Journal of Global Optimization **18:** 107–128, 2000.

Finite Exact Branch-and-Bound Algorithms for Concave Minimization over Polytopes

MARCO LOCATELLI[1] and NGUYEN V. THOAI
Universitá degli Studi di Firenze,
Dipartimento di Sistemi e Informatica,
Via di S.Marta, 3-50139 Firenze
Italy;

University of Trier,
Department of Mathematics,
D-54286 Trier, Germany

(Received 26 January 1998; accepted in revised foem 18 February 2000)

Abstract. In this paper simplicial branch-and-bound algorithms for concave minimization problems are discussed. Some modifications of the basic algorithm are presented, mainly consisting in rules to start local searches, introduction of cuts and updates of the original objective function. While some of these tools are not new in the literature, it is the first time, to the authors' knowledge, that they are used to guarantee the finiteness of a simplicial branch-and-bound approach.

Key words: Concave minimization problems, Branch-and-bound, Local searches, Concavity cuts.

1. Introduction

In this paper the problem of minimizing a concave function over a polytope is considered. The statement of the problem is the following

$$\begin{aligned} &\min\ f(x) \\ &x \in P, \end{aligned} \tag{1.1}$$

where $P = \{x \in R^n : Ax \leq b\}$, $A = (a_1, \ldots, a_m)^T \in R^{m \times n}$, $b \in R^m$, is a full-dimensional polytope, and f is a concave function over R^n. In what follows the optimal value of (1.1) will be denoted by f^*, while its solution set $\{x \in P : f(x) = f^*\}$ will be denoted by $X^*(f, P)$. It is also assumed that no constraint defining P is redundant. Even if it is known that at least one optimal solution of the problem is one of the vertices of P, this is a well known NP-hard problem even when the objective function and the feasible region have some special forms (e.g. quadratic over a hypercube, see [13]). Extensive surveys about the theoretical and algorithmic aspects of this problem together with some of its applications are given in [3, 6, 8, 9 and 12]. The algorithms for its solution

are generally subdivided in three main classes: enumerative methods, successive approximation methods and branch-and-bound methods. While algorithms from the first two classes can often be proven to return an optimal solution after finitely many iterations by exploring, in the worst case, all the vertices of P, the same is not generally true for branch-and-bound methods, which, on the other hand, have proven to work well in practice. Branch-and-bound methods often only guarantee to be finite for computing an ε-optimal solution. While it is known that if ε is smaller than the difference between the values of f at the best and the second best vertex, then a vertex x^0 which is ε-optimal is a global optimal solution, the choice of ε turns out to be a critical one: if ε is large, then few iterations are needed but the accuracy of the solution may be poor, while if ε is small, many iterations may be required. In Section 6 the dependency of the performance on the required accuracy is shown through some examples.

Some finite exact algorithms in the field of branch-and-bound methods for concave optimization have been appeared in the literature, for example [4], with the so called END process for conical branch-and-bound algorithms, [14] in the field of rectangular branch-and-bound algorithms for separable concave functions, [1, 2, 10 and 15] in the field of simplicial branch-and-bound algorithms.

In this paper modifications of convergent simplicial algorithms in order to make them finite are proposed. In Section 2 a brief introduction of simplicial branch-and-bound algorithms is given. In Section 3 some of the modifications mentioned above are introduced. The basic idea is to combine rules to decide whether to start local searches with the introduction of cuts in order to enforce finiteness of simplicial algorithms which have only been proven to be convergent. Since the finiteness proof relies on an assumption which can not be guaranteed to hold for all possible concave problems, in Section 4 conditions under which the assumption certainly holds are introduced. Moreover, possible ways to exploit the information collected by the algorithm in order to guarantee finiteness for general concave functions through updates of the objective function, are discussed. All that leads to the introduction, in Section 5, of a new finite algorithm for general concave functions, where also the objective function may be modified during the execution of the algorithm, but only when this is strictly necessary in order to ensure finiteness. Finally, in Section 6, some preliminary computational experiments are presented.

2. Simplicial algorithms for concave optimization

Let

$$S = [v_0, \ldots, v_n] = \{x : \ x = \sum_{j=0}^{n} \lambda_j v_j, \ \sum_{j=0}^{n} \lambda_j = 1, \ \lambda_j \geq 0, \ j = 0, \ldots, n\}$$

denote the n-simplex generated by the affinely independent vectors v_i's. The general scheme of a simplicial branch-and-bound algorithm is the following.

ALGORITHM 1

1. Choose $\overline{x} \in P$, $\varepsilon \geq 0$. Let $UB = f(\overline{x})$.
2. Construct a simplex $S_0 \supset P$. Set $\mathcal{P} = \{S_0\}$ and $\mathcal{M} = \mathcal{P}$.
3. For all $S = [v_0, \ldots, v_n] \in \mathcal{P}$ solve

$$\begin{aligned} \beta(S; P) = \min & \sum_{i=0}^{n} \lambda_i f(v_i) \\ & \sum_{i=0}^{n} \lambda_i v_i \in P \\ & \sum_{i=0}^{n} \lambda_i = 1 \\ & \lambda_i \geq 0. \end{aligned} \tag{2.2}$$

   Given an optimal solution $(\overline{\lambda}_0, \ldots, \overline{\lambda}_n)$ of (2.2), let $\omega(S) = \sum_{i=0}^{n} \overline{\lambda}_i v_i$.
4. Set $UB = \min\{UB, \min_{S \in \mathcal{P}}\{f(\omega(S))\}\}$.
5. In $\mathcal{M}$ delete all simplices S such that $\beta(S; P) \geq UB - \varepsilon$ (fathoming rule).
6. Let $\mathcal{R}$ be the collection of remaining simplices. If $\mathcal{R} = \emptyset$, then stop: UB is an $\varepsilon-$optimal value, i.e.

$$UB \leq f^* + \varepsilon.$$

7. Select the simplex $S^* = [v_0^*, \ldots, v_n^*] \in \mathcal{R}$, $S^* \in \arg\min\{\beta(S; P) : S \in \mathcal{R}\}$.
8. According to a given rule, to be discussed below, subdivide S^* using a point $x^* \in S^*$, $x^* = \lambda_0^* v_0^* + \cdots + \lambda_n^* v_n^*$, $\lambda_i^* \geq 0$, $i = 0, \ldots, n$ and $\sum_{i=0}^{n} \lambda_i^* = 1$. Then we obtain from S^* a new set $\mathcal{P}^*$ of simplices such that

$$S \in \mathcal{P}^*$$

$$\Updownarrow$$

$$\exists\, i,\ 0 \leq i \leq n :\ \lambda_i^* > 0,$$

$$S = [v_0^*, \ldots, v_{i-1}^*, x^*, v_{i+1}^*, \ldots, v_n^*].$$

9. Set $\mathcal{M} = \mathcal{R} \cup \mathcal{P}^* \setminus \{S^*\}$ and $\mathcal{P} = \mathcal{P}^*$. Go back to Step 3.

The deletion of simplices in Step 5 is a consequence of the fact that the objective function of Problem (2.2) is the convex envelope of f over the simplex S so that $\beta(S; P)$ is a lower bound of the optimal value of f over $S \cap P$. The choice of the rule to subdivide a simplex in Step 8 is a critical one (see also [17]). Classical subdivision rules are

Bisection which was first introduced in [5] for simplicial algorithms. It consists in subdividing the selected simplex S^* by using the midpoint of one of its longest edges.

