Proceedings of the
American Control Conference
Seattle, Washington • June 1995

Algorithms for Globally Solving D.C. Minimization Problems via Concave Programming *

Shih-Mim Liu and G. P. Papavassilopoulos
Dept of Electrical Engineering – Systems, University of Southern California
Los Angeles, CA 90089-2563
E-mail : *shihmiml@bode.usc.edu, yorgos@nyquist.usc.edu*

Abstract. Several methods have been proposed for solving a d.c. programming problem, but very few have been done on parallel approach. In this paper, three algorithms suitable for parallel implementation are presented to solve a d.c. problem via solving an equivalent concave minimization problem. To distribute the computation load as evenly as possible, a simplex subdivision process such as bisection, triangulation or other partition procedures of simplices (cf. [7]) will be employed. Some numerical test results are reported and comparison of these algorithms are given.

1. Introduction

Consider a d.c. (difference of two convex functions) global optimization problem

$$\begin{cases} \text{minimize } (f(x) - g(x)) \\ \text{subject to: } x \in D \end{cases} \qquad (DC)$$

where $D = \{x \in \mathbb{R}^n : h_i(x) \leq 0,\ (i = 1, \ldots, m)\}$, h_i, f, and g are finite convex functions on $\mathbb{R}^n$. Assume that D is compact and nonempty, then problem (DC) has a global solution.

In the literature, d.c. problems play an important role in nonconvex programming problems either from a practical or a theoretical viewpoint (cf. [7]). Indeed, d.c. problems are encountered frequently in Engineering and several methods had been proposed to solve the class of d.c. problems (cf. [7][6]). However, only a few approaches were issued in numerical test. Although some of them might be efficient, in particular, problem with a special structure such as separability of the objective function or a quadratic objective function (e.g. [13], [7] and references therein), very few have been done in parallel. Essentially, these proposed algorithms mainly use three different types of transformation of problem (DC), i.e. (1) the equivalent concave minimization problem, (2) an equivalent convex minimization problem subject to an additional reverse convex constraint, (3) canonical d.c. problem (cf. [7][3][12][8][14][1][6]).

The purpose of our paper is to propose three algorithms fitting for parallel implementation to solve the problem (DC) via solving an equivalent concave minimization problem. The first approach (Algorithm 1), which is similar to Hoffman algorithm [2], is an outer approximation method using cutting plane. Although finding the newly generated vertices is computationally expensive, the numerical experiments of the serial algorithm indicate that it is much more efficient than the others ([11],[6],[7],p525) for the tested problems given here. In addition, to investigate parallel behavior of Algorithm 1 we try a parallel simulation incorporating with the method described by [10] in solving the problem. The second algorithm (Algorithm 2), a simplicial procedure for solving the equivalent concave minimization problems, is much less efficient than the other two because of the inefficiency in detecting infeasible partition sets, the slow convergent rate of the bounds and the fast growth of the linear constraints. The efficiency, however, would be improved in its parallel implementation. The last method (Algorithm 3), originally proposed by Horst et al. [6], has similar advantages as Algorithm 2: only a sequence of linear subprograms have to be solved and both are appropriate for parallel computation with a suitable simplex partition procedure. Basically, during the parallel computation, in each iteration only the updated upper bound is required to communicate among processors for Algorithm 1, the communication should not be a problem. In both Algorithms 2 and 3, in every iteration during the parallel computation all the processors have to communicate with each other and share the following message they have obtained: the new upper bound, the new vertices created in the partition process, and the linear constraints added to define a new polyhedral set enclosing the feasible set. Since the amount of data to be passed is small, the communication overhead should not cause serious delay, compared to the time for solving a sequence of linear programs which are the main computation load of these two algorithms. Therefore, we use a sequential computer to simulate the parallel behavior of these three algorithms without considering the communication for the tested problems in Section 5.

The rest of this paper is organized as follows. The next section contains the basic idea of the methods. In Section 3 we discuss the fundamental implementation of these algorithms. Section 4 describes the details of these algorithms and their convergences are proved. Some numerical test problem are given in Section 5. Conclusion is in the final Section.

*supported in part by NSF under Grant CCR-9222734

2. Basic idea of the Methods

By introducing an additional variable t, problem (DC) can be rewritten as an equivalent global concave minimization problem which has the form

$$\begin{cases} \text{minimize } (t - g(x)) \\ \text{subject to: } f(x) \leq t \text{ and } x \in D \end{cases} \qquad (CP)$$

Let $\mathcal{D} = \{(x,t) \in \mathbb{R}^n \times \mathbb{R} : x \in D, f(x) \leq t\}$ be the feasible set of problem (CP). Given an n-simplex S with vertex set $V(S)$ containing the feasible set D, a prism (generated by S) $\mathcal{P} = \mathcal{P}(S) \subset \mathbb{R}^n \times \mathbb{R}$ is defined by

$$\mathcal{P} = \{(x,t) \in \mathbb{R}^n \times \mathbb{R} : x \in S,\ t_B \leq t \leq t_T\} \qquad (1)$$

where $t_B = \min\{f(x) : x \in D\}$ and $t_T = \max\{f(v) : v \in V(S)\}$. Note that the former is a convex minimization problem which can be done by any standard nonlinear programming method. Let $t_B = f(x_B)$ $(x_B \in D)$. The prism $\mathcal{P}$ has $n+1$ vertical lines (parallel to the $t-$axis) which pass through the $n+1$ vertices of S respectively.

2.1. An Outer Approximation Method Using Cutting Plane

Let a polyhedron $P_0 = \mathcal{P}$. Obviously, P_0 encloses $\mathcal{D}$ and $V(P_0)$ are known. Therefore, a lower bound of problem (CP) is determined by simply minimizing the functional values at all vertices of P_0. If the minimizing point is feasible to problem (CP), then that point will also be an upper bound, thus the problem (CP) is solved. Otherwise, the point violates at least one constraint of problem (CP). In this case, we should construct a hyperplane of support to some violated constraint which separates this minimizing point from P_0. In other words, this constructed hyperplane cuts through the previous polyhedron P_0 and creates a new polyhedron P_1 which will more tightly enclose the feasible set $\mathcal{D}$. All new vertices generated from cut are easily determined by the method in [5]. Denote the vertex set of P_1 by $V(P_1)$ and go to the next iteration. For its parallel computation procedure, we will follow the approach in [10].

