Pergamon

Computers chem. Engng Vol. 19, No. 5, pp. 551–566, 1995

0098-1354(94)00097-2
0098-1354/95 $9.50 + 0.00

GLOBAL OPTIMIZATION OF NONCONVEX NLPs AND MINLPs WITH APPLICATIONS IN PROCESS DESIGN

H. S. Ryoo and N. V. Sahinidis†

Department of Mechanical & Industrial Engineering, University of Illinois, Urbana-Champaign, 1206 W. Green Street, Urbana, IL 61801, U.S.A.

(Received 10 December 1993; final revision received 27 July 1994; received for publication 9 August 1994)

Abstract—This paper presents an algorithm for finding global solutions of nonconvex nonlinear programs (NLPs) and mixed-integer nonlinear programs (MINLPs). The approach is based on the solution of a sequence of convex underestimating subproblems generated by evolutionary subdivision of the search region. The key components of the algorithm are new optimality-based and feasibility-based range reduction tests. The former use known feasible solutions and perturbation results to exclude inferior parts of the search region from consideration, while the latter analyze constraints to obtain valid inequalities. Furthermore, the algorithm integrates these devices with an efficient local search heuristic. Computational results demonstrate that the algorithm compares very favorably to several other current approaches when applied to a large collection of global optimization and process design problems. It is typically faster, requires less storage and it produces more accurate results.

1. INTRODUCTION

Realistic treatments of physical and engineering systems frequently involve nonlinear models. Model nonlinearities give rise to nonconvexities which, in turn, lead to multiple local optima. Unfortunately, once a local minimum is found, the question of global optimality is often ignored by optimization practitioners. The reason is not so much a lack of understanding of the phenomena or absence of modeling techniques, but rather the absence of *efficient* global optimization algorithms. The optimization community seems to be just now starting to explore the subject of continuous global optimization in a systematic way. Although papers had addressed this topic sporadically since the 1960s (Tuy, 1964; Falk and Soland, 1969), it was not until very recently that the first systematic exposition of deterministic global optimization methods appeared (Horst and Tuy, 1990) and the first journal specializing on global optimization was established (*Journal of Global Optimization*, Kluwer Academic Publishers, 1991). In the process design literature, global optimization questions were addressed in some of the early works (e.g. Stephanopoulos and Westerberg, 1975; Westerberg and Shah, 1978; Wang and Luus, 1978). More recent papers have proposed a variety of heuristics (Kocis and Grossmann, 1988; Floudas *et al.*, 1989; Salcedo, 1992) and exact algorithms (Swaney, 1990; Floudas and Visweswaran, 1990; Manousiouthakis and Sourlas, 1992; Quesada and Grossmann, 1993).

† Author for correspondence.

Global optimization techniques can be classified as stochastic or deterministic. Stochastic methods, such as simulated annealing and Monte-Carlo minimization and their variations, do not make any assumptions for the problem functions. The methods involve random elements in their search procedure and converge to the global optimum with a probability approaching one as their running time goes to infinity (Törn and Zilinskas, 1989; Schoen, 1991). 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 (Horst and Tuy, 1993). Deterministic approaches include branch-and-bound, cutting plane algorithms and decomposition schemes. These methods were recently reviewed by Horst (1990) and Horst and Tuy (1993).

Following the development of branch-and-bound methods for integer programs, application of the same principles was suggested for continuous global optimization problems (Falk and Soland, 1969; McCormick, 1972a, b, 1983). Branch-and-bound methods employ lower and upper bounds of the optimal objective function value over subregions of the search space. Certain subregions are dynamically refined while others are excluded from consideration based on optimality and feasibility criteria. Of critical importance to the success of such methods is the accuracy (tightness) of the bounding procedures used. The bounds are developed over a range of values of the variables involved and become tighter as the search is confined to smaller

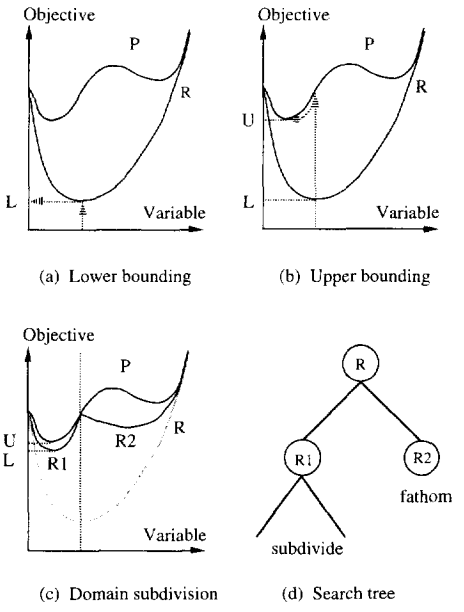

Fig. 1. Graphical interpretation of branch-and-bound for continuous global optimization.

subregions. The fundamental ideas of the approach are illustrated in Fig. 1. First, a *relaxation* of the nonconvex problem is constructed. A relaxation R of a nonconvex minimization problem P is obtained by enlarging the feasible region and/or underestimating the objective function of P. The relaxation is constructed in such a way that the difference between the optimal objective function values of problems P and P is a nonincreasing function of the size of the feasible region over which the relaxation is developed. Moreover, the relaxations typically used can be solved to their respective global minima by conventional minimization techniques (they are usually convex programs). Once the relaxed problem is solved, a valid lower bound L is thus obtained for the global minimum (Fig. 1a). Using the relaxed solution as a starting point (if some other more advantageous starting point is unavailable), local minimization techniques can be used to yield a valid upper bound, U, for P (Fig. 1b). At this stage the global minimum is known to be between L and U. If L is sufficiently close to U, the algorithm terminates. If not, the feasible region is subdivided into parts—e.g. into two parts: one to the right and one to the left of the relaxed solution. A new relaxed problem is solved for each subdivision. This time, the relaxation becomes tighter (by construction) and the lower bound comes closer to the upper bound (Fig. 1c). The process is then repeated for each subdivision until the subdivisions possess lower bounds that either exceed or are sufficiently close to the best found feasible solution of problem P. This leads to a method for searching over a tree whose nodes correspond to relaxed problems (Fig. 1d). At any time during the search, parts of the search region (tree nodes) can be excluded from further consideration by comparing their respective lower bounds to the current upper bound (e.g. node R2 in Fig. 1d).

Employing branch-and-bound techniques similar to the one described above in finding global minima of integer programs has consituted an important paradigm for the optimization community during the last two decades. It is now well understood that these problems can be solved to global optimality provided special provisions are taken at the level of modeling and algorithm design (Nemhauser and Wolsey, 1988). Tight formulations are needed and the algorithms must be designed to exploit the special mathematical structure of the problems under consideration (e.g. Sahinidis and Grossmann, 1991a, 1992). Although problems involving several hundreds and even thousands of integer variables can be solved to global optimality using today's algorithms and conventional computer technology, a great difficulty seems to arise in the context of continuous problems. Continuous problems solved to proven global optimality have so far typically involved only a few constraints and variables (see above cited references). Due to the large number of applications of continuous nonconvex programming, it is then clear that further algorithmic advances are needed.

The purpose of this paper is to present a new banch-and-bound-based method for discrete/continuous global optimization. The key components of the method are tests for reducing the ranges of the problem variables. Applied at each node of the search tree, these range reduction tests yield a branch-and-reduce algorithm for global optimization. The paper is organized as follows: Section 2 presents the main algorithm. Section 3 develops optimality-based and feasibility-based range reduction tests. Computational results with the branch-and-reduce algorithm are presented in Section 4. They involve a collection of several global optimization and process design problems. Conclusions are drawn in Section 5 and some future research directions are identified. Finally, to facilitate testing of global optimization algorithms, the test problems used in our study and complete information on their solution are provided in the Appendices.

2. THE BRANCH-AND-REDUCE GLOBAL OPTIMIZATION ALGORITHM

Although good, sometimes optimal, solutions are usually found in early stages during the search,

branch-and-bound algorithms may take a long time to verify optimality. The key element of the proposed algorithm is the introduction of devices for accelerating the convergence of a branch-and-bound algorithm. It should be noted that these devices can be incorporated in the context of other algorithms as well. Branch-and-bound is used here as the basis of the suggested approach as it is a well established and usually the preferred solution method for combinatorial global optimization problems. The global minimization problem considered here is the following:

Problem P:

$$\min f(\mathbf{x}) \text{ s.t. } \mathbf{g}(\mathbf{x}) \leqslant \mathbf{0}, \quad \mathbf{x} \in \mathbf{X}$$

where

$$f: \mathbf{X} \to \mathcal{R}, \quad \mathbf{g}: \mathbf{X} \to \mathcal{R}^{m'} \quad \text{and} \quad \mathbf{X} \subseteq \mathcal{R}^{n'}.$$

Let R be a relaxation of P. The formal statement of the proposed **branch-and-reduce algorithm** (originally sketched in Sahinidis, 1992a, b) is as follows:

1. **INITIALIZATION.** Put R on a list $\mathcal{P}$ of subproblems. Select the convergence parameter ε. Select $MAXSOLVE$: the maximum number of times a node is allowed to be solved in any iteration. Set the lower bound $L = -\infty$, and the upper bound $U = +\infty$. Goto 2.
2. **SUBPROBLEM SELECTION.** If $\mathcal{P} = \emptyset$, stop: the current best solution is optimal. Else, choose R_i from $\mathcal{P}$ according to a subproblem selection rule (node selection rule) and set $\mathcal{P} := \mathcal{P} \backslash \{R_i\}$ (i.e., delete R_i from $\mathcal{P}$). Set $SOLVE = 1$ and Goto 3.
3. **PRE-PROCESSING (optional).** Use optimality-based and feasibility-based tests to tighten variable bounds as much as possible for subproblem R_i. Goto 4.
4. **LOWER BOUNDING.** Solve R_i, or bound its solution from below. Let L_i be this lower bound. If the solution, $\mathbf{x}^i$, found for R_i is feasible for P and $f(\mathbf{x}^i) < U$ update $U \leftarrow f(\mathbf{x}^i)$, make $\mathbf{x}^i$ the current best solution, and set $\mathcal{P} := \mathcal{P} \backslash \{R_j\}$ for all those subproblems R_j for which $L_j \geqslant U - \varepsilon$. If $L_i \geqslant U - \varepsilon$, discard R_i and Goto 2. Else, let L be equal to the smallest of all L_i and Goto 5.
5. **UPPER BOUNDING (optional).** Do local search or other heuristic(s) to find a feasible solution for problem P. If found, update U and the current best solution, and set $\mathcal{P} := \mathcal{P} \backslash \{R_j\}$ for all those subproblems R_j for which $L_j \geqslant U - \varepsilon$. Goto 6.
6. **POST-PROCESSING (optional).** Use optimality-based and feasibility-based tests to strengthen the bounds of as many variables as possible. If Steps 4 or 5 were successful in improving U, apply strengthening tests to all subproblems currently in $\mathcal{P}$, otherwise apply strengthening only to R_i. If $SOLVE < MAXSOLVE$ and this step was successful in reducing the variable bounds for R_i, reconstruct R_i using the new bounds, set $SOLVE = SOLVE + 1$ and Goto 4. Else, Goto 7.
7. **BRANCHING (separation).** Apply a branching rule to R_i, i.e., generate a set of new subproblems $R_{i_1}, R_{i_2}, \ldots, R_{i_q}$, place them on the list $\mathcal{P}$, and Goto 2.

