A Finite Algorithm for Concave Minimization over a Polyhedron

Harold P. Benson
Department of Management and Administrative Sciences, University of Florida, Gainesville, Florida 32611

We present a new algorithm for solving the problem of minimizing a nonseparable concave function over a polyhedron. The algorithm is of the branch-and-bound type. It finds a globally optimal extreme point solution for this problem in a finite number of steps. One of the major advantages of the algorithm is that the linear programming subproblems solved during the branch-and-bound search each have the same feasible region. We discuss this and other advantages and disadvantages of the algorithm. We also discuss some preliminary computational experience we have had with our computer code for implementing the algorithm. This computational experience involved solving several bilinear programming problems with the code.

1. INTRODUCTION

The purpose of this article is to present a new algorithm for solving the concave minimization problem

$$\text{(P)} \qquad \begin{aligned} &\text{minimize } f(x), \\ &\text{subject to } x \in X, \end{aligned}$$

where X is a compact polyhedral set in $\mathbf{R}^n$, and f is a real-valued function that is concave and continuous, but not necessarily separable, on some open set Y in $\mathbf{R}^n$ that contains X. Problem (P) may have many locally optimal solutions. However, it is well known that for problem (P) there exists a globally optimal solution that is an extreme point of X. Let ϕ denote the optimal objective function value of problem (P).

The problem of minimizing a concave function over a polyhedron has been the subject of a great deal of research during the past 20 years. There are several reasons for this intense interest. First, since the problem may possess locally optimal solutions, special methods are required for solving it. Second, there are several other important mathematical programming models that can each be formulated as a minimization of a concave function over a polyhedron. These models include the zero–one integer linear programming problem [16,18], the linear fixed-charge problem, and the bilinear programming problem [5,22]. Finally, many practical problems, such as problems involving economies of scale and strategic weapons planning [6], can be represented as minimizations of concave functions over polyhedra.

The algorithms that have been developed for minimizing a concave function over a polyhedron each are based mainly upon either cutting planes, extreme point ranking, relaxation, branch-and-bound searches, or combinations of these approaches. The earliest of these algorithms, proposed by Tuy [24] in 1964, partitions the feasible

Naval Research Logistics Quarterly, Vol. 32, pp. 165–177 (1985)
 CCC 0028-1441/85/010165-13$04.00

domain by cone splitting. As the algorithm proceeds, portions of the feasible domain in these cones are successively eliminated from consideration. Underlying this elimination is the use of a cutting plane which is now known as a Tuy cut. Tuy's algorithm is not guaranteed to be finite.

Several other algorithms that also use cutting planes have been suggested since Tuy proposed his algorithm of 1964. Zwart [27] has developed such an algorithm. His algorithm is similar to Tuy's, but it guarantees that an approximate optimal solution will be found in a finite number of steps. Majthay and Whinston [14], using both Tuy cuts and cuts they developed called *facet* cuts, have proposed an algorithm which finds an exact optimal solution in a finite number of steps. In addition, Cabot [3] has proposed two algorithms that use cutting planes. One of these algorithms uses both Tuy cuts and cuts that are created by using convex underestimating functions. The second Cabot algorithm combines the approach of his first algorithm with extreme point ranking.

Although several algorithms that are based mainly upon cutting planes have been suggested, it seems that only one algorithm has been developed that uses mainly extreme point ranking. This algorithm, given by Taha [21], extends Murty's [15] idea of extreme point ranking for the linear fixed charge problem to the minimization of a concave function over a polyhedron.

At least three algorithms that rely mainly on the concept of relaxation have been developed. The algorithms of Falk and Hoffman [6] and of Carrillo [4] both involve solving a sequence of problems whose feasible regions are relaxations of X. Recently, Rosen [18] developed an algorithm that uses relaxations not only of X but also of certain subsets of X.

The remaining algorithms that can be used for minimizing a concave function over a polyhedron use branch-and-bound searches. In all of these algorithms, a set containing X is partitioned into smaller elements by the branching process. The earliest of these algorithms, developed by Falk and Soland [7], requires the objective function to be separable. Later, Soland [20] developed a slightly modified version of the Falk–Soland algorithm. In both of these algorithms, the branch-and-bound search yields partition elements that are rectangles. In Horst's [9] branch-and-bound algorithm, the partition elements are simplices. More recently, Thoai and Tuy [23] proposed a branch-and-bound algorithm in which the partition elements are cones. Of these four branch-and-bound algorithms, only the algorithms of Falk and Soland and of Soland guarantee that an exact optimal solution is found after a finite number of steps of the algorithm.

In this article we present a new branch-and-bound algorithm for solving the concave minimization problem (P). The algorithm finds a globally optimal extreme point solution for problem (P) in a finite number of steps. One of the major advantages of our algorithm is that the linear programming subproblems solved during the branch-and-bound search each have the same feasible region X. These subproblems differ only in their objective function coefficients. Thus, an optimal solution to one subproblem can be used as a feasible starting solution for solving the next subproblem. In addition, if X has a special structure in problem (P), this structure is preserved in the linear subproblems.

Part of the motivation for developing the algorithm given in this article came from the algorithm that Soland [20] developed for problem (P) when f is separable. In his algorithm, which is also of the branch-and-bound type, the linear programming subproblems all have the same feasible region. By using simplices instead of rectangles as partition elements, the algorithm in this article also uses linear subproblems which

all have the same feasible region. But our algorithm, unlike Soland's, does not require f to be separable.