2.2. A Parallel Simplicial Algorithm

Here, we introduce a simplicial algorithm to solve problem (CP) in parallel. In order to use simplicial algorithm, we make a prismatic triangulation of $\mathcal{P}$, i.e.

$$\mathcal{P} = \bigcup_{i=1}^{r} \mathcal{P}_i \qquad (2)$$

where r is an integer multiple of $(n+1)$, $\mathcal{P}_i$ is an n-simplex $(i = 1, \ldots, r)$ and each pair of simplices $\mathcal{P}_i$, $\mathcal{P}_j$ $(i \neq j)$ intersects at most

in common boundary points (i.e. $\mathcal{P}_i \bigcap \mathcal{P}_j = \partial\mathcal{P}_i \bigcap \partial\mathcal{P}_j$, where $\partial\mathcal{P}_i$ denotes the boundary of $\mathcal{P}_i$). Let $V(\mathcal{P}_j) = \{v_j^1, \ldots, v_j^{n+1}\}$ be the vertex set of $\mathcal{P}_j$ $(j = 1, \ldots, r)$. Obviously, we can find the solution of problem (CP) in terms of solving the following r subproblems

$$\begin{cases} \text{minimize } (t - g(x)) \\ \text{subject to: } (x,t) \in \mathcal{P}_j \bigcap \mathcal{D}, \ (j = 1, \ldots, r) \end{cases} \quad (SCP)$$

Let $\psi(x,t) = t - g(x)$, then the following linear programs can be applied to underestimate all above subproblems, i.e.

$$\begin{cases} \text{minimize} & \sum_{i=1}^{n+1} \lambda_i \psi(v_j^i) \\ \text{subject to:} & (x,t) = \sum_{i=1}^{n+1} \lambda_i v_j^i \in \mathcal{P}_j, \quad (j = 1, \ldots, r) \\ & \sum_{i=1}^{n+1} \lambda_i = 1, \ \lambda_i \geq 0, \ (i = 1, \ldots, n+1) \end{cases} \quad (SA)$$

The main idea of the parallel procedure for the simplicial algorithm is to make use of a suitable simplex subdivision technique, then solve the N linear programming subproblems as (SA) in each iteration for a case where N processors are used.

At iteration k, choose the first N_k (number of used processors) simplices from the remaining simplices (stored in increasing order of lower bounds) to perform the bounding computation and further subdivision. Let (x^k, t_k) be the best feasible point with an upper bound $(\mathcal{U}_k = t_k - g(x^k))$ obtained so far and $\mathcal{L}_k$ be a lower bound of the objective function in (CP). Obviously, if $\mathcal{U}_k - \mathcal{L}_k = 0$ then (x^k, t_k) is an optimal solution of (DC). Otherwise, delete the simplices according to the deletion laws in section 3.4 and go to the next iteration.

2.3. A Parallel Method via Linear Programs

An approach solving a D.C. programming problem by a sequence of linear programs was presented by Horst et. al [6]. This algorithm combines a new prismatic branch and bound technique with polyhedral outer approximation in such a way that only linear programming problems have to be solved. Essentially, this method can be parallelized by adopting the same simplex partition process and parallel procedure as stated in Section 2.2 because the prismatic branch and bound is created by both subdividing a prism formed by a simplex and solving the linear programming problems. Regardless of the communication, this parallel algorithm will be more efficient than that in [6]. The comparison of these two algorithms will be given in Section 5.

3. Implementation

There are four fundamental operations in the above algorithms: i) Construction of the initial simplex S, ii) Partitions of both prism and simplex, iii) Determination of bounds and cutting planes, iv) Deletion of simplices.

3.1. Construction of S

Although there are many ways (see [7, 10]) to determine a simplex $S \supset D$, we consider the following construction for Algorithms 1, 2, 3.

$$S = \{x \in \mathbb{R}^n : \alpha_i \leq x_i, \ (i = 1, \ldots, n), \ \sum_{i=1}^{n} x_i \leq \alpha\} \quad (3)$$

where $\alpha_i = \min\{x_i : x \in D\}$ $(i = 1, \ldots, n)$ and $\alpha = \max\{\sum_{i=1}^{n} x_i : x \in D\}$. Let x_i^* $(i = 1, 2, \ldots, n+1)$ be their solutions respectively. Clearly, S with a vertex set $V(S) = \{v^1, v^2, \ldots, v^{n+1}\}$ is a simplex tightly enclosing D, where $v^{n+1} = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ and $v^i = (\alpha_1, \alpha_2, \ldots, \alpha_{i-1}, \beta_i, \alpha_{i+1}, \ldots, \alpha_n)$, $(i = 1, \ldots, n)$ with $\beta_i = \alpha - \sum_{j \neq i}^{n} \alpha_j$.

3.2. Partitions of Both Prism and Simplex

As the algorithm in [6], an exhaustive subdivision process of simplex will be applied to ensure convergence of Algorithms 2, 3.

Definition 1 ([4][7][6]) *A subdivision is called exhaustive if* $\lim_{k\to\infty} \delta(S_k) \longrightarrow 0$ *($\delta(S_k)$ is the length of a longest edge of S_k) for all decreasing subsequence $\{S_k\}$ of partition elements generated by the subdivision.*

Here we also introduce an exhaustive partition process, Q-triangulation well known in operations research. Let an n-dimensional unit simplex be defined by

$$S^n = \{x \in \mathbb{R}_+^{n+1} : \sum_{i=1}^{n+1} x_i = 1\} \quad (4)$$

For $i \in I_{n+1} = \{1, \ldots, n+1\}$, $e(i)$ will denote the vector in $\mathbb{R}^{n+1}$ with i-th component equal to one and all other components equal to zero.

