and 6 present the largest gap between the two bounds and require the largest number of iterations of the algorithm.

TABLE VII. Gap between initial lower bounds and global optima for polynomial examples.

Example Number	Initial Lower Bound (L^1)	Global Optimum
1	−138,468.40	−29,763.23
2	−22,933.59	−663.50
3	−1764.29	−443.71
4	−101.82	0
5	−6.34	0
6	−1546.79	7
7	−9.14	−7.50

8.3. Linear Multiplicative Programming Problems

As there are not many LMP test problems in the literature, we generated random problems to test the algorithm. These problems varied in sizes from (m, n)=(50,50) to (200,200) with p ranging from 2 to 5. The objective cost coefficients were generated in the range [0, 10]. Finally, the elements of A and b of the constraint set $Ax \leq b$ were generated from $[-100, 0]$ to ensure a finite optimal solution. Using different seeds, ten random instances were generated and solved by *Linear Multiplicative* of Section 7 on an IBM RS/6000 66MHz-Power PC. The FORTRAN version of BARON was used with an absolute optimality tolerance of $\epsilon = 10^{-6}$.

Neither local minimization nor probing were used for these problems and the linear programming subproblems were solved using IBM's OSL (Release 2).

TABLE VIII. Computational results for $LMPs$ with $p = 2$.

Problem Size		Tree Size			CPU seconds				
m	n	N_{tot}	N_{opt}	N_{mem}	T_{tot}	T_{init}	T_{rel}	T_{feas}	T_{opt}
50	50	9.6	2.8	4.0	0.7	0.2	0.2	0.1	0.0
		3	1	3	0.4	0.1	0.1	0.0	0.0
		15	9	5	1.0	0.3	0.5	0.2	0.0
100	50	13.8	4.7	4.7	1.8	0.4	0.8	0.5	0.0
		11	1	4	1.4	0.3	0.5	0.2	0.0
		21	13	6	2.5	0.5	1.1	1.0	0.0
50	100	7.4	2.8	3.1	1.0	0.4	0.3	0.2	0.0
		1	1	1	0.5	0.2	0.0	0.0	0.0
		21	15	6	1.9	0.6	1.0	0.5	0.0
100	100	6.6	2.7	3.2	2.0	0.7	0.5	0.5	0.0
		1	0	1	0.8	0.4	0.0	0.0	0.0
		11	7	5	2.9	1.8	1.3	1.3	0.0
150	100	10.6	4.2	3.8	4.0	1.1	1.7	0.9	0.0
		1	1	1	1.5	0.6	0.1	0.0	0.0
		21	15	6	7.3	2.6	5.0	2.1	0.0
100	150	6.8	3.2	3.2	2.8	1.3	0.8	0.5	0.0
		1	1	1	1.9	0.6	0.1	0.0	0.0
		21	13	7	6.1	1.9	2.5	1.5	0.0
150	150	8.0	2.8	3.3	6.0	2.7	1.9	0.9	0.0
		1	1	1	2.7	2.0	0.2	0.1	0.0
		21	9	7	9.7	5.1	4.8	2.4	0.0
200	150	8.6	4.1	3.4	9.1	3.5	3.5	1.5	0.0
		1	1	1	5.1	1.4	1.3	0.1	0.0
		19	12	7	12.9	5.2	5.7	4.8	0.0
150	200	8.6	4.1	3.3	10.6	4.9	3.9	1.2	0.0
		1	1	1	1.9	1.1	0.1	0.1	0.0
		19	11	6	19.2	10.0	8.8	2.8	0.0
200	200	8.6	4.0	3.4	15.7	8.7	4.4	1.8	0.0
		1	1	1	8.3	3.5	0.4	0.1	0.0
		27	15	8	25.6	14.2	11.2	6.7	0.0

Tables VIII and IX provide computational results with $p = 2$ and $p = 5$. Results with $p = 3$ and $p = 4$ are similar (see Ryoo [53]). In these tables, T_{tot}, T_{init}, T_{rel}, T_{feas}, and T_{opt} denote total CPU time spent, time for the initialization, time spent on solving relaxed subproblems, time spent on feasibility-based range reduction and time spent on optimality-based range reduction, respectively. For each problem size, three rows of results are presented and correspond to the average, the best and the worst case performance of the algorithm over the 10 different random runs

TABLE IX. Computational results for LMPs with $p = 5$.