The plan of this article is as follows: In Section 2 the new algorithm for solving problem (P) is presented. In Section 3 the validity and finiteness of the algorithm are shown. Advantages and disadvantages of the algorithm and some preliminary computational experience with it are given in Section 4. This computational experience involved solving several bilinear programming problems with the computer code that we have written to implement the algorithm.

2. THE ALGORITHM

2.1. Informal Description

The branch-and-bound algorithm to be presented for solving problem (P) is in certain respects similar to the branch-and-bound algorithm of Horst [9]. However, there are some major differences. Furthermore, whereas the convergence of Horst's algorithm is not simple to establish, the validity and finiteness of our algorithm can be shown quite easily. Therefore, we will present our algorithm in its entirety without assuming any knowledge of Horst's algorithm.

The following definitions will aid in our presentation of the algorithm.

DEFINITION [17]: Let $x^0,x^1, \ldots ,x^n$ be $(n + 1)$ affinely independent points in $\mathbf{R}^n$. The convex hull of $\{x^0,x^1, \ldots ,x^n\}$ is called a *simplex* or an *n-dimensional simplex*, and the points $x^0,x^1, \ldots ,x^n$ are called *vertices* of the simplex.

DEFINITION [9]: Let S be a compact set in $\mathbf{R}^n$. A set $\{S_1,S_2, \ldots ,S_q\}$ of finitely many compact subsets S_i, $i = 1,2, \ldots ,q$, of S is said to be a *partition* of S when S equals the union of all the sets S_i, $i = 1,2, \ldots ,q$, and each pair of sets S_i and S_j, $i \neq j$, intersect only on their boundaries relative to S.

DEFINITION [6]: The *convex envelope* of a function f taken over a nonempty subset S of its domain is that function g such that (i) g is a convex function defined over the convex hull of S; (ii) $g(x) \leq f(x)$ for all $x \in S$; and (iii) if h is a convex function defined over the convex hull of S that satisfies $h(x) \leq f(x)$ for all $x \in S$, then $h(x) \leq g(x)$ for any x in the convex hull of S.

To initiate the branch-and-bound algorithm for solving problem (P), an n-dimensional simplex S^{01} is chosen that contains X and is a subset of Y. In each step k of the algorithm, branching creates a new partition $\{S^{k1},S^{k2}, \ldots ,S^{kp_k}\}$ of S^{01} where, for each $j = 1,2, \ldots ,p_k$, S^{kj} is an n-dimensional simplex. For each $j = 1,2, \ldots ,p_k$, the linear function g_{kj} that underestimates f on S^{kj} and agrees with f at the vertices of S^{kj} is either available or calculated. Problem (P) is thereby separated into p_k underestimating linear programming problems (P_{kj}), $j = 1,2, \ldots ,p_k$, where, for each j, (P_{kj}) seeks to minimize g_{kj} over X. To find a lower bound for ϕ, the algorithm computes the minimum value among the optimal objective function values of the linear programs (P_{kj}), $j = 1,2, \ldots ,p_k$. This value is a lower bound LB for ϕ. As the linear programs (P_{kj}), $j = 1,2, \ldots ,p_k$, are solved for their optimal solutions x^{kj}, $j = 1,2, \ldots ,p_k$, an update is performed, if necessary, of the minimum value that f achieves over the optimal solutions x^{kj} and extreme points of X thus far encountered in the algorithm.

This value is an upper bound for ϕ, since, for each k, the points x^{kj}, $j = 1,2, \ldots, p_k$, lie in X. If, at the end of a step k, LB = UB or no more partitioning can take place, then the algorithm terminates. Otherwise, in step $k + 1$, the branching process is again invoked to create a more refined partition of S^{01}, and the process of obtaining lower and upper bounds for ϕ is repeated.

To complete this informal description of the algorithm, we will explain in more detail how the lower bound LB is calculated. Then we will explain how, when LB $\neq$ UB, the branching process creates, if possible, a more refined partition of S^{01}.

To explain the details of how the lower bound LB is calculated in step k of the algorithm, suppose that the current partition of S^{01} is $\{S^{k1}, S^{k2}, \ldots, S^{kp_k}\}$, where, for each $j = 1,2, \ldots, p_k$, S^{kj} is an n-dimensional simplex. To calculate the lower bound LB, for each $j = 1,2, \ldots, p_k$, a linear function g_{kj} that underestimates f on S^{kj} and agrees with f at the vertices of S^{kj} must first be found. As we shall see in the algorithm, for some values of j, this function g_{kj} is available from previous steps of the algorithm. For other values of j, g_{kj} must be calculated in step k.

To explain how g_{kj} is calculated, let us temporarily fix k and j, and let $x_j^0, x_j^1, \ldots, x_j^n$ be the vertices of an n-dimensional simplex S^{kj}. Horst [9, p. 318] has shown that there exists exactly one real-valued linear function g_{kj} on $\mathbf{R}^n$ that satisfies

$$g_{kj}(x_j^i) = f(x_j^i), \qquad i = 0,1, \ldots, n.$$

Furthermore, this function satisfies $g_{kj}(x) \leq f(x)$ for all $x \in S^{kj}$ and is actually the convex envelope of f taken over S^{kj}. Therefore, to calculate g_{kj}, the algorithm solves the $(n + 1)$ linear equations

$$\langle a, x_j^i \rangle + \alpha = f(x_j^i), \qquad i = 0,1, \ldots, n, \tag{1}$$

for the unknowns a and α, and sets $g_{kj}(x) = \langle a,x \rangle + \alpha$.

