Journal of Global Optimization **12:** 1–36, 1998.

A Finite Algorithm for Global Minimization of Separable Concave Programs

J. PARKER SHECTMAN[1] and NIKOLAOS V. SAHINIDIS[2,*]
[1]*Department of Mechanical & Industrial Engineering and* [2]*Department of Chemical Engineering, The University of Illinois at Urbana-Champaign, 600 South Mathews Avenue, Urbana, IL 61801, U.S.A.*

(Received: 24 July 1995; accepted: 22 May 1997)

Abstract. Researchers first examined the problem of separable concave programming more than thirty years ago, making it one of the earliest branches of nonlinear programming to be explored. This paper proposes a new algorithm that finds the exact global minimum of this problem in a finite number of iterations. In addition to proving that our algorithm terminates finitely, the paper extends a guarantee of finiteness to all branch-and-bound algorithms for concave programming that (1) partition exhaustively using rectangular subdivisions and (2) branch on the incumbent solution when possible. The algorithm uses domain reduction techniques to accelerate convergence; it solves problems with as many as 100 nonlinear variables, 400 linear variables and 50 constraints in about five minutes on an IBM RS/6000 Power PC. An industrial application with 152 nonlinear variables, 593 linear variables, and 417 constraints is also solved in about ten minutes.

Key words: Global optimization, concave programming, branch-and-bound, domain reduction.

1. Introduction

This paper addresses the separable concave programming problem:

$$\text{(SCP):} \quad \begin{array}{l} \text{global min } f(x) \\ \text{subject to } x \in D \cap C \end{array}$$

where $x = (x_1, \ldots, x_n) \in R^n$. Given are $C = \prod_{j=1}^{n} C_j$, $C_j = [l_j, u_j]$, and $l_j, u_j \in R \cup \{-\infty, +\infty\}$; $f(x) = \sum_{j=1}^{n} f_j(x_j)$, $f_j : R \to R$, and for each j, f_j concave and bounded on C_j; $D = \{x : \sum_{j=1}^{n} a_{ij} x_j \leq b_i, i = 1, ..., m\}$, and $a_{ij}, b_i \in R$; and lastly $D \cap C$ assumed to be bounded.

Concave minimization has been a central problem in global optimization since its inception. From the viewpoint of computational complexity, SCP is *NP*-hard. Even checking whether a given feasible point is a local solution is itself an *NP*-hard problem [48]. Due to the *NP*-completeness of SCP, large classes of dissimilar problems from a variety of disciplines can be reformulated as SCPs, *e.g.*, integer programming problems [25], [27], [60]; max-min problems [19]; linear and nonlinear complementarity problems [41], [77], [79]; quadratic assignment problems

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

[2], [39]; and 3-dimensional assignment problems [23]. Yet, special properties and solution approaches can make SCPs more tractable than many *NP*-hard problems.

The industrial applications of concave programming range widely and run deeply through the gamut of product and process planning (see [33, pp. 11-14], as well as [6]). Many SCPs in operations research stem from economies of scale and fixed-charges, with applications arising in communications network planning, hydraulic network planning, chemical process network planning, plant location problems, and inventory and production scheduling. Not surprisingly, the state-of-the-art demands fast, optimal solution of SCPs.

Extensive surveys of concave programming methods are given by Horst [31], Pardalos and Rosen [50], Benson [6] and in the books by Pardalos and Rosen [51], Horst and Tuy [33], and Horst, Pardalos, and Thoai [32]. In brief, the three most tried strategies are: enumerating extreme points of the feasible domain [8], [12], [17], [18], [42], [43]; cutting-planes [2], [11], [13], [26], [29], [37], [78], [80], [83], [85], [86], [87]; and branch-and-bound [4], [6], [8], [9], [10], [11], [21], [20], [30], [36], [44], [47], [53], [54], [55], [56], [57], [58], [62], [63], [73], [74], [82]. Algorithms for the more general case subject to nonlinear constraints, such as [5], [7], [34], generally combine branch-and-bound with cutting planes.

Our algorithm is a variant of branch-and-bound that is specialized for optimization over polyhedra. The main contributions of the work are to prove that a particular branching rule ensures finiteness and to demonstrate that a collection of domain reduction techniques accelerate convergence, making routine the solution of problems with as many as 100 nonlinear variables, 400 linear variables and 50 constraints in a matter of minutes on an IBM RS/6000 Power PC. A preliminary version of this work was presented at the 1995 Princeton University Conference on the State of the Art in Global Optimization [71]. In all areas addressed by the work, the current paper includes major additional material. To wit, (1) the theory and its proof have been strengthened, in particular, the finiteness results do not require the global optima to be isolated; (2) the section on acceleration devices has been expanded with the addition of new techniques based on parametric programming; finally, (3) we display improved computational results based on a more recent implementation.

The remainder of the report is structured as follows. Section 2 comprises a statement of the proposed algorithm and Section 3 contains a proof of its finiteness. Section 4 provides finite variants of the relaxed algorithm of Falk and Soland [21], and the algorithms of Kalantari and Rosen [36], and Phillips and Rosen [54]. Section 5 describes domain reduction techniques that accelerate the solution process. Computer implementation is discussed in Section 6, along with numerical results.

2. Algorithm

2.1. Background and Outline of Algorithm

The algorithm combines standard branch-and-bound procedures with a new branching rule and accelerating devices to which we refer as *domain reduction* techniques. To emphasize the importance of the domain reduction techniques, we sometimes refer to the algorithm as a branch-and-*reduce* method (Section 5 deals with these acceleration devices in detail). The procedure will now be formally outlined. In the following algorithm, words in italic letters constitute the critical operations and will be discussed separately.

ALGORITHM 1.

Initialization

Preprocess the problem constraints $D \cap C$ to form a bounded initial hyperrectangle C^0. Add the problem min $f(x)$ s.t. $x \in D \cap C^0$ to the list $\mathcal{S}$ of open subproblems.

Set $U \longleftarrow +\infty$.

Choose an integer $2 \leq N < \infty$ to be used in *branching*.

Let $k \longleftarrow 0$. At each iteration k of the algorithm,

do (Step k)

Step *k*.1. *Select* a subproblem s_k, defined as min $f(x)$ s.t. $x \in D \cap C^{s_k}$, from the list $\mathcal{S}$ of currently open subproblems. $\mathcal{S} \longleftarrow \mathcal{S} \setminus \{s_k\}$.

Step *k*.2. *Bound* the optimum of subproblem s_k from above and below, *i.e.*, find α^{s_k} and β^{s_k} satisfying $\alpha^{s_k} \geq \min\{f(x) \text{ s.t. } x \in D \cap C^{s_k}\} \geq \beta^{s_k}$. By convention, $\alpha^{s_k} = \beta^{s_k} = +\infty$ if $D \cap C^{s_k} = \emptyset$, *i.e.*, if s_k is infeasible. If $\beta^{s_k} < +\infty$, a feasible point $x^{s_k} \in D \cap C^{s_k}$ such that $f(x^{s_k}) = \alpha^{s_k}$ will be found in the process.

Step *k*.2.a. $L \longleftarrow \min_{s \in \mathcal{S} \cup \{s_k\}} \beta^s$; **If** $f(x^{s_k}) < U$ **then** $\tilde{x} \longleftarrow x^{s_k}$ and $U \longleftarrow f(\tilde{x})$.

Step *k*.2.b. If $U = L$, **then terminate** with optimizer $\tilde{x}$.

Step *k*.2.c. $\mathcal{S} \longleftarrow \mathcal{S} \setminus \{s : \beta^s \geq U\}$ (fathoming rule). **If** $\beta^{s_k} \geq U$, **then goto Step *k*.1** (select another subproblem).

Step *k*.3. *Branch*, partitioning C^{s_k} into two new elements $C^{s_{k1}}$ and $C^{s_{k2}}$. They satisfy $C^{s_{k1}} \cup C^{s_{k2}} = C^{s_k}$ and $C^{s_{k1}} \cap C^{s_{k2}} = \partial C^{s_{k1}} \cap \partial C^{s_{k2}}$. $\mathcal{S} \longleftarrow \mathcal{S} \cup \{s_{k1}, s_{k2}\}$, *i.e.*, append the two subproblems min$\{f(x)$ s.t. $x \in D \cap C^{s_{k1}}\}$ and min$\{f(x)$ s.t. $x \in D \cap C^{s_{k2}}\}$ to the list of open subproblems.

For selection purposes, $x^{s_{k1}}, x^{s_{k2}} \longleftarrow x^{s_k}$; $\alpha^{s_{k1}}, \alpha^{s_{k2}} \longleftarrow \alpha^{s_k}$;
$\beta^{s_{k1}}, \beta^{s_{k2}} \longleftarrow \beta^{s_k}$ (inheritance of bounds).

Let $k \longleftarrow k+1$, and **goto Step** *k*.**1**.

end do

An in-depth discussion of the critical operations *preprocessing*, *bounding*, *selection*, and *branching* now follows.

2.2. PREPROCESSING

OPERATION 1. PREPROCESSING.

> For any variable x_j that is unrestricted from below, *i.e.*, $l_j = -\infty$, replace l_j in C_j with the solution to the linear program $\min x_j$ s.t. $x \in D \cap C$; and for any variable x_j that is unrestricted from above, *i.e.*, $u_j = \infty$, replace u_j in C_j with the solution to the linear program $\max x_j$ s.t. $x \in D \cap C$.

The solution of these linear programs (LPs) yields a bounded reformulation of SCP. In the process of solving these LPs, the algorithm records the feasible solutions that it encounters as preliminary bounds for use in domain reduction. Optionally, this procedure may also be applied to restricted variables, which frequently results in a tighter problem formulation.

2.3. BOUNDING

The algorithm determines bounds on each concave subproblem s_k (of Step k.2) by solving a linear programming relaxation which is formulated in the usual fashion. For each univariate concave term $f_j(x_j)$ in the objective, first construct the linear underestimator, call it $g_j(x_j)$, that intersects $f_j(x_j)$ at the current bounds $l_j^{s_k}$ and $u_j^{s_k}$ of x_j. It is well known that $g_j^{s_k}(x_j)$ is in fact the convex envelope of $f_j(x_j)$ over $[l_j^{s_k}, u_j^{s_k}]$. It is also well known that the convex envelope of a separable function $f(x) = \sum_{j=1}^{n} f_j(x_j)$ over a rectangular set C^{s_k} is the sum of the convex envelopes of its individual terms $f_j(x_j)$ taken over their respective intervals $C_j^{s_k}$ [21]. Hence, the convex envelope of $f(x)$ over C^{s_k} is $g^{s_k}(x) = \sum_{j=1}^{n} g_j^{s_k}(x_j)$.

OPERATION 2. BOUNDING (Step k.2).

> Define the LP relaxation of s_k as $\min g^{s_k}(x)$ s.t. $x \in D \cap C^{s_k}$. Let ω^{s_k} be a basic optimal solution of this LP relaxation. A lower bound on the optimum of the concave subproblem is given by $\beta^{s_k} = g^{s_k}(\omega^{s_k})$, and an upper bound may be obtained by evaluating $\alpha^{s_k} = f(\omega^{s_k})$.

2.4. SELECTION

In Step k.1 of each iteration k, the procedure selects a single subproblem to be considered for bounding—specifically, a subproblem from the list of open subproblems which has the the least lower bound.

OPERATION 3. SELECTION RULE (Step k.1).

Select any $s_k \in \mathcal{S}$ such that $\beta^{s_k} = L$.

2.5. BRANCHING

As described in Step k.3, branching replaces the partition element C^{s_k} with two new elements. In this manner, the algorithm constructs a binary tree of subproblems. Hence, every element formed in the course of the procedure belongs to a unique level of this tree. If the level of s_k is a multiple of N, the algorithm selects for partitioning a longest edge of C^{s_k} (from among those edges corresponding to nonlinear variables), and bisects this edge. This measure is included to ensure finiteness.

In a typical iteration, however, the partitioning rule selects an edge that corresponds to a variable most responsible for the gap (at the LP solution) between the concave objective $f(\omega^{s_k})$ and its local underestimator $g^{s_k}(\omega^{s_k})$. The rule then bisects the selected edge.

The partitioning rule has one additional twist. If the best solution currently known lies within the C^{s_k}, it will be used as the branching point instead of the midpoint, (provided this results in two strictly smaller elements). Branching on the best solution currently known is key to guaranteeing the finiteness of the procedure.

Below, j' is the index of the partitioning variable, p is the partitioning point, N is a user supplied parameter in the initialization step of the algorithm, $\mathcal{L}(s_k)$ gives the level of the tree to which subproblem s_k belongs, $\mathcal{J}^{s_k}$ denotes the set of indices of variables nonlinear on C^{s_k}, $\tilde{x}$ is the best solution currently known ($f(\tilde{x}) = U$). The partitioning rule is now stated more precisely.