Problem Size		Tree Size			CPU seconds				
m	n	N_{tot}	N_{opt}	N_{mem}	T_{tot}	T_{init}	T_{rel}	T_{feas}	T_{opt}
50	50	327.4	140.7	79.8	19.5	0.5	13.2	5.4	0.0
		187	7	49	12.6	0.4	5.7	2.4	0.0
		561	321	117	31.1	0.6	21.8	8.5	0.1
100	50	426.0	166.3	100.6	54.6	0.9	43.2	10.0	0.0
		165	1	59	21.3	0.7	16.1	3.9	0.0
		529	380	135	74.6	1.2	62.1	15.8	0.1
50	100	292.2	64.6	79.8	28.3	1.2	18.9	7.8	0.1
		83	1	32	6.9	0.9	3.4	2.5	0.0
		519	299	130	52.4	1.7	36.8	14.0	0.1
100	100	419.2	169.3	97.5	84.1	2.0	62.0	19.2	0.1
		123	1	40	23.3	1.7	14.3	6.9	0.0
		841	555	152	161.1	2.5	124.9	36.7	0.1
150	100	589.8	374.0	130.9	198.1	3.7	153.4	40.0	0.1
		371	235	63	100.6	2.5	68.3	24.9	0.0
		951	907	225	300.5	4.7	231.5	63.0	0.2
100	150	480.3	261.3	108.2	140.7	3.5	102.8	33.5	0.1
		113	3	33	33.9	2.1	13.8	12.3	0.0
		1311	831	214	419.6	4.7	298.0	70.6	0.3
150	150	529.5	302.0	115.8	256.1	6.4	192.2	56.2	0.1
		203	87	37	104.5	4.1	45.8	31.1	0.0
		1071	538	230	457.3	9.6	378.7	97.7	0.3
200	150	603.7	403.5	143.7	465.1	9.3	374.8	79.4	0.1
		257	29	51	168.0	5.8	124.3	34.8	0.1
		1237	796	255	1191.5	12.9	974.7	129.4	0.3
150	200	542.8	299.6	137.6	345.6	12.0	263.9	68.1	0.2
		287	1.0	86	195.2	7.6	143.3	36.7	0.1
		919	549	253	687.7	16.3	566.8	107.8	0.3
200	200	677.7	457.5	147.6	650.4	17.3	515.3	115.5	0.2
		437	131	104	333.4	12.8	242.1	71.1	0.1
		1135	905	219	1273.8	22.0	1037.1	176.7	0.4

for each performance measure. It can be seen through these tables that optimality-based range reduction tests consume but a very small fraction of the total CPU time whereas a considerable amount of time is spent at the initialization phase for preprocessing of the bounds. For a constant number of products in the objective, Tables VIII and IX indicate a weak dependence of the problem complexity on the total number of variables. Figure 1 presents average results over all problem sizes (m, n) as a function of the number of products (p). For the problems solved, there seems to be a low-order polynomial relationship between CPU time and the number of products. The generated problems were very difficult as denoted by the gap between the initial bounding LP and the optimal solution. This gap averaged from 8 to 46 % in the examples solved as shown in Table X. The initial LP bound in this table was computed after the initialization phase which includes some feasibility-based range reduction tests.

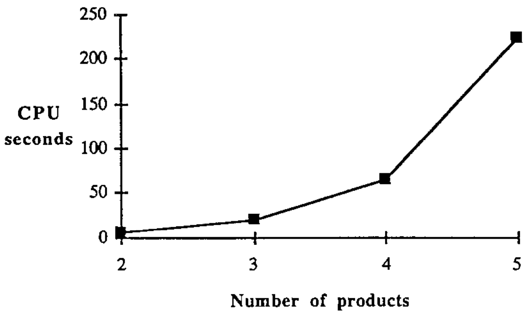

Fig. 1. CPU seconds versus the number of products for *LMP*s.

TABLE X. Gap between global optimum and initial lower bound for LMPs with various problem sizes.

