New Properties and Computational Improvement of the GOP Algorithm For Problems With Quadratic Objective Function and Constraints

V. Visweswaran and C.A. Floudas*
Department of Chemical Engineering
Princeton University
Princeton, N.J. 08544-5263

Abstract

In Floudas and Visweswaran (1990, 1992), a deterministic global optimization approach was proposed for solving certain classes of nonconvex optimization problems. An algorithm, **GOP**, was presented for the solution of the problem through a series of *primal* and *relaxed dual* problems that provide valid upper and lower bounds respectively on the global solution. The algorithm was proved to have finite convergence to an ϵ-global optimum. In this paper, new theoretical properties are presented that help to enhance the computational performance of the **GOP** algorithm applied to problems of special structure. The effect of the new properties is illustrated through application of the **GOP** algorithm to a difficult indefinite quadratic problem, a multiperiod tankage quality problem that occurs frequently in the modeling of refinery processes, and a set of pooling/blending problems from the literature. In addition, extensive computational experience is reported for randomly generated concave and indefinite quadratic programming problems of different sizes. The results show that the properties help to make the algorithm computationally efficient for fairly large problems.

Keywords : Global Optimization, Indefinite Quadratic Programming, Quadratic Constraints, Multiperiod Tankage Quality Problems.

*Author to whom all correspondence should be addressed.

1 Introduction

In recent years, global optimization of nonconvex constrained problems has received significant theoretical, algorithmic and computational attention. Surveys, books and applications for global optimization are available by Dixon and Szego (1975, 1978), Archetti and Schoen (1984), Pardalos and Rosen (1986, 1987), Törn and Zilinskas (1987), Ratschek and Rokne (1988), Hansen *et al* (1989a,b), Mockus (1989), Horst and Tuy (1990), and Floudas and Pardalos (1990, 1992).

The existing approaches for global optimization can be largely classified as deterministic or probabilistic approaches. The deterministic approaches include (a) Covering methods (e.g. Piyavskii, 1972), (b) Branch and bound methods (e.g. Al-Khayyal and Falk, 1983; Horst and Tuy, 1987; Pardalos *et al*, 1987; Al-Khayyal, 1990 ; Al-Khayyal *et al*, 1990; Hansen *et al*, 1990), (c) Cutting Plane Methods (e.g. Tuy, Thieu and Thai, 1985), (d) Interval methods (e.g. Hansen, 1979), (e) Trajectory methods (e.g. Branin, 1972), and (f) Penalty methods (e.g. Levy and Montalvo, 1985). Among probabilistic methods, the most important are (a) Random search methods (e.g. Kirkpatrick *et al*, 1983; Pinter, 1984), (b) Clustering methods (e.g. Rinnoy Kan and Timmer, 1987) and (c) Methods based on statistical models of objective functions (e.g. Zilinskas, 1986).

Floudas *et al* (1989) projected on the continuous variables so as to induce special structure in the primal and master subproblems, and used Generalized Benders' Decomposition (Geoffrion, 1972) in formulating a Global Optimum Search strategy. Though there was no guarantee of global optimality, the approach was found to be computationally quite effective for solving nonlinear and mixed-integer nonlinear programming problems. The approach was extended to bilinear, definite, indefinite and mixed-integer quadratic programming (Aggarwal and Floudas, 1990), and applied to the solution of Haverly's pooling problem (Floudas and Aggarwal, 1990).

Recently, Floudas and Visweswaran (1990, 1992) proposed a deterministic primal-relaxed dual global optimization approach for solving certain classes of nonconvex optimization problems. Making use of duality theory along with several new theoretical properties, a global optimization algorithm, **GOP**, was presented for the solution of the problem through a series of *primal* and *relaxed dual* problems that provide valid upper and lower bounds on the global solution. The algorithm was shown to attain finite ϵ-convergence and ϵ-global optimality.

In this paper, new theoretical properties are presented that enhance significantly the computational application of the **GOP** algorithm to several classes of problems that are of special structure : (a) concave quadratic problems with linear constraints, (b) indefinite quadratic problems, and (b) quadratically constrained problems. Section 2 contains a statement of the problem. A brief review of the **GOP** algorithm is provided in section 3. In section 4, new properties are presented that greatly improve the computational efficiency of the **GOP** algorithm. Section 5 addresses the application of the **GOP** algorithm to a difficult indefinite quadratic problem and illustrates the computational effect of the new properties. In section 6, the **GOP** algorithm with and without the new properties is applied to a multiperiod tankage quality problem that arises frequently in models of refineries. In section 7, the algorithm is applied to several pooling/blending problems from the literature. All three applications demonstrate that the proposed new properties improve the **GOP** algorithm by several orders of magnitude in computational performance. Finally, section 8 presents the results of applying the algorithm with the new properties to test problems available in the literature as well as randomly generated concave and indefinite quadratic programming problems.

2 Statement of the Problem

The global optimization problem addressed in this paper is stated as follows :

Determine an ϵ-globally optimal solution of the following problem:

$$\begin{array}{rrcl} & \min_{x,y} & f(x,y) & \\ \\ \textit{subject to} & g(x,y) & \le & 0 \\ & h(x,y) & = & 0 \\ & x & \in & X \\ & y & \in & Y \end{array} \tag{1}$$

where X and Y are non-empty, compact, convex sets, $g(x,y)$ is an m- vector of inequality constraints and $h(x,y)$ is a p-vector of equality constraints. It is assumed that the functions $f(x,y)$, $g(x,y)$ and $h(x,y)$ are continuous, piecewise differentiable and given in analytical form over $X \times Y$. The variables y and x are defined such that the following *Condition (A′)* is satisfied:

Condition (A') : *The functions $f(x,y)$, $g(x,y)$ and $h(x,y)$ are linear in x for every fixed y, and linear in y for every fixed x.*

Remark 1 : *Condition (A')* implies that the Lagrange function $L(x,y,\lambda^k,\mu^k)$, for fixed values of the multipliers $\lambda = \lambda^k$ and $\mu = \mu^k$, is linear in x for every fixed y, and linear in y for every fixed x.

Remark 2 : Through an appropriate use of *transformation* variables and *partitioning* of the resulting variable set (Floudas and Visweswaran, 1990), any problem that consists solely of quadratic terms in the objective function and/or constraints can be made to satisfy *Condition (A')*. Hence, the problems that are addressed by this paper include general quadratic programming problems and quadratic problems with quadratic terms in the constraints.

Remark 3 : *Condition (A')* is a relaxed form of the following *Conditions (A)* defined in Floudas and Visweswaran (1990) :

(a) $f(x,y)$ is convex in x for every fixed y, and convex in y for every fixed x.

(b) $g(x,y)$ is convex in x for every fixed y, and convex in y for every fixed x.

(c) $h(x,y)$ is affine in x for every fixed y, and affine in y for every fixed x.

Hence, the **GOP** algorithm can be directly applied to solve the classes of problems considered in this paper. However, the main objective of this paper is to exploit the special structure of *Condition (A')* so as to develop additional theoretical properties that can enhance the computational performance of the **GOP** algorithm significantly.

3 Review of the GOP Algorithm

The **GOP** algorithm can be used to solve nonconvex problems that satisfy *Conditions (A)* (Floudas and Visweswaran, 1990, 1992). The algorithm decomposes the original problem into primal and *relaxed dual* subproblems. The primal problem is solved by projecting on the y variables, and a feasible solution provides an upper bound on the global minimum and optimal multipliers for the various constraints. These multipliers are then used to formulate a Lagrange function that is used in the dual subproblem. Making use of several properties of the problem structure, the **GOP** algorithm solves the dual problem through a series of subproblems that, taken together, provide a lower bound on the global solution, and at the same time construct an underestimating function for the optimal solution. These *relaxed dual* problems are equivalent to setting a subset of the variables to a combination of their bounds, and solving for a corresponding region of the remaining variables. The algorithm obtains finite convergence to an ϵ-global solution through successive iteration between the primal and *relaxed dual* subproblems.

The major steps of the **GOP** algorithm are described below :

STEP 0 – *Initialization*

The first step is to identify the sources of nonconvexity in the problem formulation. Then, the variable set is divided into subsets of x and y variables that satisfy the necessary conditions. Once this is done, the set I_c of *connected* x variables is identified. (The Lagrange function at iteration k, for fixed values of its multipliers, can be written as $L(x, y, \lambda^k, \mu^k) = \Phi_1(y, \lambda^k, \mu^k)\Psi_1(x) + \Psi_2(x, \lambda^k, \mu^k) + \Phi_2(y, \lambda^k, \mu^k)$. The subset of variables x that participate in $\Psi_1(x)$ are called the *connected* variables). Proper bounds for all the variables are obtained, or some values are assumed. The set CB, representing the various combinations of bounds of these x variables, is defined. Next, certain storage sets and counters are defined and initialized. A counter K is set to 1, and sets K^{feas} and K^{infeas} are set to empty sets. The storage parameters $\mu_B^{stor}(K^{max}, B_j)$, $y^{stor}(K^{max}, B_j)$ and $y^k(K^{max}, B_j)$ are defined over the set of bounds CB and the maximum expected number of iterations K^{max}. Upper and lower bounds from the primal and *relaxed dual* problems, P^{UBD} and M^{LBD} are defined and suitably initialized. A convergence tolerance parameter ϵ is defined. Finally, a starting point y^1 is selected for the projected variables y.