OPERATION 4. PARTITIONING RULE (Step k.3).

if $\mathcal{L}(s_k)$ mod $N = 0$ **then**
  $j' \in$ argmax$_{j \in \mathcal{J}^{s_k}}(u_j^{s_k} - l_j^{s_k})$,
  (a nonlinear variable corresponding to a longest edge of C^{s_k}),
  $p = (u_{j'}^{s_k} - l_{j'}^{s_k})/2$,
  (bisect $[l_{j'}^{s_k}, u_{j'}^{s_k}]$).
else
  $j' \in$ argmax$_{j \in \mathcal{J}^{s_k}}[f_j(\omega_j^{s_k}) - g_j^{s_k}(\omega_j^{s_k})]$,
  (a variable with largest underestimation gap at $\omega_j^{s_k}$).
  if $\tilde{x} \in C^{s_k}$ and $\tilde{x}_{j'} \in]l_{j'}^{s_k}, u_{j'}^{s_k}[$ **then**

$p = \tilde{x}_{j'}$.
(partition through $\tilde{x}$).
else
$p = (u_{j'}^{s_k} - l_{j'}^{s_k})/2$,
(bisect $[l_{j'}^{s_k}, u_{j'}^{s_k}]$).
endif
endif
Split the domain C^{s_k}=$\prod_{j=1}^{n} C_j^{s_k}$=$\prod_{j=1}^{n}[l_j^{s_k}, u_j^{s_k}]$ into two subdomains $[l_{j'}^{s_k}, p_{j'}^{s_k}] \prod_{j\neq j'}[l_j^{s_k}, u_j^{s_k}]$ and $[p_{j'}^{s_k}, u_{j'}^{s_k}] \prod_{j\neq j'}[l_j^{s_k}, u_j^{s_k}]$.

Note that the proposed partitioning rule successively refines the initial rectangular set C^0 of variable bounds through the course of the branch-and-bound procedure. Moreover, each partition element is itself a rectangular set. In branch-and-bound algorithms, rectangular partitioning is the most natural choice when minimizing a separable function, because of the ease in bounding. Tuy [81] treats non-rectangular partitions for branch-and-bound.

3. Convergence and Finiteness

For the general branch-and-bound procedure, it can be easily shown that if the bounding operation is *consistent* and the selection operation is *bound improving* then the procedure is *convergent* (see [33, IV.2] for definitions and relevant theorems). We now show that the proposed algorithm possesses a much stronger property than convergence, namely finiteness.

The proof is by contradiction. Consider the tree of subproblems generated by the branch-and-bound procedure, and assume that the algorithm is infinite. By this assumption, the algorithm must generate at least one infinite sequence $\{C^{s_q}\}$ of subdomains C^{s_q} that are *nested*, *i.e.*, satisfying $C^{s_{q+1}} \subset C^{s_q}$, where q indicates the level of the tree to which subproblem s_q belongs. The following lemmas deal with such a sequence. They show that any path descending from the root of the tree will terminate at finite length.

LEMMA 1. $\lim_{q\to\infty}(u_j^{s_q} - l_j^{s_q}) = 0$, *for all* $j \in \mathcal{J}^{s_q}$ *(exhaustiveness).*

Proof. This property follows from the fact that every finitely many levels along the path the algorithm bisects a longest edge. It suffices to prove that $\lim_{q\to\infty} \max_{j\in\mathcal{J}^{s_q}}(u_j^{s_q} - l_j^{s_q}) = 0$, meaning that for given $\varepsilon > 0$, there exists $M > 0$ sufficiently large such that for $q \geq M$, $\max_{j\in\mathcal{J}^{s_q}}(u_j^{s_q} - l_j^{s_q}) < \varepsilon$. Let Λ denote $\max_{j\in\mathcal{J}^0}(u_j^0 - l_j^0)$—from among $j \in \mathcal{J}^0$, a longest edge of the initial box C^0. Every N levels along the path, the algorithm halves the length of a longest edge from among $j \in \mathcal{J}^{s_q}$. Hence, when $q \geq |\mathcal{J}^0| N$, $\max_{j\in\mathcal{J}^{s_q}}(u_j^{s_q} - l_j^{s_q}) \leq \frac{1}{2}\Lambda$; when $q \geq 2|\mathcal{J}^0| N$, $\max_{j\in\mathcal{J}^{s_q}}(u_j^{s_q} - l_j^{s_q}) \leq \frac{1}{4}\Lambda$; etc. In general, $\max_{j\in\mathcal{J}^{s_q}}(u_j^{s_q} - l_j^{s_q}) \leq \frac{1}{2}^{\lfloor \frac{q}{|\mathcal{J}^0|N} \rfloor}\Lambda$. If we seek $\frac{1}{2}^{\lfloor \frac{q}{|\mathcal{J}^0|N} \rfloor}\Lambda < \varepsilon$, we must have

$\lfloor \frac{q}{|\mathcal{J}^0|N} \rfloor > \log_2 \frac{\Lambda}{\varepsilon}$, in other words, $\frac{q-(q \bmod |\mathcal{J}^0|N)}{|\mathcal{J}^0|N} > \log_2 \frac{\Lambda}{\varepsilon}$. It follows that we need $q > |\mathcal{J}^0| N \log_2 \frac{\Lambda}{\varepsilon} + q \bmod |\mathcal{J}^0| N$, which is at most $|\mathcal{J}^0| N \log_2 \frac{\Lambda}{\varepsilon} + |\mathcal{J}^0| N - 1$. Let $M = |\mathcal{J}^0| N (\log_2 \frac{\Lambda}{\varepsilon} + 1)$, and Lemma 1 holds. □

LEMMA 2. $\lim_{q\to\infty}[f(\omega^{s_q}) - \beta^{s_q}] = 0$ *(consistency).*

Proof. Case (i): For some $Q < \infty$, $f(x)$ is continuous on C^{s_q} for all $q \geq Q$.

Recall from Section 2.3 that $\beta^{s_q} = g^{s_q}(\omega^{s_q})$, where g^{s_q} is the objective, and ω^{s_q} the solution of the relaxation of s_q. Consider the components f_j of f from which the respective components $g_j^{s_q}$ of g^{s_q} arise.

If for some $Q < \infty$, f_j becomes linear on $C_j^{s_Q}$, then $\forall q > Q$, f_j remains linear on $C_j^{s_q}$. By the nature of its underestimator $g_j^{s_q}$, $f_j(x_j) - g_j^{s_q}(x_j) = 0, \forall x_j \in C_j^{s_q}$, in particular for $\omega_j^{s_q}$.

If, on the other hand, f_j is continuous yet nonlinear on $C_j^{s_q}$ for all q, the argument resembles one in [81]. Since $j \in \mathcal{J}^{s_q}$, by Lemma 1, $\lim_{q\to\infty}(u_j^{s_q} - l_j^{s_q}) = 0$. Hence, as f_j is continuous on $C_j^{s_q}$, $\lim_{q\to\infty} \left| f_j(u_j^{s_q}) - f_j(l_j^{s_q}) \right| = 0$. Also, $\lim_{q\to\infty} \left| f_j(\omega_j^{s_q}) - f_j(u_j^{s_q}) \right| = 0$, since $\left| \omega_j^{s_q} - u_j^{s_q} \right| \leq \left| u_j^{s_q} - l_j^{s_q} \right|$.

Note that $\omega_j^{s_q}$ may be expressed as $\lambda l_j^{s_q} + (1-\lambda) u_j^{s_q}$, for some $0 \leq \lambda \leq 1$. As $g_j^{s_q}$ is linear, $g_j^{s_q}(\omega_j^{s_q})$ may be expressed as $\lambda g_j^{s_q}(l_j^{s_q}) + (1-\lambda) g_j^{s_q}(u_j^{s_q})$, which is the same as $\lambda f_j(l_j^{s_q}) + (1-\lambda) f_j(u_j^{s_q})$, owing to the way $g_j^{s_q}$ is constructed. Hence we find that $\left| f_j(u_j^{s_q}) - g_j^{s_q}(\omega_j^{s_q}) \right| = \lambda \left| f_j(u_j^{s_q}) - f_j(l_j^{s_q}) \right|$, which is at most $\left| f_j(u_j^{s_q}) - f_j(l_j^{s_q}) \right|$. From the triangle inequality,

$$\begin{aligned} f_j(\omega_j^{s_q}) - g_j^{s_q}(\omega_j^{s_q}) &\leq \left| f_j(\omega_j^{s_q}) - f_j(u_j^{s_q}) \right| + \left| f_j(u_j^{s_q}) - g_j^{s_q}(\omega_j^{s_q}) \right| \\ &\leq \left| f_j(\omega_j^{s_q}) - f_j(u_j^{s_q}) \right| + \left| f_j(u_j^{s_q}) - f_j(l_j^{s_q}) \right|. \end{aligned}$$

Therefore, $\lim_{q\to\infty}[f_j(\omega_j^{s_q}) - g_j^{s_q}(\omega_j^{s_q})] \leq 0$. The reverse inequality holds because, by design, $g_j^{s_q}$ underestimates f_j over $C_j^{s_q}$. Thus consistency is proved for the continuous case.

Case (ii): $f(x)$ is discontinuous on C^{s_q} for all q.

It is well known that a function concave on a closed set may have discontinuities only along the boundary of the set. Accordingly, f_j, which was assumed concave on $C_j^0 = [l_j^0, u_j^0]$, may be discontinuous only at l_j^0, at u_j^0, or at both. In the event that f_j is discontinuous at both l_j^0 and u_j^0, Lemma 1 will eventually ensure that $[l_j^{s_q}, u_j^{s_q}] \subset [l_j^0, u_j^0]$ in the proper sense, since a discontinuity implies that $j \in \mathcal{J}$.

Without loss of generality then, assume that if f_j has a discontinuity in $C_j^{s_q}$, it occurs at only one endpoint of the interval, say $l_j^{s_q}$. Let $\mathcal{D}$ be the set of indices of discontinuous variables. Then, for $j \in \mathcal{D}$, $l_j^{s_q} = l_j^0$, and $x_j \in]l_j^{s_q}, u_j^{s_q}[$ necessarily implies that $f_j(x_j) - g_j^{s_q}(x_j) > 0$.

As f_j is continuous relative to $]l_j^{s_q}, u_j^{s_q}]$, $f_j^+(l_j) = \lim_{x_j \to l_j^+} f_j(x_j)$ exists. Moreover, since $\forall j \in \mathcal{D}$, f_j is concave yet discontinuous at $l_j^{s_q}$, $\gamma_j^L := f_j^+(l_j^{s_q}) - f_j(l_j^{s_q}) > 0$. In the present case, $j \in \mathcal{D}$ implies that $j \in \mathcal{J}^{s_q}, \forall q$, so that, by Lemma 1, $\forall \delta > 0, \exists Q_1^j$ sufficiently large such that for $q > Q_1^j$, $u_j^{s_q} - l_j^0 < \delta$. Also, the extension defined by

$$\begin{cases} f_j(x_j), & \text{when } x_j \in]l_j^{s_q}, u_j^{s_q}] \\ f_j^+(l_j^{s_q}), & \text{when } x_j = l_j^{s_q} \end{cases}$$

is continuous relative to $[l_j^{s_q}, u_j^{s_q}]$, in particular at $l_j^{s_q}$. Hence, $\forall \varepsilon > 0, \exists \delta$ such that $u_j^{s_q} - l_j^0 < \delta$ implies $|f_j(u_j^{s_q}) - f_j^+(l_j^{s_q})| < \varepsilon$. Since $\gamma_j^L > 0$, we have that $\forall K$ satisfying $0 < K < \gamma_j^L, \exists Q_1^j$ such that $q > Q_1^j$ implies $|f_j(u_j^{s_q}) - f_j^+(l_j^{s_q})| < \gamma_j^L - K$.

Simultaneously, $q > Q_1^j$ also implies that $f_j(u_j^{s_q}) - f_j(l_j^{s_q}) > 0$, because $\gamma_j^L - K < \gamma_j^L = f_j^+(l_j^{s_q}) - f_j(l_j^{s_q})$ and so $q > Q_1^j$ ensures that $|f_j(u_j^{s_q}) - f_j^+(l_j^{s_q})| < f_j^+(l_j^{s_q}) - f_j(l_j^{s_q})$. By the triangle inequality,

$$\gamma_j^L \leq |f_j(u_j^{s_q}) - f_j^+(l_j^{s_q})| + [f_j(u_j^{s_q}) - f_j(l_j^{s_q})],$$

or

$$\begin{aligned} f_j(u_j^{s_q}) - f_j(l_j^{s_q}) &\geq \gamma_j^L - |f_j(u_j^{s_q}) - f_j^+(l_j^{s_q})| \\ &> \gamma_j^L - (\gamma_j^L - K) \\ &= K > 0. \end{aligned}$$

Also, by Lemma 1, $\forall K, M > 0, \exists Q_2^j$ sufficiently large such that for $q > Q_2^j$, $u_j^{s_q} - l_j^0 < K/M$. Therefore, $\forall M > 0, q > \max_{j \in \mathcal{D}}\{Q_1^j, Q_2^j\}$ implies

$$\frac{f_j(u_j^{s_q}) - f_j(l_j^{s_q})}{u_j^{s_q} - l_j^{s_q}} > M, \forall j \in \mathcal{D}.$$

Now, for all the continuous variables $x_j, j \in \mathcal{C} := \{1, ..., n\} \setminus \mathcal{D}$, f_j is continuous, bounded, and concave on the entirety of $[l_j^{s_q}, u_j^{s_q}]$. Via Theorem 24.1 of Rockafellar [61] and the Mean Value Theorem, we obtain

$$-\infty < f_{j-}'(u_j^{s_q}) \leq \frac{f_j(u_j^{s_q}) - f_j(l_j^{s_q})}{u_j^{s_q} - l_j^{s_q}} \leq f_{j+}'(l_j^{s_q}) < +\infty,$$

where f_{j-}', f_{j+}' are the left, right derivatives of f_j, respectively.

