Les Cahiers du GERAD ISSN: 0711–2440

A simple finite simplicial covering algorithm for concave minimization over a polytope

C. Meyer

G–2011–01

January 2011

A simple finite simplicial covering algorithm for concave minimization over a polytope

Christophe Meyer

GERAD - HEC Montréal
3000 chemin de la Côte-Sainte-Catherine
Montréal (Québec) H3T 2A7, Canada.
christophe.meyer@gerad.ca

January 2011

Les Cahiers du GERAD
G–2011–01

Abstract

We propose a conceptually simple, finite simplicial branch-and-bound algorithm for minimizing a concave function over a polytope. The proposed algorithm requires at each iteration the solution of two closely related linear programs of constant size for the computation of the lower bound and of the subdivision point; moreover, the objective function need to be evaluated only at extreme points of the polytope and of the initial simplex. Preliminary computational results are presented, which point to future improvements.

Key Words: concave minimization, simplicial covering algorithm.

Résumé

Nous proposons un algorithme conceptuellement simple d'énumération implicite de simplexes pour la minimisation d'une fonction concave sur un polytope en un nombre fini d'itérations. Cet algorithme requiert à chaque itération la solution de deux programmes linéaires étroitement liés et de taille constante pour le calcul de la borne inférieure et du point de subdivision; de plus la fonction objectif n'a besoin d'être évaluée qu'en des points extrêmes du polytope et du simplexe initial. Des résultats numériques préliminaires sont présentés, qui suggèrent des améliorations futures.

Mots clés : Minimisation concave; algorithme de recouvrement simplicial.

1 Introduction

The concave minimization problem

$$(CP) \qquad min\{f(x) \mid x \in P\}$$

consists in finding a global minimizer x^* of the concave function f over the polytope $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ where A is a matrix of $\mathbb{R}^{m \times n}$ and b a vector of $\mathbb{R}^m$. It is well known that this problem is NP-hard, even if the objective function is quadratic concave and the polytope is a hypercube, see for example [9]. On the other hand, this problem has the nice property that its optimal value is attained at one extreme point of the polytope (we will refer to this property as the *extreme point property*). This property suggests a finite algorithm to solve problem (CP), with the number of iterations depending only on the size of the polytope. It is the purpose of this paper to propose such an algorithm. Although our algorithm can solve in principle any instance of problem (CP), it is more specially intended for instances where the polytope may have degenerate extreme points and/or where the objective function is costly to evaluate. Examples of applications where the polytope might be degenerated include the minimum concave-cost flow problems (see e.g. [4]).

The solution methods for problem (CP) can be roughly divided in three classes: enumerative methods, successive approximation methods and branch-and-bound methods (see e.g. Benson [2] for a survey on these methods and for references to applications).
Among the enumerative methods,the extreme points ranking method exploits directly the extreme point property by enumerating the extreme points of P in nondecreasing order with respect to the value of a linear underestimator of f over P. This method suffers from the large number of extreme points to explore, due to the often bad quality of the linear underestimator, and more importantly, is very sensitive to the presence of degenerate extreme points (see [2] and references therein).
On the other side, branch-and-bound methods based on partitioning do not exploit at all the extreme point property. The first simplicial branch-and-bound method was proposed by Horst [5] in 1976. At a given iteration of this algorithm, we have at hand an incumbent solution and a collection of simplices that cover the part of the polytope that might still contain a point better that the incumbent solution. To conclude that a simplex (or more precisely, the intersection of that simplex with the polytope) cannot improve on the incumbent solution, a lower bound on f is computed by minimizing a (local) linear underestimator of f over the portion of the polytope contained in the simplex. If this lower bound is larger than the incumbent value, the simplex cannot contain a better feasible point and can therefore be eliminated from further consideration. Otherwise the simplex will eventually be partitioned into subsimplices through a so-called *radial subdivision* with respect to a point of the simplex. Special cases of this subdivision process include the bisection - the subdivision point is the middle point of a longest edge of the simplex - and the ω-partition - the subdivision point is the optimal solution ω obtained in the bounding operation. Since this algorithm and their variants cannot be showed to be finite, the focus has been made on showing convergence at infinity. The infinite convergence property implies the finitness of the algorithm if we are satisfied with an ε-optimal solution with $\varepsilon > 0$; in that case however, the number of iterations will depend on the value of ε. It was for a long time an open question whether the simplicial partitioning algorithm using only ω-partitions was convergent or not: this question has been positively answered by Locatelli and Raber [11].
Although they perform quite well in practice, the partitioning branch-and-bound algorithms are not completely satisfying from a mathematical point of view, because we know that an optimal solution to problem (CP) can be found in a finite number of steps. Several authors have tried to modify the simplicial partitioning algorithm to achieve finitness. Benson [1] and Benson and Sayin [3] propose to minimize the linear underestimator defined for the current simplex on the whole polytope, instead of on the portion of the polytope contained in the simplex. This yields an extreme point of the polytope, lying generally outside the simplex, which is stored in a list. When a simplex has to be subdivided, one looks in the list for an extreme point contained in the simplex and partition the simplex with respect to this point. Unfortunately the extreme points computed during the bounding steps do not suffice: it also necessary to add to the list all neighbours of these extreme points, a step that can be computationally costly if the polytope is degenerated. Nast [16] proposes a finite simplicial partitioning algorithm using a completely different subdivision rule: given a constraint and a simplex to be subdivided, a simplicial partition of the part of the simplex satisfying the linear inequality is built. Locatelli and Thoai [12] ensure finitness of the simplicial partitioning algorithm by a rather complex method that involves cuts generation and modification of the objective function.