ω-subdivision , which was first introduced in [16] for conical algorithms. It consists in subdividing the selected simplex S^* using the point $\omega(S^*)$ in Step 8.

Typically ω-subdivision seems to be a more natural choice with respect to bisection. The algorithm based only on bisection has been proven to be finite for $\varepsilon > 0$, and, consequently, convergent for $\varepsilon = 0$. Some algorithms have been introduced which combine bisections and ω-subdivisions. These are the so called normal algorithms, first introduced in [18] for conical algorithms, and lately extended to simplicial algorithms. They have been proved to be convergent for $\varepsilon = 0$.

For all the above mentioned algorithms, no proof of finiteness has been given for $\varepsilon = 0$, and, actually, for the algorithm based only on bisections the observation below proves that the algorithm is not finite. That justifies the search for possible modifications of Algorithm 1, which will be presented in next sections, in order to get a finite exact algorithm.

OBSERVATION 1 *Algorithm 1 employing bisections does not terminate in a finite time for $\varepsilon = 0$.*

Proof. Consider the two-dimensional simplex $S = [(0, 0), (-1, 1), (-2, 0)]$ and the two-dimensional polytope $P = S \cap \{(x_1, x_2) : x_2 + \rho x_1 \geq 0\}$ for some $\rho \in (0, 1)$. Note that $\rho \in (0, 1)$ implies that P is nonempty and full-dimensional, and, moreover, in the segment with extreme points $(-2, 0)$ and $(0, 0)$, only the point $(0, 0)$ belongs to P. Let the initial simplex S_0 coincide with S. It can be seen that, independently from the objective function, the algorithm employing only bisections generates a nested sequence $\{S_j\}$ of simplices such that for any nonnegative integer i

$$S_{2i} = \left[(0, 0),\ \left(-\frac{1}{2^{i-1}}, 0\right),\ \left(-\frac{1}{2^i}, \frac{1}{2^i}\right)\right],$$

and

$$S_{2i+1} = \left[(0, 0),\ \left(-\frac{1}{2^i}, 0\right),\ \left(-\frac{1}{2^i}, \frac{1}{2^i}\right)\right].$$

Let us consider the following objective function

$$f(x_1, x_2) = M \min\{x_2 + \rho x_1, 0\} + g(x_1, x_2),$$

where M is a positive constant and g is a concave function (possibly strictly concave) with the following properties

$$\forall\ (x_1, x_2) \in P \setminus (0, 0) \quad g(x_1, x_2) > 0 \ \text{ and } \ g(0, 0) = 0. \tag{2.3}$$

and

$$| \frac{\partial g(0,0)}{\partial x_i} | \leq \eta \quad i = 1, 2,$$

for some $\eta > 0$. A function satisfying these properties is the following

$$g(x_1, x_2) = \eta - \frac{\eta}{2}(x_1 + 1)^2 - \frac{\eta}{2}(x_2 - 1)^2.$$

In view of the concavity of g, it follows that

$$\forall\ (x_1, x_2) \in S \quad g(x_1, x_2) \leq \eta(x_2 - x_1). \tag{2.4}$$

We note that for any $(x_1, x_2) \in P$ it holds that $\min\{x_2 + \rho x_1, 0\} = 0$, so that, in view of (2.3), $f^* = 0$. In order to prove that the algorithm is not finite, we will show that the solution of the linear subproblems (2.2) over the simplices S_j's, is, for any $j \geq 0$, lower than 0. We only give the proof for a simplex S_{2i+1}, where i is a nonnegative integer. The proof for the other simplices is completely analogous. Let $q = \frac{1}{2^i}$. The objective function of the linear subproblem (2.2) is the following

$$\lambda_1 f(0,0) + \lambda_2 f(-q, 0) + \lambda_3 f(-q, q) = -\lambda_2 M\rho q + \lambda_2 g(-q, 0) + \lambda_3 g(-q, q),$$

and, in view of (2.4), it can be bounded from above by

$$-q[\lambda_2(M\rho - \eta) - 2\lambda_3\eta]. \tag{2.5}$$

Now consider the points $(-q, \lambda_3 q)$ for $\lambda_3 \in [0, 1]$. We note that if $\lambda_3 \geq \rho$, then $(-q, \lambda_3 q) \in P$. Indeed, $-\rho q + \lambda_3 q \geq 0$ for any $\lambda_3 \geq \rho$. Then, let us consider the point $\lambda_1^* = 0,\ \lambda_2^* = 1 - \rho$ and $\lambda_3^* = \rho$. The point is feasible for the linear subproblem and, in view of (2.5), its objective function value is bounded from above by

$$-q[(1 - \rho)\rho M - (1 - \rho)\eta - 2\rho\eta],$$

which is lower than 0 for M big enough, those implying $\beta(S_{2i+1}; P) < 0$, as we wanted to prove. □

3. Modifications of the basic simplicial algorithms

In this section, two basic ideas, which have already been succesfully employed in global optimization, are combined. The first one is to introduce a threshold $\delta > 0$ in order to decide when to apply a local search procedure, while the second idea is the addition of cuts. Now let $V(P)$ denote the vertex set of the polytope P, and, for a given $v \in V(P)$, let $adj(v; P) \subset V(P)$ denote the set of vertices adjacent to v in P. The local search procedure is denoted by $LS(\cdot; P)$. When applied to a

point $x \in P$, it returns a local minimum vertex of f over P. Here local minimality is intended with respect to the vertex neighborhood of a vertex, i.e. $\overline{v} \in V(P)$ is a local minimum if

$$f(v) \geq f(\overline{v}) \ \ \forall\, v \in adj(\overline{v}; P). \tag{3.6}$$

Note that local vertex minimality is a stronger requirement with respect to the usual local minimality. Indeed, (3.6) implies that v is also a local minimum of f in the usual sense, but the opposite is not necessarily true.

The local search procedure LS consists of two phases.

Phase I Given $x \in P$, a vertex $w \in V(P)$ satisfying $f(w) \leq f(x)$ must be found. There are different possibilities to implement this phase. If a subgradient $g \in \partial f(x)$ is available, then the solution $w \in V(P)$ of the linear problem $\min_{y \in P} g^T y$ satisfies the requirement. Otherwise, the following procedure can be employed.

1. Choose a direction d belonging to the null space of the active constraints in x.
2. Set

$$\eta_1 = \max\{\eta : \ \eta \geq 0, \ a_j(x + \eta d) \leq b_j, \ j = 1, \ldots, m\},$$

and

$$\eta_2 = \max\{\eta : \ \eta \geq 0, \ a_j(x - \eta d) \leq b_j, \ j = 1, \ldots, m\}.$$

3. Set $x \in \arg\min\{f(x + \eta_1 d), f(x - \eta_2 d)\}$.
4. If $x \in V(P)$, then set $w = x$ and return w, otherwise go back to 1.

Since at each iteration the number of constraints active at x increases by at least one, the procedure must stop in at most m iterations.

Phase II Given $w \in V(P)$ a vertex $v \in V(P)$ satisfying (3.6) must be found. This can be done by the following procedure which uses as a subroutine the neighbor generation procedure also employed in [2].

1. Generate all the vertices $adj(w; P)$ (neighbor generation procedure).
2. If w satisfies (3.6), then set $v = w$ and return v, otherwise choose $z \in adj(w; P)$ such that $f(z) < f(w)$, set $w = z$ and go back to 1.