Finally, then, if we examine a feasible direction d of the LP relaxation of s_q, we find that d is an *ascent direction* if

$$cd = \sum_j \frac{f_j(u_j^{s_q}) - f_j(l_j^{s_q})}{(u_j^{s_q} - l_j^{s_q})} d_j > 0.$$

Let $B := \{x : x_j = l_j^0, \forall j \in \mathcal{D}\}$. Suppose that d is a direction going from a vertex x^1 of $D \cap C^{s_q}$ through a second vertex x^2 of $D \cap C^{s_q}$, where $x^1 \in D \cap C^{s_q} \cap B$ and $x^2 \in D \cap C^{s_q} \setminus B$. Thus we have $d_j \geq 0, \forall j \in \mathcal{D}$ with $d_j > 0$, for some $j \in \mathcal{D}$. On the other hand, $\forall j \in \mathcal{C}$, d_j may be positive or negative. We have already shown that for big enough q, c_j stays positive and can be made arbitrarily large for any $j \in \mathcal{D}$, while $j \in \mathcal{C}$, c_j may be negative, but remains bounded. Therefore, $\exists Q$ such that $q > Q$ implies that

$$\sum_{j \in \mathcal{D}} \frac{f_j(u_j^{s_q}) - f_j(l_j^{s_q})}{(u_j^{s_q} - l_j^{s_q})} d_j > - \sum_{j \in \mathcal{C}} \frac{f_j(u_j^{s_q}) - f_j(l_j^{s_q})}{(u_j^{s_q} - l_j^{s_q})} d_j,$$

or $cd > 0$.

As $C_j^{s_q} \searrow l_j^{s_q} = l_j^0, \forall j \in \mathcal{D}$, C^{s_q} always contains a point of B. Furthermore, such a point will always be in $D \cap C^{s_q}$. Were this not the case, then after some finite q, we would have $D \cap C^{s_q} = \emptyset$, and the algorithm would discard s_q, contrary to the assumption that s_q was part of an infinite sequence. Thus for all $q > Q$, only vertices x of $D \cap C^{s_q}$ satisfying $x_j = l_j^{s_q} = l_j^0, \forall j \in \mathcal{D}$ solve the LP relaxation of subproblem s_q.

Therefore, for $q > Q$, $f_j(\omega^{s_q}) - g_j^{s_q}(\omega^{s_q}) = 0, \forall j \in \mathcal{D}$, because the univariate relaxations $g_j^{s_q}$ are exact at the endpoints of $C_j^{s_q}$, in particular $l_j^{s_q}$. That $\lim_{q \to \infty} f_j(\omega^{s_q}) - g_j^{s_q}(\omega^{s_q}) = 0, \forall j \in \mathcal{C}$ follows by the argument of Case (i). □

LEMMA 3. *For $q = 1, 2, \ldots$, the corresponding subdomain $D \cap C^{s_q}$ contains a global minimum of SCP (convergence).*

Proof. By contradiction. If a global minimum $x^\star$ is not in $D \cap C^{s_q}$, then $f(\omega^{s_q})$ is strictly greater than $f(x^\star)$. Since there is a finite difference between $f(x^\star)$ and $f(\omega^{s_q})$, we may choose an index q sufficiently large so that, due to Lemma 2, β^{s_q} is made arbitrarily close to $f(\omega^{s_q})$, therefore strictly greater than $f(x^\star)$, and therefore strictly greater than any current lower bound L. Due to the least-lower-bound selection rule, the algorithm will not select s_q in the first place, which contradicts the assumption that s_q was part of an infinite sequence of nested subdomains. □

REMARK 1. By the above lemma, a subproblem s in the sequence has that C^s contains a global optimum, $x^\star$, of SCP. For any direction d^A from $x^\star$ that is feasible to $D \cap C^s$, consider the ray $x^\star + \kappa d^A, \kappa > 0$. The compactness and convexity of $D \cap C$ guarantees that a unique point x^A on this ray satisfies $(1 - \lambda)x^\star + \lambda x^A \in D \cap C, 0 \leq \lambda \leq 1$ and $(1 - \lambda)x^\star + \lambda x^A \notin D \cap C, \lambda > 1$.

LEMMA 4. *After finitely many iterations, any subproblem s in the sequence has the property that for each global optimizer $x^\star \in C^s$ and each direction d^A from $x^\star$ that is feasible to $D \cap C^s$, the corresponding x^A as described in Remark 1 satisfies the following trichotomy $\forall j \in \mathcal{J}^s$:*
Either $x_j^A > u_j^s$, or $x_j^A < l_j^s$, or $x_j^A = x_j^\star$.

Proof. Suppose that for subproblem t in the sequence, $l_j^t \leq x_j^A \leq u_j^t$, yet $x_j^A \neq x_j^\star$, for a certain $j \in \mathcal{J}^t$. After a finite number of additional iterations, the algorithm will generate a descendant s of t for which either $j \notin \mathcal{J}^s$ or by Lemma 1, j satisfies $u_j^s - l_j^s < \left| x_j^A - x_j^\star \right|$, as $\mathcal{J}^s \subseteq \mathcal{J}^t$. Hence, for all $j \in \mathcal{J}^s$ for which $x_j^A \neq x_j^\star$, either $x_j^A > u_j^s$ or $x_j^A < l_j^s$, and s satisfies the desired property. □

LEMMA 5. *Given a concave function $f : R \to R$ nonlinear on $[x, y]$, for some points $x, y \in R$ with $y > x$, and given $\lambda \in R$ with $0 < \lambda < 1$, then*

$$[f(\lambda x + (1-\lambda)y) - f(x)](y - x) > [f(y) - f(x)][(\lambda x + (1-\lambda)y) - x] \quad (1)$$

and

$$[f(y) - f(x)][y - (\lambda x + (1-\lambda)y)] > [f(y) - f(\lambda x + (1-\lambda)y)](y - x). \quad (2)$$

Proof. For a nonlinear concave function, $f(\lambda x + (1-\lambda)y) > \lambda f(x) + (1-\lambda)f(y)$. Hence, we may multiply $y > x$ by $f(\lambda x + (1-\lambda)y) - \lambda f(x) - (1-\lambda)f(y)$ to obtain the valid inequality

$$\begin{aligned} &[f(\lambda x + (1-\lambda)y) - \lambda f(x) - (1-\lambda)f(y)]y \ > \\ &[f(\lambda x + (1-\lambda)y) - \lambda f(x) - (1-\lambda)f(y)]x. \end{aligned} \quad (3)$$

By adding $(1-\lambda)[f(y) - f(x)]y + [f(x) - f(\lambda x + (1-\lambda)y)]x$ to both sides of (3) we obtain (1). By adding $[f(y) - f(\lambda x + (1-\lambda)y)]y + \lambda[f(x) - f(y)]x$ to both sides of (3) we obtain (2). □

LEMMA 6. *Given a subproblem s as described in Lemma 4, $x^\star \in C^s$ solves the LP relaxation of s if and only if $x^\star$ is a global optimizer of SCP. Moreover, any pair of global optimizers $x^\star, x^{\star\star} \in C^s$, have that $x_j^\star = x_j^{\star\star}, \forall j \in \mathcal{J}^s$.*

Proof. Let $x^\star \in C^s$ globally optimize SCP. The directions d^A from $x^\star$ feasible to $D \cap C^s$ fall into two classes based on the uniquely corresponding point x^A described in Remark 1. In order to think of $x^\star$ and x^A as the 'x' and 'y' of the Lemma 5, consider the smallest rectangular set containing both C^s and x^A, that is, $\prod_{j=1}^n [\min\{l_j^s, x_j^A\}, \max\{u_j^s, x_j^A\}]$. Define $\mathcal{J}^{sA}$ to be the set of indices of variables nonlinear on this set. The first class of directions are those for which for which $\exists j \in \mathcal{J}^{sA}$ such that $x_j^A \neq x_j^\star$. The remaining directions, for which $x_j^A = x_j^\star, \forall j \in \mathcal{J}^{sA}$, fall into the second class.

Consider any direction d^A of the first class. As $x^\star \in C^s$ and $\exists j \in \mathcal{J}^{sA}$ such that $x_j^A \neq x_j^\star$, the trichotomy of Lemma 4 ensures that $x^A \notin C^s$. Also, any point $x^{\star\star} = (1-\lambda)x^\star + \lambda x^A, 0 < \lambda < 1$ cannot be a global optimizer, as

$$\begin{aligned} f[(1-\lambda)x^\star + \lambda x^A] &> (1-\lambda)f(x^\star) + \lambda f(x^A) \\ &\geq \min\{f(x^\star), f(x^A)\} = f(x^\star). \end{aligned}$$

In particular, $x^{\star\star} \in C^s$ implies that $x^{\star\star}$ is not a global optimizer. We show that the direction d^A, given as $d^A := x^A - x^\star$, is a direction of ascent from $x^\star$ for the LP.

As d^A is feasible to $D \cap C^s$, $u_j^s > l_j^s$ for all j satisfying $x_j^A \neq x_j^\star$, $x_j^\star < u_j^s$ for all j satisfying $x_j^A > u_j^s$ and $x_j^\star > l_j^s$ for all j satisfying $x_j^A < l_j^s$.

For an index $j \in \mathcal{J}^{sA}$ satisfying $x_j^A > u_j^s$, apply Lemma 5 to the relationships $x_j^A > u_j^s > l_j^s$ and $x_j^A > x_j^\star > l_j^s$ to obtain, respectively

$$[f_j(u_j^s) - f_j(l_j^s)](x_j^A - l_j^s) > [f_j(x_j^A) - f_j(l_j^s)](u_j^s - l_j^s) \tag{4}$$

and

$$[f_j(x_j^A) - f_j(l_j^s)](x_j^A - x_j^\star) > [f_j(x_j^A) - f_j(x_j^\star)](x_j^A - l_j^s). \tag{5}$$

Then multiply (4) by $x_j^A - x_j^\star$ and multiply (5) by $u_j^s - l_j^s$ to obtain, respectively

$$\begin{aligned} &[f_j(u_j^s) - f_j(l_j^s)](x_j^A - l_j^s)(x_j^A - x_j^\star) \\ &\quad > [f_j(x_j^A) - f_j(l_j^s)](u_j^s - l_j^s)(x_j^A - x_j^\star) \end{aligned} \tag{6}$$

and

$$\begin{aligned} &[f_j(x_j^A) - f_j(l_j^s)](x_j^A - x_j^\star)(u_j^s - l_j^s) \\ &\quad > [f_j(x_j^A) - f_j(x_j^\star)](x_j^A - l_j^s)(u_j^s - l_j^s). \end{aligned} \tag{7}$$

From (6) and (7), we obtain

$$\begin{aligned} &[f_j(u_j^s) - f_j(l_j^s)](x_j^A - l_j^s)(x_j^A - x_j^\star) \\ &\quad > [f_j(x_j^A) - f_j(x_j^\star)](x_j^A - l_j^s)(u_j^s - l_j^s) \end{aligned} \tag{8}$$

and since $x_j^A > l_j^s$, we find that

$$[f_j(u_j^s) - f_j(l_j^s)](x_j^A - x_j^\star) > [f_j(x_j^A) - f_j(x_j^\star)](u_j^s - l_j^s). \tag{9}$$

Note that if $x_j^\star = l_j^s$, then (4) is directly equivalent to (9) and (5) is not required.

Similarly, for an index $j \in \mathcal{J}^{sA}$ satisfying $x_j^A < l_j^s$, apply Lemma 5 to the relationships $x_j^A < l_j^s < u_j^s$ and $x_j^A < x_j^\star < u_j^s$ to obtain, respectively

$$[f_j(u_j^s) - f_j(x_j^A)](u_j^s - l_j^s) > [f_j(u_j^s) - f_j(l_j^s)](u_j^s - x_j^A) \tag{10}$$

and

$$[f_j(x_j^\star) - f_j(x_j^A)](u_j^s - x_j^A) > [f_j(u_j^s) - f_j(x_j^A)](x_j^\star - x_j^A). \tag{11}$$

Then multiply (10) by $x_j^\star - x_j^A$ and multiply (11) by $u_j^s - l_j^s$ to obtain, respectively,

$$\begin{aligned}&[f_j(u_j^s) - f_j(x_j^A)](u_j^s - l_j^s)(x_j^\star - x_j^A)\\ &\quad > [f_j(u_j^s) - f_j(l_j^s)](u_j^s - x_j^A)(x_j^\star - x_j^A)\end{aligned} \tag{12}$$

and

$$\begin{aligned}&[f_j(x_j^\star) - f_j(x_j^A)](u_j^s - x_j^A)(u_j^s - l_j^s)\\ &\quad > [f_j(u_j^s) - f_j(x_j^A)](x_j^\star - x_j^A)(u_j^s - l_j^s).\end{aligned} \tag{13}$$

From (12) and (13), we obtain

$$\begin{aligned}&[f_j(x_j^\star) - f_j(x_j^A)](u_j^s - x_j^A)(u_j^s - l_j^s)\\ &> [f_j(u_j^s) - f_j(l_j^s)](u_j^s - x_j^A)(x_j^\star - x_j^A)\end{aligned} \tag{14}$$

and since $u_j^s > x_j^A$, we find that

$$[f_j(x_j^\star) - f_j(x_j^A)](u_j^s - l_j^s) > [f_j(u_j^s) - f_j(l_j^s)](x_j^\star - x_j^A) \tag{15}$$

which is equivalent to (9). Note that if $x_j^\star = u_j^s$, then (10) is directly equivalent to (15) and (11) is not required.