The availability of the partition $\{S^{k1}, S^{k2}, \ldots, S^{kp_k}\}$ and the convex envelopes $g_{k1}, g_{k2}, \ldots, g_{kp_k}$ in step k of the algorithm allows problem (P) to be separated into p_k underestimating linear programming problems (P_{kj}) given by

$$\begin{aligned} &\text{minimize } g_{kj}(x), \\ &\text{subject to } x \in X, \qquad j = 1,2, \ldots, p_k. \end{aligned}$$

The algorithm sets the lower bound LB for ϕ equal to the minimum optimal objective function value of these p_k linear programs.

In the algorithm, at the end of step k, if LB $\neq$ UB, then a more refined partition of S^{01} is constructed, if possible, by branching at the beginning of step $k + 1$. To accomplish this, an extreme point x^k of X is found, if possible, that is not a vertex of any of the simplices of the current partition of S^{01}. (If no such extreme point exists, then no more partitioning can take place and the algorithm terminates.) Next, a simplex S^{kj} containing x^k is found. The more refined partition of S^{01} constructed in step $k + 1$ is found by forming a partition $\{S^{k+1,1} S^{k+1,2}, \ldots, S^{k+1,v}\}$ of S^{kj}, and replacing S^{kj} in the partition $\{S^{k1}, S^{k2}, \ldots, S^{kp_k}\}$ from step k with $S^{k+1,1}, S^{k+1,2}, \ldots, S^{k+1,v}$. Here, v is some positive integer.

To form the partition $\{S^{k+1,1}, S^{k+1,2}, \ldots, S^{k+1,v}\}$ of S^{kj}, the algorithm uses the following procedure. First, the vertices $x_j^0, x_j^1, \ldots, x_j^n$ of S^{kj} are identified. Next, the unique values of $\alpha_0, \alpha_1, \ldots, \alpha_n$, which satisfy

$$\sum_{i=0}^{n} \alpha_i x_j^i = x^k,$$

$$\sum_{i=0}^{n} \alpha_i = 1,$$

and

$$\alpha_i \geq 0, \qquad i = 0,1, \ldots ,n,$$

are found. Then, up to $v = n + 1$ subsimplices of S^{kj} are formed in the following manner. For each α_i that is positive, the n-dimensional simplex with vertices $x_j^0, x_j^1, \ldots, x_j^{i-1}, x^k, x_j^{i+1}, \ldots, x_j^n$ is formed. From Horst [9, p. 317], the set $\{S^{k+1,1}, S^{k+1,2}, \ldots, S^{k+1,v}\}$ of all such subsimplices of S^{kj} that can be formed in this manner forms a partition of S^{kj}. Thus, by replacing S^{kj} in the partition $\{S^{k1}, S^{k2}, \ldots, S^{kp_k}\}$ of S^{01} from step k with $S^{k+1,1}, S^{k+1,2}, \ldots, S^{k+1,v}$, the algorithm forms a more refined partition of S^{01} in step $k + 1$.

We may now give a more formal statement of the algorithm.

2.2. Formal Algorithm Statement

STEP 0:

STEP 0.1. Choose an n-dimensional simplex $S^{01} \subseteq Y$ such that $X \subseteq S^{01}$. Let the vertices of S^{01} be $\{x_1^0, x_1^1, \ldots, x_1^n\}$. If $x_1^i \in X$ for at least one $i \in \{0,1, \ldots, n\}$, set $\text{UB} = \min \{f(x_1^i) | x_1^i \in X\}$, and set $x^c = x_1^i$, where x_1^i achieves this minimum. Otherwise, set $\text{UB} = +\infty$.

STEP 0.2. Find the convex envelope $g_{01}: S^{01} \to \mathbf{R}$ of f taken over S^{01} by solving the system of linear equations (1) with $j = 1$.

STEP 0.3. Find an optimal extreme point solution x^{01} for the linear programming problem (P_{01}) given by

$$\text{minimize } g_{01}(x),$$
$$\text{subject to } x \in X.$$

STEP 0.4. Set $\text{LB} = g_{01}(x^{01})$. If $\text{UB} \neq +\infty$, set $\text{UB} = \min \{f(x_1^i), f(x^{01})\}$, and set x^c equal to either x_1^i, or x^{01}, depending upon which of these two points achieves this minimum. If $\text{UB} = +\infty$, set $\text{UB} = f(x^{01})$ and $x^c = x^{01}$. If $\text{LB} = \text{UB}$, then conclude that x^c is an optimal solution for problem (P) and terminate. Otherwise, set $k = 1$, $p_0 = 1$, and $x^0 = x^{01}$, and go to Step 1.

STEP k, $k \geq 1$: Assume without loss of generality that $x^{k-1} \in S^{k-1,p_{k-1}}$.

STEP k.1. Form a partition $\{S^{k1}, S^{k2}, \ldots, S^{kv}\}$ of $S^{k-1,p_{k-1}}$ using the method explained in Section 2.1, where, for each $j = 1,2, \ldots, v$, S^{kj} is an n-dimensional simplex. For each $j = 1,2, \ldots, v$, let the vertices of S^{kj} be $\{x_j^0, x_j^1, \ldots, x_j^n\}$.

STEP k.2. For each $j = 1,2, \ldots, v$, find the convex envelope $g_{kj}: S^{kj} \to \mathbf{R}$ of f taken over S^{kj} by solving the system of linear equations (1).