We underline at this point the difference between the use of the neighbor generation procedure in our paper, and the use of the same procedure in [2]. In our paper the neighbor generation procedure is used as a subroutine of Phase II of the local search. In [2] it is used in an essentially different way, namely it is employed to generate a collection of vertices of P with respect to which radial subdivisions are performed. While the finiteness result in [2] appears to strongly rely on the fact that the simplices are subdivided with respect to vertices of P, the same is not true for our algorithms, where it will be shown that finiteness derives from the introduction of cuts and, in some cases discussed in Section 5, from updates of the objective function.

We also recall here the concept of γ-concavity cut. Let v be a local minimum of f over P in the sense mentioned above, and let $f(v) = \gamma$. A γ-concavity cut is an inequality $\pi(x - v) \geq 1$ such that

$$x \in P, \quad \pi(x - v) \leq 1 \quad \Rightarrow \quad f(x) \geq \gamma. \tag{3.7}$$

We do not discuss here how to compute a γ-concavity cut and we refer to the existing literature (see e.g. [9]).

Before describing the modified algorithm some notation, employed in what follows, is introduced.

- $\beta(P_k) = \min_{S \in \mathcal{R}} \beta(S, P_k)$, i.e. $\beta(P_k)$ is a lower bound for Problem (1.1) with feasible region P_k (the index k for the polytope is explained below).
- Given the predefined parameter $\delta > 0$

$$Y_k^f = \{\omega(S) \mid \ S \in \mathcal{P} \ : \ f(\omega(S)) \leq UB + \delta\}, \tag{3.8}$$

  is the set of points at which local searches will be started at iteration k.
-

$$LM_k^f = \{LS(y, P_k) : \ y \in Y_k^f\}.,$$

  is the set of local minima over P_k detected by the algorithm at iteration k.
- $M_k^f = \cup_{i=1}^k LM_i^f$ is the set of all local minima detected by the algorithm up to iteration k.
- $GM_k^f = \{y \in M_k^f : \ f(y) = UB\}$ is the set of detected local minima whose value coincide with the best observed one.

Algorithm 1 is then modified in the following way.

ALGORITHM 2

1. Choose $\overline{x} \in P$, $\delta > 0$. Determine $\overline{v} = LS(\overline{x}; P) \in V(P)$. Let $UB = f(\overline{v})$. Set $M_0^f = GM_0^f = \{\overline{v}\}$, $P_0 = P$ and $k = 0$.

2. Construct a simplex $S_0 \supset P$. Set $\mathcal{P} = \{S_0\}$ and $\mathcal{M} = \mathcal{P}$.

3. For all $S = [v_0, \ldots, v_n] \in \mathcal{P}$ solve

$$\begin{aligned} \beta(S, P_k) = \min & \sum_{i=0}^{n} \lambda_i f(v_i) \\ & \sum_{i=0}^{n} \lambda_i v_i \in P_k \\ & \sum_{i=0}^{n} \lambda_i = 1 \\ & \lambda_i \geq 0. \end{aligned}$$

Given an optimal solution $(\overline{\lambda}_0, \ldots, \overline{\lambda}_n)$ of (2.2), let $\omega(S) = \sum_{i=0}^{n} \overline{\lambda}_i v_i$.

4a. Compute the set Y_k^f. If $Y_k^f = \emptyset$ then set $P_{k+1} = P_k$ and go to Step 5; otherwise go to Step 4b.

4b. Compute the set LM_k^f and let $g_k = \min_{y \in LM_k^f} f(y)$; if $g_k > UB$, then set $P_{k+1} = P_k$ and go to Step 5; otherwise if $g_k < UB$ set $P_k = P$ and $UB = g_k$. Go to Step 4c.

4c. If $UB \leq \beta(P_k)$ then stop: UB is the optimal value; otherwise go to Step 4d.

4d. Select $v \in GM_k^f \cap V(P_k)$, compute the UB-concavity cut $\pi(x - v)$, and define a new polytope $P_{k+1} \subset P_k$ by adding to the description of P_k the UB-concavity cut, i.e.

$$P_{k+1} = P_k \cap \{x \in R^n : \pi(x - v) \geq 1\}. \tag{3.9}$$

4e. Recompute the bounds $\beta(S, P_{k+1})$ and update $\beta(P_{k+1})$ accordingly

5. In $\mathcal{M}$ delete all simplices S such that $\beta(S, P_{k+1}) \geq UB$ (fathoming rule).

6. Let $\mathcal{R}$ be the collection of remaining simplices. If $\mathcal{R} = \emptyset$, then stop: UB is an optimal value.

7. Select the simplex $S^* = [v_0^*, \ldots, v_n^*] \in \mathcal{R}$, $S^* \in \arg\min\{\beta(S, P_{k+1}) : S \in \mathcal{R}\}$.

8. According to a given rule subdivide S^* using a point $x^* \in S^*$, $x^* = \lambda_0^* v_0^* + \cdots + \lambda_n^* v_n^*$, $\lambda_i^* \geq 0$, $i = 0, \ldots, n$ and $\sum_{i=0}^{n} \lambda_i^* = 1$. Then we obtain from S^* a new set $\mathcal{P}^*$ of simplices such that

$$\begin{array}{c} S \in \mathcal{P}^* \\ \Updownarrow \\ \exists\, i,\ 0 \leq i \leq n :\ \lambda_i^* > 0, \end{array}$$

$$S = [v_0^*, \ldots, v_{i-1}^*, x^*, v_{i+1}^*, \ldots, v_n^*].$$

9. Set $\mathcal{M} = \mathcal{R} \cup \mathcal{P}^* \setminus \{S^*\}$ and $\mathcal{P} = \mathcal{P}^*$. Set $k = k + 1$ and go back to Step 3.

The following remarks about these modifications hold.

REMARK 1 In Step 4b it is basically stated that local searches are started from points whose distance from the current upper bound UB is below the threshold δ. The idea is that in order to enforce finiteness of the method one possibly has to detect (by local searches in Step 4b) and cut off all global minimum vertices (see Step 4d). On the other hand it is desirable to keep as low as possible the number of cuts.

REMARK 2 Each time the polytope P_k is restricted to a new polytope P_{k+1} in Step 4d, it follows, in view of the update of the lower bounds $\beta(S, P_{k+1})$ for each $S \in \mathcal{R}$, that the bound $\beta(P_{k+1})$ is now a lower bound for the problem

$$\begin{aligned} &\min \ f(x) \\ &x \in P_{k+1}, \end{aligned} \tag{3.10}$$

with optimal value denoted by f^*_{k+1}. On the other hand, the upper bound UB is valid for the original problem. Essentially the algorithm switches to solve the restricted Problem (3.10) but keeps the original upper bound.

REMARK 3 If Update (3.9) of the feasible region is employed, it immediately follows from the definition of concavity cut that the selected vertex v of P_k does not belong to P_{k+1}, and that no point in P_k with function value lower than UB (if any exists) is cut off (see (3.7)). In particular all the newly created vertices of P_{k+1} have function value greater than or equal to UB, i.e.

$$\forall\, v \in V(P_{k+1}) \setminus V(P_k) \ \ f(v) \geq UB. \tag{3.11}$$

Now an assumption, under which the finiteness of the modified algorithm will be proved, is introduced. The assumption does not always necessarily hold, but in the next section some conditions under which it is guaranteed to hold will be presented.