STEP 1 – *Solution of a Primal problem:*

The value of y^K is stored. The following primal problem is solved :

$$\min_{x \in X} \ f(x, y^K)$$

$$\begin{aligned} \textit{subject to} \quad g(x, y^K) &\le 0 \\ h(x, y^K) &= 0 \end{aligned}$$

If the primal problem is feasible, the set K^{feas} is updated to contain K, and the optimal Lagrange multipliers λ^K and μ^K are stored. The upper bound provided by the primal problems is updated so that

$$P^{UBD} = MIN(P^{UBD}, P^K(y^K))$$

where $P^K(y^K)$ is the solution of the current (K th) primal problem.

If the primal problem is infeasible, the set K^{infeas} is updated to contain K, and the following relaxed primal subproblem is solved :

$$\min_{\substack{\alpha_i, \beta_i^+, \beta_i^- \geq 0 \\ x \in X}} \sum_i \alpha_i + \sum_i (\beta_i^+ + \beta_i^-)$$

$$\begin{aligned} \textit{subject to} \quad g(x, y^K) - \alpha &\leq 0 \\ h(x, y^K) + \beta^+ - \beta^- &= 0 \end{aligned}$$

Again, the values of the optimal Lagrange multipliers λ_1^K and μ_1^K are stored.

STEP 2- *Selection of Lagrange functions from the previous iterations:*

Before solving the *relaxed dual* problems, the set of Lagrange functions from previous iterations ($k = 1, 2 ... K - 1$) that can be present for the current *relaxed dual* problems is determined. This is done by evaluating a set of *qualifying* constraints for every such Lagrange function at y^K, the fixed value of the projected variables for the Kth (current) primal problem. The *qualifying* constraints are the gradients of the Lagrange function *with respect to* the *connected* x variables. If all the *qualifying* constraints of a particular Lagrange function are satisfied at y^K, then the Lagrangian and its accompanying qualifying constraints are selected to be constraints for the current *relaxed dual* problem.

STEP 3– *Relaxed Dual Problem:*

From the solution of the primal (or *relaxed* primal) problem of the current iteration, the Lagrange function is formulated. A form of this Lagrange function is added to every *relaxed dual* problem solved in the current iteration. These problems are solved as follows :

(a) A combination of the bounds B_l, of the *connected* variables in x, say $B_l = B_1$, is selected.

(b) The following *relaxed dual* problem is solved :

$$\min_{y \in Y, \mu_B} \mu_B$$

subject to

$$\left.\begin{array}{l}\mu_B \geq \; L(x^{B_j}, y, \lambda^k, \mu^k)|_{x^k}^{lin} \\ \qquad \nabla_{x_i} L(x, y, \lambda^k, \mu^k)|_{x^k} \leq 0 \quad if\ x_i^{B_j} = x_i^U \\ \qquad \nabla_{x_i} L(x, y, \lambda^k, \mu^k)|_{x^k} \geq 0 \quad if\ x_i^{B_j} = x_i^L \end{array}\right\} \begin{array}{c} \forall j \in UL(k, K) \\ k = 1, 2 ... K-1 \\ k \in K^{feas} \end{array}$$

$$\left.\begin{array}{l}0 \geq \; L_1(x^{B_j}, y, \lambda_1^k, \mu_1^k)|_{x^k}^{lin} \\ \qquad \nabla_{x_i} L_1(x, y, \lambda_1^k, \mu_1^k)|_{x^k} \leq 0 \quad if\ x_i^{B_j} = x_i^U \\ \qquad \nabla_{x_i} L_1(x, y, \lambda_1^k, \mu_1^k)|_{x^k} \geq 0 \quad if\ x_i^{B_j} = x_i^L \end{array}\right\} \begin{array}{c} \forall j \in UL(k, K) \\ k = 1, 2 ... K-1 \\ k \in K^{infeas} \end{array}$$

$$\left.\begin{array}{l}\mu_B \geq \; L(x^{B_l}, y, \lambda^K, \mu^K)|_{x^k}^{lin} \\ \qquad \nabla_{x_i} L(x, y, \lambda^K, \mu^K)|_{x^K} \leq 0 \quad if\ x_i^{B_l} = x_i^U \\ \qquad \nabla_{x_i} L(x, y, \lambda^K, \mu^K)|_{x^K} \geq 0 \quad if\ x_i^{B_l} = x_i^L \end{array}\right\} \quad if\ K \in K^{feas}$$

$$\left.\begin{array}{l}0 \geq \; L_1(x^{B_l}, y, \lambda_1^K, \mu_1^K)|_{x^k}^{lin} \\ \qquad \nabla_{x_i} L_1(x, y, \lambda_1^K, \mu_1^K)|_{x^K} \leq 0 \quad if\ x_i^{B_l} = x_i^U \\ \qquad \nabla_{x_i} L_1(x, y, \lambda_1^K, \mu_1^K)|_{x^K} \geq 0 \quad if\ x_i^{B_l} = x_i^L \end{array}\right\} \quad if\ K \in K^{infeas}$$

Here, the first two sets of constraints involve Lagrange functions generated from feasible primal and infeasible primal problems respectively during the previous iterations. Also included are the *qualifying* constraints for the Lagrange functions. The next two sets of constraints correspond to the Lagrange function (and its *qualifying* constraints) formulated from the primal problem of the current iteration and evaluated at a bound of $x = x^{B_l}$.

The solution of this problem, if feasible, is stored in $\mu_B^{stor}(K, B_l)$ and $y^{stor}(K, B_l)$.

(c) A new combination of bounds for x, say $B_l = B_2$, is selected.

(d) Steps (b) and (c) are repeated until the *relaxed dual* problem has been solved at each set of bounds of the *connected* variables in x, that is, for every $B_l \in CB$.

STEP 4- *Selecting a new lower bound and y^{K+1}:*

After the *relaxed dual* problem has been solved for every possible combination of the bounds x^{B_l}, a new lower bound is selected from the stored values of μ_B. This corresponds to the lowest value of the stored solutions. At the same time, the corresponding stored value of y is selected to be the fixed value of the projected variables for the next $(K + 1)$ iteration.

Once selected, the stored values are deleted from the stored sets. This ensures that the *relaxed dual* problem does not return the same value of y and μ_B during successive problems, except during the final convergence stage of the algorithm, when the solutions of the *relaxed dual* problems at successive iterations can be (and usually are) very close to each other.

STEP 5- *Check for convergence:*

IF convergence is satisfied, STOP. Else, set $K = K + 1$ and return to step 1. Finally, the check for convergence is performed. If the lower bound from the *relaxed dual* problem is within ϵ of the upper bound from the primal problems, that is, if $M^{LBD} > P^{UBD} - \epsilon$, (alternately, if both P^{UBD} and M^{LBD} are large numbers, then $(P^{UBD} - M^{LBD}) < M^{LBD}\epsilon$ can be used) then the solution corresponding to the upper bound from the primal problems is an ϵ-optimal solution to the original problem, and the algorithm is stopped. Else, $K = K+1$ and the algorithm returns to step 1. The algorithm continues with Steps 1-5 until convergence is reached.

4 New Properties for the GOP algorithm

In its original form, the **GOP** algorithm requires the solution of a *relaxed dual* problem corresponding to every combination of bounds of the *connected* x variables. This may lead to the solution of a large number of *relaxed dual* subproblems at every iteration. It has been observed, however, that a number of the subproblems are either infeasible or have identical solutions. This led to further investigation of the structure of the subproblems, and consequently, the form of the Lagrange function that is formulated from the corresponding *primal* problems and used in the solution of the *relaxed dual* subproblems. In this section, new properties of the Lagrange function are discussed. These properties help to greatly improve the computational efficiency of the **GOP** algorithm.

4.1 Linearity of the Lagrange Function

The problems that are addressed in this paper satisfy *Condition* (A') . From Remark 1, the Lagrange function that is generated from the primal problem is linear in x for all fixed y, and linear in y for all fixed x. Therefore, the linearization of the Lagrange function *w.r.t.* x is the same as the Lagrange function (that is, $L(x, y, \lambda^k, \mu^k)|_{x^k}^{lin} = L(x, y, \lambda^k, \mu^k)$). This leads to the following observation :

$$\textit{For every } i, \quad \nabla_{x_i} L(x,y,\lambda^k,\mu^k) \;\; \textit{is not a function of } x.$$

Therefore,

$$\nabla_{x_i} L(x,y,\lambda^k,\mu^k)|_{x^k}^{lin} = \nabla_{x_i} L(x,y,\lambda^k,\mu^k)$$

Based on this observation, the following properties are now stated.