The algorithm involves a search tree. Simple subproblems (relaxations) are solved at each node of the tree in Step 4. Their solution provides lower bounds, L_i, which can be used to exclude nodes in the tree from further consideration. This is achieved by comparing these lower bounds to the best current upper bound U. At all times the global minimum is bounded between the lowest lower bound L and the value U of the best found feasible solution.

Without the optional pre- and post-processing steps, the above algorithm follows closely the original developments of Falk and Soland (1969) and the branch-and-bound algorithms described elsewhere (e.g. in Horst and Tuy, 1993; McCormick, 1972a, 1976, 1983). Relaxations can be developed in more than one way. If, for example, the functions are separable, valid underestimators can be obtained by summing up underestimators of single-variable functions. If separability is not present, it can be usually (although not always) induced to problem P at the expense of introducing additional variables and constraints (see McCormick, 1972b). Moreover, it is often straightforward to develop convex underestimators of single-variable functions in an interval. For example, if a function is concave, its convex underestimation in an interval is a straight line. In any case, the form of the underestimators will depend on the interval to which the variables are restricted. As the search proceeds, these intervals should be made smaller (by the branching rule) so that the underestimators become more accurate. Node selection (Step 2) and branching (Step 7) can also be done in more than one way. The approach followed here in Step 2 of the algorithm is to always select the subproblem with the lowest lower bound. In Step 7, the solution of the relaxed subproblem R_i is followed by branching on the variable x_j appearing in the underestimating term that contributes most strongly to the deviation of the underestimators from the original functions at the current solution point. Two new subproblems are then created: one

where x_j is restricted to the interval $[x_j^L, x_j^*]$ and one with x_j in $[x_j^*, x_j^U]$. Here x_j^L, x_j^U, and x_j^* denote the lower bound, the upper bound and the solution value, respectively, for x_j in subproblem R_i. In case a local minimum is known with a solution $\bar{x}_j \in (x_j^L, x_j^U)$, $\bar{x}_j$ is made the branching point instead of x_j^*. This makes the underestimators exact at the candidate solution (Swaney, 1990). Note that, in order for variable x_j to be selected for branching by the algorithm, there must be a nonconvex term that is violated by the underestimating functions at x_j^*. As the underestimators used here are exact at endpoints, it follows that, if x_j is selected for branching, we have $x_j^L < x_j^* < x_j^U$. Therefore, the two intervals $[x_j^L, x_j^*]$ and $[x_j^*, x_j^U]$ of the descendant nodes will be strictly smaller than that of the parent node $[x_j^L, x_j^U]$. Also note that the local minimum $\bar{x}_j$ can be made the branching point only if it is in the *open* interval (x_j^L, x_j^U), i.e. only if it is stictly between the parents node's bounds, in which case the resulting descendant nodes' intervals again become smaller.

Remark 1. The above algorithm is described without any explicit reference to discrete variables. However, a binary variable x can be modeled within a continuous framework using the standard continuous reformulation $0 \leq x \leq 1$, $x(1-x) \leq 0$. Alternatively, integer variables can be dealt with explicitly as done in standard branch-and-bound algorithms (e.g. Nemhauser and Wolsey, 1988). In either case, the above algorithm applies equally well to NLP and MINLP problems.

Remark 2. When the relaxations and branching rules satisfy certain properties, it is well known (e.g. Horst and Tuy, 1993) that, *without* the optional pre- and post-processing steps, the above algorithm either terminates finitely or else produces an infinite sequence of points approaching the ε-global minimum of problem P for any $\varepsilon > 0$. Since the above algorithm prevents infinite repetition of the optional steps, the same convergence characteristics will be preserved when the optional pre- and post-processing steps are included, *provided* that the range reduction tests are valid, i.e. that these tests do not exclude the parts of the search region that contain the global optimum. Valid tests are developed in the following section.

3. RANGE REDUCTION TESTS

3.1. Optimality-based range reduction tests

Consider the relaxed problem solved at the root node of a branch-and-reduce tree:

Problem R:

$$\min \tilde{f}(\mathbf{x}) \text{ s.t. } \tilde{\mathbf{g}}(\mathbf{x}) \leq \mathbf{0}, \quad \mathbf{x} \in \tilde{\mathbf{X}}$$

where

$$\tilde{f}: \tilde{\mathbf{X}} \to \mathcal{R}, \quad \tilde{\mathbf{g}}: \tilde{\mathbf{X}} \to \mathcal{R}^m, \quad \mathbf{X} \subseteq \tilde{\mathbf{X}} \subseteq \mathcal{R}^n$$

and where for any $\mathbf{x}$ feasible in P we have $\tilde{f}(\mathbf{x}) \leq f(\mathbf{x})$, $\tilde{\mathbf{g}}(\mathbf{x}) \leq \mathbf{g}(\mathbf{x})$. Also consider the following perturbed problem:

Problem $R(\mathbf{y})$:

$$\varphi(\mathbf{y}) = \min \tilde{f}(\mathbf{x}) \text{ s.t. } \tilde{\mathbf{g}}(\mathbf{x}) \leq \mathbf{y}, \quad \mathbf{x} \in \tilde{\mathbf{X}}.$$

The perturbation function $\varphi(\mathbf{y})$ has the following well-known properties (e.g. Chap. 5 of Minoux, 1986):

Theorem 1. Assuming that R has an optimum $\mathbf{x}^*$ of finite value, λ^* is a saddle-point multiplier if and only if the hyperplane with equation $z = \varphi(\mathbf{0}) - \lambda^* \cdot \mathbf{y}$ is a supporting hyperplane at $\mathbf{y} = \mathbf{0}$ of the graph of the perturbation function $\varphi(\mathbf{y})$. In this case, we have $\varphi(\mathbf{y}) \geq \varphi(\mathbf{0}) - \lambda^* \cdot \mathbf{y}$, $\forall \mathbf{y} \in \mathcal{R}^m$. Moreover, the saddle-point exists and the perturbation function is convex if R is a convex problem satisfying standard constraint qualifications.

On the basis of Theorem 1, the following result can be derived:

Theorem 2. Assume that R is a convex optimization problem with an optimal objective function value of L and assume that the constraint $x_j - x_j^U \leq 0$ is active at the solution of problem R with a multiplier value of $\lambda_j^* > 0$. Let U be a known upper bound for problem P. The following test is valid and can then be incorporated in the pre- and post-processing steps of the branch-and-reduce algorithm:

Test 1—$\kappa_j = x_j^U - \dfrac{U-L}{\lambda_j^*}$. If $x_j^L < \kappa_j$, then set $x_j^L \leftarrow \kappa_j$.

Proof. Consider the following perturbation problem:

Problem $P(\mathbf{y})$:

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

Since, for any $\mathbf{y}$, $R(\mathbf{y})$ is a convex relaxation of $P(\mathbf{y})$, we have $\varphi(\mathbf{y}) \leq \Phi(\mathbf{y})$. Therefore, an underestimator of $\varphi(\mathbf{y})$ is also an underestimator of $P(\mathbf{y})$. Consider now the perturbed problem $R(\mathbf{y})$ where only the right hand side of constraint $x_j - x_j^U \leq y$ is perturbed. It follows from Theorem 1 that a valid underestimator for $\varphi(y)$ is $L - \lambda_j^* y$ and hence: $L - \lambda_j^* y \leq \varphi(y) \leq \Phi(y)$. By requiring that the value of $\Phi(y)$ be better than the already known upper bound U, we obtain: $L - \lambda_j^* y < U$. Finally, consider only nonpositive values for y. Since $x_j - x_j^U \leq y$ is active for $y = 0$, it will also be active in the solution of $R(y)$ for any

$y \leq 0$, i.e. $y = x_j - x_j^U$. Substituting this relationship in $L - \lambda_j^* y < U$ above and rearranging yields:

$$x_j > x_j^U - \frac{U-L}{\lambda_j^*}. \qquad \square$$

A geometric interpretation of Test 1 is provided in Fig. 2. The figure depicts the perturbation function φ and its linear underestimator z. Test 1 excludes all those values of x_j for which the linear understimator of the relaxed problem exceeds the known upper bound U.

Similarly, a valid test can be derived if the lower bounding constraint $x_j^L - x_j \leq 0$ is active at the solution of problem R with a multiplier value of $\lambda_j^* > 0$:

Test 2—Let $\pi_j = x_j^L + \dfrac{U-L}{\lambda_j^*}$. If $x_j^U > \pi_j$, then set $x_j^U \leftarrow \pi_j$.

Tests 1 and 2 can be applied only if the corresponding bound constraint is active and the corresponding dual variable is strictly positive. For variables that are not at their bounds at the relaxed solution, similar tests can be developed by probing at the bounds: that is, by temporarily fixing these variables at their bounds and solving the partially restricted relaxed problem. This leads to the following tests:

Test 3—Solve R after setting $x_j = x_j^U$. If $\lambda_j^* > 0$ continue.

Let $\pi_j = x_j^U - \dfrac{L-U}{\lambda_j^*}$. If $x_j^U > \pi_j$, then set $x_j^U \leftarrow \pi_j$.

Test 4—Solve R after setting $x_j = x_j^L$. If $\lambda_j^* > 0$ continue.

Let $\kappa_j = x_j^L - \dfrac{L-U}{\lambda_j^*}$. If $x_j^L > \pi_j$, then set $x_j^L \leftarrow \kappa_j$.