STEP k.3. For each $j = 1,2, \ldots, v$, find an optimal extreme point solution x^{kj} for the linear programming problem (P_{kj}) given by

$$\text{minimize } g_{kj}(x),$$
$$\text{subject to } x \in X.$$

STEP k.4. Set UB = min $\{\text{UB}, f(x^{k1}), f(x^{k2}), \ldots, f(x^{kv})\}$. If UB = $f(x^{k\bar{j}})$ for some $\bar{j} \in \{1,2, \ldots, v\}$, set $x^c = x^{k\bar{j}}$.

STEP k.5. Set $p_k = v + p_{k-1} - 1$. If $k \neq 1$, then, for each $j = 1,2, \ldots, p_{k-1} - 1$, set $x^{k,v+j} = x^{k-1,j}$. If $k \neq 1$, then, for each $j = 1,2, \ldots, p_{k-1} - 1$, relabel $S^{k-1,j}$ and $g_{k-1,j}$ with $S^{k,v+j}$ and $g_{k,v+j}$, respectively.

STEP k.6. Compute

$$\text{LB} = \min_j \{g_{kj}(x^{kj})\}, \tag{2}$$

where j ranges over the set $\{1,2, \ldots, p_k\}$.

STEP k.7. If LB = UB, then conclude that x^c is an optimal solution for problem (P) and terminate. Otherwise, continue.

STEP k.8. Let $\bar{j}$ be a value of j in Eq. (2) at which the minimum is achieved.

(i) If $x^{k\bar{j}}$ is not a vertex of any of the simplices S^{kj}, $j = 1,2, \ldots, p_k$, set $x^k = x^{k\bar{j}}$, set $k = k + 1$, and go to Step k. Otherwise, continue.

(ii) Let $J = \{j \in \{1,2, \ldots, p_k\} | x^{kj}$ is not a vertex of any of the simplices S^{kj}, $j = 1,2, \ldots, p_k\}$. If $J \neq \phi$, set $x^k = x^{kj}$, where j is any element of J, set $k = k + 1$, and go to Step k. If $J = \phi$, continue.

(iii) Let $Q = \{x | x$ is an extreme point of X that is not a vertex of the simplices S^{kj}, $j = 1,2, \ldots, p_k\}$. If $Q \neq \phi$, let x^k be any element of Q, set $k = k + 1$, and go to Step k. Otherwise, conclude that x^c is an optimal solution for problem (P) and terminate.

Note that at any stage of the algorithm, the vector x^c stores that extreme point of X of those encountered thus far that achieves the smallest value UB of f. This vector x^c is called the *incumbent solution*. Also note that, at the beginning of any step k, $k \geq 1$, x^{k-1} is an extreme point of X found in the previous step that is not a vertex of any simplex in the current partition of S^{01}. At the beginning of step k, $k \geq 1$, the algorithm assumes without loss of generality that $x^{k-1} \in S^{k-1,p_{k-1}}$. Of course, if this is not true, a partition of the appropriate set $S^{k-1,j}$, instead of the set $S^{k-1,p_{k-1}}$, would be formed in Step k.1, and the renaming and relabeling in Step k.5 would be adjusted accordingly.

Finally, we can now see why the algorithm requires f to be continuous, as well as concave, on some open set Y containing X. If f were discontinuous at some point on the relative boundary of X, then it might not be possible to extend the definition of f onto S^{01} in such a way so as to obtain a finite, concave function on S^{01}.

2.3. Choosing an Initial Containing Simplex

Step 0 of the algorithm calls for choosing an n-dimensional simplex $S^{01} \subseteq Y$ such that $X \subseteq S^{01}$. Since X is a compact set, this initial containing simplex S^{01} will exist as long as Y is "large" enough. In this section, two methods are suggested for finding S^{01}. Assume without loss of generality in this section that $X = \{x \in \mathbf{R}^n | Ax \leq b\}$, where A is an $m \times n$ matrix and $b \in \mathbf{R}^m$.

If $X \subseteq \{x \in \mathbf{R}^n | x \geq 0\}$, then the following simple method, suggested in [9], may suffice in constructing S^{01}. First, solve the linear programming problem

$$\beta = \max \langle e,x \rangle,$$
$$\text{subject to } x \in X,$$

where $e \in \mathbf{R}^n$ is a vector of 1's. Then, let the vertices of S^{01} be given by $x_1^0 = 0$, $x_1^i = \beta e^i$, $i = 1,2, \ldots ,n$, where $e^i \in \mathbf{R}^n$ has all of its entries equal to zero, except for the ith entry, which equals 1. If S^{01} is constructed in this way, then it can be shown that $X \subseteq S^{01}$. Therefore, as long as $S^{01} \subseteq Y$, this method suffices for finding an initial containing simplex.

The "smaller" the initial containing simplex S^{01} is, the more efficient we expect the algorithm to be. Falk and Hoffman [6] have suggested a method for constructing S^{01} which may often yield a simplex that is a strict subset of the one that the simple method above would yield. Their method is as follows. One of the vertices x_1^0 of S^{01} is chosen to be any extreme point of X. For instance, x_1^0 may be chosen to be an extreme point of X that is a locally optimal solution for problem (P), although this is not necessary. If exactly n constraints of X are binding at x_1^0, then the Falk–Hoffman method can be used. Otherwise, it cannot. Suppose, then, that exactly n constraints of X are binding at x_1^0. The cone formed by these n constraints, if truncated properly, will form S^{01}.