Property 1 : At the kth iteration of the **GOP** algorithm, let $g(y) = \nabla_{x_j} L(x,y,\lambda^k,\mu^k)$ for some j . Then, if $g(y) \geq 0$ for all y (respectively $g(y) \leq 0$ for all y), it is not necessary to solve those *relaxed dual* problems that have $g(y) \leq 0$ (respectively $g(y) \geq 0$) as a *qualifying* constraint.

Proof : The Lagrange function formulated from the kth primal problem has the following form :

$$\begin{aligned} L(x,y,\lambda^k,\mu^k) &= L(x^k,y,\lambda^k,\mu^k) + \nabla_x L(x,y,\lambda^k,\mu^k).(x-x^k) \\ &= L(x^k,y,\lambda^k,\mu^k) + \sum_{i\neq j} \nabla_{x_i} L(x,y,\lambda^k,\mu^k).(x_i - x_i^k) + g(y).(x_j - x_j^k) \end{aligned}$$

Consider the two *Relaxed Dual* **(RD)** problems that are solved at the kth iteration for two combinations of bounds of the *connected* x variables, which differ only in the bound of x_j. That is, consider two combinations B_1 and B_2 such that

$$\begin{aligned} B_1 &\equiv \{x_1^B, x_2^B, \ldots x_{j-1}^B, x_j^L, x_{j+1}^B \ldots x_n^B\} \\ B_2 &\equiv \{x_1^B, x_2^B, \ldots x_{j-1}^B, x_j^U, x_{j+1}^B \ldots x_n^B\} \end{aligned}$$

The two corresponding *relaxed dual* problems are shown below :

(1) For $x^B = x^{B_1}$: The **RD** problem is

$$\begin{aligned} &\min_{y,\mu_B} \;\; \mu_B \\ s.t.& \\ \mu_B &\geq L(x^k,y,\lambda^k,\mu^k) + \sum_{i\neq j} \nabla_{x_i} L(x,y,\lambda^k,\mu^k).(x_i^B - x_i^k) + g(y).(x_j^L - x_j^k) \qquad (2) \\ & g(y) \geq 0 \\ & F(\mu_B, y) \leq 0 \end{aligned}$$

(2) For $x^B = x^{B_2}$: The **RD** problem is

$$\begin{aligned} & \min_{y,\mu_B} \quad \mu_B \\ s.t. & \\ \mu_B \ \geq \ & L(x^k, y, \lambda^k, \mu^k) + \sum_{i \neq j} \nabla_{x_i} L(x, y, \lambda^k, \mu^k).(x_i^B - x_i^k) + g(y).(x_j^U - x_j^k) \\ & g(y) \leq 0 \\ & F(\mu_B, y) \leq 0 \end{aligned} \tag{3}$$

where the last constraint ($F(\mu_B, y) \leq 0$) has been added to represent all other constraints (such as Lagrange functions and their *qualifying* constraints from previous iterations, constraints from the original problem etc.) that are common to both the *Relaxed Dual* problems.

Consider the *qualifying* constraint corresponding to the variable x_j for the two problems. The function $g(y)$ is linear in y. Hence, assuming that the bounds on the y variables are known, it is possible to determine whether $g(y)$ is at one of its bounds (lower or upper). Suppose that such a check reveals that $g(y) \geq 0$ is always true. Then, this can be added as a constraint to both problems (2) and (3). As can be seen, problem (2) is not affected. However, problem (3) becomes

$$\begin{aligned} & \min_{y,\mu_B} \quad \mu_B \\ s.t. & \\ \mu_B \ \geq \ & L(x^k, y, \lambda^k, \mu^k) + \sum_{i \neq j} \nabla_{x_i} L(x, y, \lambda^k, \mu^k).(x_i^B - x_i^k) + g(y).(x_j^U - x_j^k) \\ & g(y) \leq 0 \\ & F(\mu_B, y) \leq 0 \\ & g(y) \geq 0 \end{aligned} \tag{4}$$

From this, it can be seen that for problem (3), the addition of the new constraint is equivalent to setting $g(y) = 0$. Hence, the first constraint in problem (4) becomes

$$\begin{aligned} \mu_B \ & \geq \ L(x^k, y, \lambda^k, \mu^k) + \sum_{i \neq j} \nabla_{x_i} L(x, y, \lambda^k, \mu^k).(x_i^B - x_i^k) + g(y).(x_j^U - x_j^k) \\ & \geq \ L(x^k, y, \lambda^k, \mu^k) + \sum_{i \neq j} \nabla_{x_i} L(x, y, \lambda^k, \mu^k).(x_i^B - x_i^k) \end{aligned}$$

Hence, problem (3) can be written as

$$\begin{aligned} &\min_{y,\mu_B} \ \mu_B \\ s.t.& \\ \mu_B \ \geq \ & L(x^k, y, \lambda^k, \mu^k) + \sum_{i \neq j} \nabla_{x_i} L(x, y, \lambda^k, \mu^k).(x_i^B - x_i^k) \\ & g(y) = 0 \\ & F(\mu_B, y) \leq 0 \end{aligned} \tag{5}$$

At the same time, problem (2) can be written as

$$\begin{aligned} &\min_{y,\mu_B} \ \mu_B \\ s.t.& \\ \mu_B \ \geq \ & L(x^k, y, \lambda^k, \mu^k) + \sum_{i \neq j} \nabla_{x_i} L(x, y, \lambda^k, \mu^k).(x_i^B - x_i^k) + g(y).(x_j^L - x_j^k) \\ & g(y) \geq 0 \\ & F(\mu_B, y) \leq 0 \end{aligned} \tag{6}$$

Since $g(y) \geq 0$ and $x_j^L \leq x_j^k$, the solution of problem (6) is always less than or equal to the solution of problem (5). Also, the feasible region of problem (6) contains the feasible region of problem (5). Hence, it is not necessary to solve the problem (5) (since, according to the **GOP** algorithm, we select the minimum of the solutions of the *relaxed dual* problems solved at all the combinations of bounds of the *connected* x variables). Moreover, since the set of bounds for the $x_i, i \in \{1, 2..n\} \backslash \{j\}$ was chosen arbitrarily, it is not necessary to solve the relaxed dual problem for any combination of bounds of the x variables for which $g(y) \leq 0$ is a *qualifying* constraint. Similarly, the opposite case (when $g(y)$ is always negative) can be proved.

Property 2 : If $\nabla_{x_j} L(x, y, \lambda^k, \mu^k) = 0$ for all y, then, it is sufficient to solve for only those combinations of bounds for which x_j is at either its lower or upper bound.

Proof : Since $g(y) = 0$, the Lagrange functions formulated from the kth iteration and used in the two *relaxed dual* problems (2) and (3) are identical. Also, the *qualifying* constraint w.r.t x_j for the two problems are the same, being of the form $g(y) = 0$. Hence, problems (2) and (3) have the same solution. Therefore, it is sufficient to solve either problem (2) or problem (3). Since this holds regardless of the set of bounds for the $x_i, i \in 1, 2..n\backslash\{j\}$, it is sufficient to solve for only those combinations of bounds that have x_j set to either its upper or its lower bound.

Property 3 : Suppose that there is a one-to-one correspondence between the x and y variable set in the feasible region of the problem, and that $y_i = x_i$. Then, the bounds on the *connected* x variables can be updated without destroying the property of ϵ-global optimality of the **GOP** algorithm.

Proof : By the assumption of one-to-one correspondence, there is a y_i for every x_i. Moreover, it is assumed that $y_i = x_i$. Then, consider the Kth iteration of the **GOP** algorithm. The fixed value of y for the primal problem is y^K. Then, before solving the relaxed dual problems for the current (Kth) iteration, the Lagrange functions from previous iterations are selected to be present in the *relaxed dual* problems for the current (Kth) iteration. Since these Lagrange functions are selected on the basis of their *qualifying* constraints being satisfied at y^K, there will be one Lagrange function from every iteration present in the *relaxed dual* problems of the Kth iteration. Along with this Lagrange function, there will be a *qualifying* constraint of the form $g(y) \geq 0$ or $g(y) \leq 0$.

Let $y^1, y^2 ... y^{K-1}$ represent the fixed values of y for all iterations prior to the Kth iteration. Then, for $y_i, i = 1, 2...n$, it is possible to determine the lower and upper bounds, that is, for every i,

$$y_i^L = MIN(y_i^1, y_i^2 ... y_i^{K-1}) \quad \text{and} y_i^U = MAX(y_i^1, y_i^2 ... y_i^{K-1})$$

can be determined. Then, it follows that $y_i^K \in [y_i^L, y_i^U] \quad \forall\, i$.