ASSUMPTION 1 If Update (3.9) of the feasible region is employed, then all the newly created vertices have function value greater than f^*, i.e.

$$\forall\, v \in V(P_{k+1}) \setminus V(P_k) \ \ f(v) > f^*.$$

We note that, in view of (3.11), Assumption 1 certainly holds when $UB > f^*$, but for $UB = f^*$ the assumption is not guaranteed to hold.

Next we introduce the concept of a normal simplicial subdivision process (see also [9] for a detailed presentation).

DEFINITION 1 Given a polytope P_k, a simplicial subdivision is normal if for any infinite nested sequence $\{S_t\}$ of simplices generated by it, the following is satisfied

$$\underline{\lim}_{t\to\infty} f(\omega(S_t)) - \beta(S_t, P_k) = 0. \tag{3.12}$$

Theorem VII.8 in [9] proves that any simplicial algorithm employing a normal subdivision rule is convergent.

First we prove the following proposition.

PROPOSITION 1 If a normal subdivision strategy is employed, then Algorithm 2 detects a global optimum after a finite number of iterations.

Proof. The current value UB is always attained at least at one vertex of P. Indeed, in view of (3.11), each time the polytope is modified in Step 4d, the newly created vertices must have a function value not smaller than UB. Therefore, either $UB = f^*$ and we are done, or

$$UB - f^* \geq \min_{v \in V(P) \setminus X^*(f,P)} f(v) - f^* = \Delta f.$$

Condition (3.12) guarantees that there exists an infinite nested sequence of simplices $\{S_t\}$ such that, possibly by passing to a subsequence

$$f(\omega(S_t)) \to f^*,$$

i.e. for t big enough $f(\omega(S_t)) - f^* < \Delta f$ so that $f(\omega(S_t)) < UB$, and, according to (3.8), a local search will be started and a vertex $\overline{v} \in V(P)$ with $f(\overline{v}) = f^*$ will be detected. □

It is also possible to prove the following proposition, stating that, under Assumption 1, the algorithm only modifies the original polytope in Step 4d a finite number of times.

PROPOSITION 2 *If Assumption 1 holds, Step 4d of Algorithm 2 is entered at most a finite number of times.*

Proof. In view of Proposition 1, we only need to prove that Step 4d is entered a finite number of times when $UB = f^*$. Note that, from Step 4b, Step 4d is entered at iteration k only if at least one local minimum over P_k with function value equal to f^* is detected. In view of Assumption 1 all local minima over P_k with function value equal to f^* are vertices of $V(P)$. Indeed, Assumption 1 guarantees that all the newly created vertices must have a function value greater than f^*. Then, each time Step 4d is entered, at least one $v \in V(P)$ is cut off, and, in view of the finite cardinality of $V(P)$, Step 4d is entered a finite number of times. □

The following theorem states the finiteness of Algorithm 2 when a normal subdivision strategy is employed.

THEOREM 1 *If in Algorithm 2 a normal subdivision strategy is employed and Assumption 1 holds, then it terminates after finitely many iterations.*

Proof. From Proposition 2 we see that the polytope P_k can be changed at most a finite number of times, i.e. $\exists\ K$ such that $\forall\ k \geq K : \ P_k = P_K$. Suppose that Algorithm 2 is infinite. At first we also assume that

$$V(P) \cap X^*(f, P) \cap P_K = \emptyset, \tag{3.13}$$

i.e. all the global minimum vertices have been already cut off. Note that we must have $UB = f^*$ because at least one global minimum vertex has certainly been detected if it has been cut off. Then it must hold that $f_K^* > f^* = UB$. Indeed,

f_K^* is attained at one local minimum vertex v of P_K and from Assumption 1 it follows that $v \in V(P_K) \setminus V(P)$ implies $f(v) > f^*$, while, if $v \in V(P_K) \cap V(P)$ it follows from (3.13) that $f(v) > f^*$. From Remark 2 we note that the lower bounds computed by the algorithm are, after the updating in Step 4e, lower bounds for f_K^* and in view of the convergence of the basic algorithm, when a normal subdivision strategy is employed, we must have $\beta(P_k) \uparrow f_K^*$ as $k \to \infty$, i.e. at some iteration k it holds that $UB < \beta(P_k)$, which contradicts the infinity of the algorithm (see Step 4c).

Next, we assume that there exists at least a global minimum vertex which is never cut off, i.e. it belongs to P_K. But, again, in view of the convergence of the algorithm, we must have $\beta(P_k) \uparrow f_K^* = f^*$. Moreover, in view of Condition (3.12), there exists an infinite nested sequence of simplices $\{S_t\}$ such that, possibly by passing to a subsequence

$$f(\omega(S_t)) \to f^*,$$

i.e. according to (3.8) a local search will be started and a vertex $\overline{v}$ of P_K will be detected and cut off in Step 4d, thus contradicting the fact that $P_k = P_K$ for any $k \geq K$. □

It must be underlined at this point that the combination of branch-and-bound approaches with concavity cuts is not new in the literature. However, to the authors' knowledge no combination of concavity cuts and branch-and-bound approaches has been proposed in order to obtain finite algorithms.

4. Conditions under which finiteness can be ensured

In this section we introduce a property under which Assumption 1 is guaranteed to hold and, consequently, the finiteness of Algorithm 2 is ensured.

PROPERTY 1 *The global optimum value f^* is only attained at vertices of the polytope, i.e.*

$$X^*(f, P) \subseteq V(P).$$

PROPOSITION 3 *If Property 1 holds, then Algorithm 2 is finite.*

Proof. In view of Theorem 1, we only need to prove that Assumption 1 holds. But this immediately follows from the observation that $v \in V(P_{k+1}) \setminus V(P_k)$ implies $v \in P \setminus V(P)$, so that, in view of Property 1, $f(v) > f^*$. □

We note that the following remark holds.

REMARK 4 If f is strictly concave, then Property 1 holds. Therefore, Algorithm 2 is finite if applied to strictly concave functions.

We are now interested in what to do in more general cases. Property 1 can not, in general, be checked in advance, so that it is not possible to guarantee in advance the finiteness of the algorithm. Nevertheless it is important to note that at each iteration k the set GM_k^f, containing all the detected points whose value is equal to UB, is available. If, at iteration k, it holds that $GM_k^f \not\subseteq V(P)$, then the algorithm should generate a WARNING, meaning that Property 1 may not hold. Since we do not know, during the execution of the algorithm, whether $UB = f^*$ or not, we are never completely guaranteed that Property 1 does not hold, but the warning acts like an alarm saying that the algorithm may experience difficulties.

Therefore, now the question is what to do when a warning arises. In order to see that, we introduce a function which can be employed to modify the original objective function. Let $M = \{1, \ldots, m\}$ and

$$g(x) = \min_{J \subseteq M,\ |J|=n} \sum_{j \in J} (b_j - a_j x).$$

The function g is concave. Moreover, it holds that

$$\forall\, x \in P,\ g(x) \geq 0. \tag{4.14}$$

We note that at a given point x, g is equal to the sum of the lowest n values of the slack variables in the constraints defining P. Therefore, the following procedure can be employed to evaluate g.