If possible, the Falk–Hoffman method uses one of the nonbinding constraints of X to truncate the above cone. To see if this is possible, the simplex tableau corresponding to the extreme point x_1^0 of X is constructed. At x_1^0, this tableau displays the equations $Tt + s = \tilde{b}$, where s consists of the current basic variables and t consists of the current nonbasic variables. Since X is compact, $m > n$. Therefore, at least one of the basic variables s is a slack variable for a constraint of X. The method calls for examining each basic variable s that is a slack variable for a constraint of X. If, for one such variable, the coefficients in its row in T are all positive, then that constraint truncates the aforementioned cone to form S^{01}. The remaining vertices x_1^i, $i = 1,2, \ldots ,n$, of S^{01} are then found by performing a linear programming pivot on each of these positive coefficients.

If there is no basic variable, among the variables s that are slack variables for the constraints of X, for which the corresponding row elements of T are all positive, then no constraint of X truncates the cone formed by the n binding constraints at x_1^0. In this case, the Falk–Hoffman method calls for solving the linear program

$$\gamma = \max \langle e,t \rangle,$$
$$\text{subject to } Tt + s = \tilde{b}; \qquad s,t \geq 0.$$

Next, the constraint $\langle e,t \rangle \leq \gamma$ is added to the system $Tt + s = \tilde{b}$. Then, the remaining vertices x_1^i, $i = 1,2, \ldots ,n$, of S^{01} are found by performing a linear programming pivot on each of the unit coefficients of the t variables in the row that was created by adding the new constraint.

As in the first method, as long as $S^{01} \subseteq Y$, the Falk–Hoffman method yields a valid initial containing simplex. When it succeeds, the Falk–Hoffman method is often preferred to the first method because it frequently yields a tighter initial containing simplex.

3. VALIDITY AND FINITENESS

The branch-and-bound algorithm given in Section 2 is valid provided that, whenever it stops, it yields an optimal solution for problem (P). By the following theorem, the algorithm is, indeed, valid.

THEOREM 1: Whenever the branch-and-bound algorithm given in Section 2 terminates, the algorithm's current incumbent solution x^c is an optimal solution for problem (P).

PROOF: Consider any step k, $k \geq 0$, of the algorithm. From Step 0.4 or, for $k \geq 1$, from Steps $k.3$ and $k.6$ of the algorithm, the lower bound LB calculated in Step k is given by

$$\text{LB} = \min_j \{\phi_{kj}^1\}, \tag{3}$$

where j ranges over the set $\{1,2, \ldots ,p_k\}$ and ϕ_{kj}^1 is, for each j, the optimal objective function value of the linear program (P_{kj}). Now consider any $j \in \{1,2, \ldots ,p_k\}$. By definition of ϕ_{kj}^1, since $(X \cap S^{kj}) \subseteq X$, ϕ_{kj}^1 is less than or equal to the optimal objective function value ϕ_{kj}^2 of the problem

$$\text{minimize } g_{kj}(x),$$
$$\text{subject to } x \in X \cap S^{kj}.$$

Also, by definition of the convex envelope, ϕ_{kj}^2 is less than or equal to the optimal objective function value ϕ_{kj}^3 of the problem

$$\text{minimize } f(x),$$
$$\text{subject to } x \in X \cap S^{kj}.$$

Therefore, $\phi_{kj}^1 \leq \phi_{kj}^3$ for any $j \in \{1,2, \ldots ,p_k\}$. Taking the minimum on both sides of this inequality over $j \in \{1,2, \ldots ,p_k\}$, we obtain, from Eq. (3),

$$\text{LB} \leq \min_j \phi_{kj}^3, \tag{4}$$

where j ranges over the set $\{1,2, \ldots ,p_k\}$. But from the definition of ϕ_{kj}^3, since $\{S^{k1},S^{k2}, \ldots ,S^{kp_k}\}$ is a partition of $S^{01} \supseteq X$, $\phi = \min_j \phi_{kj}^3$, where j ranges over $\{1,2, \ldots ,p_k\}$. Therefore, from (4), we obtain LB $\leq \phi$. Furthermore, at any point in the algorithm, $\phi \leq$ UB. This is because UB $= f(x^c)$, where x^c is that extreme point of X at which f achieves its smallest value among all extreme points of X generated to that point in the algorithm.

Suppose that the algorithm terminates in Step $k.8$ (iii) for some $k \geq 1$. Then every extreme point of X is a vertex of some simplex S^{kj}, $j \in \{1,2, \ldots ,p_k\}$. From Steps 0.1, 0.4, and $k.4$, $k \geq 1$, and from the method for forming partitions used in Steps $k.1$ for $k \geq 1$, UB equals the minimum of $f(x^{e,i})$, $i = 1,2, \ldots ,w$, where $x^{e,1},x^{e,2}, \ldots ,x^{e,w}$ are all the extreme points of X. Since problem (P) has an optimal solution which is an extreme point of X and $f(x^c) =$ UB, x^c is an optimal solution for problem (P).

Now suppose that the algorithm terminates in Step 0.4, or, for some $k \geq 1$, in Step $k.7$. Let $x \in X$. Then, from the discussion above and the definition of ϕ, LB $\leq \phi \leq f(x)$. But since the algorithm terminates in Step 0.4 or, for some $k \geq 1$, in Step $k.7$, and UB $= f(x^c)$, we have LB $=$ UB $= f(x^c)$. Taken together, the last two sentences imply that $f(x) \geq f(x^c)$. Since x was an arbitrary element of X, this completes the proof.

