A Finite Concave Minimization Algorithm Using Branch and Bound and Neighbor Generation

HAROLD P. BENSON and SERPIL SAYIN*
Department of Decision and Information Sciences, University of Florida, Gainesville, FL 32611-2017, U.S.A.

(Received: 21 May 1992; accepted: 18 October 1993)

Abstract. In this article we present a new finite algorithm for globally minimizing a concave function over a compact polyhedron. The algorithm combines a branch and bound search with a new process called neighbor generation. It is guaranteed to find an exact, extreme point optimal solution, does not require the objective function to be separable or even analytically defined, requires no nonlinear computations, and requires no determinations of convex envelopes or underestimating functions. Linear programs are solved in the branch and bound search which do not grow in size and differ from one another in only one column of data. Some preliminary computational experience is also presented.

Key words. Concave minimization, branch and bound, global optimization.

1. Introduction

The purpose of this article is to present a new algorithm for solving the concave minimization problem (P) given by

$$\min f(x), \quad \text{subject to } x \in X,$$

where X is a nonempty, compact polyhedral set in R^n, and f is a real-valued concave function on some open set Y in R^n that contains X. Problem (P) may have many locally optimal solutions which are not globally optimal, so that it represents a type of *global optimization* problem [8]. However, it is well known that there exists an extreme point of X which is a globally optimal solution of problem (P). Let m denote the optimal objective function value of problem (P).

Problem (P) has been the subject of intense interest for almost 30 years. There are at least two reasons for this. First, large classes of important decision models arising from a variety of applications lead to formulations involving the minimization of concave functions over polyhedra. Included in these applications are various problems involving economies of scale, production planning, and engineering design, to name a few. Second, many optimization problems which originally are not concave can be transformed into equivalent concave minimizations over polyhedra, including zero-one integer programming problems, bilinear programming problems, fixed charge problems, and linear complementarity problems. Surveys of the applications and importance of problem (P) can be

* Current affiliation: Management Department, Bilkent University, Ankara, Turkey.

Journal of Global Optimization **5**: 1–14, 1994.

found, for instance, in Pardalos and Rosen [14, 15], Horst [6], and Horst and Tuy [8].

From the viewpoint of computational complexity, the concave minimization problem (P) is NP-hard, even in the special case of minimizing a quadratic concave function over a very simple polyhedron such as a hypercube [16]. However, complexity results of this nature characterize worst-case instances. Hence the search for more practical algorithms for globally solving problem (P) remains an important undertaking.

Current methods for solving problem (P) usually use either cutting planes, extreme point ranking, inner approximation, outer approximation, branch and bound, or a combination of these approaches. A survey of the algorithms available for solving problem (P) will not be given here. Instead, the reader is referred to McCormick [11], Heising-Goodman [3], Rosen [18], Horst [5], Benson [1], Pardalos and Rosen [14, 15], Horst [6], and Horst and Tuy [8] for comprehensive discussions of these algorithms.

In this article we present a finite algorithm for solving the concave minimization problem (P). The algorithm finds a globally optimal extreme point solution for problem (P) by combining a branch and bound search with a neighbor generation process. The neighbor generation process helps to generate incumbent solutions and to ensure that the algorithm is implementable and convergent. In addition, the algorithm has the following advantages:

(i) It finds an exact, extreme point optimal solution.
(ii) It does not require f to be separable or even analytically defined.
(iii) It requires no nonlinear computations and no determinations of convex envelopes or other underestimating functions.
(iv) It solves linear programming problems in the branch and bound search which do not grow in size and differ from one another in only one column of data.
(v) It avoids solving some linear programs through the use of a fathoming procedure.

Part of the motivation for developing the algorithm comes from previous work by Soland [19], Horst [4], and Benson [1]. The algorithm is similar in certain respects to the algorithms presented in these papers. However, important differences exist. Furthermore, the convergence of the algorithm given here is guaranteed in a different way. Therefore, we will present the algorithm in its entirety without assuming any knowledge of these algorithms.

The plan of this article is as follows. In Section 2 certain theoretical prerequisites needed for developing the algorithm are presented. In Section 3 the new algorithm for solving problem (P) is presented, its convergence properties are shown, and its computational benefits are reviewed. Some preliminary computational experience with the algorithm is reported in the final section.

2. Theoretical Prerequisites

We shall assume henceforth without loss of generality that $X = \{x \in R^n \mid Ax \leqq b\}$,

where A is a $p \times n$ matrix and $b \in R^p$. We shall also assume familiarity with certain basic concepts commonly used in global optimization, including the notions of an *n-dimensional simplex* (or, more briefly, an *n-simplex*) a *vertex* of an n-simplex, a *partition* of a subset of R^n, a *radial subdivision* of an n-simplex, and the *convex envelope* of a lower semicontinuous function on a compact convex subset of R^n. Definitions and discussions of these concepts can be found, for instance, in [8].

One of the key procedures used in the algorithm to be presented for problem (P) involves computing a lower bound for f over $S \cap X$ for an arbitrary n-simplex S. To compute this lower bound, the algorithm finds an extreme point of X which minimizes the convex envelope f_S of f on S over X. It is well known (cf. [4, 8]) that f_S is affine over both S and R^n, and that it can be found by solving a system of linear equations. However, the algorithm to be given finds the minimum of f_S over X and an extreme point of X which achieves this minimum without requiring the explicit determination of the function f_S. To achieve this, it relies upon the following result.

THEOREM 1. *Let $S \subseteq Y$ be an n-simplex with vertices $v^0, v^1, \ldots, v^n$, and let $f_S : R^n \to R$ be the convex envelope of f on S. Consider the linear program (P_x) given by*

$$\min f_S(x), \quad \text{subject to } x \in X,$$

and the linear program (P_λ) given by

$$\min \sum_{i=0}^{n} f(v^i)\lambda_i,$$

$$\text{subject to } \sum_{i=0}^{n} (Av^i)\lambda_i \leqq b,$$

$$\sum_{i=0}^{n} \lambda_i = 1.$$

Let Λ denote the feasible region for the linear program (P_λ).