1. Compute the slack variables at each constraint, i.e. $y_i = b_i - a_i x$ for any $i \in M$ (note that the function is often evaluated at the solution of the linear subproblems, so that the value of the slack variables may be immediately available).
2. Order the y_i's in a nondecreasing way: $\{y_{i_1}, \ldots, y_{i_m}\}$.
3. Return $g(x) = \sum_{j=1}^{n} y_{i_j}$.

Now we introduce an observation which will be employed in what follows.

OBSERVATION 2 *Given a nonempty polytope $P = \{x \in R^n : a_i x \leq b_i,\ i \in M\}$, then for some $v \in P$ it holds that*

$$v \in V(P) \quad \Leftrightarrow \quad n \text{ independent constraints are active in } v. \tag{4.15}$$

Next we introduce a new proposition, proving that if all the vertices at which the global optimum value is attained are nondegenerate, then Algorithm 2 applied to the problem with objective function $f(x) + g(x)$ is finite and returns a solution of the original problem.

PROPOSITION 4 *If*

$$v \in V(P) \cap X^*(f, P) \ \Rightarrow\ v \text{ nondegenerate}, \tag{4.16}$$

then

$$X^*(f+g, P) \subseteq X^*(f, P), \tag{4.17}$$

and the concave function $f(x) + g(x)$ *satisfies Property 1.*

Proof. First we recall that $g(x) \geq 0$ for any $x \in P$. Moreover, in view of Observation 2, it holds that

$$x \in V(P) \;\Rightarrow\; g(x) = 0.$$

Therefore, the global optimum value of the problem with objective function $f+g$ is still f^* and for any $x \in P \setminus X^*(f, P)$ it holds that $f(x) + g(x) > f^*$, i.e. (4.17) is true. We note that the nondegeneracy assumption (4.16) on the global optimum vertices implies that

$$\forall\, x \in V(P) \cap X^*(f, P) \quad \text{there are exactly } n \text{ constraints active at } x. \tag{4.18}$$

Now, by contradiction, we assume that Property 1 is not satisfied, i.e. $\exists\, y \in P \setminus V(P)$ such that $f(y) + g(y) = f^*$. This implies $g(y) = 0$. Since $y \notin V(P)$ there exist $v_1, \ldots, v_p \in V(P),\ p \geq 2$ such that y belongs to the convex hull of the points $v_1, \ldots, v_p$. Since $g(y) = 0$, then there are at least n constraints active at y, which implies that there are at least $n+1$ constraints active at $v_1, \ldots, v_p$. Moreover, the concavity of f and the optimality of f^* imply that $f(v_1) = \cdots = f(v_p) = f^*$, but then (4.18) is contradicted. □

Therefore, if we substitute the original objective function with $f(x) + g(x)$ and (4.16) holds, Algorithm 2 is finite. As with Property 1, (4.16) can not be checked in advance. But we can still profit from the knowledge of the set GM_k^{f+g} by introducing a WARNING when $GM_k^{f+g} \not\subseteq V(P)$. The question is, again, what to do if the algorithm has produced a warning. In order to see that, let us assume that neither Property 1 nor (4.16) hold. Instead of adding to f the function g, we add the following function

$$\tilde{g}(x) = \min_{J \subset M,\ |J|=n,\ (a_j,\ j \in J,\ \text{independent})} \sum_{j \in J} (b_j - a_j x).$$

The function $\tilde{g}$ is concave, $\forall\, x \in P\ \ \tilde{g}(x) \geq 0$, and, in view of Observation 2, $\tilde{g}(x) = 0 \;\Leftrightarrow\; x \in V(P)$. With a proof completely analogous to that of Proposition 4, it can be seen that

$$X^*(f+\tilde{g}, P) \subseteq X^*(f, P),$$

and the concave function $f(x) + \tilde{g}(x)$ satisfies Property 1. Therefore, Algorithm 2 applied to the problem with objective function $f + \tilde{g}$ is finite. Now we need a procedure to compute the function $\tilde{g}$. We note that the set of vectors $\{a_1, \ldots, a_m\}$ forms a so called matric matroid (see, e.g., [11, page 288]). Therefore, the combinatorial optimization problem which defines $\tilde{g}(x)$ can be solved by the following greedy procedure.

1. Compute the slack variables at each constraint, i.e. $y_i = b_i - a_i x$ for any $i \in M$.
2. Order the y_i's in a nondecreasing way: $\{y_{i_1}, \ldots, y_{i_m}\}$.
3. set $E = \emptyset$ and $t = 1$
4. while ($\mid E \mid < n$) do
   { if the vectors $\{a_j,\ j \in E \cup \{a_{i_t}\}\}$ are linearly independent, set $E = E \cup \{a_{i_t}\}$; set $t := t + 1$ }.
5. Return $\tilde{g}(x) = \sum_{j \in E} y_j$.

The computation of the function $\tilde{g}$ is more expensive than the computation of the function g because at each step we need to check whether a newly selected vector is linearly independent with respect to the previously selected vectors, which can be done as follows. We start with an initial basic matrix $B = I$ where I is the $n \times n$ identity matrix, and with the nonbasic matrix $N = A$. It is then possible to implement the computation of $\tilde{g}$ by moving out of the basis the columns of the identity matrix and moving into the basis some columns of A. More precisely, step 4. of the procedure for the computation of $\tilde{g}$ can be implemented as follows.

4. while ($\mid E \mid < n$) do
{ if the vector a_{i_t} can be moved into the current basis in place of one of the columns of the identity matrix which still belong to the basis then:

- update the basis accordingly and perform the corresponding pivot operation;
- set $E = E \cup \{a_{i_t}\}$;

set $t := t + 1$ }

Therefore, we notice that the computation of $\tilde{g}$ reduces to the execution of n pivot operations. However, as it will be seen, we will try to avoid as much as possible to switch to function $f + \tilde{g}$. Indeed, in the following section all the observations collected in this section will be exploited in order to build a new algorithm which will be proved to be finite when applied to any concave function. In the algorithm not only the feasible polytope P is modified, but also the objective function f. The algorithm runs in the same way as Algorithm 2 until some special event occurs (the warnings mentioned above). In such cases the objective function is modified, but these modifications are introduced only when strictly necessary.

5. A finite exact branch-and-bound algorithm

We immediately give the description of the algorithm.

ALGORITHM 3

1. Choose $\overline{x} \in P$, $\delta > 0$. Determine $\overline{v} = LS(\overline{x}; P) \in V(P)$. Let $UB = f(\overline{v})$. Set $k = 0$, $P_0 = P$ and $h = f$. Set $M_0^h = GM_0^h = \{\overline{v}\}$.

2. Construct a simplex $S_0 \supset P$. Set $\mathcal{P} = \{S_0\}$ and $\mathcal{M} = \mathcal{P}$.

3. For all $S = [v_0, \ldots, v_n] \in \mathcal{P}$ solve

$$\begin{aligned} \beta(S, P_k, h) = \min & \sum_{i=0}^{n} \lambda_i h(v_i) \\ & \sum_{i=0}^{n} \lambda_i v_i \in P_k \\ & \sum_{i=0}^{n} \lambda_i = 1 \\ & \lambda_i \geq 0. \end{aligned}$$

Given an optimal solution $(\overline{\lambda}_0, \ldots, \overline{\lambda}_n)$ of (2.2), let $\omega(S) = \sum_{i=0}^{n} \overline{\lambda}_i v_i$.