Let S be a simplex containing D in $\mathbb{R}^n$. Thus we can triangulate S by using the fact that the S is homeomorphic to the unit simplex S^n and that every point in S^n can be represented by its barycentric coordinates. In the triangulation of S^n, the Q-triangulation is probably the best known triangulation for algorithmic purposes (cf. [15] and reference therein). Moreover, Q-triangulation seems to be a proper partition process for parallel computation here since it is easy to split a simplex into M^2 (M^{-1} is the grid size) similar subsimplices. In other words, there are M^2 linear programming subproblems of same size subdivided from a linear programming problem. Therefore if there are a large number of processors at hand, one can choose a suitable grid size for Q-triangulation.

Definition 2 ([15]) *The Q-triangulation of S^n with grid size M^{-1} is the collection of all n-simplices $\sigma(v^1, \pi)$ with vertices $v^1, \ldots, v^{n+1}$ in S^n such that*

1) each component of v^1 is a nonnegative multiple of M^{-1}.

2) $\pi = (\pi_1, \ldots, \pi_n)$ is a permutation of the elements in $I_n = \{1, \ldots, n\}$.

3) $v^{i+1} = v^i + M^{-1} q^o(\pi_i)$, $(i = 1, \ldots, n.)$ where $q^o(j) = e(j+1) - e(j)$, $j = 1, \ldots, n$.

As the prismatic triangulation in [9], the prism $\mathcal{P} = S \times T$ can be triangulated via triangulation of S and T $(t_B \leq T \leq t_T)$. Let σ_n be an n-simplex in the collection $C_n(S)$ of all n-simplices of the decomposition of S and T be a 1-simplex of the decomposition of the t-axis, i.e.

$$\sigma_n = v^0 \ldots v^n \in C_n(S) \text{ and } (t_B, t_T) = T \in C_1(t - axis)$$

The prismatic triangulation of the Cartesian is obtained by

$$\sigma_n \times T = \sum_{k=0}^{n} (-1)^k v_{t_B}^0 \ldots v_{t_B}^k v_{t_T}^k \ldots v_{t_T}^n \quad (5)$$

where $v_{t_B}^k$ is the vertex v^k of S elevated at height t_B. A Q-triangulation of S and a prismatic triangulation of $\mathcal{P}$ are illustrated in Figure 1 for n=2. For a sequential algorithm, an exhaustive

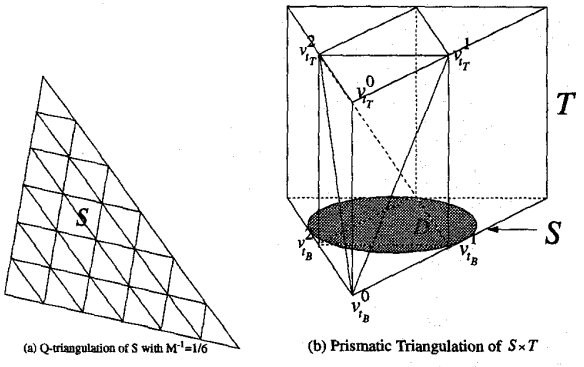

Figure 1: Q-triangulation of S and Prismatic triangulation of $\mathcal{P}$

partition process, bisection of simplex seems to be useful. Other partition procedures of simplices can be found in [7] and references therein.

Lemma 1 *Let $\{S_k\}$ be any decreasing sequence of n-simplices generated by Q-triangulation process, then* $\lim_{k\to\infty} \delta(S_k) \longrightarrow 0$, *i.e. Q-triangulation procedure is exhaustive.*

Proof: Obviously $\frac{1}{M}\delta(S_k) = \delta(S_{k+1})$, i.e. $\delta(S_{k+1}) = (\frac{1}{M})^{k+1}\delta(S_0)$ $(M \geq 2)$. Therefore, $\lim_{k\to\infty} \delta(S_k) \longrightarrow 0$. □

3.3. Computation of Bounds and Construction of Cutting Planes

Let S be an n-simplex constructed in section 3.1. Denote an initial upper bound for problem (SCP) or (CP) and the prism $\mathcal{P}$ by $\mathcal{U}_0$ and a polyhedron P_0 respectively, where $\mathcal{U}_0 = min\{f(x_i^*) - g(x_i^*), \ i = 1, \ldots, n+1\}$. Set P_k be a polyhedron enclosing $\mathcal{D}$ at iteration k and $V(P_k)$ be its vertex set.

In Algorithm 1, a point u_0 in the strict interior of $\mathcal{D}$ must be found first. At iteration k, choose $\psi(v_k) = min\ \{\psi(v),\ v \in V(P_k)\}$ as a lower bound, then solve the line searching problem $z_k = v_k + \gamma(u_0 - v_k)$ such that $z_k \in \partial\mathcal{D}$ (boundary of $\mathcal{D}$). If $\gamma = 0$ then $v_k \in \mathcal{D}$ and v_k is a solution of the problem (CP). Otherwise, update the upper bound $\mathcal{U}_{k+1} = \min\{\mathcal{U}_k, \psi(z_k)\}$ and find a constraint $\mathcal{H}(x,t) \in \{h_i(x)(i = 1,\ldots,m), f(x) - t\}$ which is binding at z_k and the linear constraint corresponding to $\mathcal{H}(x,t)$ and z_k. Set

$$P_{k+1} = P_k \bigcap \{(x,t) \in \mathbb{R}^n \times \mathbb{R} :\ C_k(x,t) \leq 0\} \tag{6}$$

where

$$C_k(x,t) = \nabla\mathcal{H}(z_k)^T[(x,t) - z_k] \ \leq 0. \tag{7}$$

Compute $V(P_{k+1})$ based on $V(P_k)$, then go to the next iteration. For the determination of both bound and cutting plane in the parallel procedure, the method in [10] will be applied.