Now, the Lagrange functions that are added from the previous iterations have their *qualifying* constraints satisfied at y^K. Since these constraints are linear in y, and are satisfied identically as an equality at the respective y^k, it follows that they are satisfied for all y lying between y^k and y^K. Moreover, the presence of these *qualifying* constraints in the *relaxed dual* problems ensures that the values of y_i are restricted to lie between y_i^L and y_i^U. Since there is a direct correspondence between the values of y and x, this means that for every i, the bounds of x_i can be updated to (x_i^L, x_i^U).

4.2 Iterations with Infeasible Primal Problems

Often, the values of y^{k+1} that are returned by step 4 of the **GOP** algorithm after solving the *relaxed dual* problems and selecting the minimum solution are such that the primal problem is infeasible. Suppose that this happens at the kth iteration, that is, the primal problem, when solved for $y = y^k$, does not have any feasible solution. Then, it is necessary to solve a *relaxed primal* problem for this fixed value of $y = y^k$. Because of the way in which the *relaxed primal* problem is solved, the Lagrange functions formulated from this problem (having the form $L_1(x, y, \lambda^k, \mu^k) \leq 0$) are used mainly as constraints defining the feasible region for the *relaxed dual* problems in terms of y. At the same time, if there are other stored solutions remaining (from the solutions of the *relaxed dual* problems of previous iterations), it is likely that some of these solutions provide values of y that are feasible for the primal problem. Therefore, for iterations when the primal problem is infeasible (such as the kth iteration), the solution of the *relaxed primal* problem is used to generate constraints to be used for future *relaxed dual* problems, but to not actually solve the *relaxed dual* problems for the current (kth) iteration. This strategy can be applied in general as long as there are stored solutions remaining at the kth iteration, since it is always possible to backtrack to the kth iteration if no feasible solutions can be found by future iterations.

5 Application 1 : Indefinite Quadratic Problem

This example is a large indefinite quadratic programming problem. It is taken from Floudas and Pardalos (1990).

$$\begin{aligned} & \min_{x,y} \quad \Phi(x,y) = \Phi_1(x) + \Phi_2(y) \\ s.t. \quad & A_1 x + A_2 y \leq b \\ x_i \geq\ & 0, \quad i = 1, 2 \ldots 10 \\ y_i \geq\ & 0, \quad i = 11, 12..20 \end{aligned}$$

where

$$\begin{aligned} \Phi_1(x) &= \frac{\theta_1}{2} \sum_{i=1}^{10} C_i (x_i - \overline{x_1})^2 \quad \theta_1 < 0,\ C_i \geq 0,\ i = 1, 2 \ldots 10 \\ \Phi_2(y) &= \frac{\theta_2}{2} \sum_{i=11}^{20} C_i (y_i - \overline{y_i})^2 \quad \theta_2 > 0,\ C_i \geq 0,\ i = 11, 12, \ldots 20 \end{aligned}$$

where $\theta_1 = -1$, $\theta_2 = 1$, and C_1, C_2, $\overline{x}$ and $\overline{y}$ are constant vectors. Hence, the function $\Phi_1(x)$ is concave, while $\Phi_2(y)$ is convex. The data for this problem is given in Table 1.

New transformation variables y_1 through y_{10} and constraints are introduced in the following manner:

$$y_1 = x_1,\ y_2 = x_2, ..., y_{10} = x_{10}$$

so that

$$\Phi_1(x, y) = \frac{\theta_1}{2} \sum_{i=1}^{10} C_i(x_i - \overline{x_i})(y_i - \overline{y_i})$$

Then, *projecting* on $y_i,\ i = 1, 2, ..., 20$ leads to the primal problem being essentially a function evaluation at $x = y^k$. This implies that the original constraints can be ignored for the primal problem. Hence, the primal problem at the kth iteration can be written as shown below :

$$\begin{aligned} &\min_{x_1, x_2 .. x_{10}} \quad \frac{\theta_1}{2} \sum_{i=1}^{10} C_i(x_i - \overline{x_i})(y_i^k - \overline{y_i}) + \frac{\theta_2}{2} \sum_{i=11}^{20} C_i(y_i^k - \overline{y_i})^2 \\ &s.t \quad x_i - y_i^k = 0 \quad i = 1, 2 ... 10 \end{aligned}$$

The constraints from the original problem are used in the *relaxed dual* problem, since they can be written with $y_i, i = 1, 2, ..., 20$ as the variables (Since y_i is equivalent to x_i for i = 1,2,...,10).

From the kth primal problem, the Lagrange function can be formulated as

$$\begin{aligned} L(x, y, \lambda^k) \ = \ & \frac{\theta_1}{2} \sum_{i=1}^{10} C_i(x_i - \overline{x_i})(y_i - \overline{y_i}) + \frac{\theta_2}{2} \sum_{i=11}^{20} C_i(y_i - \overline{y_i})^2 \\ & + \sum_{i=1}^{10} \lambda_i^k (x_i - y_i) \end{aligned}$$

The KKT gradient conditions for the kth primal problem provide :

$$\nabla_{x_i} L(x, y^k, \lambda^k) = \frac{\theta_1}{2} . C_i . (y_i^k - \overline{y_i}) + \lambda_i^k = 0$$

Use of the KKT gradient conditions leads to the following form of the Lagrange function :

$$\begin{aligned} L(x, y, \lambda^k) \ = \ & \frac{\theta_1}{2} \sum_{i=1}^{10} C_i(x_i - \overline{x_i})(y_i - y_i^k) + \frac{\theta_2}{2} \sum_{i=11}^{20} C_i(y_i - \overline{y_i})^2 \\ & - \sum_{i=1}^{10} \lambda_i^k y_i \end{aligned}$$

$$
\begin{aligned}
C &= (63, 15, 44, 91, 45, 50, 89, 58, 86, 82, \\
&\quad 42, 98, 48, 91, 11, 63, 61, 61, 38, 26) \\
\overline{x} &= (-19, -27, -23, -53, -42, 26, -33, -23, 41, 19) \\
\overline{y} &= (-52, -3, 81, 30, -85, 68, 27, -81, 97, -73) \\
b &= (380, 415, 385, 405, 470, 415, 400, 460, 400, 200)
\end{aligned}
$$

$$
A_1 = \begin{bmatrix}
3 & 5 & 5 & 6 & 4 & 4 & 5 & 6 & 4 & 4 \\
5 & 4 & 5 & 4 & 1 & 4 & 4 & 2 & 5 & 2 \\
1 & 5 & 2 & 4 & 7 & 3 & 1 & 5 & 7 & 6 \\
3 & 2 & 6 & 3 & 2 & 1 & 6 & 1 & 7 & 3 \\
6 & 6 & 6 & 4 & 5 & 2 & 2 & 4 & 3 & 2 \\
5 & 5 & 2 & 1 & 3 & 5 & 5 & 7 & 4 & 3 \\
3 & 6 & 6 & 3 & 1 & 6 & 1 & 6 & 7 & 1 \\
1 & 2 & 1 & 7 & 8 & 7 & 6 & 5 & 8 & 7 \\
8 & 5 & 2 & 5 & 3 & 8 & 1 & 3 & 3 & 5 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{bmatrix}
\qquad
A_2 = \begin{bmatrix}
8 & 2 & 4 & 1 & 1 & 1 & 2 & 1 & 7 & 3 \\
3 & 6 & 1 & 7 & 7 & 5 & 8 & 7 & 2 & 1 \\
1 & 7 & 2 & 4 & 7 & 5 & 3 & 4 & 1 & 2 \\
7 & 7 & 8 & 2 & 3 & 4 & 5 & 8 & 1 & 2 \\
7 & 5 & 3 & 6 & 7 & 5 & 8 & 4 & 6 & 3 \\
4 & 1 & 7 & 3 & 8 & 3 & 1 & 6 & 2 & 8 \\
4 & 3 & 1 & 4 & 3 & 6 & 4 & 6 & 5 & 4 \\
2 & 3 & 5 & 5 & 4 & 5 & 4 & 2 & 2 & 8 \\
4 & 5 & 5 & 6 & 1 & 7 & 1 & 2 & 2 & 4 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{bmatrix}
$$

The global solution of this problem occurs at

$$
\begin{aligned}
x^\star &= (0, 0, 0, 62.609, 0, 0, 0, 0, 0, 0) \\
y^\star &= (0, 0, 0, 0, 0, 4.348, 0, 0, 0, 0)
\end{aligned}
$$

Table 1: Data for the Indefinite Quadratic Programming Problem

Hence, since C is a positive vector, the *qualifying* constraints can be written as

$$\begin{aligned} \theta_1(y_i - y_i^k) &\geq 0 \quad \textit{if } x_i^B = x_i^L, \quad \textit{and} \\ \theta_1(y_i - y_i^k) &\leq 0 \quad \textit{if } x_i^B = x_i^U \\ &\qquad\qquad \textit{for} \quad i = 1, 2...10 \end{aligned}$$

The *relaxed dual* problem will also contain the original constraints for the problem, with the x_i, $i = 1, 2, ..., 10$ replaced by the corresponding y_i. This ensures that the relaxed dual problem never returns infeasible values of y_i for the next iteration.