Of course, for the linear variables $j \notin \mathcal{J}^{sA}$,

$$[f_j(u_j^s) - f_j(l_j^s)](x_j^A - x_j^\star) = [f_j(x_j^A) - f_j(x_j^\star)](u_j^s - l_j^s) \tag{16}$$

From the optimality of $x^\star$ to SCP,

$$\sum_j [f_j(x_j^A) - f_j(x_j^\star)] \geq 0.$$

Let $\mathcal{F} := \{j : x_j^A \neq x_j^\star\}$. Then we may write

$$\begin{aligned}&\sum_{j\in\mathcal{F}} [f_j(x_j^A) - f_j(x_j^\star)] \geq 0, \text{ so that}\\ &\sum_{j\in\mathcal{F}} [f_j(x_j^A) - f_j(x_j^\star)] \prod_{h\in\mathcal{F}} (u_h^s - l_h^s) \geq 0, \text{ or}\\ &\sum_{j\in\mathcal{F}} [f_j(x_j^A) - f_j(x_j^\star)](u_j^s - l_j^s) \prod_{h\in\mathcal{F}\setminus\{j\}} (u_h^s - l_h^s) \geq 0.\end{aligned} \tag{17}$$

Overestimating some terms of (17) with (9) and using (16), we then obtain

$$\sum_{j\in\mathcal{F}} [f_j(u_j^s) - f_j(l_j^s)](x_j^A - x_j^\star) \prod_{h\in\mathcal{F}\setminus\{j\}} (u_h^s - l_h^s) > 0$$

or

$$\sum_{j\in\mathcal{F}} \frac{f_j(u_j^s) - f_j(l_j^s)}{(u_j^s - l_j^s)}(x_j^A - x_j^\star) = \sum_j c_j d_j^A > 0.$$

Hence, d^A is a direction of ascent from $x^\star$.

Next consider any direction of the second class ($x_j^A = x_j^\star, \forall j \in \mathcal{J}^{sA}$), and observe that $f[(1-\lambda)x^\star + \lambda x^A] = (1-\lambda)f(x^\star) + \lambda f(x^A)$. Hence, a point $x^{\star\star} = (1-\lambda)x^\star + \lambda x^A, 0 < \lambda < 1$, is a global optimizer if and only if x^A is a global optimizer.

For linear variables x_j, $f_j(x_j)$ must be of the form $\alpha_j x_j$, with the coefficients $\alpha_j \in R$. As $\mathcal{J}^s \subseteq \mathcal{J}^{sA}$, for $j \notin \mathcal{J}^s$, $f_j(x_j)$ takes this form, and as g^{s_k} is also linear, so $g_j^{s_k}(x_j) \equiv f_j(x_j) = \alpha_j x_j, \forall j \notin \mathcal{J}^s$. This shows that for the LP objective c,

$$c_j = \begin{cases} 0, & j \in \mathcal{J}^s \\ \alpha_j, & j \notin \mathcal{J}^s \end{cases},$$

and

$$\begin{aligned} cd^A &= \sum_{j\notin\mathcal{J}} \alpha_j(x_j^A - x_j^\star) = \sum_{j\notin\mathcal{J}} (\alpha_j x_j^A - \alpha_j x_j^\star) \\ &= \sum_{j\notin\mathcal{J}} [f_j(x_j^A) - f_j(x_j^\star)] = f(x^A) - f(x^\star). \end{aligned}$$

Hence, if x^A is not a global optimizer, then $f(x^A) - f(x^\star) > 0$ and d^A increases the LP objective, but if x^A is a global optimizer, then $f(x^A) - f(x^\star) = 0$, so that d^A does not change the LP objective.

Therefore, any feasible direction d^A from $x^\star$ either (1) increases the LP objective, or (2) does not change the LP objective. Consider a point $x^{\star\star} = x^\star + \kappa d^A$, $\kappa > 0$. In the former case, if $x^{\star\star} \in C^s$ then $x^{\star\star}$ is not a global optimizer. In the latter case, any feasible $x^{\star\star}$

is a global optimizer. □

THEOREM 1. *The algorithm terminates finitely with a global minimum of SCP.*

Proof. Given a subproblem as described by Lemma 4, it follows from Lemma 6 that the algorithm identifies a global minimum, $x^\star$, which becomes the best solution currently known ($\tilde{x} \leftarrow x^\star$). As the trichotomy of Lemma 4 is satisfied in finitely many iterations, a global minimum is thus identified finitely. In accordance with the branching rule, s will be partitioned at $\tilde{x}$ unless the level of s is a multiple of N. In the latter case, at least one of the children of s inherits the above properties, and we consider it to be s, instead. Thus, the procedure splits s into two subproblems by introducing a partition through $\tilde{x} = x^\star$ and branching on an edge corresponding

to some nonlinear variable. In the event that s contains several global minimizers, Lemma 6 shows that the coordinates of such points differ only in the linear variables, hence the constructed partition actually passes through all global minimizers in s and the relaxation gap at all such points is simultaneously reduced.

Hence, each of the resulting partition elements also satisfies Lemmas 4 and 6. Unless the level of a subproblem is a multiple of N, the procedure branches through the incumbent, $\tilde{x}$, whenever possible. From s, the branching process thus continues at most $\lceil \frac{N}{N-1} |\mathcal{J}| \rceil$ times until some global minimizer, $x^\star$, say, is rendered gapless. If a descendant of s contains a global minimum, then a global minimum solves its LP relaxation. In other words, a descendant of s at most $\lceil \frac{N}{N-1} |\mathcal{J}| \rceil$ levels below s will have a lower bound *exactly* equal to its upper bound and will be fathomed without any further partitioning.

The same argument applies to a path of the branch-and-bound tree containing any other global minimum.

Finally, consider any inferior paths of the branch-and-bound tree. There is a finite difference between the optimum of an inferior path and the global optimum $f(x^\star)$. Consequently, for a subproblem sufficiently far down the inferior path, the lower bound will be made arbitrarily close to the optimum, therefore strictly greater than $f(x^\star) = U = L$. As the global minimum will have been found already, the fathoming rule (of Step k.2c) will then eliminate all inferior paths of the branch-and-bound tree from further consideration by the algorithm. □

4. Discussion of Related Algorithms

The proof of Theorem 1 makes it clear that a branch-and-bound algorithm based on rectangular partitions and linear underestimation terminates finitely for SCP if the following two conditions are met.

CONDITION 1. For all nested sequences $\{C^{s_q}\}$ of subdomains C^{s_q} generated by the algorithm $\lim_{q\to\infty} \max_{j\in\mathcal{J}}(u_j^{s_q} - l_j^{s_q}) = 0$.

CONDITION 2. If a subproblem contains a global solution point, the algorithm ensures that in a finite number of iterations the gap between the lower bound on the subproblem and its optimum will be reduced to zero. For the type of underestimators mentioned herein, this condition can be satisfied by constructing a partition of the subproblem through the said point.

Condition (1) guarantees that eventually a bounds box containing a global solution will satisfy Lemma 4 and therefore the corresponding LP subproblem will provide the global minimizer as its solution (Lemma 6). Condition (2) then ensures that branching will reduce the underestimation gap at the global solution point, eventually rendering the underestimator gapless. At that stage, the subproblem will be fathomed and U, L will be set to $f(x^\star)$. All inferior subproblems are eventually

fathomed, as happens in any convergent branch-and-bound algorithm. Conditions (1) and (2) in hand, we can design finite branch-and-bound algorithms for SCP.

For example, Falk and Soland give a 'relaxed' algorithm (FSR) which they prove to be convergent. Falk and Soland also claim the algorithm to be finite for SCP, although its finiteness is brought into question by Horst and Tuy ([33, p. 362]). While the finiteness of FSR remains an open question, a slightly modified version of the FSR algorithm can be proven finite by the same theorem offered above to prove finiteness of our algorithm. The existing branching rule of FSR is an ω-*partitioning*:

OPERATION 5. ω-PARTITIONING.

Variable Selection
$j' \in \text{argmax}_{j\in\mathcal{J}}[f_j(\omega_j^{s_k}) - g_j^{s_k}(\omega_j^{s_k})]$
(a nonlinear variable with largest underestimation gap).
Point Selection
$p = \omega^{s_k}$
(solution of relaxed problem).
Split C^{s_k} into two subdomains:
$[l_{j'}^{s_k}, p_{j'}^{s_k}] \prod_{j\neq j'}[l_j^{s_k}, u_j^{s_k}]$ and $[p_{j'}^{s_k}, u_{j'}^{s_k}] \prod_{j\neq j'}[l_j^{s_k}, u_j^{s_k}]$.

The rule meets Condition (2), (provided that $x^\star = \omega^{s_q}$ for some $q < \infty$), yet, it fails to meet Condition (1). The following modification, however, meets both conditions:

OPERATION 6. MODIFIED ω-PARTITIONING.

Given a positive integer N.
if $\mathcal{L}(s_k)$ mod N=0 **then**
$j' \in \text{argmax}_{j\in\mathcal{J}}(u_j^{s_k} - l_j^{s_k})$
(the variable corresponding to a longest edge of C^{s_k}).
$p = (u_{j'}^{s_k} - l_{j'}^{s_k})/2$
else
Select j' and p as in the unmodified rule.
endif
Split C^{s_k} at p, j'.

By modifying the ω-branching rule to bisect a longest edge of the selected subdomain every Nth level of the tree, Condition (1) is met. The finiteness of the modified algorithm follows, since FSR performs bounding and subproblem selection in the same manner as the algorithm proposed in Section 2.

Kalantari and Rosen specialize their algorithm [36] for quadratic concave programs that can be stated as min $\sum_{j=1}^n c_j x_j - \frac{1}{2}\lambda_j x_j^2$ s.t. $x \in D \cap C$. The algorithm is proven convergent. Therefore, it terminates finitely to an ε-*approximate solution*.

(An ε-approximate solution is a solution, call it x^ε, that satisfies $|f(x^\varepsilon) - f(x^\star)| \leq \varepsilon$, where $x^\star$ is a global optimizer and ε is a prespecified tolerance). The algorithm uses the following fathoming rule (Here, $\varepsilon > 0$ is required for finite termination; $\mathcal{S}$ is the list of currently open subproblems; U is the current least upper bound; and β^s indicates the lower bound of subproblem s):

OPERATION 7. ε-TOLERANT FATHOMING RULE.

$\mathcal{S} \leftarrow \mathcal{S} \setminus \{s \text{ s.t. } \beta^s > U - \varepsilon\}$.

This algorithm also employs the following specialized branching rule:

OPERATION 8. KALANTARI–ROSEN PARTITIONING.

Variable Selection
$j' \in \text{argmax}_{j \in \mathcal{J}} \lambda_j (u_j^{s_k} - l_j^{s_k})^2$.
Point Selection
Let $p = (u_{j'}^{s_k} - l_{j'}^{s_k})/2$.
Split C^{s_k} at p,j'.

This rule meets Condition (1) but not Condition (2). To render this algorithm finite without recourse to an ε-tolerance, one can modify the branching and fathoming rules as follows (Recall that $\tilde{x}$ indicates the best known solution in the current stage k of the procedure):

OPERATION 9. MODIFIED K–R PARTITIONING.

Variable Selection
$j' \in \text{argmax}_{j \in \mathcal{J}} \lambda_j (u_j^{s_k} - l_j^{s_k})^2$
(First choose j' according to the existing rule).
Point Selection
If $\tilde{x} \in C^{s_k}$ and $\tilde{x}_{j'} \in (l_{j'}^{s_k}, u_{j'}^{s_k})$ **then**
Choose $p = \tilde{x}_{j'}$
else
Let $p = (u_{j'}^{s_k} - l_{j'}^{s_k})/2$
(choose p according to the existing rule).
endif
Split C^{s_k} at p, j'.

OPERATION 10. MODIFIED FATHOMING RULE.

$\mathcal{S} \leftarrow \mathcal{S} \setminus \{s \text{ s.t. } \beta^s \geq U\}$.

The change in branching rule enables the algorithm to begin fathoming a subdomain which contains a global optimizer as soon as that point is known from

bounding. Moreover, the change in fathoming rule precludes the said subproblem from being prematurely fathomed. In tandem, the modified fathoming and partitioning rules enable this algorithm to converge finitely to a global minimum. The parallel algorithm of Phillips and Rosen [54] can be similarly modified to ensure finite convergence to a global minimum.

A discussion of procedures for SCP could hardly be complete without reference to the algorithm of Soland [74]. Soland's algorithm attains finiteness not by its branching strategy, which is the same ω-subdivision employed in FSR, but by means of a different bounding strategy. In formulating the LP relaxation of a given subproblem s_k, the procedure first constructs the linear underestimator $g^{s_k}(x)$ in a fashion identical to FSR and the proposed algorithm (see Bounding, Section 2.3). Soland's algorithm then determines the lower bound β^{s_k} by minimizing $g^{s_k}(x)$ over the original set of bounds $D \cap C$, rather than $D \cap C^{s_k}$. Naturally, this produces lower bounds that are weaker than the ones in the bounding procedure used here.

The proposed algorithm relies on the same subproblem for lower bounding and locating the optimum. In more general terms, the algorithm employs a simple relaxation that gradually approximates the objective, rather than requiring the gradient of the objective in explicit form. In contrast, finiteness results of Al-Khayyal and Kyparisis ([1]) require the objective function to be differentiable, and rely on the solution of additional auxiliary problems that involve the gradient of the objective. Moreover, the proof of finiteness for the proposed algorithm rests exclusively on the geometric properties of the problem domain without use of its analytic representation. As a consequence, the result is more readily applicable than that in [1], which requires the analytic specification of the polyhedron to be nondegenerate at the solution point.

5. Acceleration Devices

5.1. Use of Acceleration Devices

This section describes techniques which are not required for finiteness, nor even convergence of the algorithm. However, we have found that their incorporation in the algorithm is necessary to ensure termination in reasonable computing times. The techniques aim at reducing the domains of the problem variables. As a result of shrinking these regions, the LP bounds developed over the smaller regions are tighter and the size of the branch-and-bound tree smaller. Techniques similar to some of the ones given below have long been used in integer programming, *e.g.*, [75]. For concave programming and other continuous global optimization, similar techniques have been used by Thakur [76], Hansen, *et al.* [28], Sahinidis [67], Lamar [38], Ryoo and Sahinidis [66], [65], and Sherali and Tuncbilek [73]. For a comparative survey of this literature see [66], [65].