(i) *If $\hat{\lambda} \in \Lambda$, then $\hat{x} = \sum_{i=0}^{n} \hat{\lambda}_i v^i$ belongs to X and $f_S(\hat{x}) = \sum_{i=0}^{n} f(v^i)\hat{\lambda}_i$. Conversely, if $\hat{x} \in X$, then there exists a unique $\hat{\lambda} \in \Lambda$ such that $\hat{x} = \sum_{i=0}^{n} \hat{\lambda}_i v^i$ which, furthermore, satisfies $f_S(\hat{x}) = \sum_{i=0}^{n} f(v^i)\hat{\lambda}_i$.*

(ii) *If $\hat{\lambda}$ is an extreme point of Λ, then $\hat{x} = \sum_{i=0}^{n} \hat{\lambda}_i v^i$ is an extreme point of X. Conversely, if $\hat{x}$ is an extreme point of X, then the unique $\hat{\lambda} \in \Lambda$ such that $\hat{x} = \sum_{i=0}^{n} \hat{\lambda}_i v^i$ is an extreme point of Λ.*

(iii) *The optimal values of the linear programs (P_x) and (P_λ) are equal. If λ^* is an optimal solution for problem (P_λ), then $x^* = \sum_{i=0}^{n} \lambda_i^* v_i$ is an optimal solution for problem (P_x). If x^* is an optimal solution for problem (P_x), then the unique $\lambda^* \in \Lambda$ such that $x^* = \sum_{i=0}^{n} \lambda_i^* v^i$ is an optimal solution for problem (P_λ).*

Proof. (i) Assume that $\hat{\lambda} \in \Lambda$. Let $\hat{x} = \sum_{i=0}^{n} \hat{\lambda}_i v^i$. Since $\sum_{i=0}^{n} (Av^i)\hat{\lambda}_i \leqq b$, the definition of $\hat{x}$ implies that $A\hat{x} \leqq b$, i.e., $\hat{x} \in X$. From [4, 8], f_S is an affine

function and satisfies $f_S(v^i) = f(v^i)$, $i = 0, 1, \ldots, n$. Therefore, from the definition of $\hat{x}$, $f_S(\hat{x}) = f_S(\Sigma_{i=0}^n \hat{\lambda}_i v^i) = \Sigma_{i=0}^n \hat{\lambda}_i f_S(v^i) = \Sigma_{i=0}^n \hat{\lambda}_i f(v^i)$.

To show the converse statement in (i), assume that $\hat{x} \in X$. Since S is an n-simplex, the vectors v^i, $i = 0, 1, \ldots, n$, are affinely independent. For each $i = 0, 1, \ldots, n$, form the vector $w^i \in R^{n+1}$ whose first n entries equal those of v^i and whose $(n+1)$st entry is one. Then w^i, $i = 0, 1, \ldots, n$ are linearly independent vectors in R^{n+1}. Therefore, they form a basis for R^{n+1}. Let $\hat{y} \in R^{n+1}$ denote the vector whose first n entries equal those of $\hat{x}$ and whose $(n+1)$st entry is one. Then, since $\hat{y} \in R^{n+1}$, there exists a *unique* $\hat{\lambda} \in R^{n+1}$ such that $\hat{y} = \Sigma_{i=0}^n \hat{\lambda}_i w^i$. This implies that $\hat{x} = \Sigma_{i=0}^n \hat{\lambda}_i v^i$ and $\Sigma_{i=0}^n \hat{\lambda}_i = 1$. Furthermore, since $\hat{x} \in X$, $\Sigma_{i=0}^n (Av^i)\hat{\lambda}_i = A \Sigma_{i=0}^n \hat{\lambda}_i v^i = A\hat{x} \leqq b$. Therefore, $\hat{\lambda} \in \Lambda$. Finally, since f_S is affine and satisfies $f_S(v^i) = f(v^i)$, $i = 0, 1, \ldots, n$, we obtain $f_S(\hat{x}) = f_S(\Sigma_{i=0}^n \hat{\lambda}_i v^i) = \Sigma_{i=0}^n \hat{\lambda}_i f_S(v^i) = \Sigma_{i=0}^n \hat{\lambda}_i f(v^i)$.

(ii) Assume that $\hat{\lambda}$ is an extreme point of Λ. Let $\hat{x} = \Sigma_{i=0}^n \hat{\lambda}_i v^i$. Then, by part (i), $\hat{x} \in X$. Since $\hat{x} \in X$, again using part (i), it follows that no $\bar{\lambda} \in \Lambda$, $\bar{\lambda} \neq \hat{\lambda}$, exists such that $\hat{x} = \Sigma_{i=0}^n \bar{\lambda}_i v^i$.

Suppose that $\hat{x}$ is not an extreme point of X. Then, since $\hat{x} \in X$, for some points x^1, $x^2 \in X$ distinct from $\hat{x}$ and for some $\theta \in (0, 1)$, $\hat{x} = \theta x^1 + (1-\theta)x^2$. Since $x^1, x^2 \in X$, by part (i), for each $j = 1, 2$, $x^j = \Sigma_{i=0}^n \lambda_i^j v^i$ for some λ^1, $\lambda^2 \in \Lambda$. Therefore,

$$\begin{aligned}\hat{x} &= \theta \sum_{i=0}^{n} \lambda_i^1 v^i + (1-\theta) \sum_{i=0}^{n} \lambda_i^2 v^i \\ &= \sum_{i=0}^{n} [\theta\lambda_i^1 + (1-\theta)\lambda_i^2] v^i \\ &= \sum_{i=0}^{n} [\theta\lambda^1 + (1-\theta)\lambda^2]_i v^i ,\end{aligned}$$

where, by convexity of Λ, $[\theta\lambda^1 + (1-\theta)\lambda^2] \in \Lambda$. Since no $\bar{\lambda} \in \Lambda$, $\bar{\lambda} \neq \hat{\lambda}$, exists such that $\hat{x} = \Sigma_{i=0}^n \bar{\lambda}_i v^i$, this implies that $\hat{\lambda} = \theta\lambda^1 + (1-\theta)\lambda^2$. Therefore, $\hat{\lambda} = \lambda^1 = \lambda^2$, since $\hat{\lambda}$ is an extreme point of Λ. On the other hand, since $x^j \neq \hat{x}$, $j = 1, 2$, $\lambda^j \neq \hat{\lambda}$, $j = 1, 2$. This contradiction implies that the assumption that $\hat{x}$ is not an extreme point of X must be false.