	Gap = $(f - L^1)/L^1 \times 100$			
(m, n)	$p = 2$	$p = 3$	$p = 4$	$p = 5$
(50,50)	9	23	45	53
(100,50)	14	27	46	56
(50,100)	11	23	32	47
(100,100)	8	24	35	55
(150,100)	12	30	40	58
(100,150)	6	17	28	34
(150,150)	3	19	29	37
(200,150)	5	22	32	38
(150,200)	3	18	28	37
(200,200)	4	18	28	40
Average	**8**	**22**	**34**	**46**

8.4. Preliminary Results with MINLPs and Separable QPs

Some preliminary computational studies on a SUN SPARC Station 2 with mixed-integer nonlinear programs (MINLPs) and quadratic programs (QPs) are reported in Tables XI and XII, respectively. In addition to the MINLP Examples 13–15 of Table V, Examples 1–4 in Table XI were solved. The number of integer variables (n_i) in the problems ranged from 3 to 25. The concave quadratic programming problems (Examples 1–5 of Table XII) are from Section 2.7 of Floudas and Pardalos [18]. The problems of Tables XI and XII were solved with an absolute termination criterion of 10^{-6} using the GAMS version of BARON. No probing was used for optimality-based range reduction. The results of this subsection are only preliminary in the sense that pre-processing and post-processing are not extensive and do not exploit the special properties of the problems. These results are presented to illustrate the versatility of the proposed algorithm. Nevertheless, comparative computational results shown in Table XIII for the QPs indicate that the algorithm is competitive to the Reformulation-Linearization Technique (Sherali and Tuncbilek [58]). The preliminary implementation of the branch-and-reduce algorithm takes a larger number of iterations to converge. Yet, the relaxations used are simpler to solve and thus the resulting CPU times are competitive.

TABLE XI. Preliminary computational results for MINLPs.

Ex. No.	Source	m	n_c	n_i	N_{tot}	N_{opt}	N_{mem}	CPU sec.
1	[12]	6	3	3	3	2	2	2
2	[13]	14	6	5	7	6	4	5
3	[20]	23	9	8	9	8	8	10
4	[1]	5	5	25	83	81	14	280

TABLE XII. Preliminary computational results for QPs.

Ex. No.	Source	m	n	N_{tot}	N_{opt}	N_{mem}	CPU sec.
1	[18]	10	20	145	1	38	20
2	[18]	10	20	145	1	38	16
3	[18]	10	20	145	1	38	15
4	[18]	10	20	145	1	38	16
5	[18]	10	20	325	63	90	53

TABLE XIII. Comparative computational results for QPs.

		Reformulation-Linearization [58]				Branch-and-Reduce	
Ex. No.	Optimality Tolerance %	LD-RLT-NLP		LD-RLT-NLP(SC)			
		CPU sec. IBM 3090	No. of Iterations	CPU sec. IBM 3090	No. of Iterations	CPU sec. SUN Sparc 2	No. of Iterations
1	5	8.13	7	3.29	3	5.34	35
2	5	2.54	1	2.61	1	1.57	13
3	5	13.26	11	2.55	1	3.86	35
4	5	5.04	5	2.61	1	1.45	13
5	5	27.00	25	15.94	11	5.05	69

9. Conclusions

Range reduction techniques were presented in this paper as a means of performance improvement in global optimization algorithms. These techniques are based on optimality and feasibility criteria and were incorporated in the branch-and-bound framework to demonstrate their use. The philosophy of the resulting branch-and-reduce algorithm is to improve the lower and the upper bounds on the value of the global optimum by reducing the ranges of the continuous variables. The versatility and the efficiency of the algorithm were demonstrated by applying it to engineering design problems, standard global optimization test problems, univariate polynomial functions, mixed-integer nonlinear problems, concave quadratic programming problems and linear multiplicative programming problems.

The proposed algorithm was implemented in the global optimization software BARON. An experimental FORTRAN version of the code can be obtained by anonymous ftp from `aristotle.me.uiuc.edu`.

Acknowledgements

Partial financial support from the University of Illinois, from the EXXON Education Foundation and from the National Science Foundation under grant DMII 94-14615 is gratefully acknowledged. The authors are also thankful to Professor Panos Pardalos for helpful comments.