Note that if the algorithm stops, it finds an extreme point x^c of X that is an optimal

solution for problem (P). From the next theorem, the algorithm is guaranteed to terminate eventually.

THEOREM 2: The algorithm of Section 2 terminates in a finite number of steps.

PROOF: Since X is a polyhedron, it has a finite number of extreme points [17, Corollary 19.1.1]. Let E be the union of the set of extreme points of X and the set of vertices of S^{01}. Every n-dimensional simplex S^{kj} generated by the algorithm is defined by an appropriate set of $(n + 1)$ points in E (these points are the vertices of S^{kj}). Since the number of points in E is finite, the number of possible n-dimensional simplices S^{kj} that the algorithm can generate is finite. For each $k \geq 1$, in Step $k.1$, since x^{k-1} is not a vertex of $S^{k-1,p_{k-1}}$, at least two new simplices are created. From the latter two statements, we conclude that the algorithm must terminate in a finite number of steps.

Taken together, the two theorems and their proofs in this section imply that the branch-and-bound algorithm presented in Section 2 is guaranteed to find an optimal solution for problem (P) which is an extreme point of X after a finite number of iterations of the algorithm have been executed.

4. ADVANTAGES, DISADVANTAGES, AND PRELIMINARY COMPUTATIONAL EXPERIENCE

4.1. Advantages and Disadvantages

The branch-and-bound algorithm that we have presented has several basic advantages. Four of these are that the algorithm (a) is finite, (b) finds an exact, globally optimal solution for problem (P), (c) does not require f to be separable, and (d) does not require the construction of linear or convex underestimators of f over X or over subsets of X.

Advantage (d) is important because it allows our algorithm to be applicable to a wider variety of problems than would otherwise be possible (see Sec. 4.2 for an example). Of the algorithms we have encountered for minimizing a concave function over a polyhedron, only those by Majthay and Whinston [14], Falk and Hoffman [6], Carillo [4], and Rosen [18] possess the four basic advantages (a)–(d). However, in both the Majthay–Whinston and Falk–Hoffman algorithms, in contrast to ours, the sizes of the linear subproblems that are generated grow from iteration to iteration. In the Carillo algorithm, in contrast to ours, extra effort may be spent in generating points that do not lie in X and which, therefore, are not candidates for optimal solutions to problem (P). Rosen's algorithm requires more conditions to hold on f and X than our algorithm, including that f be twice differentiable with a bounded Hessian and that X possess no degenerate extreme points.

A unique advantage of our algorithm compared with other branch-and-bound algorithms for solving problem (P) when f is nonseparable is that the linear programming subproblems (P_{kj}) each possess the same feasible region X as problem (P). These subproblems differ only in their objective function coefficients. Thus, as mentioned in Section 1, an optimal solution to one subproblem can be used as a feasible starting solution for solving the next subproblem. In many cases, this may allow these subproblems to be solved in fewer linear programming pivots than would otherwise be

possible. In addition, if X has a special structure in problem (P), then this special structure is preserved in the linear subproblems (P_{kj}). This may allow every linear subproblem to be solved by a specialized algorithm, increasing efficiency significantly.

There are several other advantages of our algorithm. Among these are the facts that, to use the algorithm, nondegeneracy of the extreme points of X is not required, and that, at each step of the algorithm, an incumbent solution x^c that lies in 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 algorithm becomes lengthy and must be terminated prematurely.

One of the potential disadvantages of the algorithm is that, when carried out on a computer, a relatively large amount of computer storage may be required for some problems. This is to be expected in branch-and-bound algorithms of this type. However, this problem is aggravated in our algorithm by the fact that, in step k, none of the n-dimensional simplices S^{kj} can be eliminated from consideration as potentially containing an optimal solution for problem (P). In branch-and-bound algorithms, sets or categories of solutions that can be shown not to contain an optimal solution are frequently eliminated or *fathomed*. In our algorithm, however, even if the optimal objective function value for problem (P_{kj}) exceeds UB for some k and j, S^{kj} cannot be fathomed. The reason is that some extreme point $x^{w\bar{j}}$ of X in S^{kj} may be generated at a step $w > k$ in the algorithm, where, with $k = w$, $\bar{j}$ is a value of j in (2) at which the minimum in (2) is achieved. Thus, at any step k, an optimal solution for problem (P) may lie in any of the simplices $S^{k1}, S^{k2}, \ldots, S^{kp_k}$.

Another potential disadvantage of our algorithm is that, for certain problems, a relatively large number of linear subproblems (P_{kj}) might be considered. We expect that this may occur because instead of having $X \cap S^{kj}$ as the feasible region of a linear subproblem, the algorithm has X as the feasible region of (P_{kj}). Having X as the feasible region would be expected to yield smaller lower-bound values LB than would otherwise be attained. This, in turn, might increase the number of steps of the algorithm and, thus, the number of linear subproblems required to solve problem (P).

A third potential disadvantage of the algorithm is that, for certain problems, a "tight" initial containing simplex S^{01} may not exist. This might cause the algorithm to require many steps to solve such problems.

A fourth potential disadvantage of the algorithm relates to Step $k.8$ for $k \geq 1$. To solve a given problem (P), in some step $k \geq 1$, the algorithm may require the determination of whether or not the set Q in Step $k.8$ (iii) is empty. If $Q \neq \phi$, an element of Q must be found in that step. Although these tasks can be accomplished by searching among the extreme points of X for an element of Q, the computational burden involved may be great if the number of extreme points of X is large.