4a. Compute the set Y_k^h. If $Y_k^h = \emptyset$ then set $P_{k+1} = P_k$ and go to Step 5; otherwise go to Step 4b.

4b. Compute the set LM_k^h and let $g_k = \min_{y \in LM_k^h} h(y)$; if $g_k > UB$, then set $P_{k+1} = P_k$ and go to Step 5; otherwise go to Step 4c.

4c. If $g_k = UB$ go to step 4c. bis. Otherwise ($g_k < UB$) set $h = f$, $P_k = P$ and $UB = g_k$. If $UB \leq \beta(P_k, h)$ then stop: UB is the optimal value; otherwise go to Step 4d.

4c. bis If $GM_k^h \subseteq V(P)$, then go to step 4d. Otherwise

- if $h = f$, set $h = f + g$; if $GM_k^h \not\subseteq V(P)$ set $h = f + \tilde{g}$
- if $h = f + g$, set $h = f + \tilde{g}$.

(Note that for $h = f + \tilde{g}$ it certainly holds that $GM_k^h \subseteq V(P)$).
If $GM_k^h \cap V(P_k) \neq \emptyset$ go to Step 4d., otherwise set $P_{k+1} = P_k$ and go to Step 4e.

4d. Select $v \in GM_k^h \cap V(P_k)$ (note that it holds $v \in V(P)$) and compute the UB-concavity cut $\pi(x - v)$ (generated by using, indifferently, the original function f or the current function h), and define a new polytope $P_{k+1} \subset P_k$ by adding to the description of P_k the UB-concavity cut, i.e.

$$P_{k+1} = P_k \cap \{x \in R^n : \pi(x - v) \geq 1\}.$$

4e. Recompute the bounds $\beta(S, P_{k+1}, h)$ and update $\beta(P_{k+1}, h)$ accordingly

5. In $\mathcal{M}$ delete all simplices S such that $\beta(S, P_{k+1}, h) \geq UB$ (fathoming rule).

6. Let $\mathcal{R}$ be the collection of remaining simplices. If $\mathcal{R} = \emptyset$, then stop: UB is an optimal value.

7. Select the simplex $S^* = [v_0^*, \ldots, v_n^*] \in \mathcal{R}$, $S^* \in \arg\min\{\beta(S, P_{k+1}, h) : S \in \mathcal{R}\}$.

8. According to a given rule subdivide S^* using a point $x^* \in S^*$, $x^* = \lambda_0^* v_0^* + \cdots + \lambda_n^* v_n^*$, $\lambda_i^* \geq 0$, $i = 0, \ldots, n$ and $\sum_{i=0}^n \lambda_i^* = 1$. Then we obtain from S^* a new set $\mathcal{P}^*$ of simplices such that

$$\begin{array}{c} S \in \mathcal{P}^* \\ \Updownarrow \\ \exists\, i,\ 0 \leq i \leq n :\ \lambda_i^* > 0, \end{array}$$

$$S = [v_0^*, \ldots, v_{i-1}^*, x^*, v_{i+1}^*, \ldots, v_n^*].$$

9. Set $\mathcal{M} = \mathcal{R} \cup \mathcal{P}^* \setminus \{S^*\}$ and $\mathcal{P} = \mathcal{P}^*$. Set $k = k + 1$ and go back to Step 3.

The following theorem proves the finiteness of Algorithm 3.

THEOREM 2 *If f is concave and a normal subdivision strategy is employed, then Algorithm 3 is finite.*

Proof. The proof of this finiteness result is subdivided into three parts.

The first part shows that $UB = f^*$ after a finite number of iterations. By contradiction we assume that for an infinite number of iterations both $UB > f^*$ and h are not updated. In view of the finite cardinality of $V(P)$ the number of cuts introduced in Step 4d. is finite. Therefore, $\exists\ K_1 \ :\ \forall\ k \geq K_1,\ P_k = P_{K_1}$, and, after iteration K_1, the algorithm reduces to Algorithm 2 applied to problems with objective function h and feasible region given by the intersection of P_{K_1} with each non fathomed simplex into which the initial simplex has been subdivided up to iteration K_1. Then, in view of Proposition 1, after a finite number of iterations $UB = \min_{x \in P_{K_1}} h(x) = f^*$, which contradicts the fact that UB is never updated. Therefore, after a finite number of iterations, either UB is updated, or h is updated. Since in the latter case, while UB is not updated h can be updated at most twice in Step 4c.bis, it follows that after a finite number of iterations $UB > f^*$ is updated with the value of f at a new vertex $v \in V(P)$. In view of the finite cardinality of $V(P)$, after a finite number of iterations $UB = f^*$.
In the second part we prove that, while $UB = f^*$, if $h = f$ or $h = f + g$, then after a finite number of iterations either the algorithm stops or the function h is updated. By contradiction we assume that, while $UB = f^*$, the algorithm runs for an infinite number of iterations without updating $h = f$ or $h = f + g$. Again in

view of the finite cardinality of $V(P)$ the number of cuts introduced in Step 4d. is finite. Therefore,

$$\exists\, K_2 \;:\; \forall\, k \geq K_2,\;\; P_k = P_{K_2}, \tag{5.19}$$

and, after iteration K_2, the algorithm reduces to Algorithm 2 applied to problems with objective function h and feasible region given by the intersection of P_{K_2} with each nonfathomed simplex into which the initial simplex has been subdivided up to iteration K_2. Then, in view of the convergence of Algorithm 2 under the normal subdivision strategy, two cases are possible.

Case 1. for some $k \geq K_2$, $\beta(P_k, h) \geq f^*$ so that the algorithm stops, which is a contradiction.

Case 2. after a finite number of iterations we start a local search leading to a point $y \in P \;:\; f(y) = f^*$; then there are two possible subcases

Case 2a. $y \in V(P)$: then a new cut is introduced in Step 4d. and (5.19) is contradicted.

Case 2b. $y \notin V(P)$: then h is updated in Step 4c.bis, which is also a contradiction.

Thus, in each possible case we are lead to a contradiction.

The third and final part of the proof shows that, when $h = f + \tilde{g}$ and $UB = f^*$, the algorithm stops after a finite number of iterations. The proof is done by contradiction and is completely analogous to that of the second part with the only difference that Case 2b never holds because $f + \tilde{g}$ satisfies Property 1. □

We note that the algorithm tries to exploit as much as possible the information contained in the set GM_k^h, in order to introduce modifications with respect to Algorithm 2 only when this is strictly necessary to ensure finiteness. Indeed, Algorithm 3, when applied to strictly concave functions, never modifies the objective function, as the following observation proves.