Theorem 3. Tests 2–4 are valid, i.e. they do not exclude feasible solutions of P with objective function values that are lower than the upper bound U.

Proof. Similar to Theorem 2 (see Appendix A).

To apply Tests 3 and 4, the relaxed problem must be solved after fixing a variable at one of its bounds. The corresponding bound can then be improved, provided the restricted relaxed problem has a value L that is above the current upper bound U. The geometric interpretation of Tests 3 and 4 is provided in Fig. 3 where z^L and z^U are the linear underestimators of the optimal value of $R(\mathbf{y})$ obtained by probing at the lower and upper bound of variable x_j. The tests cut off parts of the feasible space where the objective function value f is guaranteed to be above the current upper bound U. The cutoff points are provided by the intersection of the linear underestimators of $\varphi(\mathbf{y})$ and the line $\varphi = U$.

Remark 3. With tighter upper bounds, the above tests become more effective. This is the rationale behind the introduction of the local search in Step 5. This step can take advantage of special approaches to the problem at hand, or it may incorporate a stochastic global optimization method. A simple heuristic is employed in our current implementation. It uses the relaxed problem solution as a starting point for local minimization of the original problem. If this point is not feasible, a random starting point is generated. Then, the original nonconvex problem is solved again with the additional constraint that the objective function value be better than the previous upper bound.

Remark 4. The tests can be used to tighten the bounds not only at the root node but at any node of the search tree. Implementation of Tests 1 and 2 requires no more effort than solving the relaxed subproblem. On the other hand, Tests 3 and 4 require successively fixing one variable at a time to its lower and upper bound. This demands the solution of at most $2n$ partially restricted relaxed problems at every node where probing is applied. To reduce the computational requirements of Tests 3 and 4, probing can be applied only at the root node and the marginal values can be stored for subsequent use whenever new, better feasible solutions are found. In this case, the derived tighter bounds will be valid for the entire tree. In addition, probing can be performed on only the important variables of the problem.

Remark 5. For global minimization of quadratic concave functions over a polyhedron, Thakur (1990)

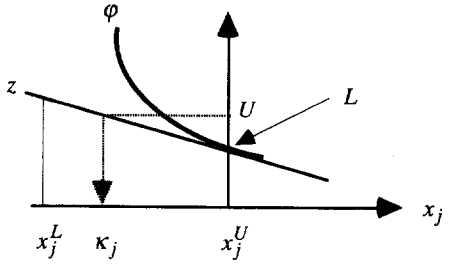

Fig. 2. Geometric interpretation of Test 1.

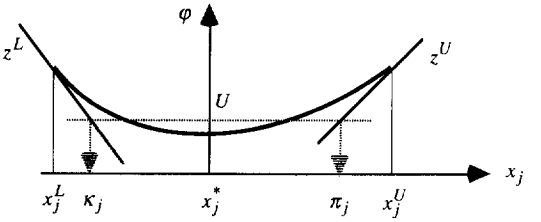

Fig. 3. Geometric interpretation of Tests 3 and 4.

uses linear underestimators and their dual solution to perform domain contraction. His approach is a special case of Tests 1 and 2 above. Tests 1–4 also generalize tests widely used in integer programming (e.g. Nemhauser and Wolsey, 1988) to the continuous case. The main difference is that only the range of a variable can be reduced in the continuous case whereas the tests lead to variable fixing in the binary case. Moreover, the tests developed here are based on nonlinear as opposed to linear programming relaxations.

3.1.1. Illustrative example. Consider the following nonconvex NLP problem (Sahinidis and Grossmann, 1991b):

$$\begin{aligned} \min \quad & -x_1 - x_2, \\ \text{s.t.} \quad & x_1 x_2 \leq 4, \\ & 0 \leq x_1 \leq 6, \\ & 0 \leq x_2 \leq 4. \end{aligned} \tag{1}$$

Due to the nonconvex constraint, the problem possesses two strong local minima at points $(x_1, x_2) = (1, 4)$ and $(x_1, x_2) = (6, 0.666667)$ with objective function values of -5 and -6.66667, respectively. Problem (1) can be made separable by introducing an additional variable $x_3 = x_1 + x_2$ and utilizing the identity $x_3^2 = x_1^2 + x_2^2 + 2x_1x_2$ to substitute for the bilinear term:

$$\begin{aligned} \min \quad & -x_1 - x_2, \\ \text{s.t.} \quad & x_3^2 - x_1^2 - x_2^2 \leq 8, \\ & x_3 = x_1 + x_2, \\ & 0 \leq x_1 \leq 6, \\ & 0 \leq x_2 \leq 4, \\ & 0 \leq x_3 \leq 10. \end{aligned} \tag{2}$$

Two straight lines, $(\alpha_1 + \beta_1 x_1)$ and $(\alpha_2 + \beta_2 x_2)$, can be now used to underestimate the concave quadratic terms $-x_1^2$ and $-x_2^2$, respectively, of the nonconvex constraint of problem (2). The coefficients of the underestimators are obtained by requiring the underestimators to match the nonconvex functions at the bounds of the variables:

$$\alpha_i = x_i^L x_i^U,$$

$$\beta_i = -(x_i^L + x_i^U) \quad \text{for } i = 1, 2.$$

In this way, a valid relaxation of (1) is the following convex NLP:

$$\begin{aligned} \min \quad & -x_1 - x_2, \\ \text{s.t.} \quad & x_3^2 - 6x_1 - 4x_2 \leq 8, \\ & x_3 = x_1 + x_2, \\ & 0 \leq x_1 \leq 6, \\ & 0 \leq x_2 \leq 4, \\ & 0 \leq x_3 \leq 10. \end{aligned} \tag{3}$$

Solving (3) provides the lower bound $L = -6.898979$ for (1). The solution of (3) occurs at the point $(x_1, x_2, x_3) = (6, 0.898979, 6.898979)$. Using this solution as a starting point for solving model (1) locally with GAMS/MINOS5.3 (Brooke *et al.*, 1988), we obtain $(x_1, x_2) = (6, 0.666667)$ and an upper bound of $U = -6.66667$. Although this solution is the global minimum, the algorithm does not terminate at this stage as the bounds L and U are not equal.

3.1.2. Application of marginal tests. In the solution of (3), variable x_1 was at its upper bound. Therefore, Test 1 can be applied for this variable [with $\lambda_1^* = 0.204124$ from the solution of (3)]:

$$\kappa_1 = x_1^U - \frac{U - L}{\lambda_1^*} = 6 - \frac{-6.666667 + 6.898979}{0.204124}$$

$$= 4.861904.$$

Then, one can safely conclude that setting $x_1^L = 4.861904$ does not remove any solutions of (1) with objective function values lower than $U = -6.666667$.

3.1.3. Application of probing. As variables x_2 and x_3 were not at one of their respective bounds in the solution of (3), we cannot apply Tests 1 or 2 to these variables. We can, however, apply probing. To probe in the direction of the lower bound of x_2, we solve (3) with the additional constraint $x_2 \leq 0$. The solution obtained is $L = -6$ and $\lambda_2^* = 1$. Test 3 now yields:

$$\kappa_2 = x_2^L + \frac{L - U}{\lambda_2^*} = 0 + \frac{-6 + 6.666667}{1} = 0.666667.$$

We can then safely set $x_2^L = \kappa_2 = 0.666667$.

3.1.4. Effect on relaxation. The new bounds ($x_1^L = 4.861904$ and $x_2^L = 0.666667$) can be used to update the relaxation in (3):

$$\begin{aligned} \min \quad & -x_1 - x_2, \\ \text{s.t.} \quad & x_3^2 + (29.171424 - 10.8619x_1) \\ & \quad + (2.666667 - 4.666667x_2) \leq 8, \\ & x_3 = x_1 + x_2, \\ & 4.861904 \leq x_1 \leq 6, \\ & 0.666667 \leq x_2 \leq 4, \\ & 0 \leq x_3 \leq 10. \end{aligned} \tag{4}$$

Comparing (4) to (3), we observe that a very large percentage of the feasible region of (3) has been fathomed by tightening the lower bounds as well as the underestimator of the nonconvex constraint. In fact, by solving (4), we can terminate the search since we obtain a lower bound $L = -6.666667$, which is exactly equal to the upper bound.

3.2. Feasibility-based range reduction tests

3.2.1. Motivating example. Once again, let us consider the nonconvex example (1) above. After

the solution of the relaxed problem (3), Test 1 provided a new lower bound for x_1: $x_1^L = 4.861904$. Observe that one can now make use of the nonconvex constraint $x_1x_2 \leq 4$ of (1) to infer a new upper bound for variable x_2:

$$x_1x_2 \leq 4 \Rightarrow x_2 \leq \frac{4}{x_1} \Rightarrow x_2^U \leq \frac{4}{x_2^L} = \frac{4}{4.861904} = 0.822723.$$

As shown in this example, range reduction tests can be developed by considering the effect of constraints. In general, assuming the feasible region of problem P is convex, it is a standard practice to solve $2n$ convex problems to find the minimum and maximum values of each variable subject to the constraints of P and the current bounds. In the case of concave minimization over a polyhedron this would require the solution of $2n$ LPs (e.g. Quesada and Grossmann, 1992). In the event that this procedure is too costly to apply or the feasible region of P is not convex, one can examine individual constraints to infer bounds. Hansen *et al.* (1991) exemplify this process and present methods that can be used for inferring bounds (monotonicity, interval analysis, solving for roots of nonlinear equations); they then develop a branch-and-bound algorithm that branches on the active constraints in order to take advantage of these tests. Here, we use the tests in the context of the branch-and-reduce algorithm. Moreover, we go one step further to develop techniques for obtaining additional, usually nonlinear, constraints that can be used for further feasibility-based tightening. Before doing so, it should be noted that particularly useful in the context of the branch-and-reduce algorithm are the following two kinds of tests for bounds tightening based on individual constraints:

Type I. The process of inducing separability in problem P requires the introduction of additional variables into the model (e.g. McCormick, 1972b). The new variables are linked to the original ones through simple relationships that are easy to analyze for implied bounds. If, for example, $x_2 = x_1^2$, an upper bound for x_1 can be used to infer an upper bound for x_2: $x_2 \leq (x_1^U)^2$.

Type II. Original problem constraints can also be used to infer bounds. For example, consider the nonconvex constraint $x_1x_2 \geq a$ that involves two positive variables. The constraint implies $x_1 \geq a/x_2$ and therefore an upper bound, x_2^U, for x can be used to yield a lower bound for x_1: $x_1^L = \max(x_1^L, a/x_2^U)$.