To show the converse statement in (ii), assume that $\hat{x}$ is an extreme point of X. Let $\hat{\lambda}$ be the unique element of Λ which satisfies $\hat{x} = \Sigma_{i=0}^n \hat{\lambda}_i v^i$.

Suppose that $\hat{\lambda}$ is not an extreme point of Λ. Then, since $\hat{\lambda} \in \Lambda$, for some points λ^1, $\lambda^2 \in \Lambda$ distinct from $\hat{\lambda}$ and for some $\theta \in (0, 1)$, $\hat{\lambda} = \theta\lambda^1 + (1-\theta)\lambda^2$. Then

$$\begin{aligned}\hat{x} &= \sum_{i=0}^{n} \hat{\lambda}_i v^i \\ &= \sum_{i=0}^{n} [\theta\lambda^1 + (1-\theta)\lambda^2]_i v^i\end{aligned}$$

$$= \theta \sum_{i=0}^{n} \lambda_i^1 v^i + (1-\theta) \sum_{i=0}^{n} \lambda_i^2 v^i$$
$$= \theta x^1 + (1-\theta) x^2 ,$$

where, for each $j = 1, 2$, $x^j = \Sigma_{i=0}^n \lambda_i^j v^i$. By part (i), $x^j \in X$, $j = 1, 2$. Also, since $\lambda^j \neq \hat{\lambda}$, $j = 1, 2$, and $\hat{\lambda}$ is the unique element of Λ satisfying $\hat{x} = \Sigma_{i=0}^n \hat{\lambda}_i v^i$, $x^j \neq \hat{x}$, $j = 1, 2$. On the other hand, since $\hat{x}$ is an extreme point of X and $\hat{x} = \theta x^1 + (1-\theta)x^2$, $\hat{x} = x^1 = x^2$. This contradiction implies that the assumption that $\hat{\lambda}$ is not an extreme point of Λ is untenable, and the proof of part (ii) is complete.

(iii) Assume that λ^* is an optimal solution for problem (P_λ). Let $x^* = \Sigma_{i=0}^n \lambda_i^* v^i$, and let $\bar{x}$ be an arbitrary element of X. Then, by part (i), $x^* \in X$ and, for some unique $\bar{\lambda} \in \Lambda$, $\bar{x} = \Sigma_{i=0}^n \bar{\lambda}_i v^i$. Since λ^* is an optimal solution for problem (P_λ),

$$\sum_{i=0}^{n} f(v^i)\lambda_i^* \leqq \sum_{i=0}^{n} f(v^i)\bar{\lambda}_i . \tag{1}$$

From part (i), $f_S(x^*) = \Sigma_{i=0}^n f(v^i)\lambda_i^*$ and $f_S(\bar{x}) = \Sigma_{i=0}^n f(v^i)\bar{\lambda}_i$. Therefore, from (1), $f_S(x^*) \leqq f_S(\bar{x})$. This implies that x^* is an optimal solution for problem (P_x). Also, since $f_S(x^*) = \Sigma_{i=0}^n f(v^i)\lambda_i^*$, the optimal values of problems (P_x) and (P_λ) are equal.

Assume that x^* is an optimal solution for problem (P_x). Let $\lambda^* \in \Lambda$ be the unique element of Λ satisfying $x^* = \Sigma_{i=0}^n \lambda_i^* v^i$. Let $\bar{\lambda}$ be an arbitrary element of Λ. Then, by part (i), $\bar{x} = \Sigma_{i=0}^n \bar{\lambda}_i v^i$ belongs to X. Since x^* is an optimal solution for problem (P_x), this implies that $f_S(x^*) \leqq f_S(\bar{x})$. From part (i), $f_S(x^*) = \Sigma_{i=0}^n f(v^i)\lambda_i^*$ and $f_S(\bar{x}) = \Sigma_{i=0}^n f(v^i)\bar{\lambda}_i$. Therefore, $\Sigma_{i=0}^n f(v^i)\lambda_i^* \leqq \Sigma_{i=0}^n f(v^i)\bar{\lambda}_i$. This implies that λ^* is an optimal solution for problem (P_λ), and the proof is complete.

Notice from Theorem 1 that to solve the linear program (P_x), the linear program (P_λ) can be solved instead. In particular, the optimal value of problem (P_λ) equals the optimal value of problem (P_x), and for any optimal extreme point solution $\lambda^* \in \Lambda$ for problem (P_λ), $x^* = [\Sigma_{i=0}^n \lambda_i^* v^i] \in X$ is an optimal extreme point solution for problem (P_x). The algorithm will make repeated use of these facts to find lower bounds and extreme points of X which achieve these lower bounds.