4.2. Preliminary Computational Experience

We have written a FORTRAN computer code, called CONMIN, which implements our proposed algorithm. To obtain a preliminary assessment of the computational properties of our algorithm, we have used the computer code to solve several bilinear programming problems. Let A be an $m \times n$ matrix, B be a $k \times s$ matrix, and Q be a $k \times n$ matrix. Also, assume $a \in \mathbf{R}^m$, $b \in \mathbf{R}^s$, $c \in \mathbf{R}^n$, and $d \in \mathbf{R}^k$. Then, the bilinear programming problem (BL) can be written

$$\text{(BL)} \qquad \begin{aligned} &\text{minimize } \langle c,x\rangle + y^TQx + \langle d,y\rangle, \\ &\text{subject to } Ax \le a, \qquad B^Ty \le b, \\ &\qquad\qquad x \ge 0, \qquad\quad y \ge 0, \end{aligned}$$

where $X = \{x \in \mathbf{R}^n | Ax \le a, x \ge 0\}$ and $Y = \{y \in \mathbf{R}^k | B^Ty \le b, y \ge 0\}$ are nonempty, compact sets. In general, the objective function for problem (BL) is neither convex nor concave. However, it is known that problem (BL) can be reformulated as a concave minimization problem (P) [5,22]. Nevertheless, we are not aware of anyone who has actually solved problem (BL) by reformulating it as a concave minimization problem and applying one of the algorithms for concave minimization. We are only aware of several special algorithms for solving problem (BL) directly, including those of Al-Khayyal and Falk [1], Altman [2], Gallo and Ulkücü [8], Konno [12], Sherali and Shetty [19], and Vaish and Shetty [25,26].

To cast problem (BL) into the form of problem (P), the particular function f that we used is defined by

$$f(x) = \langle c,x\rangle + g(x),$$

where

$$\begin{aligned} &g(x) = \min \langle y,Qx + d\rangle, \\ &\text{subject to} \qquad B^Ty \le b, \qquad y \ge 0, \end{aligned}$$

for any $x \in \mathbf{R}^n$. It can be shown that f and g are continuous, concave functions on $\mathbf{R}^n$. It can also be shown that with f, X, and Y defined as above, for any optimal solution x^* for problem (P), there exists a $y^* \in Y$ such that (x^*,y^*) is an optimal solution for problem (BL) and $f(x^*) = \langle c,x^*\rangle + y^{*T}Qx^* + \langle d,y^*\rangle$. To obtain y^*, one need only find an optimal solution for the linear program

$$\begin{aligned} &\text{minimize } \langle y,Qx^* + d\rangle, \\ &\text{subject to } B^Ty \le b, \qquad y \ge 0. \end{aligned}$$

Using the above definitions of f and X, we casted 17 bilinear programming problems (BL) into the form of problem (P) and solved them using our computer code CONMIN for the proposed branch-and-bound algorithm. In CONMIN, to solve each linear programming subproblem (P_{kj}) beyond the first, the starting solution used is the optimal solution to the previously solved subproblem. To solve the systems of equations called for by the algorithm, CONMIN uses a subroutine of IMSL [10]. In addition, it uses the subroutines of PIMPS [11] to accomplish any necessary list processing. For each of the 17 problems, the initial containing simplex called for by the algorithm was constructed by hand via the Falk–Hoffman method and was supplied to CONMIN as input data.

The computational results of solving the 17 bilinear programming problems (BL) are presented in Table 1. Those problems for which no sources are listed were constructed by the author. Column (a) gives the step number at which the algorithm terminated. Column (b) gives the total number of linear programming subproblems (P_{kj}) that were solved, and column (c) gives the total number of linear programming pivots that were required to solve these subproblems. Column (d) gives the number of linear programming pivots that were needed to make all the evaluations of the function g that the algorithm called for. The problems were solved on an IBM 3081D36 computer. Of these 17 problems, problem 17 took the largest amount of CPU time

Table 1. Computational experience.

Problem	Source	m	n	s	k	(a) Step	(b) Subproblems	No. of L.P. pivots (c) Subproblems	No. of L.P. pivots (d) Evaluations of g
1	[13]	2	2	2	2	1	3	8	10
2	[12]	3	2	3	2	3	7	18	13
3	[8]	4	2	4	2	1	3	10	17
4	[13]	4	2	4	2	0	1	4	7
5		4	2	4	2	0	1	4	7
6		4	2	4	2	1	3	6	7
7	[13]	3	3	3	3	4	11	36	31
8	[19]	5	2	3	2	2	5	13	14
9		4	4	3	3	6	14	29	15
10		4	4	3	3	4	12	22	12
11	[13]	4	4	4	4	21	63	244	74
12	[13]	5	5	5	5	52	141	507	142
13		4	7	3	3	7	17	39	24
14		4	7	3	3	3	10	24	27
15		4	7	3	3	2	5	15	24
16		4	7	3	3	0	1	12	21
17	[13]	6	6	6	6	88	209	616	195

(2.15 seconds). None of the 17 problems required the execution of the potentially computationally burdensome Step k.8 (iii) for any $k \geq 1$.