Type I and Type II feasibility-based tests capture a nonlinear, and sometimes nonconvex, relationship between variables. Their importance is that the bounds derived from these tests can be used in constructing tighter underestimators of the nonconvex terms of P. It is therefore important to utilize these tests even in the case of simple linear constraints. For example, the constraint $x_3 = x_1 + x_2$ implies bounds for variable x_3 from bounds on x_1 and x_2. Although strengthening the bounds for x_3 in this case may seem redundant, it is not if x_3 is involved in a nonconvex term (e.g. $-x_3^2$) whose relaxation is derived based on the bounds of x_3.

Remark 6. The above tests can be used at any node in the branch-and-bound tree. In fact, they become more important as branching occurs. Once branching is performed on only one variable, other variable bounds can be contracted simultaneously. This bounds contraction, in turn, tightens the relaxion and expedites the search.

Remark 7. In general, almost any nonconvex constraint can be treated as exemplified above and the tests are easy to apply. While analyzing the constraints for implied bounds, *implication lists* can be created. These lists can, in addition, be used in choosing the branching variable. By branching on the variable with the larger implication list, convergence is expected to be expedited.

Remark 8. As in the case of the optimality-based tests, an analogy can be drawn between the feasibility-based tests and procedures used in integer programming where variables are fixed by considering "integrality" requirements of certain variables (e.g. Nemhauser and Wolsey, 1988).

As the above tests operate on constraints, it is advantageous to develop techniques that can generate additional valid inequalities for the problem at hand. This is the subject of the following theorem:

Theorem 4. Assume that R is a convex optimization problem with an optimal objective function value of L and assume that the constraint $\bar{g}_i(\mathbf{x}) \leq 0$ is active at the solution of problem R with a multiplier value of $\mu_i^* > 0$. Let U be a known upper bound for problem P. Then, the constraint:

$$\bar{g}_i(\mathbf{x}) \geq -\frac{U-L}{\mu_i^*},$$

does not exclude any solutions with objective function values better than U and can be added to the current formulation of problem P.

Proof. Similar to Theorem 2.

Corollary. Assume that R is a convex optimization problem with an optimal objective function value of L and assume that the constraint $\mathbf{a}'\mathbf{x} \leq b$ is active at the solution of problem R with a multiplier

value of $\mu^* > 0$. Let U be a known upper bound for problem P. Then, the constraint:

$$\mathbf{a}'\mathbf{x} \geq b - \frac{U - L}{\mu^*},$$

does not exclude any solutions with objective function values better than U and can be added to the current relaxation R.

As the above results indicate, one can derive valid inequalities from the solution of the relaxed problem. In general, these inequalities will be nonconvex. One can then append these constraints to the original nonconvex formulation. Alternatively, these inequalities can be generated locally (at some node), then used for range reduction through Tests I and II, and eventually discarded before the solution of the next node. In this way, constraints do not accumulate during the search while the opportunity of improving variable bounds still exists. In the case of linear constraints, it might be worth to augment the original formulation by introducing the new constraints, if they do not destroy any special structure the problem might possess.

Remark 9. Whenever the tests proposed in this section strengthen the representation of the current subproblem by improving variable bounds, the algorithm of Section 2 reconstructs the relaxation and resolves the subproblem (Step 6). It is likely that the range reduction tests lead to a very small improvement in variable bounds. However, thanks to the presence of *MAXSOLVE* in the algorithm, no subproblem is solved more than *MAXSOLVE* times, thus avoiding the possibility of slow, asymptotic convergence of the bounds.

4. COMPUTATIONAL EXPERIENCE WITH THE PROPOSED ALGORITHM

4.1. Collection of test problems

To illustrate the wide applicability of the above principles, several global optimization test problems and engineering design problems are solved. The examples are described in detail in Appendix B where information is also provided on local and global optimal solutions of these problems. Some characteristics of these examples are provided in Table 1. Some of the above problems are solved here for the first time to global optimality (e.g. Examples 3, 4 and 20). Despite its small size, Example 4 had defeated previous investigations. For this problem, we have been able to find a solution with a cost of \$5195 which is better than the one reported in the literature with a cost of \$5339. It is worth noting that Example 20 constitutes a very difficult test problem as it possesses a local mimimum with an objective function value that is very close to that of the global solution (the difference between the two objective values is less than 0.001). Moreover, the two solutions correspond to two completely different process structures (see the Appendix).

4.2. Computational results with separable reformulations

To obtain a measure of the degree of nonconvexity of the above examples, the conventional local minimization package GAMS/MINOS5.3 (Brooke *et al.*, 1988) was first used to solve them. Two different initialization schemes were employed. In the first, the GAMS default starting point was used (for each variable, the zero point is used if it is

Table 1. Brief description of example problems

Example	Constraints	Variables	Description
1	1	2	Bilinear constraint
2	3	3	Design of a water pumping system
3	7	10	Alkylation process optimization
4	2	4	Design of insulated tank
5	3	5	Heat exchanger network design
6	3	3	Chemical equilibrium
7	7	10	Pooling problem
8	2	2	Bilinear and quadratic constraints
9	1	2	Bilinear constraints, bilinear objective
10	1	2	Nonlinear equality constraint
11	2	3	Bilinearities, economies of scale
12	3	4	Design of two-stage process systems
13	3	2	MINLP, process synthesis
14	10	7	MINLP, process synthesis
15	6	5	MINLP, process synthesis
16	9	12	Heat exchanger network synthesis
17	2	2	Design of a reinforced concrete beam
18	4	2	Quadratically constrained linear program
19	2	2	Quadratically constrained quadratic program
20	6	5	Reactor network design
21	6	5	Design of three-stage process system with recycle

Table 2. Solutions from GAMS/MINOS

Example	Initial solution I	Initial solution II	Global optimum
1	—	−6.6667	−6.6667
2	—	201.1593	201.1593
3	−1,161.3367	−1,161.3367	−1,161.3367
4	5,194.8662	5,194.8662	5,194.8662
5	7,049.2493	7,049.2493	7,049.2493
6	0	0	0
7	−400	−400	−400
8	—	0.7418	0.7418
9	−0.5	−0.5	−0.5
10	−16.7389	−16.7389	−16.7389
11	189.3116	189.3116	189.3116
12	−4.5142	−4.5142	−4.5142
13	—	—	2
14	7.3982	—	4.5796
15	—	—	7.6672
16	—	13,680.7910	12,292.4671
17	—	—	376.2919
18	−2.8284	−2.8284	−2.8284
19	10.3538	10.3538	−118.705
20	−0.3888	−0.387	−0.3888
21	−13.4019	−13.4019	−13.4019

— Did not converge to a feasible solution.

between the lower and upper variable bounds, or else the bound that is closer to zero is selected). In the second initialization scheme, the starting point was specified as the midpoint of the interval defined by the bounds on the variables. As seen in Table 2, the NLP solver did not converge to a feasible solution in several examples (all these problems involved nonconvex feasible regions). Among the rest, several were solved to global optimality while in some cases the solutions found were local minima from 10% to a few orders of magnitude away from the global optimum. In all cases, the CPU time was less than about 1 s on a Sun SPARC station 2. The global minima presented in Table 2 were found using the strategies described below:

Strategy 1: standard branch-and-bound with the addition of the local heuristic (Step 5) and the branching rule of Swaney (1990) for modifying the maximum deviation branching rule.

Strategy 2: Strategy 1 plus the use of marginal values (Tests 1 and 2) and feasibility-based tests (Type I and II).

Strategy 3: Strategy 2 with the addition of probing (Tests 3 and 4).

For the results of this subsection, the relaxations were developed by introducing additional variables and constraints to induce separability. This was achieved by introducing a new variable $u = x + y$ for each bilinear term xy, replacing $2xy$ by $u^2 - x^2 - y^2$ and using linear underestimators for the concave quadratics (as was done for the Illustrative example of Section 3.1). Multilinear terms and ratios were treated by recursive application of the rule for bilinear terms. For example, a term of the form xzy becomes xw when the equality constraint $w = yz$ is introduced into the problem. Subsequently, the bilinear terms xw and yz can be treated as above. Integer variables were modeled using the continuous reformulation of Remark 1 of Section 2. Although, for the purpose of this study, the relaxations were derived manually, it is possible to automate this process using symbolic manipulation packages.

Computational results are presented in Table 3 where N_{tot}, N_{opt} and N_{mem} respectively denote the total number of nodes in the search tree, the node where the global optimum was found, and the maximum number of nodes stored at any point during the

Table 3. Computational results with different search strategies

	Strategy 1				Strategy 2				Strategy 3			
Example	N_{tot}	N_{opt}	N_{mem}	T	N_{tot}	N_{opt}	N_{mem}	T	N_{tot}	N_{opt}	N_{mem}	T
1	3	1	2	0.8	1	1	1	0.5	1	1	1	0.7
2	1062[a]	1	1001[a]	150[a]	1	1	1	0.3	1	1	1	0.3
3	2122[a]	1	113[a]	1245[a]	13	1	5	5.3	9	1	4	14
4	1000[a]	1	1000[a]	140[a]	31	1	11	7	7	1	3	33
5	1000[a]	1	1000[a]	417[a]	115	1	21	14	55	1	14	31
6	1	1	1	0.3	1	1	1	0.3	1	1	1	0.3
7	205	1	37	43	37	1	14	11	37	1	14	23
8	43	1	8	10	1	1	1	0.8	1	1	1	0.8
9	2192[a]	1	1000[a]	330[a]	21	1	8	5.2	21	1	8	6.9
10	1	1	1	0.4	1	1	1	0.4	1	1	1	0.4
11	81	1	24	19	3	1	2	0.6	1	1	1	0.7
12	3	1	2	0.6	1	1	1	0.2	1	1	1	0.2
13	7	2	3	1.3	3	1	2	0.7	1	1	1	0.7
14	7	3	3	3.4	7	3	3	2.7	3	3	2	2.7
15	15	8	5	3.4	1	1	1	0.3	1	1	1	0.3
16	2323[a]	1	348[a]	1211[a]	1	1	1	2.2	1	1	1	2.4
17	1000[a]	1	1001[a]	166[a]	1	1	1	3.7	1	1	1	4
18	1	1	1	0.5	1	1	1	0.5	1	1	1	0.6
19	85	1	14	11.4	9	1	4	1.8	1	1	1	1.4
20	3162[a]	1	1001[a]	778[a]	541	1	88	125	413	1	82	205
21	7	1	4	1.2	1	1	1	0.5	1	1	1	0.5

[a] Did not converge within limits of $T \leq 1200$ (= 20 min), and $N_{mem} \leq 1000$ nodes.