Also partially relevant to this paper are works that do not seek a finite algorithm but that propose either a covering algorithm or to minimize the linear underestimator of the current simplex over the whole polytope. Horst, Thoai and De Vries [7] propose a covering simplicial algorithm with the purpose of eliminating rapidly large parts of the interior of the polytope. The extreme points of P do not play any role in the definition of the subsimplices, and only infinite convergence is proven. Kuno and Nagai [10] minimize the linear underestimator over the whole polytope, obtaining an extreme point of P. This extreme point is however only used to try to improve the incumbent value, since bisections are used to partition the simplex.

There also exist branch-and-bound algorithms to solve problem (CP) that use other geometrical objects in place of simplices, more particularly rectangles and cones. Several finite conical covering algorithms have already been proposed [14, 15]. The algorithm that is proposed here for simplices is inspired from [14].

In this paper, we propose a conceptually simple, finite covering algorithm for solving problem (CP). Instead of subdividing a simplex with respect to a point that lies inside the simplex as proposed in Horst [5], we allow the subdivision point to be outside the simplex. This will yield a covering of the current simplex, instead of a partitioning. The subdivision point is obtained by minimizing the linear underestimator of f defined for this simplex over the face of P containing ω (we recall that ω is the optimal solution obtaining in the bounding procedure, which consists in minimizing the linear underestimator over the intersection of the current simplex with the polytope). At first look, it might seem a total nonsense to replace a simplex by subsimplices which together are strictly larger (in terms of inclusion) than the simplex! It turns out however that we are able to prove a monotonicity result on the lower bound: the lower bound of a subsimplex can never be worst than the lower bound of its parent simplex. We then get the same monotonicity result than in simplicial partitioning algorithms, with the advantage that simplices are defined by extreme points of the polytope. An interesting property of this algorithm is that we need to evaluate the objective function only at extreme points of the polytope. One can therefore expect that the number of evaluations of the objective function will be much smaller than in a partitioning algorithm, which may be particularly interesting if the function is costly to evaluate.

The remaining of this paper is organized as follows. In Section 2 we describe the basic operations that are needed by the algorithm, while the algorithm itself is described in Section 3. The finitness result is stated and proved in Section 4. In Section 5, we discuss the implementation and present some very preliminary computational results, that suggest some future improvements. Finally we conclude in Section 6.

2 Basic operations

In this section, the basic operations of the algorithm are described. The construction of an initial simplex (Section 2.1) and the computation of a lower bound (Section 2.2) are two standard operations from the literature, that we recall for completeness. As for the subdivision of a simplex (Section 2.3), we generalize the radial subdivision of Horst [5] to the case where the subdivision point lies outside the simplex. The computation of the subdivision point is the main novelty of this paper. The monotonicity result is also given in this subsection. Finally we need to check for cycling: this is explained in Section 2.4.

2.1 Initial simplex

The first task of the algorithm is to construct a simplex S that contains the polytope P. We use the following simple method: we look for a simplex S of the following form: $S = \{x \in \mathbb{R}^n : x_i \geq d_i (i = 1, \ldots, n); \sum_{i=1}^{n} x_i \leq d_0\}$ where $d_i (i = 0, \ldots, n)$ are constants that are determined by solving the following $n + 1$ linear programs: $\min_{x \in P} x_i (i = 1, \ldots, n)$ and $\max_{x \in P} \sum_{i=1}^{n} x_i$.

2.2 Lower bound

Given a simplex S, a lower bound of f over $S \cap P$ is obtained by minimizing f_S over $S \cap P$, where f_S denotes the convex envelope of f over S. Roughly speaking, the convex envelope of f over S is the greatest convex

underestimator of f over S, see e.g. Horst and Tuy [8] for a formal definition. It is well known that the convex envelope f_S of f over the simplex $S = conv\{v^0, v^1, \ldots, v^n\}$ spanned by the vertices $v^0, v^1, \ldots, v^n$ admits the following simple representation:

$$f_S(x) = \sum_{j=0}^{n} \lambda_j f(v^j)$$

where the $\lambda_j (j = 0, \ldots, n)$ are the *barycentric coordinates* of x, i.e., the unique solution to the system

$$x = \sum_{j=0}^{n} \lambda_j f(v^j)$$
$$\sum_{j=0}^{n} \lambda_j = 1$$

see, e.g., [8, Theorem IV.6].