5.1 The GOP Algorithm Without the New Properties

When the **(GOP)** algorithm was applied in its original form to solve this problem the algorithm took 3-4 iterations to converge to a the global solution. The algorithm needed to solve one primal, and 1024 *relaxed dual* subproblems (2^{10}) at every iteration. From a starting point of 0 for all the y variables, the algorithm needed 4 iterations to converge, solving a total of 4096 *relaxed dual* problems.

5.2 The GOP Algorithm With Property 1

Consider the starting point of $y = 0$ for the **GOP** algorithm. The primal problem, for this fixed value of y^1, is given below :

$$\min_{x_1, x_2 .. x_{10}} \quad -\frac{\theta_1}{2} \sum_{i=1}^{10} C_i (x_i - \overline{x_i}) \overline{y_i} + \frac{\theta_2}{2} \sum_{i=11}^{20} C_i \overline{y_i}^2$$

$$\textit{subject to} \quad x_i = 0 \quad i = 1, 2...10$$

The solution of this problem takes place at $x = 0$, and the value of the objective function is 547663.5 . The optimal multipliers for the constraints are

$$\lambda = \{598.5, 202.5, 506, 2411.5, 945, -650, 1468.5, 667, -1763, -779\}$$

Hence, the Lagrange function formulated from the first primal problem has the form

$$L(x, y, \lambda^1) = -0.5 \sum_{i=1}^{10} C_i (x_i - \overline{x_i}) y_i + 0.5 \sum_{i=11}^{20} C_i (y_i - \overline{y_i})^2 - \sum_{i=1}^{10} \lambda_i^k y_i$$

From this, it can be seen that all the *qualifying* constraints are of the form $y_i \geq 0$ or $y_i \leq 0$. Now, the lower bound for all the y variables is 0. Hence, instead of solving 1024 *relaxed dual* problems, it is sufficient to solve only the *relaxed dual* problem for which *qualifying* constraints are all of the form $y_i \geq 0$. This corresponds to the combination of bounds $x^B = \{x_i^U\}, \quad i = 1, 2...10$. Setting the x variables to their upper bounds and solving the resulting convex problem in y, the solution obtained is

$$y = \{5.936, 0, 0, 58.9954, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8.219, 0, 0, 0, 0\}$$

with a value of $\mu_B = 43855.57$. These are the fixed value of y for the second iteration and the lower bound on the global optimum respectively.

At the second iteration, the primal problem has an objective value of 63469.1141, and the following set of multipliers is obtained :

$$\lambda = \{785.4, 202.5, 506, 5095.8, 945, -650, 1468.5, 667, -1763, -779\}$$

From these values, the Lagrange function for the second iteration can be written as

$$\begin{aligned} L(x, y, \lambda^2) &= -0.5 \sum_{i \neq 1 \ or \ 4} C_i(x_i - \overline{x_i})y_i - 0.5C_1(x_1 - \overline{x_1})(y_1 - 5.936) \\ &\quad -0.5C_4(x_4 - \overline{x_4})(y_4 - 58.995) + 0.5 \sum_{i=11}^{20} C_i(y_i - \overline{y_i})^2 - \sum_{i=1}^{10} \lambda_i^k y_i \end{aligned}$$

Thus, for this iteration, all but two of the x variables (x_1 and x_4) have the *qualifying* constraint $y_i \geq 0$ or $y_i \leq 0$ associated with them. Hence, since the lower bound of all the y variables is 0, these eight x variables can be set to their upper bounds, which corresponds to the *qualifying* constraints $y_i \geq 0, \quad i = \{1, 2...10\} \backslash \{1, 4\}$. Therefore, it is necessary to solve only for the four combinations of bounds of the variables x_1 and x_4, with all other x_i at their upper bounds. Thus, only four *relaxed dual* problems need to be solved at the second iteration. Of these four problems, three provide solutions with a value of μ_B that is greater than the global solution. The fourth *relaxed dual* problem, corresponding to $x_1 = x_1^L$ and $x_4 = x_4^U$ gives a solution of $\mu_B = 49188.9$, with

$$y = \{0, 0, 0, 62.609, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4.348, 0, 0, 0, 0, 0\}$$

This is the global solution for the problem. At the third iteration, the **GOP** algorithm converges to this solution, needing to solve one primal and two *relaxed dual* problems at the third iteration.

Thus, by making use of Property 1, the **GOP** algorithm needs to solve only 7 *relaxed dual* problems, as opposed to the 4096 problems solved originally. Thus, there is an improvement in the performance of the algorithm by almost three orders of magnitude.

6 Application 2 : A Multiperiod Tankage Quality Problem

Chemical process design and control involves optimization of models that are nonconvex in nature. Often, these nonconvexities occur in the form of bilinear constraints in the plant model. Examples of this can be found in the design of heat exchanger networks, pooling and blending units,and distillation sequences. In all these cases, the bilinearities occur due to the necessity of introducing the flow rate (stock) and composition (quality), associated with some or all streams (products), as variables in the model. Hence, optimization of such models need to be treated as global optimization problems.

The following example concerns the application of the **(GOP)** algorithm to a multiperiod tankage quality problem that arises often in the operations of refineries. The following sets are defined for the mathematical formulation of the problem :

$$\begin{aligned} PR &= \{p\} \ : \ \text{set of products} \\ CO &= \{c\} \ : \ \text{set of components} \\ T &= \{t\} \ : \ \text{set of time periods} \\ QL &= \{l\} \ : \ \text{set of qualities} \end{aligned}$$

For this problem, there are 3 products $(p1, p2, p3)$, 2 components $(c1, c2)$, and 3 time periods $(t0, t1, t2$ - where $t0$ is the time period corresponding to the starting point). The following variables are defined :

$$\begin{aligned} x_{c,p,t} &: \text{ amount of component } c \text{ allocated to product } p \text{ at period } t \\ s_{p,t} &: \text{ stock of product } p \text{ at end ofperiod } t \\ q_{p,l,t} &: \text{ quality } l \text{ of product } p \text{ at period } t \end{aligned}$$

The objective of the problem is to maximize the total value at the end of the last time period. The terminal value of each product (val_p) is given. Lower and upper bounds on the

$$PR \equiv \{p1, p2, p3\} \qquad CO \equiv \{c1, c2\}$$
$$T \equiv \{t0, t1, t2\} \qquad QL \equiv \{q1, q2\}$$

Terminal Value of products : $val_p = (60, 90, 40)$.

Arrivals of components $AR_{c,t}$

Component	*Time Period*		
	$t0$	$t1$	$t2$
$c1$	.200	.250	.150
$c2$	.200	.150	.250

Product Lifting $LF_{p,t}$

Product	*Time Period*	
	$t1$	$t2$
$p1$	.080	.120
$p2$	.150	.100
$p3$	.150	.200

Lower Bounds on Product Quality $q^L_{p,l}$

Products	*Qualities*	
	$q1$	$q2$
$p1$	70	50
$p2$	80	70
$p3$	60	40

Upper Bounds on Product Quality $q^U_{p,l}$

Products	*Qualities*	
	$q1$	$q2$
$p1$	100	100
$p2$	100	100
$p3$	100	100

Qualities in Components $QU_{c,l}$

Components	*Qualities*	
	$q1$	$q2$
$c1$	40	80
$c2$	100	50

Initial Quality Values $qinit_{c,l}$ at time $t0$

Components	*Qualities*	
	$q1$	$q2$
$p1$	70	50
$p2$	90	70
$p3$	60	40

Limits on Product Stocks ($stock_{p,t}$)

Product	*Time Period*		
	$t0$	$t1$	$t2$
$p1$	.050	.100	.100
$p2$	.050	.100	.100
$p3$	.050	.100	.100

Table 2: Data for the Multiperiod Tankage Quality Problem

quality variables are provided, as well as initial quality values. Limits on product stocks ($stock_{p,t}$) for each time period are also provided, along with the qualities in each component ($QU_{c,l}$) and the product lifting ($LF_{p,t}$) for every period. The data for this problem is provided in Table 2.

The complete mathematical formulation for this problem, consisting of 39 variables and 22 inequality constraints (of which 12 are nonconvex), is given below :

$$\max \quad \sum_{p \in PR} val_p . s_{p,'t2'}$$

subject to

$$\begin{aligned}
\sum_{p \in PR} x_{c,p,t} &\leq AR_{c,t} && t \in \{t1, t2\},\ c \in CO \\
s_{p,t} + \sum_{c \in CO} x_{c,p,t+1} - s_{p,t+1} &\geq LF_{p,t+1} && t \in \{t0, t1\},\ p \in PR \\
s_{p,t} . q_{p,l,t} + \sum_{c \in CO} x_{c,p,t+1} . QU_{c,l} &\geq \\
(s_{p,t+1} + LF_{p,t+1}) . q_{p,l,t+1} & && t \in \{t0, t1\},\ p \in PR,\ l \in QL
\end{aligned}$$

The sources of nonconvexities in this problem are the bilinear terms $s_{p,t} . q_{p,l,t}$ in the last set of constraints. Thus, fixing either the set of s or q variables makes the problem linear in the remaining variables.