Table 4. CPU times (s) from FORTRAN implementation

	Time (% of total time)						
Example	Relaxed	Local	Tightening	Marginals	Probing	Other	Total
1	0.09 (24%)	0.21 (55%)	0	0	0	0.08 (21%)	0.38
2	0.53 (16%)	0.04 (1%)	0	0	2.58 (79%)	0.12 (4%)	3.27
4	1.49 (74%)	0.26 (13%)	0.01 (0.5%)	0	0.17 (8%)	0.08 (4.5%)	2.01
14	1.16 (24%)	2.96 (62%)	0	0	0.52 (11%)	0.13 (3%)	4.77

search. The termination criterion used throughout the computations was $\varepsilon = 10^{-6}$. Finally, T is the CPU time in seconds on a Sun SPARC station 2. The results of Table 3 motivate the following discussion:

- Conventional branch-and-bound (Strategy 1) did not converge within the preset time and memory limits, even for the small problems. This is partly due to the fact that the way the relaxations were constructed in this computational study only guarantees asymptotic convergence for problems with nonconvex constraints. Additional runs using the standard maximum deviation subdivision rule and-or excluding the local search heuristic gave much worse results than Strategy 1.
- Compared to Strategy 1, the more sophisticated branch-and-reduce strategies (2 and 3) reduce the total number of nodes (N_{tot}), as well as the maximum number of nodes needed to be stored (N_{mem}) during the search.
- Strategy 3 sometimes takes more CPU time than Strategy 2 since probing requires solving additional relaxed problems. However, Strategy 3 reduced the memory requirements (N_{mem}) of Strategy 2 in eight examples (3, 4, 5, 11, 13, 14, 19, 20).
- The local search heuristic was capable of identifying the global optimum at an early stage of the search. In 20 of the 21 examples, the heuristic found the global optimum at the root node when Strategy 2 or 3 was used. For Strategy 1, this happened in 18 of the 21 examples. In terms of time, the heuristic consumed only a very small portion of the total time required. This indicates that incorporating the local minimization heuristic is highly advantageous. Not only is it computationally inexpensive, but it also provides the means for faster fathoming of inferior nodes.

The above computations were carried out using GAMS/MINOS5.3 (Brooke *et al.*, 1988) and the CPU times presented in Table 3 account for solving all the optimization subproblems (including those for probing). Preliminary computational results with a FORTRAN implementation of the algorithm are presented in Table 4 (using MINOS5.4 as the NLP solver). The table provides information on the distribution of the CPU time among the various tasks of the algorithm. Apparently, most of the time is spent on solving the relaxations, while very little time is spent on applying the pre- and post-processing steps, performing local minimization and bookkeeping.

4.3. *Impact of tighter relaxations*

The computational results of the previous subsection indicated that the branch-and-reduce strategy solved most of the problems in a very small number of nodes. Exceptions were problems 5, 7, 9 and 20. For these problems, this subsection considers the effect of a tighter relaxation. In particular, bilinear terms of the form xy were approximated by the following constraint set (see, e.g. McCormick, 1983):

$$xy \geq \max\{y^U x + x^U y - x^U y^U, y^L x + x^L y - x^L y^L\},$$

$$xy \leq \min\{x^L y + y^U x - x^L y^U, y^L x + x^U y - y^L x^U\}.$$

It can be easily verified that the above relationships give rise to tighter relaxations than the ones based on the separable reformulations of Section 4.2. The effect of the tighter relaxations on the performance of the algorithm is shown in Table 5. The algorithm this time required a smaller number of nodes, as expected due to the tighter lower bounds. Strategy 3 once again required the smallest number of nodes and reduced the memory requirements of Strategy 2 for Examples 9 and 20.

4.4. *Comparisons with other approaches*

As most of the examples used in this study have also been used as test problems for other global optimization approaches, they can serve as a basis for preliminary comparisons.

- Examples 7, 10, 12, 13, 14, 15, 16 and 18 were used in the testing of the decomposition global

Table 5. Impact of tighter relaxations

Example	Strategy 1				Strategy 2				Strategy 3			
	N_{tot}	N_{opt}	N_{mem}	T	N_{tot}	N_{opt}	N_{mem}	T	N_{tot}	N_{opt}	N_{mem}	T
5	85	1	12	10	23	1	6	2	15	1	6	3
7	3	1	2	1	3	1	2	1	3	1	2	1.2
9	7	1	4	1.1	7	1	4	1.1	5	1	3	1.7
20	4293	1	274	470	179	1	23	21	91	1	20	30

optimization approaches of Floudas *et al.* (1989) and Visweswaran and Floudas (1990). As seen in Table 3, Strategy 3 of the proposed algorithm solved six of these eight problems at the root node, whereas the approaches of Floudas and co-workers took several iterations—with several subproblems solved at each iteration.

- Three of the examples (6, 10 and 18) can be solved to optimality when the nonconvexities are ignored. For this reason, Strategy 1 converged at the root node for these problems. This result demonstrates that the approach is capable of recognizing certain convex problems. This property is not shared by other global optimization approaches, including the decomposition methods of Floudas and coworkers whose number of iterations seems to depend on the starting point. It is also not present in other branch-and-bound-based global optimization approaches (e.g. Swaney, 1990).
- Example 20 is from Manousiouthakis and Sourlas (1992) whose branch-and-bound algorithm took 7950 nodes. As seen in Table 5, for the same example, Strategies 2 and 3 required only 179 and 91 nodes, respectively.
- Examples 7–12 were also solved in Swaney (1990) whose branch-and-bound algorithm required fewer nodes for Eamples 7 and 9. This was achieved by solving a number of additional cover LPs for testing for optimality of the incumbent.
- Example 3 is nontrivial since it involves several nonconvexities and nonlinear equality constraints. Due to these difficulties, the conventional branch-and-bound strategy (Strategy 1 of Table 3) did not converge after two thousand nodes. For this example, the branch-and-bound algorithm of Quesada and Grossmann (1993) found the global optimum but provided a lower bound within only 5% of the upper bound after 53 nodes. On the other and, the branch-and-reduce Strategies 2 and 3 (Table 3) took only 13 and 9 nodes, respectively, to prove global optimality with $\varepsilon = 10^{-6}$.
- Finally, in comparing the above computational results to those found in the above-cited papers, it should be noted that the convergence criterion used in other works is rarely smaller than $\varepsilon = 10^{-3}$. The computational results reported here were obtained by requiring a much stricter termination criterion: $\varepsilon = 10^{-6}$. Therefore, the lower bounds obtained are much more accurate. Interestingly enough, *for all the examples solved in this study, no subproblems were fathomed with objective function values lower than the upper bounds at termination.*

4.5. *Results with large-scale problems*

To further demonstrate the potential of the algorithm, preliminary results with two large-scale problems are presented. In both cases, Strategy 2 was used with an absolute termination criterion of $\varepsilon = 10^{-6}$.

The first problem is the fourth MINLP example of Duran and Grossmann (1986). This model involves 30 variables, 25 of which are binary, and 30 constraints, 25 of which are nonlinear. The physical problem corresponds to determining the optimal positioning of a new product in a multiattribute space. The algorithm required 85 nodes and 190 seconds. Despite the small number of nodes, the CPU time is larger than the examples of the previous sections due to the large number of nonlinearities in the model.

To obtain a second large-scale test problem for the algorithm, we generated a problem with 200 variables and 200 constraints. All the constraints were linear and the objective was constructed by taking the product of five linear functions of all the 200 variables. In this way, all 200 variables appeared in nonconvex terms. All coefficients were generated randomly and the constraint set was fully dense. The global minimum was found at the root node and the search was complete after 183 nodes and 231 CPU s. The proposed algorithm was capable of handling these two large-scale problems very effectively both in terms of number of nodes as well as in terms of CPU times.

5. CONCLUSIONS

Optimality-based and feasibility-based tests have been developed for range reduction in continuous global optimization and their use has been illustrated in the context of a branch-and-bound-based algorithm. The former tests are based on dual solutions and probing while the latter on feasibility considerations and simultaneous bounds contraction during branching. The proposed tests aim at tightening variable bounds. Based on these tight bounds, the relaxation of a non-convex problem can itself be tightened by deriving tighter underestimators. The entire process is applied at each node of the search tree thus leading to a branch-and-reduce algorithm. In each node of the tree, the algorithm essentially generates new cuts: the new variable bounds and the tighter underestimators. Unlike other cutting plane methods for global optimization, however, the cuts derived by this process do not destroy any special structure the relaxation might possess and they are very easy to implement.

The proposed algorithm was tested on a collection of global optimization and process design problems (NLPs and MINLPs). The approach was found to be highly efficient for all these problems. Therefore, future work should be directed towards the development of specialized branch-and-reduce algorithms for particular classes of global optimization problems. A general purpose, FORTRAN-based, branch-and-reduce optimizer is currently under development. Future work will include the use of one-parametric nonlinear programming techniques to solve the perturbed problems and to apply tighter tests based on calculating the intersection of the graph of the perturbation function with the $\varphi = U$ line of Figs 2 and 3. Also, systematic techniques will be developed to improve the feasibility-based tests after branching.

Finally, it should be noted that the proposed tests can be incorporated in the context of other (not necessarily branch-and-bound-based) global optimization algorithms. Particularly appealing also seems to be the integration of tight formulations (e.g. Quesada and Grossmann, 1993) and sufficient tests for optimality (e.g. Swaney, 1990) within the branch-and-reduce scheme.

Acknowledgements—Partial financial support from the University of Illinois and a DuPont young faculty research initiation grant is gratefully acknowledged. This paper has benefitted from discussions with Professors Udatta Palekar and Bruce Lamar.

REFERENCES

Bracken J. and G. P. McCormick, *Selected Applications of Nonlinear Programming*. Wiley, New York (1968).

Brooke A., D. Kendrick and A. Meeraus, *GAMS—A User's Guide*. The Scientific Press, Redwood City (1988).

Duran M. A. and I. E. Grossmann, 'An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Math. Program.* **36**, 307 (1986).

Falk J. E., Conditions for global optimality in nonlinear programming. *Opers Res.* **21**, 337–340 (1969).