References

1. Albers, S. and K. Brockhoff (1977), "A Procedure for New Product Positioning in an Attribute Space," *European Journal of Operational Research*, **1**, 230-238.
2. Al-Khayyal, F. and J. E. Falk (1983), "Jointly Constrained Biconvex Programming," *Mathematics of Operations Research*, **8**(2), 273-286.
3. Anagnostou, G., E. M. Ronquist, and A. T. Patera (1991), "A Computational Procedure for Part Design," in J. P. Mesirov (ed.), *Very Large Scale Computation in the 21st Century*, SIAM, Philadelphia.
4. Balakrishnan, V. and S. Boyd (1992), *Global Optimization in Control System Analysis and Design*, in Leondes, C. T. (ed.), *Control and Dynamic Systems: Advances in Theory and Applications*, vol. 53, Academic Press, New York.
5. Bracken, J. and G. P. McCormick (1968), *Selected Applications of Nonlinear Programming*, Wiley, New York.
6. Brayton, R. K., G. D. Hachtel, and A. L. Sangiovanni-Vincentelli (1981), "A Survey of Optimization Techniques for Integrated-Circuit Design," *Proceedings of the IEEE*, **69**, 1334-1362.
7. Brooke, A., D. Kendrick, and A. Meeraus (1988), *GAMS - A User's Guide*, The Scientific Press, Redwood City.
8. Colville, A. R. (1968), "A Comparative Study of Nonlinear Programming Codes," IBM Scientific Report 320-2940, New York.
9. Danninger, G. (1992), "Role of Copositivity in Optimality Criteria for Nonconvex Optimization Problems," *Journal of Optimization Theory and Applications*, **75**(3), 535-538.
10. Dixon, L. C. W. (1990), "On Finding the Global Minimum of a Function of One Variable," *SIAM National Meeting* , Chicago, IL.
11. Dixon, L. C. W. and G. P. Szegø (1975), *Towards Global Optimization*, North Holland, Amsterdam.

12. Durán, M. A. (1984), "A Mixed-integer Nonlinear Programming Approach for the Systematic Synthesis of Engineering Systems," Ph.D. Thesis, Department of Chemical Engineering, Carnegie Mellon University.
13. Durán, M. A. and I. E. Grossmann (1986), "A Mixed-integer Nonlinear Programming Algorithm for Process Systems Synthesis," *American Institute of Chemical Engineers Journal*, **32**, 592-606.
14. Durán, M. A. and I. E. Grossmann (1986), "An Outer-Approximation Algorithm for a Class of Mixed-Integer Nonlinear Programs," *Mathematical Programming*, **36**, 307-339.
15. Falk, J. E. (1973), "Conditions for Global Optimality in Nonlinear Programming," *Operations Research*, **21**, 337-340.
16. Falk, J. E. and R. M. Soland (1969), "An Algorithm for Separable Nonconvex Programming Problems," *Management Science*, **15**, 550-569.
17. Floudas, C. A. and A. R. Ciric (1989), "Strategies for Overcoming Uncertainties in Heat Exchanger Network Synthesis," *Computers & Chemical Engineering*, **13**, 1133-1152.
18. Floudas, C. A. and P. M. Pardalos (1990), *A Collection of Test Problems for Constrained Global Optimization Algorithms*, Springer-Verlag, Berlin.
19. Goldstein, A. A. and J. F. Price (1971), "On Descent from Local Minima," *Mathematics of Computation*, **25**, 569-574.
20. Grossmann, I. E. (1985), "Mixed-integer Programming Approach for the Synthesis of Integrated Process Flow-sheets," *Computers & Chemical Engineering*, **9**, 463-482.
21. Haftka, R. T. and Gurdal, Z. (1992), *Elements of Structural Optimization*, Kluwer Academic Publishers, Dordrecht.
22. Hamed, A. S. E. and G. P. McCormick (1993), "Calculation of Bounds on Variables Satisfying Nonlinear Inequality Constraints," *Journal of Global Optimization*, **3**(1), 25-47.
23. Hansen, P., B. Jaumard, and S.-H. Lu (1991), "An Analytical Approach to Global Optimization," *Mathematical Programming*, **52**(2), 227-254.
24. Hansen, P., B. Jaumard, and J. Xiong (1993), "Decomposition and Interval Arithmetic Applied to Global Minimization of Polynomial and Rational Functions," *Journal of Global Optimization*, **3**(4), 421-437.
25. Haverly, C. A. (1978), "Studies of the Behaviour of Recursion for the Pooling Problem," *SIGMAP Bull.*, **25**, 19.
26. Hillestad, R. J. and S. E. Jacobsen (1980), "Reverse Convex Programming," *Applied Mathematics and Optimization*, **6**, 63-78.
27. Hiriart-Urruty, J.-B. (1986), "When Is a Point x satisfying $\nabla f(x) = 0$ a global optimum of f?," *American Mathematics Monthly*, **93**, 556-558.
28. Horst, R. and H. Tuy (1993), *Global Optimization: Deterministic Approaches*, Springer-Verlag, 2nd ed., Berlin.
29. Kalantari, B. and J. B. Rosen (1987), "An Algorithm for Global Minimization of Linearly Constrained Convex Quadratic Functions," *Mathematics of Operations Research*, **12**(3), 544-561.
30. Kocis, G. R. and I. E. Grossmann (1988), "Global Optimization of Nonconvex MINLP Problems in Process Synthesis," *Industrial and Engineering Chemistry Research*, **27**(8), 1407-1421.
31. Konno, H. and T. Kuno (1990), "Generalized Linear Multiplicative and Fractional Programming," *Annals of Operations Research*, **25**, 147-162.
32. Konno, H., T. Kuno, and Y. Yajima (1992), "Parametric Simplex Algorithms for a Class of *NP*-Complete Problems Whose Average Number of Steps is Polynomial," *Computational Optimization and Applications*, **1**, 227-239.
33. Kuno, T. and H. Konno (1991), "A Parametric Successive Underestimation Method for Convex Multiplicative Programming Problems," *Journal of Global Optimization*, **1**(3), 267-285.
34. Lamar, B. W. (1993), "An Improved Branch and Bound Algorithm for Minimum Concave Cost Network Flow Problems," *Journal of Global Optimization*, **3**(3), 261-287.
35. Liebman, J., N. Khachaturian, and V. Chanaratna (1981), "Discrete Structural Optimization," *Journal of Structural Division, ASCE*, **107**, no. ST11, Proceedings paper 16643 (Nov.), 2177-2197.
36. Liebman, J., L. Lasdon, L. Schrage, and A. Waren (1986), *Modeling and Optimization with GINO*, The Scientific Press, Palo Alto, CA.