The acceleration devices given in Section 5.2 employ linear parametric programming methods, that is, extensive dual information obtained through exchange of basis in the relaxed problem. Section 5.3 deals with acceleration devices based

on sensitivity analysis only, that is, dual multipliers that are obtained directly from the solution of the relaxed problem, without basis exchange. Finally, Section 5.4 provides techniques that do not require dual information at all.

Consider the LP relaxation $\min g^s(x)$ s.t. $x \in D \cap C^s$ of concave subproblem s. Let ω^s be a point that solves this relaxation, let $\beta^s := g^s(\omega^s)$, and let U be a known upper bound on the global solution.

5.2. Acceleration Devices Employing Linear Parametric Programming

TECHNIQUE 1. Generating valid inequalities by using linear parametric programming to change the left-hand sides of existing constraints:

Widely used methods, discussed in, *e.g.*, [24], track the optimum of an LP when a particular constraint $\sum_{j=1}^{n} a_{ij}x_j \leq b_i$ is perturbed from its current value $\sum_{j=1}^{n} a_{ij}\omega_j^s$ by a quantity μ_i. For an LP minimization problem, this parametric function $\beta^\pi(\mu_i)$ is well known to be convex. As μ_i is decreased from zero, let l_i^π denote the value of $(\sum_{j=1}^{n} a_{ij}\omega_j^s) + \mu_i$ for which $\beta^\pi(\mu_i) = U$ or the perturbed LP becomes infeasible, whichever of the two is greater. Similarly, as μ_i is increased from zero, let u_i^π denote the value of $(\sum_{j=1}^{n} a_{ij}\omega_j^s) + \mu_i$ for which $\beta^\pi(\mu_i) = U$ or the perturbed LP becomes infeasible, whichever of the two is lesser. Since $\beta^\pi(\mu_i)$ is convex, it must be greater than U for any feasible values of $(\sum_{j=1}^{n} a_{ij}\omega_j^s) + \mu_i$ less than l_i^π or greater than u_i^π, if such values exist. Hence, the inequalities

$$\sum_{j=1}^{n} a_{ij}x_j \geq l_i^\pi$$

and

$$\sum_{j=1}^{n} a_{ij}x_j \leq u_i^\pi$$

are valid for s.

TECHNIQUE 2. Tightening bounds by using linear parametric programming to change the values of variables:

Consider a variable x_j, to be perturbed from its current value ω_j^s by λ_j. As λ_j is decreased from zero, let l_j^π denote the value of $\omega_j^s + \lambda_j$ for which $\beta^\pi(\lambda_j) = U$ or the perturbed LP becomes infeasible, whichever of the two is greater. Similarly, as λ_j is increased from zero, let u_j^π denote the value of $\omega_j^s + \lambda_j$ for which $\beta^\pi(\lambda_j) = U$ or the perturbed LP becomes infeasible, whichever of the two is lesser. Since $\beta^\pi(\lambda_i)$ is convex, it must be greater than U for any feasible values of $\omega_j^s + \lambda_i$ less than l_j^π or greater than u_j^π, if such exist. Hence, the constraints

$$x_j \geq l_j^\pi$$

and

$$x_j \leq u_j^\pi$$

are valid for s.

5.3. ACCELERATION DEVICES BASED ON SENSITIVITY ANALYSIS ONLY

The domain reduction techniques in this subsection are special cases of those developed by Ryoo and Sahinidis in [66], [65].

5.3.1. *Domain Reduction by Use of Available Marginal Values*

TECHNIQUE 3. Generating mirror inequalities by using sensitivity analysis with respect to b_i:

Consider a linear constraint $\sum_{j=1}^{n} a_{ij}x_j \leq b_i$ that is active at ω^s with a dual multiplier value of $\mu_i^s < 0$, and consider further, perturbing its current right-hand b_i by a quantity μ_i. For all values of μ_i for which the perturbed LP remains feasible, the affine function $\mu_i^s\mu_i + \beta^s$ is less than or equal to the parametric function $\beta^\pi(\mu_i)$, since the latter is convex. As μ_i is decreased from zero, let l_i^σ denote the value of μ_i for which $\mu_i^s\mu_i + \beta^s = U$. It follows that $\beta^\pi(\mu_i)$ must be greater than U for any feasible values of $b_i + \mu_i$ less than l_i^σ, if such exist. Hence, the mirror inequality

$$\sum_{j=1}^{n} a_{ij}x_j \geq l_i^\sigma = b_i + \frac{U - \beta^s}{\mu_i^s}$$

is valid for s.

TECHNIQUE 4. Tightening bounds by using sensitivity analysis with respect to l_j^s or u_j^s:

Consider a domain-bound $x_j \leq u_j^s$ that is active at ω^s with a dual multiplier value of $\lambda_j^s < 0$, and consider further, perturbing the bounding value u_j^s by λ_j. For all values of λ_j for which the perturbed LP remains feasible, the affine function $\lambda_j^s\lambda_j + \beta^s$ is less than or equal to the parametric function $\beta^\pi(\lambda_j)$, since the latter is convex. As λ_j is decreased from zero, let l_j^σ denote the value of λ_j for which $\lambda_j^s\lambda_j + \beta^s = U$. It follows that $\beta^\pi(\lambda_j)$ must be greater than U for any feasible values of $u_j^s + \mu_i$ less than l_j^σ, if such exist. Hence, the constraint

$$x_j \geq l_j^\sigma = u_j^s + \frac{U - \beta^s}{\lambda_j^s}$$

is valid for C^s. Similarly, consider a domain-bound $x_j \geq l_j^s$ that is active at ω^s with a dual multiplier value of $\lambda_j > 0$, and consider further, perturbing the bounding value l_j^s by λ_j. For all values of λ_j for which the perturbed LP remains feasible, the affine function $\lambda_j^s\lambda_j + \beta^s$ is less than or equal to the parametric function $\beta^\pi(\lambda_j)$, since the latter is convex. As λ_j is increased from zero, let u_j^σ denote the value of λ_j for which $\lambda_j^s\lambda_j + \beta^s = U$. It follows that $\beta^\pi(\lambda_j)$ must be greater than U for any feasible values of $l_j^s + \mu_i$ greater than u_j^σ, if such exist. Hence, the constraint

$$x_j \leq u_j^\sigma = l_j^s + \frac{U - \beta^s}{\lambda_j^s}$$

is valid for s.

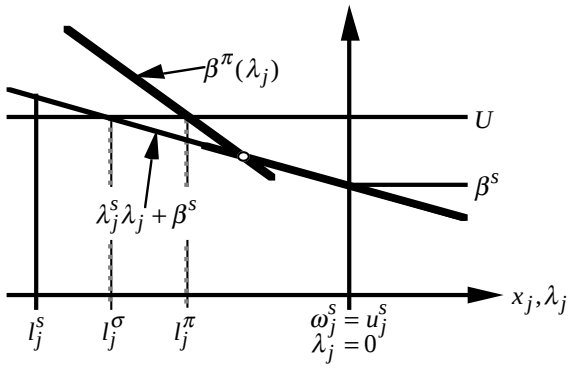

Figure 1. Comparison of Techniques 2 and 4 for improving a lower bound.

5.3.2. *Domain Reduction Using Probing to Induce Marginal Values*

TECHNIQUE 5. Generating mirror inequalities by probing the slack domain of constraint i:

Consider a linear constraint $\sum_{j=1}^n a_{ij}x_j \le b_i$ that is inactive at ω^s. Solve the partially restricted relaxed problem r, defined as $\min g^s(x)$ s.t. $x \in D \cap C^s \cap \{\sum_{j=1}^n a_{ij}x_j \ge b_i\}$, to obtain a solution point ω^r of value $\beta^r = g^s(\omega^r)$. If the constraint has a dual multiplier value of $\mu_i^r > 0$ in the solution of r, apply Technique 3 to the added constraint of problem r, to find that the mirror inequality

$$\sum_{j=1}^n a_{ij}x_j \ge l_i^\sigma = b_i - \frac{U - g^s(\omega^r)}{\mu_i^r}$$

is valid for s.

TECHNIQUE 6. Tightening bounds by probing the existing domain of variable x_j:

Consider a domain-bound $x_j \le u_j^s$ that is inactive at ω^s. Solve the partially restricted relaxed problem r, defined as $\min g^s(x)$ s.t. $x \in D \cap C^s \cap \{x_j \ge u_j^s\}$, to obtain a solution point ω^r of value $\beta^r = g^s(\omega^r)$. If the bound has a dual multiplier value of $\lambda_j^r > 0$ in the solution of r, apply Technique 4 to the added constraint of problem r, to find that

$$x_j \le u_j^\sigma = u_j^s - \frac{U - g^s(\omega^r)}{\lambda_j^r}$$

is valid for s. Similarly, consider a domain-bound $x_j \ge l_j^s$ that is inactive at ω^s. Solve the partially restricted relaxed problem r, defined as $\min g^s(x)$ s.t. $x \in D \cap C^s \cap \{x_j \le l_j^s\}$, to obtain a solution point ω^r of value $\beta^r = g^s(\omega^r)$. If the bound has a dual multiplier value of $\lambda_j^s < 0$ in the solution of r, apply Technique 4 to the added constraint of problem r, to find that

$$x_j \ge l_j^\sigma = l_j^s - \frac{U - g^s(\omega^r)}{\lambda_j^r}$$

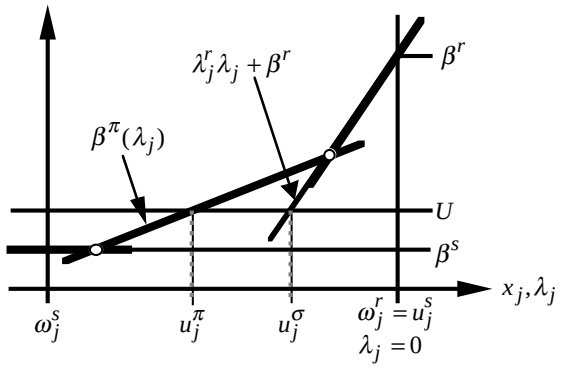

Figure 2. Comparison of Techniques 2 and 6 for improving an upper bound.

is valid for s.

5.4. Acceleration Devices That Do Not Require Dual Information

5.4.1. *Optimality-Based Tightening*

Optimality-based tightening uses the current upper and lower bounds on the global solution (U and L, respectively,) to generate constraints that may trim-off inferior portions of C^s.

TECHNIQUE 7. OBT

Use the least upper bound.

Compute $\Delta_h^U = U - \sum_{j \neq h} \min f_j(u_j^s), f_j(l_j^s)$.

CASE (A). There exists a point $x_h^{(A)}$ such that $\Delta_h^U = f_h(x_h^A)$ and f_h is decreasing at $x_h^{(A)}$.

CASE (B). There exists a point $x_h^{(B)}$ such that $\Delta_h^U = f_h(x_h^B)$ and f_h is increasing at $x_h^{(B)}$.

If CASE (A)$\wedge\neg$CASE (B) **then**

$x_h \geq x_h^{(A)}$

is valid for s.

If CASE (B)$\wedge\neg$CASE (A) **then**

$x_h \leq x_h^{(B)}$

is a valid for s.

If CASE (B)$\wedge$CASE (A) **then**

$x_h \leq x_h^{(B)} \vee x_h \geq x_h^{(A)}$

is valid for s.

Use the least lower bound.

Compute $\Delta_h^L = L - \sum_{j \neq h} \max_{[l_j^s, u_j^s]} f_j(x_j)$.

CASE (C). There exists a point $x_h^{(C)}$ such that $\Delta_h^L = f_h(x_h^C)$ and f_h is decreasing at $x_h^{(C)}$.

CASE (D). There exists a point $x_h^{(D)}$ such that $\Delta_h^L = f_h(x_h^D)$ and f_h is increasing at $x_h^{(D)}$.

If CASE (C) **then**
$x_h \leq x_h^{(C)}$
is valid for s.
If CASE (D) **then**
$x_h \geq x_h^{(D)}$
is a valid for s.

Note that the inverse f^{-1} of a concave function $f : R \to R$ is itself not necessarily a function. For example, consider the case where the relation f_h^{-1} is one-to-two and let f_h^{-1U} and f_h^{-1L} denote the upper and lower forks of f_h^{-1}, respectively. In this case $[l_h^s, u_h^s]$ can be pared-down to $[l_h^s, u_h^s] \cap \{[f_h^{-1L}(\Delta_h^L), f_h^{-1L}(\Delta_h^U)] \cup [f_h^{-1U}(\Delta_h^U), f_h^{-1U}(\Delta_h^L)]\}$, which may be disjoint.

5.4.2. *Feasibility-Based Tightening*

Feasibility-based tightening generates constraints that cut-off infeasible portions of the solution space.

TECHNIQUE 8. FBT

Consider the constraints $\sum_{j=1}^n a_{ij}x_j \leq b_i$, $i = 1, ..., m$. Then one of the constraints

$$\begin{cases} x_h \leq \dfrac{1}{a_{ih}} \left(b_i - \displaystyle\sum_{j \neq h} \min\{a_{ij}u_j^s, a_{ij}l_j^s\} \right), & a_{ih} > 0 \\ x_h \geq \dfrac{1}{a_{ih}} \left(b_i - \displaystyle\sum_{j \neq h} \max\{a_{ij}u_j^s, a_{ij}l_j^s\} \right), & a_{ih} < 0 \end{cases} \tag{18}$$

is also valid for each pair (i, j) that satisfies $a_{ij} \neq 0$.