As a consequence, the minimum $\alpha(S)$ of f_S over $S \cap P$ is the optimal value of the following linear program:

$$\begin{array}{ll} \min & \displaystyle\sum_{j=0}^{n} \lambda_j f(v^j) \\ \text{s.t.} & \left\{ \begin{array}{ll} \displaystyle a_i \left(\sum_{j=0}^{n} \lambda_j v^j \right) \leq b_i & i = 1, \ldots, m \\ \displaystyle\sum_{j=0}^{n} \lambda_j = 1 & \\ \lambda_j \geq 0, & j = 0, \ldots, n \end{array} \right. \end{array} \tag{1}$$

where a_i denote the i^{th} row of A, $i = 1, \ldots, m$.

We will denote by $\omega(S)$ the point of $P \cap S$ defined by $\omega(S) = \sum_{j=0}^{n} \lambda_j^* v^j$, where λ^* is an optimal solution of the linear program (1).

2.3 Subdivision

This subsection is divided into 3 parts: we first extend the notion of radial subdivision of a simplex to the case of a covering in Section 2.3.1. Then in Section 2.3.2, we define the new subdivision point. Finally in Section 2.3.3 we state and derive the monotonicity result on the lower bound of a subsimplex with respect to the one of its parent simplex.

2.3.1 Radial subdivision

We slightly generalize the notion of radial subdivision introduced in Horst [5] to the covering of a simplex as follows. Let again $S = conv\{v^0, v^1, \ldots, v^n\}$ be a simplex and let $v = \sum_{i=0}^{n} \lambda_i v^i$ be a point of $\mathbb{R}^n$ with $\sum_{i=0}^{n} \lambda_i = 1$. The only assumption on v is that v does not coincide with any of the vertices of S. In particular, v may lie outside S, a situation that will be characterized by some λ_i being negative.
Let $J_> = \{j \,|\lambda_j > 0\}$. For all $j \in J_>$, define S^j to be the simplex obtained by replacing v^j by v in S.

Proposition 1 *The set of simplices $S^j (j \in J_>)$ forms a covering of the simplex S..*

Proof. We have to prove that $S \subseteq \bigcup_{j \in J_>} S^j$. Let x be any point of S. We want to show that there exists some ℓ with $\lambda_\ell > 0$ such that $x \in S^\ell$. Since x belongs to S, there exists $\mu \geq 0$ such that $x = \sum_{j=0}^{n} \mu_j v^j$, $\sum_{j=0}^{n} \mu_j = 1$.

Let ℓ be such that $\frac{\mu_\ell}{\lambda_\ell} = \min_{j \in J_>} \frac{\mu_j}{\lambda_j}$. We have

$$\begin{aligned} x &= \sum_{j=0, j\neq\ell}^{n} \mu_j v^j + \frac{\mu_\ell}{\lambda_\ell}\left(v - \sum_{j=0, j\neq n}^{n} \lambda_j v^j\right) \\ &= \sum_{j=0, j\neq\ell}^{n} \left(\mu_j - \frac{\mu_\ell}{\lambda_\ell}\lambda_j\right) v^j + \frac{\mu_\ell}{\lambda_\ell} v. \end{aligned}$$

By definition of ℓ, $\mu_j - \frac{\mu_\ell}{\lambda_\ell}\lambda_j \geq 0$ for $j = 0, \ldots, n, j \neq \ell$ and $\frac{\mu_\ell}{\lambda_\ell} \geq 0$. Moreover

$$\sum_{j=0, j\neq\ell}^{n} \left(\mu_j - \frac{\mu_\ell}{\lambda_\ell}\lambda_j\right) + \frac{\mu_\ell}{\lambda_\ell} = (1 - \mu_\ell) - \frac{\mu_\ell}{\lambda_\ell}(1 - \lambda_\ell) + \frac{\mu_\ell}{\lambda_\ell} = 1.$$

Hence x belongs to S^ℓ. □

We will say that a simplex S^j $(j \in J_>)$ is a *son* of the simplex S by the subdivision process and that S is a *father* of S^j. Note that since we are dealing with covering, a same simplex could have several fathers.

2.3.2 A new subdivision point

To show the finitness of our algorithm we need that all simplices are subdivided with respect to an extreme point of the polytope. If $\omega(S)$ is an extreme point of P, we choose $v = \omega(S)$. We now explain how to define the subdivision point v if $\omega(S)$ is not an extreme point of P.