This problem was first studied through the application of a Global Optimum Search (GOS) approach (Floudas *et al.*, 1989 ; Floudas and Aggarwal, 1990). The GOS was fairly successful in identifying the global minimum for the problem, but from some of the starting points, it converged to either a local solution, or a non-local, non-global solution, even though the special feature of "Radial Search" was applied.

In order to reduce the problem to a form where the **GOP** algorithm can be applied, it is necessary to identify the set of x and y variables. The sources of nonconvexities in this problem are the bilinear terms $s_{p,t} . q_{p,l,t}$ in the last set of constraints. Thus, fixing either the set of s or q variables makes the problem linear in the remaining variables. Here, the q variables are chosen to be the y variables, that is, they are fixed for the primal problem. Then, the variables s form the set of *connected* x variables.

For a fixed $q = q^k$, the primal problem is given by :

$$\min \quad \sum_{p \in PR} -val_p . s_{p,'t2'}$$

subject to

$$\sum_{p \in PR} x_{c,p,t} \leq AR_{c,t} \qquad t \in \{t1, t2\},\ c \in CO \tag{7}$$

$$-s_{p,t} - \sum_{c \in CO} x_{c,p,t+1} + s_{p,t+1} \leq -LF_{p,t+1} \qquad t \in \{t0, t1\},\ p \in PR \tag{8}$$

$$\begin{aligned} -s_{p,t} . q^k_{p,l,t} - \sum_{c \in CO} x_{c,p,t+1} . QU_{c,l} &\leq \\ -(s_{p,t+1} + LF_{p,t+1}) . q^k_{p,l,t+1} & \qquad t \in \{t0, t1\},\ p \in PR,\ l \in QL \end{aligned} \tag{9}$$

$$0 - s_{p,t} \leq 0 \qquad p \in PR, t \in \{t0, t1\} \tag{10}$$

$$s_{p,t} - stock_{p,t} \leq 0 \qquad p \in PR, t \in \{t0, t1\} \tag{11}$$

where the bounds on the stocks s have been explicitly incorporated in the problem. The problem has been written as a minimization problem by multiplying the objective function by -1. It should be noted that $s(p,' t0')$, the stock of product p at the beginning of the first period, is fixed.

The KKT gradient conditions for the x variables are given as

$$\nabla_{x_{c,p,t}} L(x, s, q^k, \mu^k) = \mu^k_{1_{c,t}} - \mu_{2_{p,t-1}} - \sum_{l \in QL} \mu_{3_{p,l,t-1}} . QU_{c,l} = 0$$

The KKT gradient conditions for the *connected* variables s in this problem are given by

(i) For $t =' t1'$, $p \in PR$,

$$\nabla_{s_{p,t}} L(x, s, q^k, \mu^k) = \mu^k_{2_{p,t-1}} - \mu^k_{2_{p,t}} + \sum_{l \in QL} (\mu_{3_{p,l,t-1}} - \mu_{3_{p,l,t}}) . q^k_{p,l,t} - \mu^k_{L_{p,t}} + \mu^k_{U_{p,t}} = 0$$

(ii) For $t =' t2'$, $p \in PR$,

$$\nabla_{s_{p,t}} L(x, s, q^k, \mu^k) = -val_p + \mu^k_{2_{p,t-1}} - \mu^k_{2_{p,t}} + \sum_{l \in QL} (\mu_{3_{p,l,t-1}} - \mu_{3_{p,l,t}}) . q^k_{p,l,t} - \mu^k_{4_{p,t}} + \mu^k_{5_{p,t}} = 0$$

where $\mu_1^k, \mu_2^k, \mu_3^k, \mu_4^k$ and μ_5^k correspond to the constraint sets (7) through (11) .

The Lagrange function formulated from the kth primal problem is given by

$$\begin{aligned} L(x,s,q,\mu^k) \; = \; & \sum_{p\in PR} -val_p . s_{p,'t2'} + \sum_{\substack{t\in\{t1,t2\}\\ c\in CO}} \mu_{1_{c,t}} (\sum_{p\in PR} x_{c,p,t} - AR_{c,t}) \\ & + \sum_{\substack{t\in\{t0,t1\}\\ p\in PR}} \mu^k_{2_{p,t}} (-s_{p,t} - \sum_{c\in CO} x_{c,p,t+1} + s_{p,t+1} + LF_{p,t+1} \\ & + \sum_{\substack{t\in\{t0,t1\}\\ p\in PR\\ l\in QL}} \mu_{3_{p,l,t}} (-s_{p,t} . q_{p,l,t} - \sum_{c\in CO} x_{c,p,t+1} . QU_{c,l} + [s_{p,t+1} + LF_{p,t+1}] . q_{p,l,t+1}) \\ & + \sum_{\substack{t\in\{t1,t2\}\\ p\in PR}} \mu_{4_{p,t}} (0 - s_{p,t}) + \sum_{\substack{t\in\{t1,t2\}\\ p\in PR}} \mu_{5_{p,t}} (s_{p,t} - stock_{p,t}) \end{aligned}$$

Using the KKT gradient conditions for the x variables, it can be seen that the terms in x in the Lagrange function will vanish (due to the fact that they are not *connected* variables). This, along with the use of the KKT gradient conditions for the s variables, enables the Lagrange function to be written in the following form :

$$\begin{aligned} L(x,s,q,\mu^k) \; = \; & - \sum_{\substack{t\in\{t1,t2\}\\ c\in CO}} \mu_{1_{c,t}} - AR_{c,t} + \sum_{\substack{t\in\{t0,t1\}\\ p\in PR}} \mu^k_{2_{p,t}} . LF_{p,t+1} \\ & + \sum_{\substack{t\in\{t0,t1\}\\ p\in PR\\ l\in QL}} \mu_{3_{p,l,t}} . LF_{p,t+1} . q_{p,l,t+1} - \sum_{\substack{t\in\{t1,t2\}\\ p\in PR}} \mu_{5_{p,t}} . stock_{p,t} \\ & + \sum_{\substack{t\in\{t1,t2\}\\ p\in PR\\ l\in QL}} (\mu^k_{3_{p,l,t-1}} - \mu^k_{3_{p,l,t}}) . (q_{p,l,t} - q^k_{p,l,t}) . s_{p,t} \end{aligned}$$

Thus, the *qualifying* constraints to be added along with the Lagrange function to the *relaxed dual* problem are of the form

$$\begin{aligned} \sum_{l\in QL} (\mu^k_{3_{p,l,t-1}} - \mu^k_{3_{p,l,t}}) . (q_{p,l,t} - q^k_{p,l,t}) \; & \geq 0 \quad if \; s_{p,t} = s^L_{p,t} \\ \sum_{l\in QL} (\mu^k_{3_{p,l,t-1}} - \mu^k_{3_{p,l,t}}) . (q_{p,l,t} - q^k_{p,l,t}) \; & < 0 \quad if \; s_{p,t} = s^U_{p,t} \end{aligned}$$

for all $t \in \{t1, t2\}, \; p \in PR$.

There are six s variables (corresponding to three products at two time periods $t1$ and $t2$). Hence, there are 64 (2^6) *relaxed dual* problems solved at every iteration.

6.1 The GOP Algorithm Without the New Properties

When the **GOP** algorithm was applied to the problem in this form, it identified a global solution of -9.5316 from all considered starting points. Starting from the lower bound, the algorithm took 16 iterations to converge, solving a total of 16 primal and 1024 *relaxed dual* subproblems. In addition, the algorithm needed to solve 3 *relaxed primal* problems.

6.2 The GOP Algorithm With the New Properties

When the **GOP** algorithm was applied while making use of Property 2 and avoiding the solution of the *relaxed dual* problems for iterations when the primal problem was infeasible, it required 11 iterations to converge, and needed to solve a total of 240 *relaxed dual* subproblems (vs. 1024 *relaxed dual* problems before). In addition, when Property 1 was also used, the number of *relaxed dual* problems that were solved was reduced from 1024 to 8. From other starting points, similar improvements were obtained upon making use of the new properties.