Of course, to tighten variable bounds at subproblem s, one could simply solve the $2n$ LPs

$$\{\min \pm x_j \text{ s.t. } x \in D \cap C^s\}, \tag{19}$$

which would provide tightening that is optimal, albeit computationally expensive. In this regard, the former cuts (18) function as 'poor man's linear programs,' particularly when they are applied iteratively, looping over the set of variables several times.

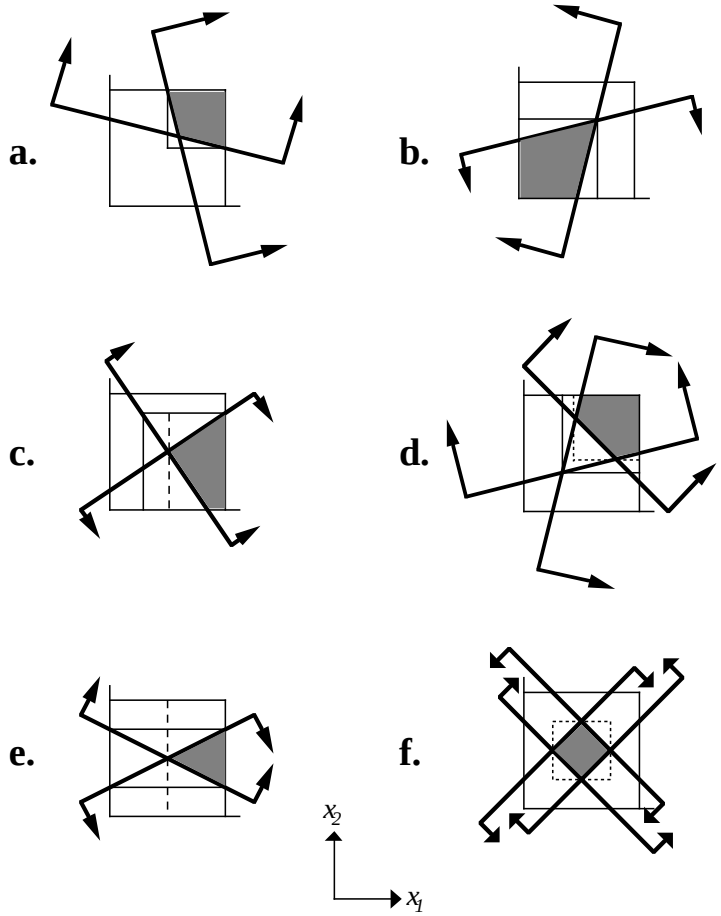

Figure 3. Poor Man's LPs.

Figure 3 shows how an implementation of (18) compares to the solution of the LPs for different two-dimensional examples. In each instance, the outer box represents the bound set before tightening begins, with constraints shown in bold lines and the feasible region shaded. Bounds improved by (18) are shown in solid line; improvements by (19) are shown in dashed line, when they differ from those of (18).

In Figures 3a and 3b, techniques (18) and (19) give the same result. In Figure 3c, the effects of (18) agree with the effect of (19) for variable x_2, while also improving the bounds on x_1, albeit not to the maximum possible extent. In Figure 3d, (18) improves bounds on both variables, although neither bound is improved to the maximum possible extent. Figure 3e shows the bounds on x_2 tightened to their full extent by (18), but here the heuristic fails to improve bounds on x_1, at all. Figure 3f is particularly insightful as a pathological case for the heuristic. In the latter case, the bounds are not improved at all, whereas a great deal of bounds reduction is possible, illustrated by the four LP solutions (dashed lines).

One can see why Figure 3f is pathological for the 'poor man's linear programs.' The heuristic can make use of only one bound and one constraint at a time, while linear programming considers the entire constraint set simultaneously. In practice, a case such as Figure 3f would not occur if all of the $2n$ LPs are solved initially, in preprocessing, on a one-time basis (see Section 2.2, also Sketch, below). Thereafter, for each subdomain C^s, at least one bound acts as a non-redundant constraint of $D \cap C^s$, *i.e.*, $\partial(D \cap C^s) \setminus \partial D \neq \emptyset$. Finally, note that sometimes the heuristic achieves its maximum domain reduction asymptotically, *e.g.*, Figure 3b and Figure 3d, where improved bounds on variable x_1 then enable bound improvement on variable x_2 that, in turn, facilitate further tightening of x_1 on the next pass, etc.

The following sketch illustrates how the Algorithm employs the various tightening techniques given above (see Section 2 for portions omitted here).

ALGORITHM 2. (SKETCH) (Branch-and-Reduce Algorithm).

Initialization Preprocess the problem as detailed in Section 2.2: Solve LPs to bound unbounded variables and (optionally) to tighten existing bounds.
Using feasible points found in Preprocessing, determine an initial value of U and **apply Techniques 7 and 8 to the original domain $D \cap C$.**

$\vdots$

Let $k \longleftarrow 0$. At each iteration k of the algorithm,

do (Step k). Step k.1. Select a subproblem s_k.

$\vdots$

Step k.2. Apply all of the Techniques 1-8 to subproblem s_k. *Bound* the optimum of subproblem s_k from above and below, *i.e.*, find α^{s_k} and a revised β^{s_k} satisfying $\alpha^{s_k} \geq \{\min f(x) \text{ s.t. } x \in D \cap C^{s_k}\} \geq \beta^{s_k} \ldots$.

Step k.2.a. $L \longleftarrow \min_{s \in \mathcal{S} \cup \{s_k\}} \beta^s$; **If** $f(x^{s_k}) < U$ **then** $\tilde{x} \longleftarrow x^{s_k}$ and $U \longleftarrow f(\tilde{x})$.

If U was improved in Step k.2.a **then**
Apply Techniques 1-8 to the entire set $\mathcal{S}$ of open subproblems.
else
Apply Techniques 1-8 only to subproblem s_k.
endif
If the domain reduction techniques succeeded **goto** the beginning of Step k.2.

$\vdots$

end do

6. Computational Results

6.1. IMPLEMENTATION

The proposed algorithm was implemented using BARON [68], a general purpose global optimization software package for solving nonlinear and mixed integer-nonlinear programs (see [16, 40, 66, 65], and consult `http://archimedes.me.uiuc.edu` for the software and further details). The strategy used in the package closely follows the presentation of the previous section. It employs the branch-and-reduce optimization strategy, integrating conventional branch-and-bound with a wide array of domain reduction tools and a combination of branching rules. The implementation was done on an IBM RS/6000 Power PC in FORTRAN. IBM's OSL (Release 2 [35]) was used to solve the relaxed LPs.

Table 1 provides results for 36 small to medium-sized problems from the literature. For each problem, the table provides the size in terms of constraint rows and variables, the source, and results obtained with the algorithm. These experiments used an absolute optimality tolerance of $\varepsilon = 10^{-6}$ throughout, meaning that at any given iteration, the algorithm deleted all subproblems with lower bounds greater than or equal to $U - \varepsilon$. Here, N_{tot}, N_{opt} and N_{mem} denote the total number of iterations, the node at which the optimal solution was found, and the maximum number of nodes stored in memory during the search, respectively. It can be seen that the algorithm solves many of the problems at the root node. Also in the table, T_{tot}, T_{pre}, and T_{bar} denote the total time taken to solve the problem, the time spent on preprocessing, and the time for branch-and-reduce ($T_{tot} = T_{pre} + T_{bar}$). T_{bar} is broken down further into time spent solving the LP relaxations (T_{rel}), time spent applying the problem-specific domain reduction Techniques 7 and 8 (T_{red}), and time spent applying Techniques 3 and 4 which use marginals (T_{mar}). No probing (Techniques 5 and 6) was employed. Evidently, domain reduction using marginals takes a negligible amount of time. In total, domain reduction took but a small fraction of the total CPU time.

The results of Table 1 were obtained using the partitioning rule proposed in Section 2.5, and without any domain reduction based on probing. Table 2 compares results for different strategies. The omega partitioning rule is considered first. For the larger problems solved, it appears to construct search trees that are larger than those formed by other strategies. Omega partitioning could not solve the last problem (from [45], see also [46]) even after 10000 iterations. On the other hand, the proposed bisection of the most violated variable provides more balanced trees and smaller memory requirements. The table also illustrates the effect of bisection with three different probing strategies considered: no probing, probing the bounds of a single variable, and probing the bounds of three variables. Clearly, increasingly sophisticated search strategies lead to smaller search trees and memory requirements although the CPU times actually may increase, as probing entails the solution of additional LPs.

Table 1. Computational results for miscellaneous test problems.

Problem	m	n	Source	Number of Nodes			CPU sec IBM RS/6000 Power PC					
				N_{tot}	N_{opt}	N_{mem}	T_{tot}	T_{pre}	T_{bar}	T_{rel}	T_{red}	T_{mar}
Pan1	4	3	[49]	3	3	2	0.1	0	0.1	0	0	0
Pan2	1	5	[49]	7	2	3	0.1	0	0.1	0	0	0
Z	5	3	[86]	7	0	4	0.1	0	0.1	0	0	0
BSJ2	5	3	[3]	7	0	4	0.1	0	0.1	0	0	0
BSJ4	4	6	[3]	1	1	1	0.1	0	0	0	0	0
KR	5	2	[36]	3	1	2	0.1	0	0.1	0	0	0
PhR1	5	6	[52]	1	0	1	0.1	0	0	0	0	0
PhR2	5	6	[52]	0	0	0	0	0	0	0	0	0
PhR3	5	6	[52]	1	1	1	0.1	0	0	0	0	0
PhR4	4	3	[52]	3	0	2	0.1	0	0.1	0	0	0
PhR11	4	3	[52]	1	1	1	0.1	0	0.1	0	0	0
PhR12	4	3	[52]	1	1	1	0.1	0	0.1	0	0	0
PhR13	10	3	[52]	1	1	1	0.1	0	0.1	0	0	0
PhR14	10	3	[52]	1	0	1	0.1	0	0.1	0	0	0
PhR15	4	4	[52]	3	0	2	0.1	0	0.1	0	0	0
PhR20	9	3	[52]	1	1	1	0.1	0	0.1	0	0	0
FP1	1	5	[22]	7	7	3	0.2	0	0.1	0	0	0
FP2	2	6	[22]	1	0	1	0.1	0	0	0	0	0
FP3	10	13	[22]	1	0	1	0.1	0	0	0	0	0
FP4	5	6	[22]	1	1	1	0.1	0	0	0	0	0
FP5	11	10	[22]	5	1	3	0.2	0.1	0.1	0	0	0
FP6	5	10	[22]	9	8	4	0.2	0.1	0.1	0	0	0
FP7a	10	20	[22]	73	30	9	1.2	0.2	0.9	0.6	0.3	0
FP7b	10	20	[22]	83	32	9	1.2	0.2	0.9	0.6	0.2	0
FP7c	10	20	[22]	67	32	9	1.3	0.2	1	0.7	0.3	0
FP7d	10	20	[22]	59	28	9	0.9	0.2	0.7	0.5	0.2	0
FP7e	10	20	[22]	181	66	21	2.6	0.2	2.3	1.8	0.5	0
FP8	10	24	[22]	3	3	2	0.2	0.1	0.1	0.1	0	0
RV1	5	10	[64]	33	2	4	0.3	0	0.2	0.2	0	0
RV2	10	20	[64]	99	13	12	1.1	0.2	0.9	0.7	0.2	0
RV3	20	20	[64]	203	23	24	2.2	0.2	2	1.7	0.2	0
RV7	20	30	[64]	161	5	11	2.9	0.4	2.4	1.4	1	0
RV8	20	40	[64]	165	49	18	3.2	0.8	2.4	1.8	0.5	0
RV9	20	50	[64]	439	90	65	8	1	7	4.8	2.1	0
M1	11	20	[45]	187	4	5	2.5	0.2	2.2	1.8	0.3	0
M2	21	30	[45]	281	1	5	7.1	1	6.2	4.8	1.3	0

Table 3 provides comparative computational results for test problems that appear in Floudas and Pardalos [22]. The table presents CPU times in seconds with number of nodes in parentheses. Variants of two different branch-and-bound algorithms are considered, the Reformulation-Linearization Technique (RLT) of Sherali and

Table 2. Computational results with different search strategies.