Falk J. E. and R. M. Soland, An algorithm for separable nonconvex programming problems. *Mgmt Sci.* **15**, 550–569 (1969).

Floudas C. A. and A. R. Ciric, Strategies for overcoming uncertainties in heat exchanger network synthesis. *Computers chem. Engng* **13**, 1133–1152 (1989).

Floudas C. A. and V. Visweswaran, A global optimization algorithm (GOP) for certain classes of nonconvex NLPs—I. Theory. *Computers chem. Engng* **14**, 1397–1417 (1990).

Floudas C. A., A. Aggarwal and A. R. Ciric, Global optimum search for nonconvex NLP and MINLP problems. *Computers chem. Engng* **13**, 1117–1132 (1989).

Hansen P., B. Jaumard and S.-H. Lu, An analytical approach to global optimization. *Math. Program.* **52**, 227–254 (1991).

Haverly C. A., Studies of the behaviour of recursion for the pooling problem. *SIGMAP Bull.* **25**, 19 (1978).

Horst R., Deterministic methods in constrained global optimization: some recent advances and new fields of application. *Naval Res. Logist.* **37**, 433–471 (1990).

Horst R. and H. Tuy, *Global Optimization: Deterministic Approaches*. Springer-Verlag, Berlin (1990).

Horst R. and H. Tuy, *Global Optimization: Deterministic Approaches*, 2nd Edn. Springer-Verlag. Berlin (1993).

Kocis G. R. and I. E. Grossmann, Global optimization of nonconvex MINLP problems in process synthesis. *Ind. Engng Chem.—Res.* **27**, 1407 (1988).

Lasdon L. S., A. D. Waren, S. Sarkar and F. Palacios, Solving the pooling problem using generalized reduced gradient and successive linear programming algorithms. *SIGMAP Bull.*, **77**, 9–15 (1979).

Liebman J., N. Khachaturian and V. Chanaratna, Discrete structural optimization *J. Struct. Div. ASCE*, **107**, No. ST11, *Proceedings Paper 16643* (Nov.), 2177–2197 (1981).

Liebman J., L. Lasdon, L. Schrage and A. Waren, *Modeling and Optimizatin with GINO* The Scientific Press, Palo Alto, CA (1986).

Manousiouthakis M. and D. Sourlas, A global optimization approach to rationally constrained rational programming. *Chem. Engng. Commun.* **115**, 127–147 (1992).

McCormick G. P., Attempts to calculate global solutions of problems that may have local minima. *Numerical Methods for Non-linear Optimization* (Lootsma F. A., Ed.). Academic Press, London (1972a).

McCormick G. P., Converting general nonlinear programming problems to separable nonlinear programming problems. Report T-267, The George Washington University (1972b).

McCormick G. P., Computability of global solutions to factorable nonconvex programs. Part I—convex underestimating problems. *Math. Program.* **10**, 147–175 (1976).

McCormick G. P., *Nonlinear Programming. Theory, Algorithms, and Applications*. Wiley Interscience, New York (1983).

Minoux M., *Mathematical Programming. Theory and Algorithms*. Wiley, New York (1986).

Nemhauser G. L. and L. A. Wolsey, *Integer and Combinatorial Optimization*. Wiley, New York (1988).

Quesada I. and I. E. Grossmann, A global optimization algorithm for linear fractional and bilinear programs. *ORSA/TIMS Meeting*, Chicago, IL (1993).

Sahinidis N. V., Branch and bound experiments in global optimization. *Annual AIChE Mtg*, Miami Beach, FL (1992a).
Sahinidis N. V., Accelerating branch-and-bound in continuous global optimization. University of Illinois at Urbana-Champaign, Department of Mechanical and Industrial Engineering, Research Report UILU ENG 92-4031 (1992b).
Sahinidis N. V. and I. E. Grossmann, MINLP model for cyclic multiproduct scheduling on continuous parallel production lines. *Computers chem. Engng* **15**, 85–103 (1991a).
Sahinidis N. V. and I. E. Grossmann, Convergence properties of Generalized Benders Decomposition. *Computers chem. Engng* **15**, 481–491 (1991b).
Sahinidis N. V. and I. E. Grossmann, Reformulation of the multi-period MILP model for capacity expansion of chemical processes. *Opers Res.* **40**, S127–S144 (1992).
Salcedo R. L., Solving nonconvex nonlinear programming and mixed-integer nonlinear programming problems with adaptive random search. *Ind. Engng Chem. Res.* **31**, 262–273 (1992).
Schoen F., Stochastic techniques for global optimization: a survey of recent advances. *J. Global Optim.* **1**, 207–228 (1991).
Soland R. M., An algorithm for separable nonconvex programming problems II: nonconvex constraints. *Mgmt Sci.* **17**, 759–773 (1971).
Sourlas D., Private communication (1993).
Stoecker W. F., *Design of Thermal Systems*. McGraw-Hill, New York (1971).
Stephanopoulos G. and A. W. Westerberg, The use of Hestenes' Method of multipliers to resolve dual gaps in engineering system optimization. *J. Optim. Theory Applic.* **15**, 285–309 (1975).
Swaney R. E., Global solution of algebraic nonlinear programs. *AIChE Annl Mtg*, Chicago, IL (1990).
Thakur L. S., Domain contraction in nonlinear programming: minimizing a quadratic concave function over a polyhedron. *Math. Opers Res.* **16**, 390–407 (1990).
Törn A. and A. Zilinskas, *Global Optimization, Lecture Notes in Computer Science*. p. 350. Springer-Verlag, Berlin (1989).
Tuy H., Concave programming under linear constraints. *Sov. Math.* **5**, 1437–1440 (1964).
Visweswaran V. and C. A. Floudas, A. global optimization algorithm (GOP) for certain classes of nonconvex NLPs—II. Application of theory and test problems. *Computers chem. Engng* **14**, 1419–1434 (1990).
Wang B.-C. and R. Luus, Reliability of optimization procedures for obtaining global optimum. *AIChE. Jl* **24**, 619–626 (1978).
Westerberg A. W. and J. V. Shah, Assuring a global optimum by the use of an upper bound on the lower (dual) bound. *Computers chem. Engng* **2**, 83–92 (1978).
Yuan X., S. Zhang, L. Piboulcau and S. Domenech, Une méthode d'optimisation non linéaire en variables mixtes pour la conception de procédés. *Reche. Opérat/Opers Res.* **22**, 331–346 (1988).

APPENDIX A

Proof of Theorem 3

Proof. We will prove that test 4 is valid. Consider the following perturbation problem after solving R with $x_j = x_j^L$ (i.e. after including $x_j \leq x_j^L$ in R):

Problem $P(\mathbf{y})$:

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

Since, for any $\mathbf{y}$, $R(\mathbf{y})$ is a convex relaxation of $P(\mathbf{y})$, we have $\varphi(\mathbf{y}) \leq \Phi(\mathbf{y})$. Therefore, an underestimator of $\varphi(\mathbf{y})$ is also an underestimator of $P(\mathbf{y})$. Consider now the perturbed problem $R(\mathbf{y})$ where only the right-hand side of constraint $x_j - x_j^L \leq y$ is perturbed. It follows from Theorem 1 that a valid underestimator for $\varphi(y)$ is $L - \lambda_j^* y$ and hence: $L - \lambda_j^* y \leq \varphi(y) \leq \Phi(y)$. By requiring that the value of $\Phi(y)$ be better than the already known upper bound U, we obtain: $L - \lambda_j^* y < U$. Finally, consider only nonpositive values for y. Since $x_j - x_j^L \leq y$ is active for $y = 0$, it will also be active in the solution of $R(y)$ for any $y \leq 0$, i.e. $y = x_j - x_j^L$. Substituting this relationship in $L - \lambda_j^* y < U$ and rearranging yields the required result $x_j > x_j^L + ((L - U)/\lambda_j^*)$. The cases of Tests 2 and 3 follow directly from Tests 1 (Theorem 2) and Test 4, respectively. □

APPENDIX B

Test Problems and Solutions

This Appendix describes 21 global optimization test problems. For all the problems, we state the global solution as well as known local solutions (local minima and local maxima). Additional local solutions might exist unless otherwise stated.

Example 1

Sahinidis and Grossmann, 1991b):

$$\begin{aligned} \min \quad & -x_1 x_2, \\ \text{s.t.} \quad & x_1 x_2 \leq 4, \\ & \mathbf{0} \leq \mathbf{x} \leq (6, 4). \end{aligned}$$

The global optimum is $\mathbf{x} = (6,\ 0.666667)$ with $f = -6.666667$. There is only one local minimum is at $\mathbf{x} = (1, 4)$ with $f = -5$.

Example 2—Water pumping system

Stoecker (1971), in Liebman *et al.* (1986) modified:

$$\begin{aligned} \min \quad & x_3, \\ \text{s.t.} \quad & x_3 = 250 + 30x_1 - 6x_1^2, \\ & x_3 = 300 + 20x_2 - 12x_2^2, \\ & x_3 = 150 + 0.5(x_1 + x_2)^2, \\ & \mathbf{0} \leq \mathbf{x} \leq (9.422, 5.903, 267.42). \end{aligned}$$

The global optimum is $\mathbf{x} = (6.293429,\ 3.821839,\ 201.159334)$ with $f = 201.159334$.

Example 3—Alkylation process optimization

Bracken and McCoarmick (1968), modified in Liebman *et al.* (1986) and Quesada and Grossmann (1993) Fig. 4):

$$\begin{aligned} \min \quad & 5.04x_1 + 0.035x_2 + 10x_3 + 3.36x_5 - 0.063x_4x_7, \\ \text{s.t.} \quad & x_1 = 1.22x_4 - x_5, x_9 + 0.222x_{10} = 35.82, \\ & x_9 + 0.222x_{10} = 35.82, \\ & 3x_7 - x_{10} = 133, \end{aligned}$$

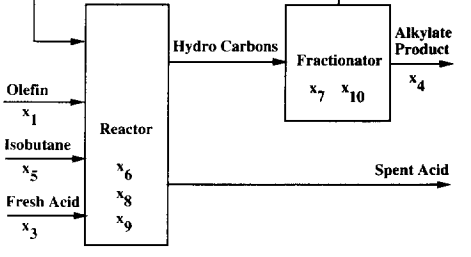

Fig. 4. Alkylation process optimization.

$$x_7 = 86.35 + 1.098x_8 - 0.038x_8^2 + 0.325(x_8 - 89),$$
$$x_4x_9 + 1000x_3 = 98{,}000x_3/x_6,$$
$$x_2 + x_5 = x_1x_8,$$
$$1.12 + 0.13167x_8 - 0.00667x_8^2 \geq x_4/x_1,$$
$$(1, 1, 0, 1, 0, 85, 90, 3, 1.2, 145) \leq \mathbf{x} \leq (2000, 16{,}000, 120, 5000, 2000, 93, 95, 12, 4, 162).$$

The global optimum is $\mathbf{x} = (1728.310416, 16{,}000, 98.133457, 3055.992144, 2000, 90.618812, 94.189777, 10.414796, 2.615609, 149.569330)$ with $f = -1161.336694$.