7 Application 3: Pooling/Blending Problems

In this section, we consider the application of the proposed algorithm to several pooling/blending problems arising in chemical process models. These problems are taken from Ben-Tal and Gershovitz (1992), and have the following form :

$$\max \quad -\sum_i \sum_l c_i x_{il} + \sum_l \sum_j d_j y_{lj} + \sum_i \sum_j (d_j - c_i) z_{ij}$$

subject to

$$\begin{aligned}
\sum_l x_{il} + \sum_j z_{ij} &\leq A_i \\
\sum_i x_{il} + \sum_j y_{lj} &= 0 \\
\sum_i x_{il} &\leq S_l \\
-\sum_i q_{ik} x_{il} + p_{lk} \sum_j y_{lj} &= 0 \\
\sum_l y_{lj} + \sum_i z_{ij} &\leq D_j
\end{aligned}$$

$$\begin{aligned}\sum_l (p_{lk} - Q_{jk}) y_{lj} + \sum_i (q_{ik} - Q_{jk}) z_{ij} &\le 0 \\ p_{lk}, x_{il}, y_{lj}, z_{ij} &\ge 0\end{aligned}$$

where

$$\begin{aligned}\{1, 2, \cdots, i, \cdots, I\} &\equiv \text{set of components} \\ \{1, 2, \cdots, j, \cdots, J\} &\equiv \text{set of products} \\ \{1, 2, \cdots, k, \cdots, K\} &\equiv \text{set of qualities} \\ \{1, 2, \cdots, l, \cdots, L\} &\equiv \text{set of pools}\end{aligned}$$

and the following variables represent :

x_{il}	–	amount of component i allocated to pool l
y_{lj}	–	amount going from pool l to product j
z_{ij}	–	amount of component i going to product j
p_{lk}	–	level of quality k in pool l

The following parameters represent :

A_i	–	Upper bounds for component availabilities
D_j	–	Upper bounds for product demands
S_l	–	Upper bounds for pool sizes
Q_{jk}	–	Upper bounds for product qualities
q_{ik}	–	Level of quality k in component$i$
c_i	–	Unit price of component i
d_j	–	Unit price of product j

Four different instances of this problem were considered. The data for these problems can be found in Ben-Tal and Gershovitz (1992). The results of application of the enhanced **GOP** algorithm to these problems is given in Table 3.

Problem No.	*Problem Size*				*GOP Algorithm*	
	I	J	K	L	Iterations	CPU (HP730)
1.	4	2	1	1	7	0.95
2.	5	5	2	1	19	3.19
3.	5	5	2	3	47	44.54
4.	5	5	2	3	42	40.31

Table 3: Pooling Problems From Ben-Tal and Gershovitz (1992)

8 Concave and Indefinite Quadratic Problems

The algorithm was applied to a number of standard test problems from the literature and a large number of randomly generated concave and indefinite quadratic programming problems. This section presents the results for these runs. All the results reported in this section were obtained by running a version of the **GOP** algorithm coded in C. The algorithm was run on an HP 9000/730 workstation with 64 MB of memory. The linear programming solver CPLEX was used to solve the linear primal and relaxed dual subproblems.

8.1 Example Problems From Literature

The **GOP** algorithm was applied to 11 small-size concave programming problems from Phillips and Rosen (1988.) These problems have the following form :

$$\min_{x,y\in\Omega} \ \psi(x,y) = \theta_1\varphi(x) + \theta_2 d^t y$$

where

$$\begin{aligned}
\varphi &= 0.5\sum_{i=1}^{n}\lambda_i(x_i-\overline{\omega_i})^2, \\
\Omega &= \{(x,y) : A_1x + A_2y \le b, x \ge 0, y \ge 0\}, \\
x,\ \lambda,\ \overline{\omega} &\in \Re^n, \\
y,\ d &\in \Re^k, \\
A_1 &\in \Re^{m\times n} \\
A_2 &\in \Re^{m\times k}
\end{aligned}$$

Problem	*Problem Size*			*GOP Algorithm*		P&R
	M	N	K	Iterations	CPU (HP730)	CPU (CRAY2)
example	5	2	0	9	1.09	0.026
prob1	5	6	0	3	0.54	0.022
prob2	5	6	0	3	0.55	0.020
prob3	5	6	0	2	0.45	0.026
prob10	4	2	0	10	1.17	0.017
prob11	4	3	0	12	1.48	0.015
prob12	4	3	0	12	1.50	0.014
prob13	10	3	0	5	0.68	0.022
prob14	10	3	0	6	0.82	0.020
prob15	4	4	0	10	2.03	0.029
prob20	9	2	1	48	8.98	0.023

Table 4: Example Problems from Phillips and Rosen (1988) ($\epsilon = 0.001$)

$$\theta_1,\ \theta_2\ \in\ \Re.$$

Here, m is the number of linear constraints, n is the number of concave variables (x), and k is the number of linear variables (y). The parameters θ_1 and θ_2 are -1 and 1 respectively, and the relative tolerance for convergence between the upper and lower bounds (ϵ) is 0.001.

The results of the application of the algorithm to these problems are given in Table 4. The CPU times for the **GOP** algorithm and the Phillips and Rosen algorithm (denoted by P&R) are given in seconds. However, it should be noted that the P&R algorithm was run on a CRAY2. As can be seen, the algorithm solves problems of this size very fast, usually taking about 2-3 iterations to identify the optimal solution and then about 5-10 iterations to converge.

8.2 Randomly Generated Concave Problems

The algorithm was also applied to randomly generated problems of the form given above. Such problems have earlier been studied by Phillips and Rosen (1988). The data for the

Run	Problem size			Iterations		CPU (sec.)	
	m	n	k	N_I	S_I	N_C	S_C
1	20	25	0	1.00	0.000	0.456	0.010
2	20	25	50	2.10	0.300	1.662	1.614
3	20	25	100	3.00	0.447	16.508	19.711
4	20	25	200	3.18	0.833	33.149	28.441
5	20	25	400	3.64	0.771	82.026	57.834

Table 5: Random runs for m = 20, n = 25 and k ranging from 0 to 400 ($\epsilon = 0.1$)

Run	Problem size			Iterations		CPU (sec.)	
	m	n	k	N_I	S_I	N_C	S_C
1	20	50	0	1.00	0.000	0.554	0.012
2	20	50	50	2.40	0.489	17.436	30.908
3	20	50	100	2.70	0.455	46.761	49.195
4	20	50	200	3.06	0.249	108.968	80.490

Table 6: Random runs for m = 20, n = 50 and k ranging from 0 to 200 ($\epsilon = 0.1$)

constants $\lambda, \varpi, d, A_1, A_2$ and b was generated by the same routines used by Phillips and Rosen. The values of the parameters θ_1 and θ_2 are -0.001 and 0.1 respectively.

Table 5 gives the results of the runs for $m = 20$, $n = 25$ and k going from 0 to 400, while Table 6 gives the results of the runs for $m = 20$, $n = 50$ and k going from 0 to 200. In both cases, the tolerance was $\epsilon = 0.1$, and 10 different seeds were used. In the table, N_I represents the average number of iterations taken to converge within ϵ, and S_I is the standard deviation for the ten runs. N_C is the average time taken in CPU seconds and S_C is the standard deviation over ten runs.

In all the cases, the algorithm requires very few iterations for the upper and lower bounds to be within 0.1 of the optimal solution. In no instance does the algorithm take more than 5 iterations to converge. Moreover, as the number of linear variables increases, the total CPU time increases almost linearly.

Run	Problem size			Iterations		CPU (sec.)	
	m	n	k	N_I	S_I	N_C	S_C
1	20	25	0	2.90	1.609	0.871	0.702
2	20	25	50	6.57	5.518	6.601	7.430
3	20	25	100	11.75	9.601	39.415	33.078

Table 7: Random runs for m = 20, n = 25 and k ranging from 0 to 100 ($\epsilon = 0.001$)

Run	Problem size			Iterations		CPU (sec.)	
	m	n	k	N_I	S_I	N_C	S_C
1	40	25	0	1.00	0.000	0.465	0.017
2	40	25	50	1.75	0.433	0.970	0.566
3	40	25	100	2.00	0.000	2.708	4.211
4	40	25	200	2.40	0.489	25.142	26.127

Table 8: Random runs for m = 40, n = 25 and k ranging from 0 to 200 ($\epsilon = 0.1$)

Table 7 gives the results of runs for $m = 20$, $n = 25$ and k going from 0 to 100 for a tolerance of $\epsilon = 0.001$. The number of iterations as well as the time taken to converge increases considerably to get to solutions of higher accuracy. Contrary to this, when the number of constraints is increased to $m = 40$ while keeping the tolerance at $\epsilon = 0.1$, (Table 8) the algorithm does not take any appreciably longer time to converge. This indicates that for these sizes of problems, the algorithm is very fast in converging to a solution of low accuracy. What is more interesting to note is that in almost all cases, even when the tolerance specified is 0.1, the actual upper bound found by the algorithm is very close (and often identical) to the global optimum for the problem.