	Omega			Bisection			Probing-1			Probing-3		
Problem	N_{tot}	N_{mem}	T_{tot}	N_{tot}	N_{mem}	T_{tot}	N_{tot}	N_{mem}	T_{tot}	N_{tot}	N_{mem}	T_{tot}
Pan1	3	2	0.1	3	2	0.1	3	2	0.1	3	2	0.2
Pan2	7	3	0.1	7	3	0.1	7	3	0.1	7	3	0.1
Z	7	4	0.1	7	4	0.1	3	2	0.2	3	2	0.2
BSJ2	7	4	0.1	7	4	0.1	3	2	0.2	3	2	0.1
BSJ4	1	1	0.1	1	1	0.1	1	1	0.1	1	1	0.1
KR	3	2	0.1	3	2	0.1	1	1	0.1	1	1	0.1
PhR1	1	1	0.1	1	1	0.1	1	1	0.1	1	1	0.1
PhR2	1	1	0	0	0	0	0	0	0	0	0	0
PhR3	1	1	0.1	1	1	0.1	1	1	0.1	1	1	0.1
PhR4	3	2	0.1	3	2	0.1	1	1	0.1	1	1	0.1
PhR11	1	1	0.1	1	1	0.1	1	1	0.1	1	1	0.1
PhR12	1	1	0.1	1	1	0.1	1	1	0.1	1	1	0.1
PhR13	1	1	0.1	1	1	0.1	1	1	0.1	1	1	0.1
PhR14	1	1	0.1	1	1	0.1	1	1	0.1	1	1	0.1
PhR15	3	2	0.1	3	2	0.1	3	2	0.1	3	2	0.1
PhR20	1	1	0.1	1	1	0.1	1	1	0.1	1	1	0.1
FP1	7	3	0.1	7	3	0.2	7	3	0.2	7	3	0.2
FP2	1	1	0.1	1	1	0.1	1	1	0	1	1	0.1
FP3	1	1	0.1	1	1	0.1	1	1	0.1	1	1	0.1
FP4	1	1	0.1	1	1	0.1	1	1	0.1	1	1	0.1
FP5	3	2	0.2	5	3	0.2	3	2	0.2	3	2	0.2
FP6	9	6	0.2	9	4	0.2	9	4	0.3	9	4	0.4
FP7a	81	17	1.3	73	9	1.2	23	4	2	31	3	2.6
FP7b	87	20	1.4	83	9	1.2	15	3	1.7	11	3	1.8
FP7c	83	21	1.5	67	9	1.3	17	3	1.8	11	3	1.7
FP7d	71	13	1.3	59	9	0.9	21	3	2	13	3	1.9
FP7e	169	36	2.9	181	21	2.6	79	12	6.6	79	12	9.6
FP8	3	2	0.3	3	2	0.2	3	2	0.4	1	1	0.5
RV1	17	3	0.2	33	4	0.3	5	3	0.3	5	3	0.3
RV2	75	23	0.8	99	12	1.1	13	5	1.1	9	3	1.1
RV3	271	66	3.1	203	24	2.2	25	6	1.7	17	5	1.7
RV7	177	63	2.8	161	11	2.9	13	3	1.8	11	3	1.9
RV8	201	28	4.3	165	18	3.2	19	5	3.2	13	4	3.3
RV9	561	113	9.5	439	65	8	55	11	7.2	39	10	7.5
M1	2713	757	39.5	187	5	2.5	117	3	8.1	41	3	7.3
M2	$>$10000	9959	$>$213	281	5	7.1	115	3	22.4	113	3	27

Table 3. Computational results for the Floudas and Pardalos test problems.

Problem	m	n	Tolerance	Reformulation-Linearization [73] IBM 3090 supercomputer		Branch-and-Reduce IBM RS/6000 Power PC	
			(%)	LD-RLT-NLP	LD-RLT-NLP(SC)	R&S [66]	proposed
FP5	11	10	1	1.12 (1)	1.17 (1)		0.2 (1)
FP6	5	10	1	1.61 (5)	1.72 (5)		0.2 (8)
FP7a	10	20	5	8.13 (7)	3.29 (3)	6.88 (35)	0.5 (15)
FP7b	10	20	5	2.54 (1)	2.61 (1)	2.05 (13)	0.4 (11)
FP7c	10	20	5	13.26 (11)	2.55 (1)	5.71 (35)	0.4 (13)
FP7d	10	20	5	5.04 (5)	2.61 (1)	2.10 (13)	0.6 (23)
FP7e	10	20	5	27.00 (25)	15.94 (11)	11.37 (69)	0.9 (59)

Table 4. CPU seconds for Rosen and van Vliet problems ($\varepsilon = 0.001$).

Problem	m	n	G&R86 Cyber 845	R&vV87 CRAY2	P&R87 CRAY2	S&S95 RS/6000
RV1	5	10	10.34	1.50	0.11	0.3
RV2	10	20	20.47	18.69	1.43	1.1
RV3	20	20	211.87	73.84	3.21	2.2
RV7	20	30	417.26	118.76	9.16	2.9
RV8	20	40	328.55	195.53	16.52	3.2
RV9	20	50				8.0

Tuncbilek [73], (see also [72]), and the branch-and-reduce algorithm. RLT is known to produce stronger lower bounds than the more straightforward linear programming underestimation used by branch-and-reduce. As a result, RLT requires the solution of fewer nodes. On the other hand, the algorithm proposed in this paper solves much simpler relaxations and therefore requires smaller CPU times than RLT. In addition, our approach can accommodate larger problems, as the RLT would introduce prohibitively larger numbers of constraints and variables in the subproblem relaxations. The algorithm of Ryoo and Sahinidis [66] is slower than the one presently proposed, since it incorporates only a subset of the domain reduction techniques used here, it does not perform extensive preprocessing, and it uses omega subdivision as the branching rule.

Computational results for the Rosen and van Vliet [64] problems are presented in Table 4. We note that the parallel algorithm P&R87 of Phillips and Rosen [54] serves as the current benchmark for quadratic programming. One can draw an important conclusion by comparing the entries of this table. Today on a standard engineering workstation, we can solve in a few seconds the same problems which, until recently, required several minutes of CPU time on large mainframes.

The algorithm was also applied to randomly generated large-scale problems of the form:

$$\begin{aligned} \text{min} \quad & \frac{1}{2}\theta_1 \sum_{j=1}^{n} \lambda_j (x_j - \bar{\omega}_j)^2 + \theta_2 \sum_{j=1}^{k} d_j y_j \\ \text{subject to} \quad & \begin{cases} A_1 x + A_2 y \leq b \\ x \geq 0, y \geq 0 \end{cases}, \end{aligned}$$

where

$$\begin{aligned} & x, \lambda, \bar{\omega} \in R^n, \\ & y, d \in R^k, \\ & b \in R^m, \\ & A_1 \in R^{m \times n}, \\ & A_2 \in R^{m \times k}, \\ & \theta_1, \theta_2 \in R. \end{aligned}$$

The values of the parameters θ_1 and θ_2 are 0.001 and 0.1, respectively. Such problems have been studied by Phillips and Rosen [54] and Visweswaran and Floudas [84]. The data for the constants λ, $\bar{\omega}$, d, b, A_1, and A_2 were generated by the same routines used by [54] and [84].

Table 5 presents computational results for problems of different sizes of m, n and k. The data for the GOP algorithm are taken from [84] and the data for the P&R algorithm are taken from [54]; (in the more recent [58], a variant of the latter algorithm is applied to smaller test problems than those studied by Table 5). These strategies used *relative* optimality criteria of 0.1 and 0.001, respectively. Our algorithm was applied with an *absolute* optimality criterion of $\varepsilon = 10^{-6}$. It should be noted that the codes used to generate the test problems of Table 5 were the same for the three algorithms. Also, each row of this table was generated from a total of 10 different random runs. Apparently, our sequential implementation on a standard engineering workstation provides more accurate results for this class of problems in very reasonable computing times. Even the largest problems with 100 nonlinear variables, 400 linear variables and 50 constraints can be solved within a matter of minutes.

For each pair of values taken by m and n in Table 5, Figure 4 depicts how the number of nodes requires by the branch & bound algorithm increases with the number k of linear variables. The increase seems to follow an approximately quadratic relationship.

Regarding the comparative results of Tables 3 through 5, it is also insightful to compare the relative capabilities of the hardware, which allows us to directly compare CPU times on different computers at different points in time. Table 6, extracted from Dongarra [15], gives three measures of computing speed for each

Table 5. CPU times (sec) for the Phillips and Rosen test problems.

			GOP93 [84] E=0.1 (relative) HP 730		P&R87 [54]* E=0.001 (relative) CRAY 2 (parallel)			P&R87 [54]* E=0.001 (relative) CRAY 2 (serial)			proposed algorithm $\varepsilon = 10^{-6}$ (absolute) IBM RS/6000 PC		
m	n	k	avg	std dev	min	avg	max	min	avg	max	min	avg	max
20	25	0	0.456	0.012	1	2	4	1	6	9	0.2	0.4	0.5
20	25	50	1.662	1.614	1	1	1	1	2	3	0.8	1	1.2
20	25	100	16.508	19.711	1	2	3	3	4	5	1	2	3
20	25	200	33.149	28.441	1	7	18	5	20	47	2	4	7
20	25	400	82.026	57.834	7	14	32	20	44	93	4	9	14
20	50	0	0.554	0.012	3	6	13	9	17	37	1.2	1.4	1.6
20	50	50	17.436	30.908	1	2	3	4	5	7	2	3	4
20	50	100	46.761	49.195	2	5	14	7	17	43	3	4	7
20	50	200	108.968	80.490	4	9	29	11	29	79	5	9	21
20	50	400			20	32	51	68	105	163	8	20	31
40	25	0	0.465	0.017							0.2	0.4	0.5
40	25	50	0.970	0.566							0.3	1	1.2
40	25	100	2.708	4.211							1	2	3
40	25	200	25.142	26.127							2	3.5	4
40	25	400									11	17	24
50	100	0									5.5	8	23
50	100	50									8	9.5	11
50	100	100									7	14	29
50	100	200									13	45	137
50	100	400									38	168	380

* Data Estimated From Figures 5-8 of [54].

Table 6. Comparative Speed of Computers* in Mflop/s**

Computer	Experimental		Theoretical
	LINPACK, n=100	Custom, n=1000	
Cray-2/4-256 (4 processors, 4.1 ns)	62	1226	1951
IBM 3090 supercomputer	9.6-16	71-97	116-138
HP 9000/730 (66 MHz)	24	49	66
IBM RS/6000-N40 (PowerPC601 50MHz)	6.7	NA	50

*Extracted from [15].
*Millions of floating point operations per second.

machine previously mentioned. For each machine listed in the first column of the table, the second column provides the speed at which a dense system of 100 linear equations is solved using standard programs from the LINPACK [14] libraries in a FORTRAN environment. The third column shows results based on solving a dense system of 1000 linear equations using a custom implementation on each computer. Finally, the fourth column indicates the theoretical peak rate of execution based

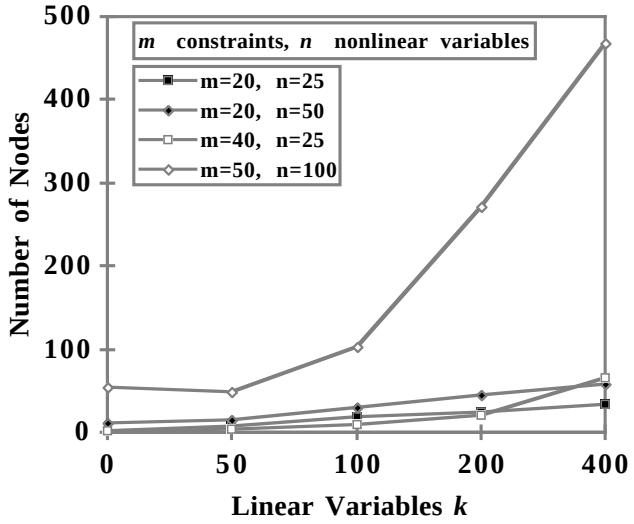

Figure 4. Number of branch & bound nodes for the Phillips and Rosen test problems.

on the cycle time of the hardware, as given by the manufacturer. By examining Table 6, one can easily see that the other studies use faster processors. Yet, from the computational results in Tables 3-5, one finds that our algorithm is substantially faster than existing ones.

6.2. Application to Network Problems with Concave Costs

The problem of selecting processes and planning expansions of an industrial chemical complex to maximize net present value has traditionally been formulated as a multiperiod, mixed-integer linear program [70], [69]. However, the fixed-charges in such problems allow for reformulation as concave programs. In fact, a concave programming approach to these problems seems computationally more expedient than solving the equivalent MILP; for solving actual process planning problems, solution of the SCP using the algorithm proposed in this paper requires about one-third of the time required to solve the MILP using OSL [40].

For example, consider the petrochemical complex planning problem given in [70]. The problem involves thirty-eight processes and twenty-eight chemicals over four time periods, making for an MILP formulation in 897 variables (152 binaries and 745 continuous variables), and 569 constraints. An equivalent SCP formulation has only 745 continuous variables (152 concave and 593 linear), and 417 constraints. While solving the MILP on OSL required 13,289 nodes and 770 CPU seconds, solving the SCP on BARON with the algorithm developed in this paper required just 5,237 nodes and 638 CPU seconds [40].

7. Conclusions

We draw two. One conclusion is of practical concern, the other of theoretical value. Practically speaking, domain reduction techniques are an exceedingly compelling way to accelerate computer implementations of branch-and-bound for nonlinear programming. For concave programming in particular, this increase in computing speed facilitates the solution of large industrial problems that had previously been solved only by integer programming.

From the theoretical standpoint, a branch-and-bound algorithm using rectangular partitions can solve SCP globally and finitely by

(a) branching at the best known solution whenever possible, and

(b) partitioning exhaustively in the search process.

The degree to which results of the current paper can be extended to the general NLP to provide finite termination with a global solution remains a major open question.

Acknowledgements

Partial financial support from the EXXON Education Foundation and from the National Science Foundation under grants DMII 94-14615 and CAREER award DMII 95-02722 to N.V.S. is gratefully acknowledged. We wish to thank Dr. A. Phillips for providing us his test problem generator. The paper has benefited greatly from anonymous review.