3. The Algorithm

3.1. ALGORITHM STATEMENT

In each step k of the algorithm, a global lower bound LB_k for m is found via branch and bound. To accomplish this, first an initial n-simplex S^0 is chosen that contains X and is a subset of Y. Then, in a typical step k, a radial subdivision of a sub-n-simplex S^{k-1} of S^0 is performed. This yields a partition T^k of S^{k-1} and a

new partition Q^k of S^0 of the form $Q^k = (Q^{k-1} \backslash S^{k-1}) \cup T^k$, where Q^{k-1} is the partition of S^0 obtained from step $k-1$. The partition Q^k of S^0 consists of a set of n-simplices denoted $\{S^j \mid j \in I(Q^k)\}$, where $I(Q^k)$ is a finite index set. For each n-simplex S^j in Q^k that does not belong to T^k, a lower bound w_j for the minimum of f over $S^j \cap X$ is available from some previous step. For each n-simplex S^j in Q^k that belongs to T^k, a lower bound w_j for the minimum of f over $S^j \cap X$ is computed in step k. Whether or not the value of w_j is computed prior to or during step k, a linear programming problem of the form (P_λ) (see Theorem 1) is solved to help compute it. To find the global lower bound LB_k for m, the algorithm computes the minimum $w_{\hat{j}}$ of the values w_j, $j \in I(Q^k)$, and sets $\mathrm{LB}_k = w_{\hat{j}}$.

The neighbor generation process used in the algorithm helps primarily to ensure that the algorithm is implementable and convergent. As a by-product, however, it helps to generate incumbent solutions.

The neighbor generation process is invoked in a typical step whenever an extreme point of X is found which will serve or is being considered for serving as the basis of a radial subdivision of some sub-n-simplex in the next step. When such an extreme point x is found, the neighbor generation process is invoked to search among all of the neighboring extreme points of x in X for those which have not yet served as vertices of any sub-n-simplex of S^0 created thus far in the algorithm. L is maintained throughout the algorithm which, at any given time, stores all extreme points of X found via the neighbor generation process which have not yet served as vertices of a sub-n-simplex of S^0 thus far created by the algorithm.

In a typical step k, the extreme point $x_{\hat{j}}$ of X found from solving the linear program (P_λ) that was solved to help compute $\mathrm{LB}_k = w_{\hat{j}}$ is used as the basis of the next radial subdivision in the next step. However, if $x_{\hat{j}}$ has already served as a vertex of some sub-n-simplex of S^0, the algorithm examines the list L. If $L = \emptyset$, the algorithm terminates since, as we shall see (Theorem 2), this guarantees that all extreme points of X have been found. If $L \neq \emptyset$, elements $\bar{x}$ of L are individually removed from L until one is found, if possible, which can serve as the basis of the next radial subdivision. To qualify for this, $\bar{x}$ must fail a fathoming test. In this way, each step k either finds an extreme point of X for use in a radial subdivision in the next step, or, failing to do so, terminates with $L = \emptyset$.

When the algorithm must remove points from L for *consideration for use* as the basis of a radial subdivision, the neighbor generation process is invoked for each such point. The neighbor generation process is also invoked whenever an extreme point of X is successfully found for *actual* use in the next radial subdivision, regardless of whether it was found from computing LB_k or from the list L.

As linear programs of the form (P_λ) are solved and as the neighbor generation process is invoked, the algorithm identifies various extreme point feasible solutions of problem (P). Each step k computes an upper bound UB_k for m given by the minimum objective function value in problem (P) of all such points thus far encountered. Throughout the algorithm, for each k, a record is kept of an

extreme point x^c of X (the *incumbent solution*) which satisfies $f(x^c) = \mathrm{UB}_k$. Thus, as a by-product of the neighbor generation process, incumbent solutions may be found which might not otherwise have been detected. Whenever $\mathrm{LB}_k = \mathrm{UB}_k$, the algorithm terminates.

We may now give a formal statement of the algorithm. In the algorithm statement, VL is a list maintained by the algorithm which, at any given time, stores all points that have served or have been considered for serving as a vertex of a subsimplex of S^0 created by the algorithm.

Step 0

0.1. Choose an n-simplex $S^0 \subseteq Y$ such that $X \subseteq S^0$ and let $Q^0 = \{S^0\}$. Let the vertices of S^0 be $\{v^0, v^1, \ldots, v^n\}$. Set $N = \emptyset$ and $\mathrm{VL} = \{v^0, v^1, \ldots, v^n\}$. For each $i \in \{0, 1, \ldots, n\}$ for which $v^i \in X$, set $N = N \cup \{v^i\}$ and generate the set E_i of extreme points of X adjacent to v^i in X which do not belong to VL. For each $i \in \{0, 1, \ldots, n\}$ for which $v^i \notin X$, set $E_i = \emptyset$. Set $L = \bigcup_{i=0}^n E_i$. If $N = \emptyset$, set $\mathrm{UB} = +\infty$ and go to step 0.2. If $N \neq \emptyset$, choose $x^c \in \operatorname{argmin}\{f(x) \mid x \in N \cup L\}$ and set $\mathrm{UB} = f(x^c)$.

0.2. Find the optimal value q_0 and an extreme point optimal solution $\lambda^* \in \Lambda$ for the linear program (P_λ) (see Theorem 1). Let $x^* = \sum_{i=0}^n \lambda_i^* v^i$, set $w_0 = q_0$, and set $\bar{x}^0 = x^*$. Generate the set E of extreme points of X adjacent to $\bar{x}^0$ which do not belong to VL. If $E \subseteq L$, go to Step 0.3. Otherwise, set $L = L \cup E$ and continue.

0.3. Set $\mathrm{LB}_0 = w_0$. Set $\mathrm{UB}_0 = \min\{\mathrm{UB}, \{f(x) \mid x \in \{\bar{x}^0\} \cup E\}\}$. If $\mathrm{UB} \neq +\infty$, choose $\hat{x} \in \operatorname{argmin}\{f(x^c), \{f(x) \mid x \in \{\bar{x}^0\} \cup E\}\}$ and set $x^c = \hat{x}$. Otherwise, choose $\hat{x} \in \operatorname{argmin}\{f(x) \mid x \in \{\bar{x}^0\} \cup E\}$ and set $x^c = \hat{x}$. If $\mathrm{LB}_0 = \mathrm{UB}_0$, conclude that x^c is an optimal solution for problem (P) and stop. Otherwise, set $k = 1$ and go to step k.