For the simplicial algorithm, let P_0 be a polyhedron containing $\mathcal{D}$ which is defined by

$$P_0 = \{\tilde{x} \in \mathbb{R}^n \times \mathbb{R} : A\tilde{x} \leq b,\ \tilde{x} = (x,t)\}, \tag{8}$$

where A is a real $m \times (n+1)$ matrix and $b \in \mathbb{R}^m$. Construct a sequence of polyhedral convex sets $P_0, P_1, P_2, \ldots$ such that $P_0 \supset P_1 \supset \ldots \supset \mathcal{D}$. The transition from P_k to P_{k+1} $(k = 0,1,\ldots)$ is done by adding some appropriate cutting planes $C_k^j(x,t) \leq 0$ $(j = 1,\ldots,q)$ $(q \leq N_k$ (number of processors at iteration $k))$ to the constraint set which defines P_k, i.e.,

$$P_{k+1} = P_k \bigcap \{(x,t) \in \mathbb{R}^n \times \mathbb{R} :\ C_k^j(x,t) \leq 0,\ j = 1,\ldots,q\} \tag{9}$$

At the iteration k, let upper bound $\mathcal{U}_k$ and lower bound $\mathcal{L}_k$ for P_k. Denote the number of the remaining collection of simplices $\mathcal{R}_k$ by $|\mathcal{R}_k|$. Assume that there is a simplex S_k^l with lower bound $\mathcal{L}(S_k^l)$ $(l = 1,\ldots,|\mathcal{R}_k|)$ in $\mathcal{R}_k$ and its vertex set $V(S_k^l) = \{v_l^i, i = 1,\ldots,n+1\}$. Since every point $\tilde{x} \in S_k^l$ is uniquely represented as

$$\tilde{x} = \sum_{i=1}^{n+1} \lambda_i v^i = V_l\lambda,\ \sum_{i=1}^{n+1} \lambda_i = 1,\ \lambda_i \geq 0\ \ (i = 1,\ldots,n+1), \tag{10}$$

where V_l denotes the matrix with columns $v_l^1,\ldots,v_l^{n+1}$ and λ is a vector with components $\lambda_1,\ldots,\lambda_{n+1}$. Every subproblem

$$\text{minimize } (t - g(x)) \tag{11}$$

$$\text{subject to: } (x,t) \in S_k^l \bigcap \mathcal{D},\ (l = 1,\ldots,|\mathcal{R}_k|) \tag{12}$$

can be underestimated by the following linear program with the constraint set $S_k^l \bigcap P_k$.

$$\min \sum_{i=1}^{n+1} \lambda_i \psi(v_l^i) \tag{13}$$

$$\text{s.t.: } AV_l\lambda \leq b,\ \sum_{i=1}^{n+1} \lambda_i = 1,\ \lambda_i \geq 0\ (i = 1,\ldots,n+1) \tag{14}$$

Since $\lambda_{n+1} = 1 - \sum_{i=1}^{n} \lambda_i \geq 0$, (13) and (14) become

$$\min \sum_{i=1}^{n} \lambda_i[\psi(v_l^i) - \psi(v_l^{n+1})] + \psi(v_l^{n+1}) \tag{15}$$

$$\text{s.t.: } A\tilde{V}_l\tilde{\lambda} \leq \tilde{b},\ \ \sum_{i=1}^{n} \lambda_i \leq 1,\ \lambda_i \geq 0\ \ (i = 1,\ldots,n) \tag{16}$$

where $\tilde{V}_l$ denotes the matrix with columns $(v_l^i - v_l^{n+1})$ $(i = 1,\ldots,n)$, $\tilde{\lambda}$ is a vector with components $\lambda_1,\ldots,\lambda_n$, and $\tilde{b} = b - Av^{n+1}$.

Let $\tilde{\lambda}^*$ be an optimal solution of the linear program (15), (16), thus we have $\tilde{x}^* = (x_k^l, t_k^l)$ by (10). If $(x_k^l, t_k^l) \in \mathcal{D}$, then an upper bound of S_k^l, $\mathcal{U}(S_k^l) = t_k^l - g(x_k^l)$ is obtained. Otherwise, we have a new lower bound, $\max\{\mathcal{L}(S_k^l), \sum_{i=1}^{n+1} \lambda_i^* \psi(v_l^i)\}$. Let $\mathcal{H}(x,t) := \max\{h_i(i = 1,\ldots,m), f(x) - t\}$ and add a constraint

$$C_k^l(x,t) = \nabla\mathcal{H}(x_k^l, t_k^l)^T[(x,t) - (x_k^l, t_k^l)] + \mathcal{H}(x_k^l, t_k^l) \ \leq 0 \tag{17}$$

Remark 1: Let u_0 be a strictly interior point in $\mathcal{D}$, and find a point z_k^l between $[u_0, (x_k^l, t_k^l)]$ such that $z_k \in \mathcal{D}$. If we replace the constraint (17) by

$$C_k^l(x,t) = \nabla\mathcal{H}(z_k^l)^T[(x,t) - z_k^l] \ \leq 0 \tag{18}$$

Then this cut is better because it is closer to $\mathcal{D}$. Note that a variety of other cuts can also be employed (cf. Horst and Tuy [7]).