OBSERVATION 3 *If f is strictly concave, then Algorithm 3 coincides with Algorithm 2.*

Proof. We only need to prove that the condition $GM_k^f \subseteq V(P)$ is always satisfied if f is strictly concave. By contradiction we assume that, at some iteration k, there exists $v \in GM_k^f \setminus V(P)$. Note that the local vertex minimality of v implies that

$$\forall\, w \in adj(v; P_k) \quad f(w) \geq UB. \tag{5.20}$$

From $v \in V(P_k)$, it follows that $v \in (v_1, v_2)$, where $v_1, v_2 \in V(P_h)$ for some $h < k$, and there exists a UB-concavity cut $\pi(x - v') \geq 1$, where $v' \in V(P_h)$ and

$$\pi(v_1 - v') < 1 \quad \pi(v_2 - v') > 1 \quad \pi(v - v') = 1.$$

Note that it must hold $v_2 \in adj(v; P_k)$. Moreover, in view of (3.7), $f(v_1) \geq UB$, and the strict concavity of f implies

$$UB = f(v) > \min\{f(v_1), f(v_2)\} = f(v_2),$$

so that (5.20) is contradicted. □

We also recall that any branch-and-bound algorithm has two phases. In phase I the global optimum value is searched for ($UB > f^*$). In phase II optimality of the current incumbent is checked ($UB = f^*$). Of course, it is not possible, during the execution of the algorithm, to know in which of the two phases we are, but it is typical for branch-and-bound algorithms to have a phase II longer than phase I. What we note is that if Property 1 is satisfied, then during phase I the objective function may be modified, but during phase II the objective function is equal to f and is never changed. If Property 1 is not satisfied but (4.16) holds, then in phase II the algorithm only modifies f into $f + g$ and then it does not change it any more. Therefore, at least in phase II, multiple modifications of the objective function are introduced only when none of the above mentioned conditions and properties holds.

We finally underline that it is possible to consider a variant of the above algorithm in which during the computations not all the cuts are kept in memory but only the last T of them, where T is some predefined positive integer. This may have the disadvantage that the same vertex is cut more than once, but, on the other hand, it avoids solving at each iteration linear subproblems with, possibly, a very high number of constraints. A good balance between the computational loss of time due to repeated cuts and the computational gain of time due to smaller linear subproblems has to be found. It can be seen that if Property 1 holds, this modification does not affect the finiteness result for Algorithm 2, at least when exhaustive (shrinking to a point) subdivision processes are used. We do not give a detailed proof of this fact here, but we note that exhaustiveness implies that when the iteration counter k is big enough, all the simplices $S \in \mathcal{M}$ containing a global optimum vertex v have a small diameter. As soon as the vertex is detected through a local search, the concavity cut also cuts off all these simplices. In particular, for each of them the feasible region of the linear subproblem (2.2) becomes empty and, consequently, the fathoming rule excludes them from further consideration.

6. Numerical experiments

In the numerical computations the basic simplicial algorithm, Algorithm 1, employing the bisection subdivision strategy, has been compared with Algorithm 2, employing the same subdivision strategy. The following objective functions, taken from the literature (see [2 and 7]), have been considered:

$$f_1(x) = -\mid x_1 + \sum_{j=2}^{n} \frac{j-1}{j} x_j \mid^{\frac{3}{2}};$$

Table I. CPU times for problems with different values of n, different objective functions and different accuracy

Obj.Func.	n	ε	Alg. 1	Alg. 2
f_2	5	10^{-1}	0.48	0.07
f_2	5	10^{-10}	3.42	0.07
f_2	6	10^{-1}	2.67	0.07
f_2	6	10^{-2}	23.48	0.07
f_2	6	10^{-10}	153.82	0.07
f_3	5	10^{-1}	13.45	12.21
f_3	5	10^{-4}	17.80	12.37
f_3	6	10^{-1}	319.61	137.42
f_3	6	10^{-2}	444.09	141.28
f_3	6	10^{-10}	1213.67	146.70
f_4	6	10^{-1}	133.55	93.08
f_4	6	10^{-10}	346.02	93.11

$$f_2(x) = -\mid \sum_{j=1}^{n} \frac{1}{j} x_j \mid \log \left(1 + \mid \sum_{j=1}^{n} \frac{1}{j} x_j \mid \right);$$

$$f_3(x) = -3 \sum_{j=1}^{n} x_j^2 + 2(\sum_{j=1}^{n-1} x_j x_{j+1});$$

$$f_4(x) = -(\sum_{j=1}^{n} x_j^2) \log(1 + \sum_{j=1}^{n} x_j^2),$$

We note that functions f_3, f_4 are strictly concave. Therefore, Algorithm 3 performs in the same way as Algorithm 2. Functions f_1, f_2 are not strictly concave, but the two algorithms still behave in the same way. Indeed, being these two functions indirectly linear, it can be proved that the first local search detects the global optimum and the first concavity cut eliminates the whole polytope. Therefore, after at most one iteration Algorithm 2 stops for any possible accuracy ε.

All computations have been performed on a SUN MICROSPARC II. Two different kinds of experiments have been carried out.

The first kind of experiments show through some examples the dependency of the performance of the simplicial algorithm based on bisections from the choice of the predefined accuracy and that the proposed modifications often seem to considerably reduce this dependency, as it is intuitively clear in view of the finiteness

result. The results are presented in Table 1. The feasible region is the set $[0, 1]^n$. The first column indicates the objective function used, the second one the value of n, the third one the predefined accuracy and the fourth and fifth one the CPU times respectively for Algorithms 1 and 2. It is interesting to note that, while the results for the basic simplicial algorithms are strongly dependent on the choice of the accuracy, the modified algorithm is almost independent from this choice.

The second kind of experiments, presented in the third subsection, is performed on randomly generated problems. Feasible regions have been randomly generated for different values of the number of variables n and the number of constraints m. To each problem, nonnegativity constraints for the variables have been added. The procedure employed to generate the random problems is the one presented in [10]. We do not report here the comparison between the two algorithms for the functions f_1 and f_2 but we underline that while the modified algorithm is able to exploit as much as possible the peculiar structure of these objective functions and stops after at most one iteration, the basic simplicial branch-and-bound algorithm based on bisections has not this ability and may perform quite badly. Indeed, for some instances the basic algorithm did not stop after 500000 iterations and more than 9000 seconds of CPU time.

Also the computations with functions f_3 and f_4, whose results are reported in Tables 2 and 3, show that in most cases the modified algorithm outperforms the original one. This is always true from the point of view of the maximum memory requirements and the total number of iterations, which must necessarily hold, and most of the times from the point of view of the required CPU times, which could not be said in advance in view of the increased computational effort due to the computation of cuts and the larger linear subproblems. Tables 2 and 3 report the results obtained for the functions f_3 and f_4. For each table the following notation has been employed.

n denotes the dimension of the problem (varying between 5 and 9);

m denotes the number of constraints (varying between 8 and 16) without considering the nonnegative constraints;

T_1 denotes the CPU times for Algorithm 1;

T_2 denotes the CPU times for Algorithm 2;

M_1 denotes the memory requirements, i.e. the maximum number of nodes in the branch-and-bound tree, for Algorithm 1;

M_2 denotes the memory requirements for Algorithm 2;

I_1 denotes the total number of iterations for Algorithm 1;

Table II. CPU times, memory requirements and number of iterations for function f_3

n	m	T_1	T_2	M_1	M_2	I_1	I_2
5	8	1.16	0.14	13	1	24	4
5	10	1.24	0.28	11	4	132	13
5	12	4.42	2.38	51	34	312	147
5	14	2.51	1.33	25	17	174	76
5	16	4.06	2.49	44	35	276	143
6	8	3.12	6.44	61	39	541	248
6	10	10.61	5.66	122	79	803	399
6	12	13.76	12.49	222	205	923	766
6	14	21.04	17.33	256	214	1430	1064
6	16	20.18	12.70	152	106	1184	645
7	8	2.75	1.56	31	18	186	72
7	10	160.43	115.66	1221	1055	10774	6178
7	12	382.96	458.22	2993	2984	22798	22319
7	14	89.25	84.69	810	779	4529	4001
7	16	108.64	97.58	1028	954	5487	4590
8	8	11.05	0.12	33	1	839	1
8	10	482.70	412.29	3355	3105	25532	19276
8	12	483.15	286.16	3378	2229	24746	13719
8	14	318.54	273.17	2693	2323	15435	11749
8	16	112.29	84.31	479	428	4759	2692
9	8	11.75	1.03	64	8	874	38
9	10	704.86	621.47	5634	4801	37552	29943
9	12	731.08	311.55	4079	2062	38351	14248
9	14	534.16	568.15	4411	4223	24420	23509
9	16	1376.74	261.22	5249	1989	72143	9450

I_2 denotes the total number of iterations for Algorithm 2.