Step k, $k \geqq 1$. At the beginning of step k, a partition Q^{k-1} of S^0 is available from the previous step. Also available for each n-simplex $S^j \in Q^{k-1}$ are a lower bound w_j for the minimum of f over $S^j \cap X$ and an extreme point $x^j \in X$ found in the process of computing this lower bound. Assume that $S^{k-1} \in Q^{k-1}$ denotes an element of Q^{k-1} which contains the point $\bar{x}^{k-1}$ computed in the previous step.

k.1. If $\bar{x}^{k-1} \in L$, remove $\bar{x}^{k-1}$ from L. Using $\bar{x}^{k-1} \in S^{k-1}$ as a basis, perform a radial subdivision of S^{k-1} to obtain a partition T^k of S^{k-1}. Set $\mathrm{VL} = \mathrm{VL} \cup \{\bar{x}^{k-1}\}$.

k.2. For each n-simplex $S^j \in T^k$:

(i) With v^i, $i = 0, 1, \ldots, n$ set equal to the vertices of S^j, find the optimal value q_j and an extreme point optimal solution $\lambda^* \in \Lambda$ for the linear program (P_λ) (see Theorem 1);

(ii) Set $x^j = \sum_{i=0}^n \lambda_i^* v^i$; and

(iii) Set $w_j = \max\{q_j, w_{k-1}\}$.

k.3. Set $\mathrm{UB}_k = \min\{\mathrm{UB}_{k-1}, \{f(x^j) \mid S^j \in T^k\}\}$. If $\mathrm{UB}_k = f(x^j)$ for some j such that $S^j \in T^k$, set $x^c = x^j$.

k.4. Set $Q^k = (Q^{k-1} \backslash S^{k-1}) \cup T^k$.

k.5. Let $w_{\hat{j}} = \min\{w_j \mid j \in I(Q^k)\}$, and set $\mathrm{LB}_k = w_{\hat{j}}$.

k.6. If $\mathrm{LB}_k = \mathrm{UB}_k$, conclude that x^c is an optimal solution for problem (P) and stop. Otherwise, continue.

k.7. (i) If $x^{\hat{j}} \notin \mathrm{VL}$, set $\bar{x}^k = x^{\hat{j}}$ and go to step *k.8.* Otherwise, continue.

(ii) If $L = \emptyset$, conclude that x^c is an optimal solution for problem (P) and stop. Otherwise, remove any point $\bar{x}$ from L and continue.

(iii) Find any n-simplex $S^{\bar{j}} \in Q^k$ which contains $\bar{x}$. If $w_{\bar{j}} \leqq \mathrm{UB}_k$, set $\bar{x}^k = \bar{x}$ and go to step *k.8.* Otherwise, continue.

(iv) Generate the set E of extreme points of X adjacent to $\bar{x}$ which do not belong to VL. Set $\mathrm{VL} = \mathrm{VL} \cup \{\bar{x}\}$. If $E \subseteq L$, go to step *k.7*(ii). Otherwise, set $L = L \cup E$, find $\hat{x} \in \operatorname{argmin}\{f(x^c), \{f(x) \mid x \in E\}\}$, set $x^c = \hat{x}$, set $\mathrm{UB}_k = f(x^c)$, and go to step *k.7*(ii).

k.8. Generate the set E of extreme points of X adjacent to $\bar{x}^k$ which do not belong to VL. If $E \subseteq L$, set $k = k + 1$ and go to step k. Otherwise, set $L = L \cup E$, find $\hat{x} \in \operatorname{argmin}\{f(x^c), \{f(x) \mid x \in E\}\}$, set $x^c = \hat{x}$, set $\mathrm{UB}_k = f(x^c)$, set $k = k + 1$, and go to step k.

The set N in step 0.1 stores each vertex v^i of S^0 which is also an extreme point of X. Since $S^0 \supseteq X$, it is easily shown that for any vertex v^i of S^0, if $v^i \in X$, then v^i is an extreme point of X. Step 0.1 relies on this fact to construct N.

In step *k.7*(iii), if the lower bound $w_{\bar{j}}$ for the simplex $S^{\bar{j}}$ containing the point $\bar{x}$ removed from L exceeds UB_k, then control is passed to step *k.7*(iv). In step *k.7*(iv), the usual neighbor generation process for $\bar{x}$ and the associated incumbent update are performed, and $\bar{x}$ is added to the list VL. However, control then passes to step *k.7*(ii) rather than to step $k + 1$. In particular, no radial subdivision of the simplex $S^{\bar{j}}$ is performed, and no linear programming problems of the form (P_λ) that would be associated with this radial subdivision are solved. In this sense, the simplex $S^{\bar{j}}$ is *fathomed.* Notice that the point $\bar{x}$ is not used at this stage as a basis of a radial subdivision in the next step. However, since it was a candidate for such a use, $\bar{x}$ is added to the set VL.

3.2. CONVERGENCE

We will now show that the algorithm finds an optimal extreme point solution for problem (P) in a finite number of steps. To show this, the following result is needed.

THEOREM 2. *For any* $k \geqq 1$, *if* $L = \emptyset$ *in step k.7*(ii) *of the algorithm, then* VL *contains every extreme point of* X.

Proof. Assume that $k \geqq 1$. To prove the theorem, we will prove the contrapositive. Therefore, assume at step *k.7*(ii) that an extreme point $\hat{x} \in X$ exists which does not belong to VL. Then either (1) at least one extreme point of X adjacent to $\hat{x}$ belongs to VL or (2) no extreme point of X adjacent to $\hat{x}$ belongs to VL.