Let U with vertices $v_u^1,\ldots,v_u^{n+1}$ be a subsimplex generated by a exhaustive subdivision process from an initial n-simplex $S \supset D$. Hence the lower bound in Algorithm 3 is primarily calculated by the following linear program in (λ, t) (cf. Horst et al. [6]):

$$\text{maximize } (\sum_{i=1}^{n+1} t_i\lambda_i - t) \tag{19}$$

$$\text{subject to: } AV_u\lambda + at \leq b \tag{20}$$

$$\sum_{i=1}^{n+1} \lambda_i = 1,\ \lambda_i \geq 0\ \ (i = 1,\ldots,n+1) \tag{21}$$

where V_u denotes the matrix with columns $v_u^1,\ldots,v_u^{n+1}$, $a \in \mathbb{R}^m$ and A, b are given in (8). If c^* is the optimal objective function value in (19), (20), (21), then the lower bound of U is provided by

$$\mathcal{L}(U) = \begin{cases} +\infty, & \text{if (19),(20),(21) have no feasible point} \\ \mathcal{L}(U), & \text{if } c^* \leq 0 \\ \mathcal{L}(U) - c^*, & \text{if } c^* > 0 \end{cases} \tag{22}$$

With regard to the upper bound of U, it can be obtained from both evaluating new vertex in the corresponding partition of simplex and solving the linear programming problem (19), (20), (21).

Besides the computation of bounds, the construction of the cutting planes is the same as Algorithm 2. For parallel process the determinations of both bound and cutting plane will also follow the way as described in simplicial algorithm.

3.4. Deletion of Simplices

At iteration k of the simplicial algorithm we try to delete the simplices that do not contain any feasible solution better than (x^k, t_k). Here, we have the following deletion laws:

(a) Delete any simplex S_k^l $(l = 1,\ldots,|\mathcal{R}_k|)$ if all of its vertices locate outside the current polyhedral set P_k or (13), (14) has no feasible solution. In this case $S_k^l \bigcap \mathcal{D} = \emptyset$

(b) Delete simplex S_k^l if $\mathcal{L}(S_k^l) \geq \mathcal{U}_k$.

(c) Delete simplex S_k^l if its optimal objective function value obtained from (13), (14) is greater than upper bound $\mathcal{U}_k$.

4. The Algorithms

In this Section, we describe these algorithms in detail for solving problem (DC) in terms of solving problem (CP) by cutting plane method or a sequence of linear programs. Assume that there are N processors throughout the following algorithms.

Algorithm 1

Construct a prism $\mathcal{P} \supset \mathcal{D}$ associated with a simplex $S \supset D$ and let a polyhedron $P_0 = \mathcal{P}$ as described in Section 3.3. Obviously, the vertex set $V(P_0)$ of P_0 is known and consists of the $2(n+1)$ points. Let u_0 be a strictly interior point and $\epsilon > 0$. Set

$$\text{upper bound: } \mathcal{U}_0 = \min\{0, (f(x_i^*) - g(x_i^*)), i = 1,\ldots,n+1\}, \tag{23}$$

$$\text{lower bound: } \mathcal{L}_0 = \min\{\psi(v) :\ v \in V(P_0)\}, \tag{24}$$

Iteration $(k = 0,1,2,\ldots)$

Let $(x^k, t_k) = v_k$ satisfy $\mathcal{L}_k = \psi(v_k)$, and find a γ such that $z_k = v_k + \gamma(u_0 - v_k) \in \partial\mathcal{D}$. If $\gamma = 0$, then v_k is an optimal solution of (CP). Otherwise, set $\mathcal{U}_{k+1} = \min\{\mathcal{U}_k, \psi(z_k)\}$, add a constraint according to (7) and form (6). Compute $V(P_{k+1})$ (only the vertices having objective function value lower than the current upper bound need to be stored, let V^{k+1} denote these vertices), then let $\mathcal{L}_{k+1} = \min\{\psi(v) : v \in V^{k+1}\}$ and (x^{k+1}, t_{k+1}) be the point satisfying $\mathcal{L}_{k+1} = \psi(x^{k+1}, t_{k+1})$. If $\mathcal{U}_{k+1} - \mathcal{L}_{k+1} \leq \epsilon$, then stop and z_k is an optimal solution of problem (CP). Otherwise, go to the next iteration.

Remark 2:

(1) If let t_T be large enough such that $\psi(x, t_T) > \mathcal{U}_0$, where $\forall x \in V(S)$, then at most $n+1$ vertices (x, t_B) need to be stored for the next iteration. Therefore, by adopting the method as described in [10] we have a parallel algorithm using $n+1$ processors. One

numerical test will be given in Section 5.
(2) At iteration k, we also can delete the vertices yielding an objective function value great than $\mathcal{U}_k - \epsilon$. In this case, if $V^{k+1} = \emptyset$, then algorithm terminates: z_k is an optimal solution.

Lemma 2 ([7]) *Let L_k, $k = 1, 2, \ldots$ be a sequence of arbitrary set in $\mathbb{R}^n$. If $\{x^k\}$ is a bounded sequence of points satisfying*

$$x^k \in \bigcap\nolimits_{h<k} L_h$$

then $d(x^k, L_k) \longrightarrow 0$ $(k \longrightarrow \infty)$, where d is the distance function in $\mathbb{R}^n$.

Theorem 1 (cf. Hoffman [2]) *In the Algorithm 1 every accumulation point of the sequence $\{(x^k, t_k)\}$ is a global optimal solution of problem (CP).*

Proof: The algorithm 1 generates a sequence of points $\{(x^k, t_k)\}$ and a corresponding sequence of lower bounds $\{\psi(x^k, t_k)\}$ which is monotonically nondecreasing (since $S^k \supset S^{k+1}$) and bounded from above by the value of $\psi(x^*, t^*)$, where (x^*, t^*) is any feasible solution in $\mathcal{D}$. Let the half-space $L_k = \{(x,t) \in \mathbb{R}^n \times \mathbb{R} : C_k(x,t) \leq 0\}$, and the hyperplane $H_k = \{(x,t) \in \mathbb{R}^n \times \mathbb{R} : C_k(x,t) = 0\}$, then we have

$$C_k(x^k, t_k) > 0, \quad \text{while} \quad C_h(x^k, t_k) \leq 0 \ \ (h = 0, 1, \ldots, k-1).$$