Example 4—Insulated steel tank design

Stoecker (1971) in Liebman *et al.* (1986):

$$\begin{aligned} \min \quad & 400x_1^{0.9} + 1000 + 22(x_2 - 14.7)^{1.2} + x_4, \\ \text{s.t.} \quad & x_2 = \exp[-3950/(x_3 + 460) + 11.86], \\ & 144(80 - x_3) = x_1x_4, \\ & (0, 14.7, -459.67, 0) \leq \mathbf{x} \leq (15.1, 94.2, 80, \infty). \end{aligned}$$

The global optimum is $\mathbf{x} = (0, 94.177866, 80, 0)$ with $f = 5194.866243$. There is a local minimum at $\mathbf{x} = (15.954, 29.404, 5.864, 669.148)$ with $f = 5339.253$.

Example 5—Heat exchanger network design

Liebman *et al.* (1986) (Fig. 5):

$$\begin{aligned} \min \quad & x_1 + x_2 + x_3, \\ \text{s.t.} \quad & 100{,}000(x_4 - 100) = 120x_1(300 - x_4), \\ & 100{,}000(x_5 - x_4) = 80x_2(400 - x_5), \\ & 100{,}000(500 - x_5) = 40x_3(600 - 500), \\ & (0, 0, 0, 100, 100) \\ & \quad \leq \mathbf{x} \leq (15{,}834, 36{,}250, 10{,}000, 300, 400). \end{aligned}$$

The global optimum is $\mathbf{x} = (579.306743, 1359.9712666, 5109.971263, 182.017600, 295.601150)$ with $f = 7049.249$.

Example 6—Chemical equilibrium problem

Liebman *et al.* (1986) modified:

$$\begin{aligned} \min \quad & 0, \\ \text{s.t.} \quad & x_3^2/x_1x_2^3 = 0.000169, \\ & x_1 + x_2 + x_3 = 50, \\ & x_2/x_1 = 3, \\ & \mathbf{0} \leq \mathbf{x} \leq (12.5, 37.5, 50). \end{aligned}$$

The global optimum is $\mathbf{x} = (10.601856, 31.805569, 7.592574)$ with $f = 50$. Note that the original problem is a feasibility problem which has been stated here as an optimization problem by using a constant as the objective.

Example 7—Pooling problem

Haverly (1978), in Lasdon *et al.* (1979), in Liebman *et al.* (1986), in Swaney (1990) and in Visweswaran and Floudas (1990) (Fig. 6):

$$\begin{aligned} \min \quad & -9x_5 - 15x_9 + 6x_1 + 16x_2 + 10x_6, \\ \text{s.t.} \quad & x_1 + x_2 = x_3 + x_4, \\ & x_3 + x_7 = x_5, \\ & x_4 + x_8 = x_9, \\ & x_7 + x_8 = x_6, \\ & x_{10}x_3 + 2x_7 \leq 2.5x_5, \end{aligned}$$

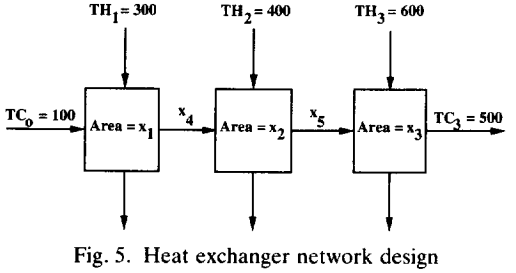

Fig. 5. Heat exchanger network design

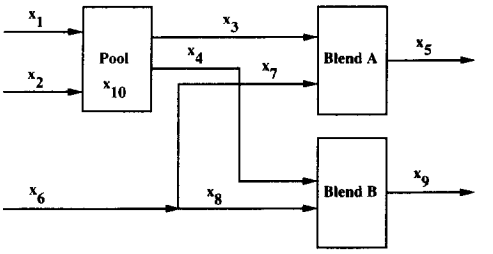

Fig. 6. Pooling problem

$$x_{10}x_4 + 2x_8 \leq 1.5x_9,$$
$$3x_1 + x_2 = x_{10}(x_3 + x_4),$$
$$(0, 0, 0, 0, 0, 0, 0, 0, 0, 1) \leq \mathbf{x} \leq (300, 300, 100, 200, 100, 300, 100, 200, 200, 3).$$

The gobal optimum is $\mathbf{x} = (0, 100, 0, 100, 0, 100, 0, 100, 200, 100)$ with $f = -400$. One local minimum occurs at $\mathbf{x} = (50, 0, 50, 0, 100, 50, 50, 0, 0, 3)$ with $f = -100$. There are infinite local solutions to this problem with an objective function value of zero at $\mathbf{x} = (0, 0, 0, 0, 0, 0, 0, 0, 0, a)$ where $1 \leq a \leq 3$.

Example 8

Swaney (1990):

$$\begin{aligned} \min \quad & 2x_1 + x_2, \\ \text{s.t.} \quad & -16x_1x_2 + 1 \leq 0, \\ & -4x_1^2 - 4x_2^2 + 1 \leq 0, \\ & \mathbf{0} \leq \mathbf{x} \leq \mathbf{1}. \end{aligned}$$

The global optimum is $\mathbf{x} = (0.129409, 0.482963)$ with $f = 0.741782$.

Example 9

Swaney (1990):

$$\begin{aligned} \min \quad & -2x_1x_2, \\ \text{s.t.} \quad & 4x_1x_2 + 2x_1 + 2x_2 - 3 \leq 0, \\ & \mathbf{0} \leq \mathbf{x} \leq \mathbf{1}. \end{aligned}$$

The global optimum is $\mathbf{x} = (0.5, 0.5)$ with $f = -0.5$.

Example 10

Soland (1971), in Stephanopoulos and Westerberg (1975), in Floudas *et al.* (1989), in Swaney (1990):

$$\begin{aligned} \min \quad & -12x_1 - 7x_2 + x_2^2, \\ \text{s.t.} \quad & -2x_1^4 - x_2 + 2 = 0, \\ & \mathbf{0} \leq \mathbf{x} \leq (2, 3). \end{aligned}$$

The global optimum is $\mathbf{x} = (0.717536, 1.469842)$ with $f = -16.738893$.

Example 11

Westerberg and Shah (1978), in Swaney (1990):

$$\begin{aligned} \min \quad & 35x_1^{0.6} + 35x_2^{0.6}, \\ \text{s.t.} \quad & 600x_1 - 50x_3 - x_1x_3 + 5000 = 0, \\ & 600x_2 + 50x_3 - 15{,}000 = 0, \\ & (0, 0, 100) \leq \mathbf{x} \leq (34, 17, 300), \\ & (0, 0, 100) \leq \mathbf{x} \leq (34, 17, 300). \end{aligned}$$

The global optimum is $\mathbf{x} = (0, 16.666667, 100)$ with $f = 189.311627$. There is a local solution at $\mathbf{x} = (33.333, 0, 300)$ with $f = 286.943$.

Example 12—Design of two-stage process system

Stephanopoulos and Westerberg (1975), in Floudas *et al.* (1989), in Swaney (1990) (Fig. 7):

$$\begin{aligned}\min\ & x_1^{0.6}+x_2^{0.6}-6x_1-4x_3+3x_4,\\ \text{s.t.}\ & -3x_1+x_2-3x_3=0,\\ & x_1+2x_3\leq 4,\\ & x_2+2x_4\leq 4,\\ & \mathbf{0}\leq\mathbf{x}\leq(3,4,2,1).\end{aligned}$$

The global optimum is $\mathbf{x}=(1.333333, 4, 0, 0)$ with $f=-4.514202$.

Example 13—Process synthesis MINLP

Kocis and Grossmann (1988), in Floudas *et al.* (1989):

$$\begin{aligned}\min\ & 2x+y,\\ \text{s.t.}\ & 1.25-x^2-y\leq 0,\\ & x+y\leq 1.6,\\ & 0\leq x\leq 1.6,\\ & y=\{0,1\}.\end{aligned}$$

The global optimum is $x=0.5$ and $y=1$ with $f=2$. There is a local minimum at $x=1.118$ and $y=0$ with $f=2.236$.

Example 14—Process synthesis MINLP

Yuan *et al.* (1888), in Floudas *et al.* (1989):

$$\begin{aligned}\min\ & (y_1-1)^2+(y_2-2)^2+(y_3-1)^2-\log(y_4+1)\\ & +(x_1-1)^2+(x_2-2)^2+(x_3-3)^2,\\ \text{s.t.}\ & y_1+y_2+y_3+x_1+x_2+x_3\leq 5,\\ & y_3^2+x_1^2+x_2^2+x_3^2\leq 5.5,\\ & y_1+x_1\leq 1.2,\\ & y_2+x_2\leq 1.8,\\ & y_3+x_3\leq 2.5,\\ & y_4+x_1\leq 1.2,\\ & y_2^2+x_2^2\leq 1.64,\\ & y_3^2+x_3^2\leq 4.25,\\ & y_2^2+x_3^2\leq 4.64,\\ & \mathbf{0}\leq\mathbf{x}\leq(1.2, 1.8, 2.5),\\ & y_i=\{0,1\},\quad i=1,2,3,4.\end{aligned}$$

The global optimum is $\mathbf{x}=(0.2, 0.8, 1.907878)$ and $\mathbf{y}=(1, 1, 0, 1)$ with $f=4.579582$. We have found the following feasible integer solutions (local minima):

$\mathbf{x}=(0, 0.5, 1.5)$ and $y=(1, 1, 1, 1)$ with $f=5.807$.
$\mathbf{x}=(0.2, 1.281, 2.062)$ and $\mathbf{y}=(1, 0, 0, 0)$ with $f=7.038$.
$\mathbf{x}=(0.2, 1.281, 2.062)$ and $\mathbf{y}=(0, 0, 0, 0)$ with $f=7.398$.
$\mathbf{x}=(0.2, 1.281, 2.062)$ and $\mathbf{y}=(1, 0, 0, 1)$ with $f=6.345$.
$\mathbf{x}=(0.2, 0.800, 1.908)$ and $\mathbf{y}=(1, 1, 0, 0)$ with $f=5.273$.
$\mathbf{x}=(0.20, 0.800, 1.908)$ and $\mathbf{y}=(1, 1, 0, 1)$ with $f=4.580$.
$\mathbf{x}=(0.7, 0.800, 1.500)$ and $\mathbf{y}=(0, 1, 1, 0)$ with $f=5.780$.
$\mathbf{x}=(1, 1.281, 1.5)$ and $\mathbf{y}=(0, 0, 1, 0)$ with $f=7.768$.
$\mathbf{x}=(0.2, 1.281, 1.5)$ and $\mathbf{y}=(1, 0, 1, 0)$ with $f=7.408$.