Case 1. At least one extreme point $\hat{\hat{x}}$ of X adjacent to $\hat{x}$ belongs to VL. For any

extreme point y of X added to VL through step $k.7$(ii) of the algorithm, from steps 0.1 and 0.2 and from steps w.1, w.7, and w.8, $1 \leqq w \leqq k$, the neighbor generation process guarantees that all extreme points of X adjacent to y which do not belong to VL are contained in L. Since $\hat{\hat{x}}$ is an extreme point of X adjacent to $\hat{x}$ which belongs to VL in step $k.7$(ii), and $\hat{x} \notin$ VL, this implies that $\hat{x} \in L$. Hence, in this case, $L \neq \emptyset$.

Case 2. No extreme point of X adjacent to $\hat{x}$ belongs to VL. Consider the extreme point $\bar{x}^0 \in X$. From step 1.1, $\bar{x}^0 \in$ VL. Since X is a polyhedron, there exist a finite number of extreme points $y^1, y^2, \ldots, y^t$ of X such that $\hat{x}$ is adjacent to y^1, y^h is adjacent to y^{h+1} for each $h = 1, 2, \ldots, t-1$, and y^t is adjacent to $\bar{x}^0$. Let $\hat{Y} = \{y \in \{y^1, y^2, \ldots, y^t\} \mid y \notin \text{VL}\}$. Since y^1 is adjacent to $\hat{x}$ and no extreme point of X adjacent to $\hat{x}$ belongs to VL, $y^1 \notin$ VL. Therefore, $\hat{Y} \neq \emptyset$ and we may choose an integer h^* such that

$$h^* = \max\{h \in \{1, 2, \ldots, t\} \mid y^h \notin \text{VL}\} .$$

If $h^* < t$, then y^{h^*} is adjacent to $y^{h^*+1} \in$ VL. If $h^* = t$, then y^{h^*} is adjacent to $\bar{x}^0 \in$ VL. In either case, y^{h^*} is an extreme point of X which does not belong to VL but which is adjacent to an extreme point of X which *does* belong to VL. Then, by the same reasoning as used in Case 1 for $\hat{x}$, it follows that $y^{h^*} \in L$. Since this implies that $L \neq \emptyset$, the proof is complete.

We may now show the following result.

THEOREM 3. *Whenever the algorithm terminates, the algorithm's current incumbent solution x^c is an extreme point optimal solution for problem (P).*

Proof. Consider any step k, $k \geqq 0$, of the algorithm. From step 0.3 or, for $k \geqq 1$, from step $k.5$,

$$\text{LB}_k = \min\{w_j \mid j \in I(Q^k)\} . \tag{2}$$

For each $j \in I(Q^k)$, by Theorem 1, q_j, calculated in step $\hat{k}.2$(i) for some $\hat{k} \leqq k$, is a lower bound for the minimum of f over $X \cap S^j$. In addition, for any sets A and B such that $A \subseteq B$, any lower bound for the minimum of f over B is a lower bound for the minimum of f over A. The latter two statements imply that for each $j \in I(Q^k)$, w_j, calculated in step $\hat{k}.2$(iii) for some $\hat{k} \leqq k$, is a lower bound for the minimum m_j of f over $X \cap S^j$. Therefore,

$$\min\{w_j \mid j \in I(Q^k)\} \leqq \min\{m_j \mid j \in I(Q^k)\} . \tag{3}$$

The set $\{S^j \mid j \in I(Q^k)\}$ is a partition of $S^0 \supseteq X$. This implies that the right-hand-side of inequality (3) is identical to m. From (2), this implies that $\text{LB}_k \leqq m$.

Suppose that the algorithm terminates in step k. Then either (1) $\text{LB}_k = \text{UB}_k$ or (2) $L = \emptyset$.

Case 1. $\text{LB}_k = \text{UB}_k$. Since $\text{UB}_k = f(x^c)$, where x^c is the current incumbent

extreme point solution, it follows that in this case, $LB_k = f(x^c)$. From the discussion above, $LB_k \leqq m$. Therefore, $f(x^c) \leqq m$. Since $x^c \in X$, this implies that $f(x^c) = m$, so that x^c is an extreme point optimal solution for problem (P).

Case 2. $L = \emptyset$. Then $k \geqq 1$, the algorithm terminates in step $k.7$(ii), and, from Theorem 2, VL contains every extreme point of X. From steps 0.1, 0.3 and steps $\hat{k}.1$, $\hat{k}.3$, $\hat{k}.7$, and $\hat{k}.8$, $1 \leqq \hat{k} \leqq k$, for any extreme point $\hat{x}$ of X contained in VL by step $k.7$, $UB_k \leqq f(\hat{x})$. Since $UB_k = f(x^c)$, where x^c is the current incumbent extreme point solution, and since VL contains every extreme point of X, this implies that $f(x^c) \leqq f(x)$ for all extreme points x of X. Since problem (P) has an optimal solution which is an extreme point of X, it follows that x^c is an extreme point optimal solution for problem (P), and the proof is complete.

The algorithm always eventually terminates by the following result.

THEOREM 4. *The algorithm terminates in a finite number of steps.*

Proof. Suppose, to the contrary, that the algorithm does not terminate in a finite number of steps. Then step $k.7$ is executed an infinite number of times. Therefore, in an infinite number of executions of step $k.7$, either (1) the point x^j in step $k.7$(i) does not lie in VL, or (2) a point $\bar{x}$ in step $k.7$(ii) is removed from L.

Case 1. In an infinite number of executions of step $k.7$, x^j in step $k.7$(i) does not lie in VL. From steps $k.1$ and $k.7$(i), whenever $x^j \notin$ VL in some step, it is added to the set VL in the next step. But each point x^j is an extreme point of X, and X contains a finite number of extreme points. The latter two statements imply that it is impossible for x^j not to belong to VL in an infinite number of executions of step $k.7$(i). Therefore, this case cannot occur.