Let $I_=$ be the subset of the constraints of P satisfied at equality by the optimal solution λ^* of proble (1). We consider the new linear program:

$$\begin{aligned} \min \quad & \sum_{j=0}^{n} \lambda_j f(v^j) \qquad (2) \\ \text{s.t.} \quad & \begin{cases} a_i \left(\sum_{j=0}^{n} \lambda_j v^j\right) = b_i & i \in I_= \\ a_i \left(\sum_{j=0}^{n} \lambda_j v^j\right) \leq b_i & i \in \{1, \ldots, m\} \setminus I_= \\ \sum_{j=0}^{n} \lambda_j = 1. \end{cases} \end{aligned}$$

Observe the differences with the linear problem (1): the constraints of P satisfied at equality by λ^* were transformed to equalities, and non-negativity constraints on the variables λ were removed. Let λ'^* be a basic optimal solution of (2), and let $\alpha'(S)$ be its optimal value. We claim the following:

- Problem (2) is feasible and $\alpha'(S) \leq \alpha(S)$. Indeed λ^* is a feasible solution to (2) with value $\alpha(S)$, and (2) is a minimization problem.
- $\omega'(S) = \sum_{j=0}^{n} \lambda'^*_j v^j$ is an extreme point of P.

We take $v = \omega'(S)$ for the radial subdivision of the simplex S. Note that if $\omega(S)$ is an extreme point of P, $\omega'(S) = \omega(S)$.

2.3.3 Monotonicity result

The following result is crucial for our finitness result:

Proposition 2 *Assume that $f(\omega'(S)) > \alpha(S)$. Then $\alpha(S^\ell) \geq \alpha(S)$ for every son S^ℓ of simplex S.*

Proof. The dual of (1) is

$$\begin{array}{ll} \max & -\sum_{i=1}^{m} \mu_i b_i + \pi \\ \text{s.t.} & \begin{cases} -\sum_{i=1}^{m} \mu_i a_i v^j + \pi \leq f(v^j), & j = 0, \ldots, n \\ \mu_i \geq 0 & i = 1, \ldots, m. \end{cases} \end{array} \tag{3}$$

Let (μ^*, π^*) be an optimal solution. Let us show that (μ^*, π^*) is a feasible solution to the dual associated with simplex S^ℓ. We have to show that

$$-\sum_{i=1}^{m} \mu_i^* a_i \omega'(S) + \pi^* \leq f(\omega'(S)).$$

Since $a_i\omega(S) = b_i$ for all i such that $\mu_i^* > 0$ by the complementary slackness conditions (see e.g. Luenberger [13]), we have that $a_i\omega'(S) = b_i$ for such i by construction of the linear program (2), hence

$$-\sum_{i=1}^{m} \mu_i^* a_i \omega'(S) + \pi^* = -\sum_{i=1}^{m} \mu_i^* b_i + \pi^*.$$

By duality, the right-hand side is equal to $\alpha(S)$, which by assumption, is strictly less than $f(\omega'(S))$. It follows that (μ^*, π^*) with objective value $-\sum_{i=1}^{m} \mu_i^* b_i + \pi^* = \alpha(S)$ is feasible for the dual (3) associated with the simplex S^ℓ. Since the dual is a maximization problem, we conclude that $\alpha(S^\ell) \geq \alpha(S)$. □

Note that the assumption in Proposition 2 is not restrictive. Indeed if the assumption is not satisfied, i.e., if $f(\omega'(S)) \leq \alpha(S)$, then the simplex S can be eliminated and will therefore not be subdivided.
Note also that the proof did not exploit the fact that λ'^* is an optimal solution of (2). Actually the proof of Proposition 2 remains valid if instead of $\omega'(S)$, we take any feasible point v of the face F of P containing $\omega(S)$. However in order to get a finite algorithm, we need that v is an extreme point of the face F which satisfies $f_S(v) < f(v)$.

2.4 Anti-cycling

It could happen that the algorithm generates a sequence of simplices $S^p, S^{p+1}, \ldots, S^{p+t}$ for some integers p and t where for each $k = p, \ldots, p+t-1$, S^{k+1} is a son of S^k, and such that $S^{p+t} = S^p$. This is what we call a *cycle*. In order to avoid generating again an already subdivided simplex that would cause the algorithm to enter in an infinite loop, we need to test for each subsimplex whose lower bound equals the one of its father if that subsimplex was not already subdivided. This testing can be done relatively easily if we use the standard practice from the literature that consists in selecting for subdivision at each iteration the simplex waiting to be subdivided that has the smallest lower bound. Indeed let $\underline{f}_h$ denote the lower bound of the simplex selected to be subdivided at iteration h. By Proposition 2, $\{\underline{f}_h\}$ defines a non-decreasing sequence, therefore cycling can only involve simplices corresponding to the same value of $\underline{f}_h$. Assume that at iteration h_1, a new value for $\underline{f}_h$ is obtained, i.e., $\underline{f}_{h_1} > \underline{f}_{h_1-1}$. We will store in a list all subdivided simplices that have $\underline{f}_{h_1}$ as lower bound. When generating a subsimplex of lower bound equal to $\underline{f}_{h_1}$, we first scan the list to check that the subsimplex was not already subdivided; if yes we do not generate it again. The list is emptied when we arrive at an iteration h_2 where $\underline{f}_{h_2} > \underline{f}_{h_1}$.