Since we have computational experience with only 17 special problems, we can only make tentative conclusions about the computational properties of our proposed algorithm. First, it seems that the algorithm is quite practical for relatively small concave minimization problems. For instance, the maximum CPU time required to solve the 17 bilinear programs was 2.15 seconds, and no computer storage problems were encountered. Second, it appears that the number of linear subproblems that must be solved may grow rather rapidly as problem size increases. However, these subproblems seem to be solved relatively quickly. For instance, the 209 linear subproblems solved in problem 17 each have six constraints (besides nonnegativity conditions) and six variables, but, on the average, were solved in only about three linear programming pivots. This is, no doubt, at least partially because the algorithm exploits the fact that the subproblems differ only in their objective function coefficients.

To draw more definite conclusions concerning the computational properties of the algorithm would require more elaborate experimentation. We are in the process of conducting this experimentation.

ACKNOWLEDGMENT

The author is sincerely indebted to Selcuk Erenguc for his suggestions and help with the convergence properties of the algorithm and with the computer code CONMIN.

REFERENCES

[1] Al-Khayyal, F., and Falk, J. E., "Jointly Constrained Biconvex Programming," *Mathematics of Operations Research*, **8**, 273–286 (1983).

[2] Altman, M., "Bilinear Programming," *Bulletin de l'Academie Polonaise des Sciences*, **16,** 741–746 (1968).
[3] Cabot, A. V., "Variations on a Cutting Plane Method for Solving Concave Minimization Problems with Linear Constraints," *Naval Research Logistics Quarterly*, **21,** 265–274 (1974).
[4] Carrillo, M. J., "A Relaxation Algorithm for the Minimization of a Quasiconcave Function on a Convex Polyhedron," *Mathematical Programming*, **13,** 69–80 (1977).
[5] Carvajal-Moreno, R., "Minimization of Concave Functions Subject to Linear Constraints," Report ORC-3, Operations Research Center, University of California, Berkeley, CA, 1972.
[6] Falk, J. E., and Hoffman, K. R., "A Successive Underestimation Method for Concave Minimization Problems," *Mathematics of Operations Research*, **1,** 251–259 (1976).
[7] Falk, J. E., and Soland, R. M., "An Algorithm for Separable Nonconvex Programming Problems," *Management Science*, **15,** 550–569 (1969).
[8] Gallo, G., and Ulkücü, A., "Bilinear Programming: An Exact Algorithm," *Mathematical Programming*, **12,** 173–194 (1977).
[9] Horst, R., "An Algorithm for Nonconvex Programming Problems," *Mathematical Programming*, **10,** 312–321 (1976).
[10] International Mathematical and Statistical Libraries, Inc., *The IMSL Library Reference Manual*, IMSL, Houston, TX, 1982.
[11] Koehler, G. J., *Program for Interactive Multiple Process Simulation (PIMPS) Users Guide*, mimeographed notes, Gainesville, FL, 1976.
[12] Konno, H., "A Cutting Plane Algorithm for Solving Bilinear Programs," *Mathematical Programming*, **11,** 14–27 (1976).
[13] Konno, H., "Maximization of a Convex Quadratic Function Subject to Linear Constraints," *Mathematical Programming*, **11,** 117–127 (1976).
[14] Majthay, A., and Whinston, A., "Quasiconcave Minimization Subject to Linear Constraints," *Discrete Mathematics*, **9,** 35–59 (1974).
[15] Murty, K. G., "Solving the Fixed Charge Problem by Ranking the Extreme Points," *Operations Research*, **16,** 268–279 (1968).
[16] Raghavachari, M., "On Connections Between Zero-One Integer Programming and Concave Programming under Linear Constraints," *Operations Research*, **17,** 680–684 (1969).
[17] Rockafellar, R. T., *Convex Analysis*, Princeton U.P., Princeton, NJ, 1970.
[18] Rosen, J. B., "Global Minimization of a Linearly Constrained Concave Function by Partition of Feasible Domain," *Mathematics of Operations Research*, **8,** 215–230 (1983).
[19] Sherali, H. D., and Shetty, C. M., "A Finitely Convergent Algorithm for Bilinear Programming Problems Using Polar Cuts and Disjunctive Face Cuts," *Mathematical Programming*, **19,** 14–31 (1980).
[20] Soland, R. M., "Optimal Facility Location with Concave Costs," *Operations Research*, **22,** 373–382 (1974).
[21] Taha, H. A., "Concave Minimization Over a Convex Polyhedron," *Naval Research Logistics Quarterly*, **20,** 533–548 (1973).
[22] Thieu, T. V., "Relationship Between Bilinear Programming and Concave Minimization under Linear Constraints," *Acta Mathematica Vietnamica*, **5,** 106–113 (1978).
[23] Thoai, N. V., and Tuy, H., "Convergent Algorithms for Minimizing a Concave Function," *Mathematics of Operations Research*, **5,** 556–566 (1980).
[24] Tuy, H., "Concave Programming Under Linear Constraints," *Soviet Mathematics*, **5,** 1437–1440 (1964).
[25] Vaish, H., and Shetty, C. M., "The Bilinear Programming Problem," *Naval Research Logistics Quarterly*, **23,** 303–309 (1976).
[26] Vaish, H., and Shetty, C. M., "A Cutting Plane Algorithm for the Bilinear Programming Problem," *Naval Research Logistics Quarterly*, **24,** 83–94 (1977).
[27] Zwart, P. B., "Global Maximization of a Convex Function with Linear Inequality Constraints," *Operations Research*, **22,** 602–609 (1974).