Case 2. In an infinite number of executions of step $k.7$, a point $\bar{x}$ in step $k.7$(ii) is removed from L. From steps 0.1, 0.2, $k.7$ and $k.8$, L contains only extreme points of X, and the only points ever added to L by the algorithm are extreme points of X not already contained in L. Since X contains a finite number of extreme points, this implies that it is impossible to execute the removal of a point $\bar{x}$ in step $k.7$(ii) from L an infinite number of times. Therefore, this case cannot occur.

It follows that the assumption that the algorithm does not terminate in a finite number of steps is false, and the proof is complete.

Taken together, Theorems 3 and 4 imply that the algorithm is guaranteed to find an optimal solution for problem (P) in a finite number of steps. Furthermore, the optimal solution that it finds is an extreme point of X.

3.3. COMPUTATIONAL BENEFITS

The branch and bound-neighbor generation algorithm that we have presented for problem (P) has several attractive computational benefits.

First, as shown in Section 3.2, it is guaranteed to find an exact, extreme point optimal solution for problem (P) in a finite number of steps.

Second, it does not require f to be separable or even analytically defined. The only requirement for f is that it be a real-valued concave function on some open set Y in R^n that contains X.

Third, the algorithm requires no nonlinear computations and no determinations of convex envelopes or other underestimating functions.

Fourth, the linear programming problems solved during the branch and bound search do not grow in size and differ from one another in only one column of data. In particular, these linear programs are all of the form of problem (P_λ) given in Theorem 1. Thus, they all contain $(p+1)$ constraints and $(n+1)$ variables. Furthermore, for each n-simplex $S^j \in T^k$ in step *k.2*, the vertices of S^j are identical to those of its parent n-simplex S^{k-1}, except that in S^j the vertex $\bar{x}^{k-1}$ replaces one of the vertices $v^{\hat{i}}$ of S^{k-1}. As a result, the linear program (P_λ) associated with S^j which is solved in step *k.2* differs from the one solved earlier for S^{k-1} only in the $(p+1)$ coefficients for $\lambda_{\hat{i}}$ in the objective function and in the first p constraints. This implies that if the simplex method is used, for instance, to solve the linear programs (P_λ), the optimal basis found previously for the linear program (P_λ) associated with S^{k-1} can be used as a starting basis for the simplex method solution of the problem (P_λ) associated with S^j. In this way, the linear programs (P_λ) can, in general, be more quickly solved than they would be by using the traditional Phase I–Phase II approach of the simplex method [13].

Fifth, the algorithm incorporates a fathoming procedure to save unnecessary computations. In particular, in step *k.7*(iii), if $w_{\bar{j}} > \mathrm{UB}_k$, the simplex $S^{\bar{j}}$ is not partitioned and no linear programs of the form (P_λ) that would be associated with this partitioning process are solved.

The algorithm has other advantages. For instance, at each step of the algorithm, an incumbent extreme point solution x^c of X is available. This incumbent solution, while not necessarily an optimal solution, may often provide a user of the algorithm with a very good feasible solution for problem (P) when the number of steps executed grows large and the algorithm must therefore be prematurely terminated. In particular, the neighbor generation process can be expected to enhance the quality of this incumbent solution.

4. Preliminary Computational Experience

We have written a computer code in *C* which implements the proposed algorithm. We constructed 60 test problems, and used the code to solve these problems on an IBM 3090 Model 600J mainframe computer. In this section, we describe the code, the sources of the test problems, and the computational effort that the code required to solve the problems.

It has been pointed out [2, 7, 15] that the difficulty of finding a globally optimal solution to a concave minimization problem limits the sizes of problems solvable

by reasonable effort. Furthermore, the amount of reported information about the computational behavior of existing algorithms is limited. Since there are not generally-accepted test problems, we have constructed various test problems partially based on some data in the literature. Our goals in constructing and solving these test problems are limited. We merely sought to make some preliminary conclusions about the algorithm.

The code uses the simplex-based subroutines of the Optimization Subroutine Library [9] to solve the linear programming problems (P_λ) called for in the algorithm. In each run, the initial simplex S^0 that contains X was constructed separately using a method proposed by Horst [4]. The linear programming problems required by Horst's method are also solved by the simplex method procedures given in the subroutines of the Optimization Subroutine Library [9]. The branch and bound tree and the lists L and VL were maintained and processed using dynamic data structures.

The test problems were constructed as follows. Using data from Pegden and Petersen [17], we constructed eight nonempty, compact polyhedra of five different sizes. Each size is defined by the number of rows (m) and the number of columns (n) in the matrix B, where $X = \{x \in R^n \mid Bx \leqq g,\ x \geqq 0\}$, and $g \in R^m$. Next, we obtained six different types of concave functions from the literature to use as objective functions. The forms and sources of these functions are given in Table I. In function number 4, K represents any positive integer. We also obtained some variations of these functions by appending linear terms, negative quadratic terms, or both. Last, we combined the compact polyhedra with the concave functions in various combinations to create five categories of 12 problems each, where a category of problems is defined by the common size (m, n) of the feasible regions of the problems in the category.

Table I. Some objective function forms

Function no.	Functional form	Source
1	$-\left\lvert x_1 + \sum_{j=2}^{n} \frac{(j-1)}{j} x_j \right\rvert^{3/2}$	[7]
2	$-\left[1 + \left(\sum_{j=1}^{n} j x_j\right)^2\right]^{1/2}$	[7]
3	$-\left\lvert \sum_{j=1}^{n} \frac{1}{j} x_j \right\rvert - \ln\left[1 + \left\lvert \sum_{j=1}^{n} \frac{1}{j} x_j \right\rvert\right]$	[7]
4	$-K \sum_{j=1}^{n} x_j^2 + 2\left(\sum_{j=1}^{n-1} x_j x_{j+1}\right)$	[10]
5	$-129x_1^2 + 242x_1x_2 - 129x_2^2 + 1258x_1 - 1242x_2$	[20]
6	$-\left(\sum_{j=1}^{n} x_j^2\right) \ln\left(1 + \sum_{j=1}^{n} x_j^2\right)$	[7]