3 Algorithm

We now proceed to the description of our algorithm. At the beginning of an iteration, the following quantities are available:

- *Cover*: the set of simplices that have yet to be processed;

- $\underline{f} = \min_{S \in \mathcal{C}over} \alpha(S)$: lower bound on the optimal value f^*.
- $\overline{x}$ and $\overline{f}$: respectively the current incumbent solution and the current incumbent value.
- $\mathcal{D}one$: set of simplices S that have been processed and whose lower bound $\alpha(S)$ is equal to $\underline{f}$: this set will have to be checked when generating a new simplex in order to prevent cycling, see Section 2.4.
- $\mathcal{N}ew$: set of new simplices resulting from the subdivision of the selected simplex at the previous iteration (at the first iteration, $\mathcal{N}ew$ will contain the initial simplex).

The algorithm is now as follows:

Step 1 (initialization) : construct an initial simplex S_0 containing P as explained in Section 2.1. Let $\mathcal{N}ew = \{S_0\}$. Set $\underline{f}$ to $-\infty$ and $\overline{f}$ to $+\infty$. Initialize the sets $\mathcal{C}over$ and $\mathcal{D}one$ to $\emptyset$.

Step 2 (fathoming) : for each simplex S in $\mathcal{N}ew$, solve the linear program (1), obtaining the optimal value $\alpha(S)$. Remove from $\mathcal{N}ew$ all simplices S for which $\alpha(S) \geq \overline{f}$.

Step 3 (anti-cycling) : remove from $\mathcal{N}ew$ all simplices that are in $\mathcal{D}one$.

Step 4 (subdivision point computation) : for each simplex S in $\mathcal{N}ew$, let $\omega'(S)$ be the extreme point of P corresponding to a basic optimal solution of problem (2). If $f(\omega'(S)) < \overline{f}$, update $\overline{x}$ and $\overline{f}$.. If $\underline{f} = \overline{f}$: stop and return $\overline{x}$; otherwise add to $\mathcal{C}over$ the simplices of $\mathcal{N}ew$.

Step 5 (simplex selection) : if $\mathcal{C}over = \emptyset$, stop: return $\overline{x}$. Otherwise select the simplex $\tilde{S} = \arg\min_{S \in \mathcal{C}over} \alpha(S)$ and remove it from $\mathcal{C}over$. If $\alpha(\tilde{S}) > \underline{f}$, reset $\mathcal{D}one$ to $\emptyset$ and $\underline{f}$ to $\alpha(\tilde{S})$. Add $\tilde{S}$ to $\mathcal{D}one$.

Step 6 (subdivision) : subdivide the simplex $\tilde{S}$ via the point $\omega'(\tilde{S})$ as indicated in Section 2.3. Let $\mathcal{N}ew$ be the set of sons of $\tilde{S}$. Return to Step 2.

4 Finitness result

In this section, we show that the algorithm described in Section 3 is finite and that it provides an optimal solution of problem (CP).

Theorem 1 *The algorithm described in Section 3 terminates after a finite number of iterations with a global minimizer to problem (CP).*

Proof. We first observe that the simplices generated by the algorithm are in finite number: indeed the vertices defining the simplices are the extreme points of P and the $n+1$ vertices of the initial simplex S_0. It might be convenient to view the algorithm as generating a graph G defined as follows: the vertices of G correspond to the simplices generated by the algorithm, and there is an arc from simplex S^i to simplex S^j if and only if S^j was obtained from S^i through a subdivision (note that G may not be a directed tree since a same simplex could be the sons of several different simplices). Proposition 2 ensures that if we progress along a path of G, the lower bound α of the encountered simplices will never decrease. In principle, we could have a cycle between simplices of same lower bound, but Step 3 makes this impossible. Since no cycle is possible, the algorithm will indeed be finite. The main effort in the proof will therefore be to prove that when the algorithm stops, it has found an optimal solution of problem (CP).