37. Manousiouthakis, M. and D. Sourlas (1992), "A Global Optimization Approach to Rationally Constrained Rational Programming," *Chemical Engineering Communications*, **115**, 127-147.
38. McCormick, G. P. (1972), "Converting General Nonlinear Programming Problems to Separable Nonlinear Programming Problems," Technical Report Serial T-267, The George Washington University, Washington, D.C.
39. McCormick, G. P. (1976), "Computability of Global Solutions to Factorable Nonconvex Programs: Part I - Convex Underestimating Problems," *Mathematical Programming*, **10**, 147-175.
40. McCormick, G. P. (1983), *Nonlinear Programming. Theory, Algorithms, and Applications*, Wiley Interscience, New York.
41. Minoux, M. (1986), *Mathematical Programming. Theory and Algorithms*, Wiley, New York.
42. Moore, R. (1966), *Interval Analysis*, Prentice Hall, Englewood Cliffs, New Jersey.
43. Murtagh, B. A. and M. A. Saunders (1986), *MINOS 5.0 User's Guide*, Technical Report SOL 83-20, Systems Optimization Laboratory, Department of Operations Research, Stanford University, CA.
44. Murty, K. G. and S. N. Kabadi (1987), "Some *NP*-Complete Problems in Quadratic and Nonlinear Programming," *Mathematical Programming*, **39**, 117-129.
45. Neumaier, A. (1992), "An Optimal Criterion for Global Quadratic Optimization," *Journal of Global Optimization*, **2**(2), 201-208.
46. Papalambros, P. Y. and D. J. Wilde (1988), *Principles of Optimal Design*, Cambridge University Press.
47. Pardalos, P. M. (1990), "Polynomial Time Algorithms for Some Classes of Constrained Quadratic Problems," *Optimization*, **21**(6), 843-853.
48. Pardalos, P. M. and R. Horst (1994), *Handbook of Global Optimization*, Kluwer Academic Publishers, Norwell (Massachusetts).
49. Pardalos, P. M. and G. Schnitger (1988), "Checking local optimality in constrained quadratic programming is *NP*-hard," *Operations Research Letters*, **7**, 33-35.
50. Pardalos, P. M., D. Shalloway, and G. Xue (1994), "Optimization Methods for Computing Global Minima of Nonconvex Potential Energy Functions," *Journal of Global Optimization*, **4**(2), 117-133.
51. Phillips, A. T. and J. B. Rosen (1990), "Guaranteed ϵ-Approximate Solution for Indefinite Quadratic Global Minimization," *Naval Research Logistics*, **37**, 499-514.
52. Rozvany, G. I. N. (1989), *Structural Design via Optimality Criteria*, Kluwer Academic Publishers, Dordrecht.
53. Ryoo, H. S. (1994), "Range Reduction as a Means of Performance Improvement in Global Optimization: A Branch-and-Reduce Global Optimization Algorithm," Master's Thesis, University of Illinois at Urbana-Champaign, IL.
54. Ryoo, H. S. and N. V. Sahinidis (1995), "Global Optimization of Nonconvex NLPs and MINLPs with Applications in Process Design," *Computers & Chemical Engineering*, **19**(5), 551-566.
55. Sahinidis, N. V. and I. E. Grossmann (1991), "Convergence Properties of Generalized Benders Decomposition," *Computers & Chemical Engineering*, **15**(7), 481-491.
56. Schoen, F. (1991), "Stochastic Techniques for Global Optimization: A Survey of Recent Advances," *Journal of Global Optimization*, **1**(3), 207-228.
57. Sherali H. D. and A. Alameddine (1992), "A new Reformulation-Linearization Technique for Bilinear Programming Problems," *Journal of Global Optimization*, **2**(4), 379-410.
58. Sherali H. D. and C. H. Tuncbilek (1994), "Tight Reformulation-Linearization Technique Representations for Solving Nonconvex Quadratic Programming Problems," Technical Report, Virginia Polytechnic Institute and State University, Blacksburg, Virginia.
59. Soland, R. M. (1971), "An Algorithm for Separable Nonconvex Programming Problems II: Nonconvex Constraints," *Management Science*, **17**(11), 759-773.
60. Stephanopoulos, G. and A. W. Westerberg (1975), "The Use of Hestenes' Method of Multipliers to Resolve Dual Gaps in Engineering System Optimization," *Journal of Optimization Theory and Applications*, **15**(3), 285-309.
61. Stoecker, W.F. (1971), *Design of Thermal Systems*, McGraw-Hill Book Co., New York.
62. Swaney, R. E. (1990), "Global Solution of Algebraic Nonlinear Programs," *AIChE Annual Meeting*, Chicago, IL.