Both the algorithms have been stopped when accuracy $\varepsilon = 10^{-1}$ has been reached. In the computations it has been chosen the following value for δ

$$\delta = \max\{0.01 \mid UB \mid, 0.01\},$$

however it is possible to consider alternative choices, such as an adaptive update of this value. A small value of δ may cause the algorithm to need a lot of iterations before starting a local search leading to a vertex of P with function value equal to UB, which is then cut off. Therefore, in some cases the value of δ should be increased. For instance, if the algorithm is generating many points with function

Table III. CPU times, memory requirements and number of iterations for function f_4

n	m	T_1	T_2	M_1	M_2	I_1	I_2
5	8	1.17	0.28	11	4	140	16
5	10	1.38	0.43	11	7	145	24
5	12	6.19	3.33	80	44	452	229
5	14	1.94	0.70	18	12	135	36
5	16	3.53	4.60	38	38	236	225
6	8	7.43	5.08	95	71	609	389
6	10	12.60	8.33	151	144	927	559
6	12	9.89	9.24	144	139	637	534
6	14	24.15	21.79	282	280	1601	1336
6	16	63.11	31.61	572	260	3843	1778
7	8	8.15	4.22	77	48	598	214
7	10	268.13	243.53	2618	2420	17512	13802
7	12	332.27	304.77	3142	3034	19230	16568
7	14	101.19	96.20	852	795	5063	4496
7	16	233.73	225.23	2339	2269	12199	10995
8	8	15.60	0.11	33	1	1167	1
8	10	432.15	331.61	3543	2783	21706	15353
8	12	566.76	416.86	4993	3465	28544	20618
8	14	565.40	506.73	4528	4493	27117	21940
8	16	275.62	166.50	1167	1034	12178	6082
9	8	19.38	1.24	64	13	1473	46
9	10	1355.01	1268.29	10076	7904	68341	57714
9	12	864.09	804.53	4743	3599	44885	33745
9	14	1249.05	1304.46	7781	7699	51433	50582
9	16	441.41	331.30	2275	1860	17339	11513

value close to UB but greater than $UB + \delta$, so that no local search is started, then δ should be increased in order to start at least one local search and, in case this leads to a local minimum vertex with function value not greater than UB, to cut this vertex. On the other hand, if the value of δ is too large, many local searches are started leading to noninteresting local minima, i.e. local minima with function value greater than UB. In such a case it seems to be sensitive for the algorithm to decrease the value of δ.

Finally, we recall that only bisections have been employed. More sophisticated subdivision rules should be tested in the future.

7. Conclusion

In this paper simplicial branch-and-bound algorithms for concave optimization have been considered. After having introduced the concave optimization problem and the basic structure of simplicial branch-and-bound algorithms, some modifications of the basic simplicial algorithms, consisting of local searches and concavity cuts, have been introduced. The modified algorithm has been proven to be finite if a given assumption holds. Since the assumption can not be guaranteed to hold for general concave functions, conditions under which it certainly holds have been presented. Moreover, some ideas have been discussed to deal with the cases in which the conditions are not fulfilled. The ideas consist in modifying the objective function, but only when, according to the information collected by the algorithm, this becomes necessary in order to ensure finiteness. All that has led to the description of a new algorithm which has been proved to be finite for general concave functions. Some preliminary computational results are presented aiming at showing that the proposed modifications are not only of theoretical interest since they enforce finiteness, but are also of practical interest. The preliminary computations seem to show that the modified algorithm has a lower dependency on the predefined accuracy, and that it often outperforms the basic simplicial algorithm when applied to randomly generated problems.

Finally, we also point out that, even if only simplicial branch-and-bound algorithms have been considered, it seems to be possible to extend the ideas presented here to other branch-and-bound algorithms. Indeed, the modifications of convergent simplicial branch-and-bound algorithms, that we have introduced here in order to enforce finiteness, i.e. concavity cuts and, when necessary, updates of the objective function, are not dependent on the geometrical objects (simplices, cones, rectangles) through which the feasible region is partitioned.

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 Sayin, S. (1994) A finite concave minimization algorithm using branch and bound and neighbor generation, *Journal of Global Optimization*, 5: 1–14.
3. Benson, H.P. (1995) Concave minimization: theory, applications and algorithms in R.Horst, P.Pardalos (eds.), *Handbook of Global Optimization*, pp. 43–148, Kluwer Academic Publishers, Dordrecht, The Netherlands
4. Hamami M., Jacobsen S.E., (1988) Exhaustive nondegenerate conical processes for concave minimization on convex polytopes, *Mathematics of Operations Research* 13, 479–487.
5. Horst R., (1976) An algorithm for nonconvex programming problems, *Mathematical Programming* 10: 312–321.
6. Horst R., (1984) On the global minimization of concave functions. Introduction and survey, *Operations Research Spektrum* 6: 195–205.
7. Horst R., Thoai N.V., (1988) Modification, implementation and comparison of three algorithms for globally solving concave minimization problems, *Computing*, 42: 271–289

8. Horst R., Pardalos P.M. and Thoai N.V., (1995) 'Introduction to global optimization', Kluwer Academic Publishers, Dordrecht, The Netherlands
9. Horst R. and Tuy H., (1996) *Global optimization: deterministic approaches*, (third edition), Springer-Verlag Berlin Heidelberg New York (1996)
10. Nast M., (1996) Subdivision of simplices relative to a cutting plane and finite concave minimization, *Journal of Global Optimization*, 9: 65–93
11. Papadimitriou C.H., Steiglitz K., (1982) *Combinatorial optimization: algorithms and complexity*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey.
12. Pardalos P.M., Rosen J.B., (1986) Methods for global concave optimization: a bibliographic survey, *SIAM Review* 26, 367–379.
13. Pardalos P.M., Schnitger G., (1988) Checking local optimality in constrained quadratic programming is NP-hard, Operations Research Letters, 7: 33–35.
14. Shectman J.P., Sahinidis N.V., (1998) A finite algorithm for global minimization of separable concave function, *Journal of Global Optimization* 12: 1–36
15. Tam B.T., Ban V.T., (1985) Minimization of a concave function under linear constraints, *Economika i Mathematicheskie Metody* 21: 709–714, in Russian
16. Tuy, H. (1991) Concave programming under linear constraints, *Soviet Mathematics* 5: 1437–1440
17. Tuy, H. Effect of the subdivision strategy on convergence and efficiency of some global optimization algorithms, *Journal of Global Optimization*, 1: 23–36
18. Tuy, H. (1991) Normal conical algorithmfor concave minimization over polytopes, *Mathematical Programming* 51: 229–245