Assume that the incumbent value was $\overline{f}$ when the algorithm stopped, and assume by contradiction that there exists $x' \in P$ with $f(x') < \overline{f}$. Let $\gamma = \overline{f}$ and $\gamma' = f(x')$. Since f is concave on $\mathbb{R}^n$, it is also continuous (see, e.g., Rockafellar [17]), hence by definition

$$\forall \varepsilon > 0 \quad \exists \delta_\varepsilon > 0 \quad | \quad x \in B(x', \delta_\varepsilon) \Rightarrow |f(x) - \gamma'| < \varepsilon$$

where $B(x', \delta_\varepsilon)$ denote the ball of center x' and radius δ_ε.
We fix ε to $\frac{\gamma - \gamma'}{2}$, which implies that $f(x) \leq \frac{\gamma + \gamma'}{2} < \gamma$ for all $x \in B(x', \delta_\varepsilon)$. To any subset E_i of $vert(P) \cup vert(S_0)$ with $|E_i| \leq n$, we associate the affine subspace H_i spanned by the elements of E_i. Since the H_i are in finite number and of dimension $< n$, the set $B(x', \delta_\varepsilon) \cap P \setminus \left(\bigcup_i H_i \right)$ is non-empty. Let $\tilde{x} \in B(x', \delta_\varepsilon) \cap P \setminus$

$\left(\bigcup_i H_i\right)$: then $f(\tilde{x}) < \gamma$. We consider the sequence $\{S^q\}, q = 0, 1, \ldots$ of simplices containing $\tilde{x}$ such that for all q, S^{q+1} is a son of S^q by the subdivision process of Step 6. Note that $\alpha(S^q) < \gamma$ for all q, since otherwise $f(x) \geq \gamma$ for all $x \in S^q \cap P$, contradicting the fact that $f(\tilde{x}) < \gamma$. Hence all simplices of the sequence belong to the set of simplices generated by the algorithm.. Moreover since the subdivision process is always defined, the sequence is infinite. This infinite sequence could be interrupted in our algorithm by the anti-cycling rule of Step 3. To show that this is not the case, and to get a contradiction that will allow us to conclude, we will show that all simplices in the sequence must be distinct. To show that all simplices in the sequence are distinct, assume that $S^q = conv\{v^{q0}, v^{q1}, \ldots, v^{qn}\}$ contains $\tilde{x}$. Then there exists a unique vector $\tilde{\zeta}^q > 0$ such that

$$\tilde{x} = \sum_{j=0}^{n} \tilde{\zeta}_j^q v^{qj}$$
$$\sum_{j=0}^{n} \tilde{\zeta}_j^q = 1$$

(the fact that no component of the vector $\tilde{\zeta}^q$ is null is a consequence of the assumption $\tilde{x} \notin \bigcup_i H_i$). We claim that $\sum_{j=0}^{n} \tilde{\zeta}_j^q f(v^{qj}) < \gamma$. Indeed, if $\sum_{j=0}^{n} \tilde{\zeta}_j^q f(v^{qj}) \geq \gamma$, by concavity of f, $f(\tilde{x}) = f\left(\sum_{j=0}^{n} \tilde{\zeta}_j^q v^{qj}\right) \geq \sum_{j=0}^{n} \tilde{\zeta}_j^q f(v^{qj}) \geq \gamma$, a contradiction. It is now not difficult to show that the simplex S^{q+1} is the son of S^q obtained by replacing the point $v^{q\ell_q}$ by $w^q = \omega'(S^q) = \sum_{j=0}^{n} \lambda_j^q v^{qj}$ with ℓ_q satisfying $\frac{\tilde{\zeta}_{\ell_q}^q}{\lambda_{\ell_q}^q} = \min_{j|\lambda_j^q>0} \left\{\frac{\tilde{\zeta}_j^q}{\lambda_j^q}\right\}$ (see the proof of Proposition 1). We then have $S^{q+1} = conv\{v^{q+1,0}, \ldots, v^{q+1,n}\}$ with $v^{q+1,j} = v^{qj}$ for $j = 0, \ldots, n$, $j \neq \ell_q$ and $v^{q+1,\ell_q} = w^q$, and $\tilde{x} = \sum_{j=0}^{n} \tilde{\zeta}_j^{q+1} v^{q+1,j}$ with

$$\tilde{\zeta}_j^{q+1} = \begin{cases} \tilde{\zeta}_j^q - \frac{\tilde{\zeta}_{\ell_q}^q}{\lambda_{\ell_q}^q} \lambda_j^q & \text{if } j \neq \ell_q \\ \frac{\tilde{\zeta}_{\ell_q}^q}{\lambda_{\ell_q}^q} & \text{if } j = \ell_q. \end{cases}$$