63. Thakur, L. S. (1990), "Domain Contraction in Nonlinear Programming: Minimizing a Quadratic Concave Function Over a Polyhedron," *Mathematics of Operations Research*, **16**(2), 390-407.
64. Thoai, N. V. (1991), "A Global Optimization Approach for Solving the Convex Multiplicative Programming Problem," *Journal of Global Optimization*, **1**(4), 341-357.
65. Törn, A., and A. Zilinskas (1989), *Global Optimization*, Lecture Notes in Computer Science, 350, Springer-Verlag, Berlin.
66. Tuy, H. (1964), "Concave Programming Under Linear Constraints," *Doklady Akademic Nauk*, **159**, 32-35. Translated *Soviet Mathematics*, **5**, 1437-1440.
67. Tuy, H. (1987), "Convex Programs with an additional reverse convex constraint," *Journal of Optimization Theory and Applications*, **52**, 463-486.
68. Tuy, H. (1991), "Effect of the Subdivision Strategy on Convergence and Efficiency of Some Global Optimization Algorithms," *Journal of Global Optimization*, **1**(1), 23-36.
69. Tuy, H. and R. Horst (1988), "Convergence and Restart in Branch-and-Bound Algorithms for Global Optimization. Application to Concave Minimization and D.C. Optimization Problems," *Mathematical Programming*, **41**(2),161-183.
70. Tuy, H., V. Khatchaturov, and S. Utkin (1987), "A Class of Exhaustive Cone Splitting Procedures in Conical Algorithms for Concave Minimization," *Optimization*, **18**(6), 791-807.
71. Visweswaran, V. and C. A. Floudas (1990), "A Global Optimization Algorithm (GOP) for Certain Classes of Nonconvex NLPs – II. Application of Theory and Test Problems," *Computers & Chemical Engineering*, **14**(2), 1419-1434.
72. Westerberg, A. W. and J. V. Shah (1978), "Assuring a Global Optimum by the Use of an Upper Bound on the Lower (Dual) Bound," *Computers & Chemical Engineering*, **2**, 83-92.
73. Wilde, D. J. (1978), *Globally Optimal Design*, J. Wiley & Sons, New York.
74. Wilkinson, J. H. (1963), *Rounding Errors in Algebraic Processes*, Prentice Hall, Englewood Cliffs, New Jersey.
75. Wingo, D. R. (1985), "Globally Minimizing Polynomials without Evaluating Derivatives," *International Journal of Computer Mathematics*, **17**, 287-294.
76. Yuan, X., S. Zhang, L. Pibouleau, and S. Domenech (1988), "Une méthode d'optimisation non linéaire en variables mixtes pour la conception de procédés," *Recherche Opérataionnelle/Operations Research*, **22**(4), 331-346.