Now the sequence $\{(x^k, t_k)\}$ is bounded (since the enclosing polyhedron P_k is bounded), by Lemma 2, we have $d((x^k, t_k), L_k) \to 0$ $(k \to \infty)$. Hence $d((x^k, t_k), H_k) \to 0$, where $H_k = \{(x,t) \in \mathbb{R}^n \times \mathbb{R} : \nabla\mathcal{H}(z_k)^T[(x,t) - z_k] = 0\}$, i.e.

$$d((x^k, t_k), H_k) \quad = \quad \frac{|\nabla\mathcal{H}(z_k)^T[(x^k, t_k) - z_k]|}{(1 + \|\nabla\mathcal{H}(z_k)\|^2)^{1/2}} \longrightarrow 0 \quad (25)$$

Let $(\bar{x}, \bar{t})$ be an accumulation point for the sequence $\{(x^k, t_k)\}$ and a subsequence $\{(x^{k_i}, t_{k_i})\}$ of $\{(x^k, t_k)\}$ converge to $(\bar{x}, \bar{t})$. By Lemma 2,3 in [2], we have two sequences $\{\lambda_k\}$, $\{z_k\}$ and their accumulation points $\bar{\lambda}$, $\bar{z}$ such that $z_k = (x^k, t_k) + \lambda_k[u_0 - (x^k, t_k)]$ and $\bar{z} = (\bar{x}, \bar{t}) + \bar{\lambda}[u_0 - (\bar{x}, \bar{t})]$. Let subsequences $\{\lambda_{k_i}\}$, $\{z_{k_i}\}$ of the sequences $\{\lambda_k\}$, $\{z_k\}$ converge to $\bar{\lambda}$, $\bar{z}$ respectively, then we also have $z_{k_i} = (x^{k_i}, t_{k_i}) + \lambda_{k_i}[u_0 - (x^{k_i}, t_{k_i})]$ From (25) we obtain

$$|\mathcal{H}(z_{k_i})^T[(x^{k_i}, t_{k_i}) - z_{k_i}]| \longrightarrow 0 \quad \Longrightarrow \quad |\mathcal{H}(\bar{z})^T[(\bar{x}, \bar{t}) - \bar{z}]| = 0$$
$$\Longrightarrow |\mathcal{H}(\bar{z})^T\{\bar{\lambda}[u_0 - (\bar{x}, \bar{t})]\}| = 0$$

Since u_0 is a strictly interior point, only $\bar{\lambda} = 0$ can satisfy, i.e. $(\bar{x}, \bar{t}) = \bar{z} \in \mathcal{D}$. ∎

Algorithm 2

Initialization (Given an $\epsilon > 0$)
Construct a prism $\mathcal{P}$ associated with a simplex $S \supset D$, and let a polyhedral convex set $P_0 = \mathcal{P} \supset \mathcal{D}$ as described in Section 3.3. According to the process stated in Section 3.2, partition $\mathcal{P}$ into r simplices S_0^l, $(l = 1, \ldots, r)$ and denote the vertex set of S_0^l by $V(S_0^l)$. Let

$$\mathcal{U}_0' = min\{0, (f(x_i^*) - \mathrm{g}(x_i^*)),\ i = 1, \ldots, n+1\} \quad (26)$$

$$\mathcal{U}_0 = \min\{\mathcal{U}_0',\ \min\{\psi(v) : v \in (\bigcup_{l=1}^{r} V(S_0^l)) \bigcap \mathcal{D}\}\} \quad (27)$$

(cf. section 3.3). Set $(x^0, t_0) \in \mathcal{D}$ such that $t_0 - \mathrm{g}(x^0) = \mathcal{U}_0$ and let $\mathcal{L}_0 = -\infty$. Delete all simplices S_k^l $(l = 1, \ldots, r)$ which satisfy the deletion law (a).
Iteration k $(k = 0, 1, 2, \ldots)$
At the begining of iteration k we have a polyhedron $P_k \supset \mathcal{D}$, the best feasible point (x^k, t_k) obtained so far, and an associated upper bound $\mathcal{U}_k = t_k - \mathrm{g}(x^k)$. Furthermore, we have a set $\mathcal{R}_k$ of simplices generated from the initial partition of $\mathcal{P}$ by deletion and subdivision according to the rules described in Section 3.2, 3.4, and their lower bounds $\mathcal{L}(S_k^l)$, $l = 1, \ldots, |\mathcal{R}_k|$.

Step 1: Choose the first N_k simplices in $\mathcal{R}_k$ (if $|\mathcal{R}_k| > N$, then $N_k = N$). Solve these linear programs (15), (16) corresponding to both S_k^j $(j \in I_k = \{1, \ldots, N_k\})$ and P_k in parallel. Hence, let (x_k^j, t_k^j) $(j =\in I_k)$ be their optimal solutions. If any simplex S_k^j $(j \in I_k)$ which satisfies deletion laws (a), (b), or (c), delete it. Set $I_k = I_k/\{j\}$.

Step 2: For $\forall j \in I_k$, if $(x_k^j, t_k^j) \in \mathcal{D}$ $(j \in I_k)$, then set $\mathcal{U}(S_k^j) = \min\{\mathcal{U}_k, t_k^j - \mathrm{g}(x_k^j)\}$ and $I_k = I_k/\{j\}$. Otherwise, let $\mathcal{U}(S_k^j) = \mathcal{U}_k$ and add a constraint $C_k^j(x,t) \leq 0$ according to (17). Set $P_{k+1} = \{(x,t) \in P_k : C_k^j(x,t) \leq 0,\ j \in I_k\}$

Step 3: Subdivide the simplices S_k^j, $j \in I_k$ into a finite number of subsimplices and delete the subsimplices satisfying deletion law (a). Let $S_k^{j_i}$ $(j_i \in J_k$, where J_k is the index set of these remaining subsimplices) be the remaining subsimplices. Let $\mathcal{V}_k^j$ be the new feasible vertices of the corresponding partition of the simplices S_k^j, $(j \in I_k)$. Set $\mathcal{U}(S_k^j) = \min\{\mathcal{U}_k, \min\{\psi(v) : v \in \mathcal{V}_k^j\}\}$ if $\mathcal{V}_k^j \neq \emptyset$