Then

$$\begin{aligned} f_{S^{q+1}}(\tilde{x}) = \sum_{j=0}^{n} \tilde{\zeta}_j^{q+1} f(v^{q+1,j}) &= \sum_{j=0}^{n} \left(\tilde{\zeta}_j^q - \frac{\tilde{\zeta}_{\ell_q}^q}{\lambda_{\ell_q}^q} \lambda_j^q\right) f(v^{q,j}) + \frac{\tilde{\zeta}_{\ell_q}^q}{\lambda_{\ell_q}^q} f(w^q) \\ &= \sum_{j=0}^{n} \tilde{\zeta}_j^q f(v^{q,j}) + \frac{\tilde{\zeta}_{\ell_q}^q}{\lambda_{\ell_q}^q} \left(f(w^q) - \sum_{j=0}^{n} \lambda_j^q f(v^{q,j})\right) \\ &= \sum_{j=0}^{n} \tilde{\zeta}_j^q f(v^{q,j}) + \frac{\tilde{\zeta}_{\ell_q}^q}{\lambda_{\ell_q}^q} \left(f(w^q) - f_{S^q}(w^q)\right) \\ &> f_{S^q}(\tilde{x}), \end{aligned}$$

the last inequality holding because $f(w^q) > \alpha^q \geq \alpha'^q = f_{S^q}(w^q)$ (if $f(w^q) \leq \alpha^q$, the simplex S^{q+1} would have been eliminated) and $\frac{\tilde{\zeta}_{\ell_q}^q}{\lambda_{\ell_q}^q} > 0$ by the choice of $\tilde{x}$ and ℓ_q. Since the value of $f_S(\tilde{x})$ depends only on $\tilde{x}$ and on the simplex, this shows that a same simplex cannot repeat. We therefore have an infinite sequence of distinct simplices, which is not possible because there are a finite number of different simplices. Therefore our assumption that there exists a point x' of P with $f(x') < \overline{f}$ was false, which concludes the proof. □

It is not known presently if there can be cycles, i.e., if Step 3 of the algorithm is really necessary. A proof of the non-existence of a cycle would simplify the algorithm and its implementation and would make the proof

Table 1: Numerical results

Instance	algorithm	#iter.	max$\|\mathcal{C}over\|$	max$\|\mathcal{D}one\|$	#eval_f
(15,5)	covering	137 (19)	47	9	41
	ω-partition	107 (15)	41	7	50
(30,6)	covering	137680 (176)	20157	4693	201
	ω-partition	37013 (37)	7702	337	2862
(30,7)	covering	$>$ 230000 (186)	$\geq$ 182239	$\geq$ 27378	$\geq$ 481
	ω-partition	163034 (63)	37450	2129	9161

of Theorem 1 straightforward. It is not difficult to show that if a cycle exists, the intersection of the simplices involved in the cycle must contain at least one vertex of the initial simplex that is not in P. Moreover the current proof of Theorem 1 can be adapted to show that the dimension of the intersection must be less than n.

5 Implementation and computational experiments

In the covering algorithm described in Section 3, a same simplex may be generated several times. To ensure finitness, we need only to check that a generated simplex is not already in the set $\mathcal{D}one$. But for efficiency purpose, we need also to check that a generated simplex is not already in the set $\mathcal{C}over$, in order to avoid duplicate work. A simplex can be viewed as a list of $n+1$ pointers to vertices. In order to compare 2 simplices, it suffices then to compare their lists of pointers. This however assume that a same vertex is not generated more than once. In practice we have to add therefore a fourth list to the description of the algorithm, $\mathcal{V}ertex$, which contains the vertices already generated. Each time a vertex $\omega'(S)$ is generated in Step 4, we check if it is already in the list or not. In a first step, these lists were implemented in a very straightforward manner, implying that a search in a list involves traversing the whole list each time.
In fact, our main objective with this non-optimized implementation was to evaluate the number of iterations needed by the algorithm, as well as the number of function evaluations, and to compare these numbers with those of the simplicial partitioning algorithm using only ω-subdivisions. To this effect, we generated 3 instances of problem (CP) with the following objective function:

$$f(x) = -\left(\sum_{j=1}^{n} x_j^2\right) \log\left(1 + \sum_{j=1}^{n} x_j^2\right).$$

This objective function has been used in particular in [6, 3, 12]. The three instances correspond to $(m, n) = (15, 5)$, $(m, n) = (30, 6)$ and $(m, n) = (30, 7)$.

Table 5 shows the results that we obtained. The first number in the column #iter is the number of iterations, while the number in parenthesis has a different meaning depending on the algorithm: for the covering algorithm, it corresponds to the number of iterations for which the subdivision point was inside the simplex; for the ω-partitioning algorithm, it corresponds to the number of iterations for which the subdivision point was an extreme point of the polytope P. The columns $max|\mathcal{C}over|$ and $max|\mathcal{D}one|$ report the largest size of the sets $\mathcal{C}over$ and $\mathcal{D}one$ respectively (the ω-partitioning algorithm does not require the use of an explicit set $\mathcal{D}one$; the number in this column therefore corresponds to the maximum number of iterations with the same value of the lower bound $\underline{f}$). Finally $\#eval\_f$ reports the number of evaluations of the function f. It must be noted that the covering algorithm for instance $(m, n) = (30, 7)$ was still running, so we report bound on the corresponding values when available. The ω-partitioning algorithm returns ε-optimal solution with $\varepsilon = 10^{-6}$. To allow a fair comparison of the number of evaluations of the objective function, the list $\mathcal{V}ertex$ is also used in the ω-partitioning algorithm to ensure that the objective function is not evaluated more than once on a same vertex.