Table II. Computational results: averages

Category						
m	n	Iterations	Nodes	Pivots	Pseudo-pivots	CPU time
4	5	13.08	44.08	89.83	82.50	1.94
8	6	6.67	20.75	42.92	100.50	1.83
3	7	0.42	2.00	10.08	24.50	1.58
3	8	0.50	2.42	10.25	28.70	1.57
5	10	10.25	42.83	97.58	330.80	2.42

For simplicity, we solved only problems in which each extreme point of X that was encountered was nondegenerate. This shortened the code required to execute the neighbor generation process. This part of the code uses simplex-type pivots, which we call pseudo-pivots. Although the degenerate case can be handled by pseudo-pivots as well, the process in this case can become rather complicated [12, 13]. For this reason, in these initial computational experiments, we decided to eliminate the problems in which degenerate extreme points were encountered. For each solved problem, the computer code was executed until either an incumbent solution was found with an objective function value guaranteed to be within five percent of the optimal objective function value, or until the list L became empty.

Statistics summarizing the results of our computations are given in Tables II and III. In each table, five measures are used to evaluate the results: Number of iterations of the algorithm, number of nodes created in the branch and bound search (which equals the number of linear programs (P_λ) solved), total number of simplex method pivots required to solve the problems (P_λ), number of simplex-type pseudo-pivots, and the CPU time in seconds. Table II gives the averages of these measures by category. In Table III, the minimum and the maximum for each measure in each category are reported.

From Table II, we see that the computational effort required to solve these 60 problems does not necessarily monotonically increase as m or n increases. Also, from Table III, we note that the variance in the computational effort needed to solve the problems within a category can be relatively small or relatively large. These two observations seem to suggest that structural factors other than problem size significantly contributed to the computational effort required to solve these 60

Table III. Computational results: extremes

Category		Iterations		Nodes		Pivots		Pseudo-pivots		CPU time	
m	n	min	max	min	max	min	max	min	max	min	max
4	5	0	22	1	73	6	148	5	120	1.57	2.19
8	6	0	20	1	58	7	130	6	198	1.57	2.21
3	7	0	3	1	8	7	27	21	42	1.57	1.63
3	8	0	2	1	6	8	19	24	40	1.48	1.61
5	10	0	62	1	283	11	756	30	1230	1.57	6.58

problems. This is consistent with preliminary results obtained for other types of global optimization problems (see, e.g., [2, 7]).

Although the code successfully solved each of the 60 test problems, we cannot as yet draw any conclusions concerning its practicality for larger problems. More computational testing will be required to investigate this issue.

References

1. Benson, H. P. (1985), A Finite Algorithm for Concave Minimization over a Polyhedron, *Naval Research Logistics Quarterly* **32**, 165–177.
2. Benson, H. P. and Erenguc, S. S. (1990), An Algorithm for Concave Integer Minimization over a Polyhedron, *Naval Research Logistics* **37**, 515–525.
3. Heising-Goodman, C. D. (1981), A Survey of Methodology for the Global Minimization of Concave Functions Subject to Convex Constraints, *Omega* **9**, 313–319.
4. Horst, R. (1976), An Algorithm for Nonconvex Programming Problems, *Mathematical Programming* **10**, 312–321.
5. Horst, R. (1984), On the Global Minimization of Concave Functions: Introduction and Survey, *Operations Research Spektrum* **6**, 195–205.
6. Horst R. (1990), Deterministic Methods in Constrained Global Optimization: Some Recent Advances and New Fields of Application, *Naval Research Logistics* **37**, 433–471.
7. Horst, R., Thoai, N. V., and Benson, H. P. (1991), Concave Minimization via Conical Partitions and Polyhedral Outer Approximation, *Mathematical Programming* **50**, 259–274.
8. Horst, R. and Tuy, H. (1993), *Global Optimization (Deterministic Approaches)*, 2nd Edition, Springer, Berlin.
9. International Business Machines (1990), *Optimization Subroutine Library Guide and Reference*, International Business Machines, Mechanicsburg, Pennsylvania.
10. Konno, H. (1976), Maximization of a Convex Quadratic Function Subject to Linear Constraints, *Mathematical Programming* **11**, 117–127.
11. McCormick, G. P. (1972), Attempts to Calculate Global Solutions of Problems that may have Local Minima, in F. Lootsma (ed.), *Numerical Methods for Nonlinear Optimization*, Academic Press, London, 209–221.
12. Murty, K. G. (1968), Solving the Fixed Charge Problem by Ranking the Extreme Points, *Operations Research* **16**, 268–279.
13. Murty, K. G. (1983), *Linear Programming*, Wiley, New York.
14. Pardalos, P. M. and Rosen, J. B. (1986), Methods for Global Concave Minimization: A Bibliographic Survey, *SIAM Review* **28**, 367–379.
15. Pardalos, P. M. and Rosen, J. B. (1987), *Constrained Global Optimization: Algorithms and Applications*, Springer, Berlin.
16. Pardalos, P. M. and Schnitger, G. (1987), Checking Local Optimality in Constrained Quadratic Programming is NP-Hard, *Operations Research Letters* **7**, 33–35.
17. Pegden, C. D. and Petersen, C. C. (1979), An Algorithm (GIPC2) for Solving Integer Programming Problems with Separable Nonlinear Objective Functions, *Naval Research Logistics Quarterly* **26**, 595–609.
18. Rosen, J. B. (1983), Global Minimization of a Linearly Constrained Concave Function by Partition of Feasible Domain, *Mathematics of Operations Research* **8**, 215–230.
19. Soland, R. M. (1974), Optimal Facility Location with Concave Costs, *Operations Research* **22**, 373–382.
20. Tuy, H., Thieu, T. V., and Thai, N. Q. (1985), A Conical Algorithm for Globally Minimizating a Concave Function over a Closed, Convex Set, *Mathematics of Operations Research* **10**, 498–514.