It should be noted here that all the relaxed dual subproblems are actually solved as linear subproblems. In fact, this is not really necessary. Due to the branch and bound nature of the algorithm, it is certainly possible to devise a method to enumerate the solutions of the subproblems implicitly. Using such a scheme to solve the relaxed dual problem at every iteration would considerably reduce the solution time for the algorithm.

Run	Problem size			Iterations		CPU (sec.)	
	m	n	k	N_I	S_I	N_C	S_C
1	20	25	0	2.61	2.519	21.552	80.381
2	20	25	50	6.87	4.781	50.703	65.973
3	20	25	100	11.32	8.710	121.718	134.201

Table 9: Random runs of indefinite quadratic problems for m = 20, n = 25 and k ranging from 0 to 100 ($\epsilon = 0.1$)

8.3 Randomly Generated Indefinite Quadratic Problems

The algorithm was also applied to randomly generated indefinite quadratic problems. These problems have the same form as for the concave problems, and are generated in the same form except for the fact that some of the λ's are generated positive and some are generated negative. Due to the nature of the generation, the number of positive and negative λ's generated are roughly equal.

Table 9 gives the results of the runs for $m = 20$, $n = 25$ and k going from 0 to 400, while Table 6 gives the results of the runs for $m = 20$, $n = 50$ and k going from 0 to 200. In both cases, the tolerance was $\epsilon = 0.1$. These problems take considerably more time to solve than the corresponding pure concave problems. The main reason for this is that in both cases, the relaxed dual problems are being solved as linear programming problems. For the case of concave problems, the linear underestimators are sufficient to provide fairly tight lower bounds for the optimal solution. However, in the case of indefinite programming problems, the linear underestimators bound the convex terms in the objective very loosely.

Conclusions

The **GOP** algorithm is a rigorous approach toward global optimization of mathematical programming problems that satisfy certain conditions. As part of the **GOP** algorithm, a number of *relaxed dual* problems need to be solved at every iteration. In this paper, theoretical properties based on the linearity of the Lagrange function that is used in the *relaxed dual* problems are presented for (a) concave quadratic problems, (b) indefinite quadratic problems, and (c) quadratically constrained problems. These properties are illustrated through three applications as well as a large number of randomly generated problems. As the results for the applications show, the new properties result in a large decrease in the number of *relaxed dual* problems that need to be solved at every iteration. At the same time, this results in a decrease in the number of iterations required for convergence, due to the decrease in the number of stored solutions that have to be investigated by the **GOP** algorithm. As a result, the new properties significantly improve the computational efficiency of the **GOP** algorithm.

It is expected that similar properties can be developed for other classes of problems. Research work on the development of additional theoretical properties that can further enhance the computational efficiency of the **GOP** algorithm for other classes of problems is currently underway, and results will be reported in a future publication.

References

[1] Aggarwal, A., and Floudas, C.A., A Decomposition Approach for Global Optimum Search in QP, NLP and MINLP problems, *Annals of Operations Research*, **25**, 119 (1990).

[2] Al-Khayyal, F.A., and Falk, J.E., Jointly Constrained Biconvex Programming, *Mathematics of Operations Research*, **8**, (1983).

[3] Al-Khayyal, F.A., Jointly Constrained Bilinear Programs and Related Problems : An Overview , *Computers in Mathematical Applications*, **19**, 53 (1990).

[4] Al-Khayyal, F.A., Horst, R., and Pardalos, P.M., Global Optimization of Concave Functions Subject to Quadratic Constraints : An Application in Nonlinear Bilevel Programming , *Annals of Operations Research, Special Issue on Hierarchical Optimization*, To appear (1990).

[5] Archetti, F., and Schoen, F., A Survey on the Global Minimization Problem : General Theory and Computational Approaches, *Annals of Operations Research*, **1**, 87 (1984).

[6] Ben-Tal, Aharon, and Gershovitz, Vladimir, Computational Methods for the Solution of the Pooling/Blending Problem, *Technical Report*, Technion-Israel Institute of Technology, Haifa, Israel (1992).

[7] Branin, F.H., Widely Convergent Methods for Finding Multiple Solutions of Simultaneous Nonlinear Equations, *IBM Journal of Research Developments*, 504 (1972).

[8] Dixon, L.C.W., and Szego, G.P., Editors, *Towards Global Optimization*, North Holland, Amsterdam (1975).

[9] Dixon, L.C.W., and Szego, G.P., Editors, *Towards Global Optimization 2.*, North-Holland, Amsterdam (1978).

[10] Floudas, C.A., and Aggarwal, A., A Decomposition Strategy for Global Optimum Search in the Pooling Problem, *Operations Research Journal on Computing*, **2(3)**, in Press (1990).

[11] Floudas, C.A., Aggarwal, A., and Ciric, A.R., Global Optimum Search for Nonconvex NLP and MINLP problems, *Computers and Chemical Engineering*, **13**, 1117 (1989).

[12] Floudas, C.A., and Pardalos, P.M., A Collection of Test Problems for Constrained Global Optimization Algorithms, *Lecture Notes in Computer Science*, Eds. G. Goos and J. Hartmanis, Vol. 455, Springer-Verlag, Berlin (1990).

[13] Floudas, C.A., and Pardalos, P.M., Recent advances in Global Optimization, *Princeton Series in Computer Science*, Princeton University Press, Princeton, New Jersey (1992).

[14] Floudas, C.A., and Visweswaran, V., A Global Optimization Algorithm (GOP) for Certain Classes of Nonconvex NLPs - I. Theory, *Computers and Chemical Engineering*, **14(12)**, 1419 (1990).

[15] Floudas, C.A., and Visweswaran, V., A Primal-Relaxed Dual Global Optimization Approach : Theory , *Journal of Optimization Theory and Applications*, In Press (1992).

[16] Geoffrion, A.M., Generalized Benders Decomposition, *Journal of Optimization Theory and Applications*, **10**, 237 (1972).

[17] Hansen, E.R., Global Optimization using Interval Analysis: The One-Dimensional Case, *Journal of Optimization Theory and Applications*, **29**, 331 (1979).

[18] Hansen, P., Jaumard, B., and Lu, S-H., Global Optimization of Univariate Lipschitz functions : I. Survey and Properties, *Mathematical Programming*, **55**, 251-272 (1992).

[19] Hansen, P., Jaumard, B., and Lu, S-H., Global Optimization of Univariate Lipschitz functions : II. New Algorithms and Computational Comparisons, *Mathematical Programming*, **55**, 273-292 (1992).

[20] Hansen, P., Jaumard, B., and Lu, S-H., An Analytical Approach to Global Optimization, *Mathematical Programming*, **52**, 227 (1991).

[21] Horst, R., and Tuy, H., On the Convergence of Global Methods in Multiextremal Optimization, *Journal of Optimization Theory and Applications*, **54**, 253 (1987).

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

[23] Kirkpatrick, S., Gelatt, C.D, and Vecchi, M.P., Optimization by Simulated Annealing, *Science*, **220**, 671 (1983).

[24] Levy, A.V., and Montalvo, A., The Tunneling Algorithm for the Global Minimization of Functions , *SIAM J. of Sci. Stat. Comp.*, **6**, 15 (1985).

[25] Mockus, J., Bayesian Approach to Global Optimization - Theory and Applications, Kluwer Academic Publishers, The Netherlands (1989).

[26] Pardalos, P.M., Glick, J.H., and Rosen, J.B., Global Minimization of Indefinite Quadratic Problems, *Computing*, **39**, 281 (1987).

[27] Pardalos, P.M., and Rosen, J.B., Methods for Global Concave Minimization: A Bibliographic Survey, *SIAM Review*, **28**, 367 (1986).

[28] Pardalos, P.M., and Rosen, J.B., Constrained Global Optimization : Algorithms and Applications, *v. 268 of Lecture Notes in Computer Science*, Springer Verlag (1987).

[29] Phillips, A. T. and Rosen, J. B., A Parallel Algorithm for Concave Quadratic Global Optimization, *Mathematical Programming*, **42**, 421 (1988).

[30] Piyavskii, S.A., An Algorithm for finding the Absolute Extremum of a Function, *USSR Comput. Math. Phys.*, **12**, 57 (1972).

[31] Ratschek, H., and Rokne, J., *New Computer Methods for Global Optimization*, Halsted Press (1988).

[32] Rinnooy Kan, A.H.G., and Timmer, G.T., Stochastic Global Optimization Methods. Part I: Clustering Methods, *Mathematical Programming*, **39**, 27 (1987).

[33] Törn, A., and Zilinskas, A., Global Optimization , *Lecture Notes in Computer Science, No. 350*, Springer-Verlag (1987).

[34] Tuy, H., Thieu, T.V., and Thai, N.Q., A Conical Algorithm for Globally Minimizing a Concave Function Over a Closed Convex Set, *Mathematics of Operations Research*, **10**, 498 (1985).

[35] Zilinskas, A., *Global Optimization – Axiomatics of Statistical Models, Algorithms and Their Applications*, Moklas, Vilnius (1986)