References

1. F. Al-Khayyal and J. Kyparisis. Finite convergence of algorithms for nonlinear programs and variational inequalities. *Journal of Optimization Theory and Applications*, 70(2):319–332, 1991.
2. M. S. Bazaraa and H. D. Sherali. On the use of exact and heuristic cutting plane methods for the quadratic assignment problem. *Journal Operational Society*, 33:991–1003, 1982.
3. S. Ben Saad and S. E. Jacobsen. A level set algorithm for a class of reverse convex programs. *Annals of Operations Research*, 25:19–42, 1990.
4. H. P. Benson. A finite algorithm for concave minimization over a polyhedron. *Naval Research Logistics Quarterly*, 32:165–177, 1985.
5. H. P. Benson. Separable concave minimization via partial outer approximation and branch and bound. *Operations Research Letters*, 9:389–394, 1990.
6. H. P. Benson. Concave minimization: Theory, applications and algorithms. In, P. M. Pardalos and R. Horst (eds.) Handbook of Global Optimization, Chapter 3, Hingham, Massachusetts, 1994.
7. H. P. Benson and R. Horst. A branch and bound-outer approximation algorithm for concave minimization over a convex set. *Computers Math Applications*, 21(6/7):67–76, 1991.
8. H. P. Benson and S. Sayin. A finite concave minimization algorithm using branch and bound and neighbor generation. *Journal of Global Optimization*, 5:1–14, 1994.
9. I. M. Bomze and G. Danninger. A global optimization algorithm for concave quadratic programming problems. *SIAM Journal of Optimization*, 3:826–842, 1993.
10. I. M. Bomze and G. Danninger. A finite algorithm for solving general quadratic problems. *Journal of Global Optimization*, 4:1–16, 1994.
11. K. M. Bretthauer and A. V. Cabot. A composite branch and bound, cutting plane algorithm for concave minimization over a polyhedron. *Computers in Operations Research*, 21(7):777–785, 1994.

12. A. V. Cabot and R. L. Francis. Solving certain nonconvex quadratic minimization problems by ranking the extreme points. *Operations Research*, 18:82–86, 1970.
13. R. Carvajal-Moreno. Minimization of concave functions subject to linear constraints. Technical Report ORC 72–3, Operations Research Center, University of California, Berkeley, 1972.
14. J. Dongarra, J. Bunch, C. Moler, and G. W. Stewart. *LINPACK User's Guide*. SIAM, Philadelphia, PA, 1979.
15. Jack J. Dongarra. Performance of various computers using standard linear equations software. Technical Report CS–89–85, Computer Science Department, University of Tennessee, Knoxville, and Mathematical Sciences Section, Oak Ridge National Laboratory, Oak Ridge, 1997.
16. M. C. Dorneich and N. V. Sahinidis. Global optimization algorithms for chip layout and compaction. *Engineering Optimization*, 25(2):131–154, 1995.
17. M. E. Dyer. The complexity of vertex enumeration methods. *Mathematics of Operations Research*, 8:381–402, 1983.
18. M. E. Dyer and L. G. Proll. An algorithm for determining all extreme points of a convex polytope. *Mathematical Programming*, 12:81–96, 1977.
19. J. E. Falk. A linear max–min problem. *Mathematical Programming*, 5:169–188, 1973.
20. J. E. Falk and K. R. Hoffman. A successive underestimation method for concave minimization problems. *Mathematics of Operations Research*, 1(3):251–259, 1976.
21. J. E. Falk and R. M. Soland. An algorithm for separable nonconvex programming problems. *Management Science*, 15(9):550–569, 1969.
22. C. A. Floudas and P. M. Pardalos. *A Collection of Test Problems for Constrained Global Optimization Algorithms*. Number 268 in Lecture Notes in Computer Science. Springer–Verlag, Berlin– Heidelberg, 1990.
23. A. M. Frieze. A bilinear programming formulation of the 3–dimensional assignment problem. *Mathematical Programming*, 7:376–379, 1974.
24. T. Gal. *Postoptimal Analyses, Parametric Programming, and Related Topics*. McGraw-Hill Internaional, London, 1979.
25. F. Gianessi and F. Niccolucci. Connections between nonlinear and integer programming problems. In, Symposia Mathematica XIX, Istituto Nazionale Di Alta Mathematica, pp. 161–176, New York, 1976.
26. F. Glover. Convexity cuts and cut search. *Operations Research*, 21:123–134, 1973.
27. F. Glover and D. Klingman. Concave programming applied to a special class of 0-1 integer programs. *Operations Research*, 21:135–140, 1973.
28. P. Hansen, B. Jaumard, and S.-H. Lu. An analytical approach to global optimization. *Mathematical Programming, Series B*, 52:227–254, 1991.
29. K. L. Hoffman. A method for globally minimizing concave functions over convex sets. *Mathematical Programming*, 22:22–32, 1981.
30. R. Horst. An algorithm for nonconvex programming problems. *Mathematical Programming*, 10:312–321, 1976.
31. R. Horst. On the global minimization of concave functions–introduction and survey. *OR Spektrum*, 6:195–205, 1984.
32. R. Horst, P. M. Pardalos, and N. V. Thoai. *Introduction to Global Optimization*. Nonconvex Optimization and its Applications. Kluwer Academic Publishers, Norwell, MA, 1995.
33. R. Horst and H. Tuy. *Global Optimization: Deterministic Approaches*. Springer–Verlag, Berlin, third edition, 1996.
34. N. V. Horst, R., Thoai and H. P. Benson. Concave minimization via conical partitions and polyhedral outer approximation. *Mathematical Programming*, 50:259–274, 1991.
35. IBM. *Optimization Subroutine Library Guide and Reference Release 2*. International Business Machines Corporation, Kingston, NY, third edition, July 1991.
36. B. Kalantari and J. B. Rosen. An algorithm for global minimization of linearly constrained convex quadratic functions. *Mathematics of Operations Research*, 12(3):544–561, 1987.
37. S. L. Krynski. Minimization of a concave function under linear constraints (modification of tuy's method). In, Survey of Mathematical Programming, Proceedings of the Ninth International Mathematical Programming Symposium, Mathematicl Programming Society, Budapest, 1976, volume 1, pp. 479–493, Amsterdam, 1979.

38. B. W. Lamar. An improved branch and bound algorithm for minimum concave cost network flow problems. *Journal of Global Optimization*, 3(3):261–287, 1993.
39. E. L. Lawler. The quadratic assignment problem. *Management Science*, 9:586–699, 1963.
40. M. L. Liu and N. V. Sahinidis and J. P. Shectman. Planning of chemical process networks via global concave minimization. In, I. E. Grossmann (ed.), Global Optimization in Engineering Design, Boston, MA, 1996.
41. O. L. Mangasarian. Characterization of linear complementarity problems as linear programs. *Mathematical Programming Study*, 7:74–87, 1978.
42. T. H. Matheiss. An algorithm for determining unrelevant constraints and all vertices in systems of linear inequalities. *Operations Research*, 21:247–260, 1973.
43. T. H. Matheiss and D. S. Rubin. A survey and comparison of methods for finding all vertices of convex polyhedral sets. *Mathematics of Operations Research*, 5:167–185, 1980.
44. G. P. McCormick. Attempts to calculate global solutions of problems that may have local minima. In, F. A. Lootsma (ed.), Numerical Methods for Non–Linear Optimization, pp. 209–221, New York, 1972.
45. K. Moshirvaziri. Personal Communication, 1994.
46. K. Moshirvaziri. A generalization of the construction of test problems for nonconvex optimization. *Journal of Global Optimization*, 5:21–34, 1994.
47. B. M. Mukhamediev. Approximate methods of solving concave programming problems. *USSR Computational Mathematics and Mathematical Physics*, 22(3):238–245, 1982.
48. K. G. Murty and S. N. Kabadi. Some NP–complete problems in quadratic and nonlinear programming. *Mathematical Programming*, 39:117–129, 1987.
49. P. M. Pardalos. *Integer and Separable Programming Techniques for Large–Scale Global Optimization Problems*. PhD thesis, Computer Science Department, University of Minnesota, Minneapolis, 1985.
50. P. M. Pardalos and J. B. Rosen. Methods for global concave minimization: A bibliographic survey. *SIAM Review*, 28:367–379, 1986.
51. P. M. Pardalos and J. B. Rosen. *Constrained Global Optimization: Algorithms and Applications*. Number 268 in Lecture Notes in Computer Science. Springer–Verlag, Berlin–Heidelberg, 1987.
52. A. T. Phillips. *Parallel Algorithms for Constrained Optimization*. PhD thesis, University of Minnesota, Minneapolis, 1988.
53. A. T. Phillips and J. B. Rosen. A parallel algorithm for constrained concave quadratic global minimization. Technical Report 87–48, Computer Science Department, Institute of Technology, University of Minnesota, Minneapolis, 1987.
54. A. T. Phillips and J. B. Rosen. A parallel algorithm for constrained concave quadratic global minimization. *Mathematical Programming*, 42:421–448, 1988.
55. A. T. Phillips and J. B. Rosen. Guaranteed ε–approximate solution for indefinite quadratic global minimization. *Naval Research Logistics*, 37:499–514, 1990.
56. A. T. Phillips and J. B. Rosen. A parallel algorithm for partially separable non–convex global minimization: Linear constraints. *Annals of Operations Research*, 25:101–118, 1990.
57. A. T. Phillips and J. B. Rosen. Sufficient conditions for solving linearly constrained separable concave global minimization problems. *Journal of Global Optimization*, 3:79–94, 1992.
58. A. T. Phillips and J. B. Rosen. Computational comparison of two methods for constrained global optimization. *Journal of Global Optimization*, 5(4):325–332, 1994.
59. J. B. Phillips, A. T., Rosen and M. van Vliet. A parallel stochastic method for the constrained concave global minimization problem. *Journal of Global Optimization*, 2(3):243–258, 1992.
60. M. Raghavachari. On connections between zero–one integer programming and concave programming under linear constraints. *Operations Research*, 17:680–684, 1969.
61. R. T. Rockafellar. *Convex Analysis*. Princeton University Press, Princeton, second edition, 1972.
62. J. B. Rosen. Global minimization of a linearly constrained concave function by partition of feasible domain. *Mathematics of Operations Research*, 8(2):215–230, 1983.
63. J. B. Rosen and P. M. Pardalos. Global minimization of large–scale constrained concave quadratic problems by separable programming. *Mathematical Programming*, 34:163–174, 1986.

64. J. B. Rosen and M. van Vliet. A parallel stochastic method for the constrained concave global minimization problem. Technical Report 87–31, Computer Science Department, Institute of Technology, University of Minnesota, Minneapolis, 1987.
65. H. S. Ryoo and N. V. Sahinidis. Global optimization of nonconvex nlps and minlps with applications in process design. *Computers & Chemical Engineering*, 19(5):551–566, 1995.
66. H. S. Ryoo and N. V. Sahinidis. A branch-and-reduce approach to global optimization. *Journal of Global Optimization*, 8(2):107–138, March 1996.
67. N. V. Sahinidis. Accelerating branch–and–bound in continuous optimization. Research Report UILU ENG 92–4031, University of Illinois, Urbana, 1992.
68. N. V. Sahinidis. Baron: A general purpose global optimization software package. *Journal of Global Optimization*, 8(2):201–205, March 1996.
69. N. V. Sahinidis and I. E. Grossmann. Reformulation of the multiperiod MILP model for capacity expansion of chemical processes. *Operations Research*, 40, Supp. No. 1:S127–S144, 1992.
70. N. V. Sahinidis, I. E. Grossmann, R. E. Fornari, and M. Chathrathi. Optimization model for long range planning in the chemical industry. *Computers and Chemical Engineering*, 13:1049–1063, 1989.
71. J. P. Shectman and N. V. Sahinidis. A finite algorithm for global minimization of separable concave programs. In, C. A. Floudas and P. M. Pardalos (eds.), State of the Art in Global Optimization: Computational Methods and Applications, Boston, MA, 1996.
72. H. D. Sherali and A. Alameddine. A new reformulation–linearization technique for bilinear programming problems. *Journal of Global Optimization*, 2(4):379–410, 1992.
73. H. D. Sherali and C. H. Tuncbilek. A reformulation-convexification approach for solving nonconvex quadratic programming problems. *Journal of Global Optimization*, 7(1):1–31, July 1995.
74. R. M. Soland. Optimal facility location with concave costs. *Operations Research*, 22:373–382, 1974.
75. U. H. Suhl and R. Szymanski. Supernode processing of mixed–integer models. *Computational Optimization and Applications*, 3:317–331, 1994.
76. N. V. Thakur. Domain contraction in nonlinear programming: Minimizing a quadratic concave function over a polyhedron. *Mathematics of Operations Research*, 16(2):390–407, 1990.
77. T. V. Thieu. Relationship between bilinear programming and concave programming. *Acta Mathematica Vietnamica*, 2:106–113, 1980.
78. N. V. Thoai and H. Tuy. Convergent algorithms for minimizing a concave function. *Mathematics of Operations Research*, 5:556–566, 1980.
79. N. V. Thoai and H. Tuy. Solving the linear complementarity problem through concave programming. *USSR Computational Mathematics and Mathematical Physics*, 23(3):55–59, 1983.
80. H. Tuy. Concave programming under linear constraints. *Soviet Mathematics*, 5:1437–1440, 1964.
81. H. Tuy. Effect of the subdivision strategy on convergence and efficiency of some global optimization algorithms. *Journal of Global Optimization*, 1(1):23–36, 1991.
82. H. Tuy and R. Horst. Convergence and restart in branch–and–bound algorithms for global optimization. application to concave minimization and DC optimization problems. *Mathematical Programming*, 41:161–183, 1988.
83. T. V. Tuy, H., Thieu and Thai N. Q. A conical algorithm for globally minimizing a concave function over a closed convex set. *Mathematics of Operations Research*, 10:498–514, 1985.
84. V. Visweswaran and C. A. Floudas. New properties and computational improvement of the gop algorithm for problems with quadratic objective functions and constraints. *Journal of Global Optimization*, 3:439–462, 1993.
85. P. B. Zwart. Computational aspects on the use of cutting planes in global optimization. In, Proceedings of the 1971 Annual Conference of the ACM, Association for Computing Machinery, pp. 457–465, 1971.
86. P. B. Zwart. Nonlinear programming: Counterexamples to global optimization algorithms. *Operations Research*, 21:1260–1266, 1973.
87. P. B. Zwart. Global maximization of a convex function with linear inequality constraints. *Operations Research*, 22:602–609, 1974.