Step 4: For $j_i \in J_k$, solve the linear program (15), (16) corresponding to both P_{k+1} and $S_k^{j_i}$. Delete any subsimplex $S_k^{j_i}$ by deletion laws (a), (b), or (c). Let $\mathcal{R}_k'$ denote the collection of remaining subsimplices $S_k^{j_i}$ $(j_i \in J_k)$. For each $S_k^{j_i} \in \mathcal{R}_k'$, Set $\mathcal{L}(S_k^{j_i}) = \max\{\mathcal{L}(S_k^j), \mathcal{L}(S_k^{j_i})\}$

Step 5: Let $\mathcal{F}_k$ denote the set of new feasible points obtained from solving the linear programs in step 4, and set $\mathcal{U}_{k+1} = \min\{\min\{\mathcal{U}(S_k^j),\ j = 1, \ldots, N_k\},\ \min\{\psi(v) : v \in \mathcal{F}_k\}\}$ Let $(x^{k+1}, t_{k+1}) \in \mathcal{D}$ such that $t_{k+1} - \mathrm{g}(x^{k+1}) = \mathcal{U}_{k+1}$. Set $\mathcal{R}_{k+1} = (\mathcal{R}_k \backslash \{S_k^j : j = 1, \ldots, N_k\}) \bigcup \mathcal{R}_k'$ If $|\mathcal{R}_{k+1}| = 0$, then stop (x^{k+1}, t_{k+1}) is an optimal solution of problem (CP). Otherwise, set $\mathcal{L}_{k+1} = \min\{\mathcal{L}(S_{k+1}^l) : l = 1, \ldots, |\mathcal{R}_{k+1}|\}$

Step 6: If $\mathcal{U}_{k+1} - \mathcal{L}_{k+1} \leq \epsilon$, then stop: (x^{k+1}, t_{k+1}) is an optimal solution of problem (CP). Otherwise, go to the next iteration.

Remark 3: I_k is an index set of active processors at iteration k. Note that not all processor are active throughout an iteration, some of them might be idle after some step.

Theorem 2 *Any accumulation point of the sequence (x^k, t_k) generated by Algorithm 2 is a global optimal solution.*

Proof: From Lemma 1, we know that any infinite nested sequence of simplices S_r, $r \in Q \subset \{0, 1, 2 \ldots\}$, obtained from Algorithm 2 by means of a Q-triangulation subdivision is exhaustive. Then the proof can be seen in [7]. ∎

Algorithm 3

This algorithm is to execute the approach proposed by [6] in parallel. Since a prism in [6] is always generated by an n-simplex, the prismatic partition in [6] is similar to the simplex subdivision in Algorithm 2. By employing both parallel procedure and subdivision process as indicated in Algorithm 2, we have a parallel process for the algorithm in [6].

5. Numerical Tests

Problem 1: We applied the Algorithm 1 (serial process) to the following problem which was solved in Muu and Oettli [11] by a branch-and-bound method.

$$\text{minimize } (4x_1^2) - (0.1x_1^4 - x_2^{1/2}) \quad (28)$$
$$\text{subject to: } 0 \leq x_1 \leq 1, \quad 0 \leq x_2 \leq 2, \quad x_1 + x_2 \geq 1, \quad (29)$$

Set $f(x) = 4x_1^2$, $\mathrm{g}(x) = 0.1x_1^4 - x_2^{1/2}$, then (28), (29) can be rewritten as a (CP) problem

$$\text{minimize } \{t - \mathrm{g}(x) : x \in D, \quad f(x) - t \leq 0\} \quad (30)$$

where $D = \{x \in \mathbb{R}^2 : 0 \leq x_1 \leq 1,\ 0 \leq x_2 \leq 2,\ x_1 + x_2 \geq 1\}$

Choose the convergence criterion $\epsilon = 0.01$ and construct a simplex $S = \{x \in \mathbb{R}^2 : x_1 \geq 0,\ x_2 \geq 0,\ x_1 + x_2 \leq 3\}$ and its corresponding prism $\mathcal{P}(= P_0)$ with vertices $(3, 0, t_B)$, $(3, 0, t_T)$, $(0, 3, t_B)$, $(0, 3, t_T)$, $(0, 0, t_B)$, $(0, 0, t_T)$, where $t_B = 0$ and $t_T = 10000000$. Thus, these vertices have objective function values of -36, 9999964, 0, t_T, 0, t_T respectively. From (23), we obtain $\mathcal{U}_0 = 1$ and $(x^0, t_0) = (0, 1, 0)$.
Iteration 0: choose $v_0^* = (3, 0, 0)$, then find a $\gamma = 0.9168$ such that $z_0 = (0.4788, 0.9168, 0.9168)$ and $\psi(z_0) = 1.8691$. Set $\mathcal{U}_1 = \mathcal{U}_0 = 1$, and form $P_1 = P_0 \bigcap \{(x,t) : C_0(x,t) = 3.83x_1 - t - 0.9168 \leq 0\}$.

With $\mathcal{U}_k - \mathcal{L}_k \leq \epsilon$, the algorithm terminates after 5 iterations at an approximate optimal solution $(x^*, t_*) =$

(0.0683,0.9364,0.0186) with the objective function value 0.9863. Throughout the computation, 14 vertices are generated by cuts and maximum number of vertices stored in memory is only 2. But, the algorithm in [11] showed $x_k = (0.125, 0.875)$ with objective function value 0.9979 after 8 iterations.