We observe that the number of iterations is much larger with the covering algorithm than with the ω-partitioning algorithm. This seems to be due to an increase of the maximum number of iterations with the

same lower bound, see column $\max|\mathcal{D}one|$. This in turn can be explained by the fact that the point $\omega(S)$ may still be feasible for some subsimplex, resulting in no change in their lower bound. This suggests to modify the subdivision process in order to guarantee that the point $\omega(S)$ is not feasible for any subsimplex of S. A possible way to do that is to allow simplices with more than $n+1$ extreme points (i.e., polytopes), in a similar way to what was proposed for cones in Jaumard and Meyer [15].
As for the number of evaluations of the objective function, our algorithm performs much better than the ω-partitioning algorithm as expected. For the second instance, the number of evaluations is divided by more than 10, while the number of iterations is multiplied by 4. Of course it is not possible to conclude on one instance, but if these ratios were similar for other instances, we could possibly accept the increase in the number of iterations if the cost to evaluate the objective function is high.
We did not report the computational time here, but it is clear that for the covering algorithm to be efficient in practice, we will have to implement very carefully the various lists needed by the algorithm, in order to allow efficient searches in these lists. For the second instance for example, more than 60% of the computational time is spent in operations on the lists, and it is believed that this number is much larger for the third instance.

6 Conclusion

We have presented a finite simplicial covering algorithm for the problem of minimizing a concave function over a polytope. Unlike most of the existing algorithms, simplices are all subdivided via an extreme point of the polytope, which allow finitness of the algorithm and reduction in the number of evaluations of the objective function. More work is needed to obtain a competitive algorithm, in particular the implementation of this new algorithm set new challenges to programmers to efficiently check that a simplex was not already generated.

References

[1] H. P. Benson. A finite algorithm for concave minimization over a polyhedron. *Naval Research Logistics Quarterly*, 32:165–177, 1985.

[2] H. P. Benson. Concave minimization: Theory, applications and algorithms. In R. Horst and P. M. Pardalos, editors, *Handbook of Global Optimization*. Kluwer Academic Publishers, 1996.

[3] H. P. Benson and S. Sayin. A finite concave minimization algorithm using branch and bound and neighbor generation. *Journal of Global Optimization*, 5(1):1–14, 1994.

[4] G. M. Guisewite and P. M. Pardalos. Minimum concave-cost network flow problems: applications, complexity, and algorithms. *Ann. Oper. Res.*, 25(1-4):75–99, 1990. Computational methods in global optimization.

[5] R. Horst. An algorithm for nonconvex programming problems. *Mathematical Programming*, 10:312–321, 1976.

[6] R. Horst, N. V. Thoai, and H. P. Benson. Concave minimization via conical partitions and polyhedral outer approximation. *Mathematical Programming*, 50:259–274, 1991.

[7] R. Horst, N. V. Thoai, and J. D. Vries. A new simplicial cover technique in constrained global optimization. *Journal of Global Optimization*, 2:1–19, 1992.

[8] R. Horst and H. Tuy. *Global Optimization (Deterministic Approaches)*. Springer-Verlag, Berlin, second edition, 1993.

[9] B. Kalantari and A. Bagchi. An algorithm for quadratic zero-one programs. *Naval Research Logistics*, 37(4):527–538, 1990.

[10] T. Kuno and H. Nagai. A simplicial branch-and-bound algorithm conscious of special structures in concave minimization problems. *Comput. Optim. Appl.*, 39(2):219–238, 2008.

[11] M. Locatelli and U. Raber. On convergence of the simplicial branch-and-bound algorithm based on ω-subdivisions. *Journal of Optimization Theory and Applications*, 107(1):69–79, 2000.

[12] M. Locatelli and N. V. Thoai. Finite exact branch-and-bound algorithms for concave minimization over polytopes. *Journal of Global Optimization*, 18:107–128, 2000.

[13] D. G. Luenberger. *Linear and Nonlinear Programming*. Addison-Wesley Publishing Company, second edition, 1989.

[14] C. Meyer. A simple finite cone covering algorithm for concave minimization. *Journal of Global Optimization*, 18(4):357–365, 2000.

[15] C. Meyer and B. Jaumard. A new finite cone covering algorithm for concave minimization. In *Combinatorial and global optimization (Chania, 1998)*, volume 14 of *Ser. Appl. Math.*, pages 237–249. World Sci. Publishing, River Edge, NJ, 2002.

[16] M. Nast. Subdivision of simplices relative to a cutting plane and finite concave minimization. *Journal of Global Optimization*, 9:65–93, 1996.

[17] R. T. Rockafellar. *Convex Analysis*. Princeton University Press, Princeton, New Jersey, 1970.