Example 15

Kocis and Grossmann (1988), in Floudas *et al.* (1989):

$$\begin{aligned}\min\ & 2x_1+3x_2+1.5y_1+2y_2-0.5y_3,\\ \text{s.t.}\ & x_1^2+y_1=1.25,\\ & x_2^{1.5}+1.5y_2=3,\\ & x_1+y_1\leq 1.6,\\ & 1.333x_2+y_2\leq 3,\\ & -y_1-y_2+y_3\leq 0,\\ & y_1,y_2,y_3=\{0,1\},\\ & \mathbf{0}\leq\mathbf{x}\leq(1.12, 2.10).\end{aligned}$$

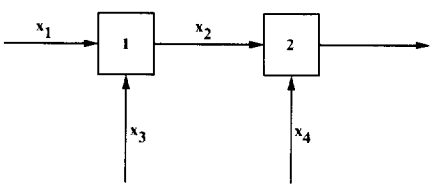

Fig. 7. Design of two-stage process system.

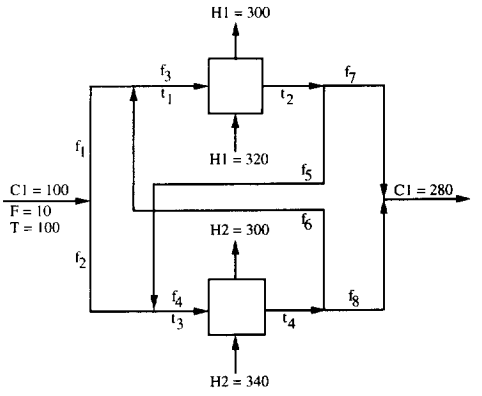

Fig. 8. Heat exchanger network design.

The global optimum is $\mathbf{x}=(1.118034, 1.310371)$ and $\mathbf{y}=(0, 1, 1)$ with $f=7.667180$.

Example 16—Heat exchanger network synthesis

Floudas and Ciric (1989), in Floudas *et al.* (1989) (Fig. 8):

$$\begin{aligned}\min\ & 1200\left[800\Big/2.5\left(2/3\sqrt{(320-t_2)(300-t_1)}+\frac{(320-t_2)+(300-t_1)}{6}\right)\right]^{0.6}\\ & +1200\left[1000\Big/0.2\left(2/3\sqrt{(340-t_4)(300-t_3)}+\frac{(340-t_4)+(300-t_3)}{6}\right)\right]^{0.6},\\ \text{s.t.}\ & f_1+f_2=10,\\ & f_1+f_6=f_3,\\ & f_2+f_5=f_4,\\ & f_5+f_7=f_3,\\ & f_6+f_8=f_4,\\ & 100f_1+t_2f_6=t_1f_3,\\ & 100f_2+t_2f_5=t_3f_4,\\ & f_3(t_2-t_1)=800,\\ & f_4(t_4-t_3)=1000,\\ & \mathbf{0}\leq\mathbf{f}\leq\mathbf{10},\\ & (100, 100, 100, 100)\leq\mathbf{t}\leq(290, 310, 290, 330).\end{aligned}$$

The global optimum is $\mathbf{x}=(0, 10, 10, 10, 0, 10, 10, 0)$ and $\mathbf{t}=(200, 280, 100, 200)$ with $f=12292.467132$. A local solution is present at: $\mathbf{f}=(10, 0, 10, 10, 10, 0, 0, 10)$ and $\mathbf{t}=(100, 180, 180, 280)$ with $f=15{,}446.916$. A local minimum is at: $\mathbf{f}=(4.067, 5.933, 4.067, 5.933, 0, 0, 4.067, 5.933)$ with $\mathbf{t}=(100, 296.720, 100, 268.540)$ with $f=13{,}680.791$.

Example 17—Design of a reinforced concrete beam

Liebman *et al* (1981, 1986):

$$\begin{aligned}\min\ & 29.4x_1+18x_2,\\ \text{s.t.}\ & x_1-0.2458x_1^2/x_2\geq 6,\\ & (0, 0.00001)\leq\mathbf{x}\leq(115.8, 30.0).\end{aligned}$$

The global optimum is $\mathbf{x}=(8.170018, 7.56074)$ with $f=376.291932$.

Example 18

Visweswaran and Floudas (1990):

$$\begin{aligned}\min\ & x_1+x_2,\\ \text{s.t.}\ & x_1^2+x_2^2\leq 4,\\ & x_1^2+x_2^2\geq 1,\end{aligned}$$

$$x_1 - x_2 \leq 1,$$
$$x_2 - x_1 \leq 1,$$
$$-2 \leq \mathbf{x} \leq 2.$$

The global optimum is $\mathbf{x} = (-1.414214, -1.414214)$ with $f = -2.828427$. Two local solutions are at: $\mathbf{x} = (-1, 0)$ with $f = -1$ and at $\mathbf{x} = (1, 0)$ with $f = 1$.

Example 19

Manousiouthakis and Sourlas (1992):

$$\begin{aligned} \min \quad & x_1^4 - 14x_1^2 + 24x_1 - x_2^2, \\ \text{s.t.} \quad & -x_1 + x_2 - 8 \leq 0, \\ & x_2 - x_1^2 - 2x_1 + 2 \leq 0, \\ & (-8, 0) \leq \mathbf{x} \leq (10, 10). \end{aligned}$$

The global optimum is $\mathbf{x} = (-3.173599, 1.724533)$ with $f = -118.704860$. Local solutions occur at $\mathbf{x} = (2.702, 10.702)$ with $f = -98.597$, $\mathbf{x} = (0.840, 0.386)$ with $f = 10.631$ and at $\mathbf{x} = (0.732, 0.000)$ with $f = 10.354$.

Example 20—Reactor network design

Manousiouthakis and Sourlas (1992) (Fig. 9):

$$\begin{aligned} \min \quad & -x_4, \\ \text{s.t.} \quad & x_1 - 1 + k_1 x_1 x_5 = 0, \\ & x_2 - x_1 + k_2 x_2 x_6 = 0, \\ & x_3 + x_1 - 1 + k_3 x_3 x_5 = 0, \\ & x_4 - x_3 + x_2 - x_1 + k_4 x_4 x_6 = 0, \\ & x_5^{0.5} + x_6^{0.5} \leq 4, \\ & \mathbf{0} \leq \mathbf{x} \leq (1, 1, 1, 1, 16, 16), \end{aligned}$$

where $k_1 = 0.09755988$, $k_2 = 0.99k_1$, $k_3 = 0.0391908$, $k_4 = 0.9k_3$.

The global optimum is $\mathbf{x} = (0.771462, 0.516997, 0.204234, 0.388812, 3.036504, 5.096052)$ with $f = -0.388812$. The values of the kinetic constants ($k_1 k_2, k_3, k_4$) and some signs in the constraints were misprinted in the original paper of Manousiouthakis and Sourlas (1992). These authors have used the model as stated here for their computations (Sourlas, 1993).

This example constitutes a very difficult test problem as it possesses a local minimum with an objective function value that is very close to that of the global solution. The local solutions are at: $\mathbf{x} = (0.390, 0.390, 0.375, 0.375, 16, 0)$ with $f = -0.375$ and at $\mathbf{x} = (1, 0.393, 0, 0.388, 0, 16)$ with $f = -0.3881$. Interestingly enough, the two local solutions utilize only one of the two reactors wheras the global solution makes use of both reactors.

Example 21—Design of three-stage process system with recycle

Stephanopoulos and Westerberg (1975) (Fig. 10):

$$\begin{aligned} \min \quad & x_1^{0.6} + x_2^{0.6} + x_3^{0.4} - 4x_3 + 2x_4 + 5x_5 - x_6, \\ \text{s.t.} \quad & -3x_1 + x_2 - 3x_4 = 0, \\ & -2x_2 + x_3 - 2x_5 = 0, \\ & 4x_4 - x_6 = 0, \\ & x_1 + 2x_4 \leq 4, \\ & x_2 + x_5 \leq 4, \\ & x_3 + x_6 \leq 6, \\ & \mathbf{0} \leq \mathbf{x} \leq (3, 4, 4, 2, 2, 6). \end{aligned}$$

The global optimum is $\mathbf{x} = (0.166667, 2, 4, 0.5, 0, 2)$ with $f = -13.401904$. There is a local solution at $\mathbf{x} = (0, 0)$ with $f = -4.259$.

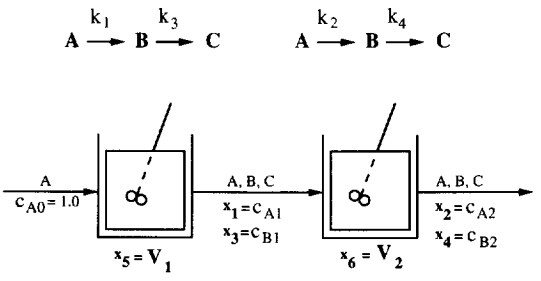

Fig. 9. Reactor network design.

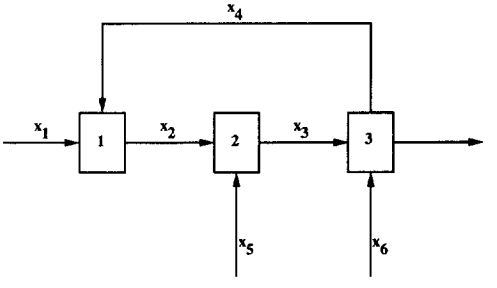

Fig. 10. Design of three-stage process system with recycle.