Problem 2 [6]: In this problem, we use Algorithms 1,2,3 to solve it and let $N_k = N$ in parallel algorithms.

$$\begin{array}{ll} \text{minimize} & (4x_1^4 + 2x_2^2) - (4x_1^2) \\ \text{subject to:} & x_1^2 - 2x_1 - 2x_2 - 1 \le 0, \\ & -1 \le x_1 \le 1, \qquad -1 \le x_2 \le 1, \end{array}$$

Let $f(x) = 4x_1^4 + 2x_2^2$, $g(x) = 4x_1^2$, then we have (CP) problem

$$\text{minimize } \{t - g(x) : (x,t) \in \mathcal{D}\}$$

where $\mathcal{D} = \{(x,t) \in \mathbb{R}^3 : f(x) - t \le 0, x_1^2 - 2x_1 - 2x_2 - 1 \le 0, -1 \le x_1 \le 1, -1 \le x_2 \le 1\}$ and $t_B = 0$. Let $\epsilon = 0.05$ and construct an initial simplex $S = \{x \in \mathbb{R}^2 : x_1 \ge -1, x_2 \ge -1, x_1 + x_2 \le 2\}$ with vertices (3,−1), (−1,3), (−1,−1). We obtain $\mathcal{U}_0 = 0$, $(x^0, t_0) = (0,0)$ according to (23), (26).

Algorithm 1: Choose u_0 =(0,0,0.5) and t_T =10000000. We obtain v_0^* =(3,−1,0) by evaluating $t - g(x)$ at (3,−1,0), (−1,3,0), (−1,−1,0). After a line searching, we find $\gamma = 0.8194$, z_0 =(0.5417,−0.1806,0.4097) and $\psi(z_0) = -0.7642$. Thus, $\mathcal{U}_1 = -0.7642$. Form a new polyhedron

$$P_1 = P_0 \bigcap \{(x,t) \in \mathbb{R}^3 : 2.5437x_1 - 0.7223x_2 - t - 1.0987 \le 0\}$$

Continuing with the same procedure, the algorithm will stop at an approximate optimal solution (x^*, t_*) =(0.7171,−0.0103,1.0580) with objective function value −0.9990 after 16 iterations. There are 90 generated vertices during the computation.

If we employ the parallel method described in [10], then the algorithm only needs 10 iterations to obtain the same accuracy.

Algorithm 2: (Let $N_k = N = 1$, i.e. sequential algorithm) Partition the prism ($t_T = 36$) into 3 simplices according to (5), then subdivide them into 6 subsimplices by bisection.

Iteration 0: we have $\mathcal{U}_0 = 0$, $(x^0, t_0) = (0,0,0)$, $P_1 = P_0 \bigcap \{(x,t) : C_0^j(x,t) \le 0,\ j = 1,2,3\}$, where $C_0^1(x,t) = 16x_1 + 4x_2 - t - 14$, $C_0^2(x,t) = -16x_1 - 4x_2 - t - 14$, $C_0^3(x,t) = 16x_1 - 4x_2 - t - 14$. and $|\mathcal{R}_1|$=8 with lower bounds -15.2, -4, -4, -4, -1.84, -1.84, -1.1579, -1.1579. Obviously, $\mathcal{L}_1 = -15.2$.

Iteration 1: Solving the linear program (15), (16), we obtain $\lambda = (0.35, 0.15, 0.5, 0)$. Thus (x^1, t_1) =(0.7,−0.7,0). Since $(x^1, t_1) \notin \mathcal{D}$, form $P_2 = P_1 \bigcap \{(x,t) : C_1(x,t) = 5.4880x_1 - 2.800x_2 - t - 3.8612 \le 0\}$. At the end of this iteration, $|\mathcal{R}_2| = 8$, $\mathcal{L}_2 = -11.4541$, $\mathcal{U}_2 = 0$. After 114 iterations, the algorithm will stop at an approximate optimal solution.

Table 1 presents some computational results of Algorithms 2,3 using different number of processors (results of Algorithm 2 by using equation (18) are shown in parentheses), where $N_k = N$ and bisection is applied in subdivision process of simplices.

Table 1: Results for Algorithms 2, 3 with N Processors

	Algorithm 2			Algorithm 3		
N	Iter	Cont_no	Max	Iter	Cont_no	Max
1	114(98)	109(91)	32(21)	34	29	24
2	63(51)	114(91)	33(20)	22	40	35
3	43(34)	112(92)	35(21)	14	36	28
4	26(26)	92(93)	28(23)	10	33	27
5	22(19)	98(87)	28(22)	9	36	29
10	14(13)	104(104)	13(13)	8	55	18

Iter:	number of iterations
N:	number of processors
Cont_no:	number of added constraints
Max:	maximal number of simplices ($\mathcal{R}_k$) stored

Problem 3 [6]: Here, we apply Algorithm 1 (use serial procedure only) to solve the problem

$$\begin{array}{ll} \text{minimize} & (x_1^4 + x_2 + x_3) - (x_1 + x_2^2 - x_3) \\ \text{subject to:} & (x_1 - x_2 - 1.2)^2 + x_2 \le 4.4, \\ & x_1 + x_2 + x_3 \le 6.5, \qquad x_1 \ge 1.4, \\ & x_2 \ge 1.6, \qquad x_3 \ge 1.8, \end{array}$$

which was solved by Thoai ([7],p525) and Horst et al. [6] with an $\epsilon = 0.01$. The former needs 81 iterations to obtain an approximate solution and the latter terminates after 18 iterations. But Algorithm 1 proposed in this paper only requires 4 iterations to achieve the approximate solution of the same precision.

First, transform the original (DC) problem into a (CP) problem, where $f(x) = x_1^4 + x_2 + x_3$, $g(x) = x_1 + x_2^2 - x_3$. With an interior point $u_0 = (1.4, 1.6, 1.8, 8.0)$ and the same ϵ , Algorithm 1 has an approximate optimal solution (x^*, t_*) =(1.4000,1.8095,1.8000,7.4511) at the end of the 4-*th* iteration.

6. Conclusion

From our numerical results, the serial version of Algorithm 1 seems to be very efficient, at least for tested problems, compared with other methods ([11],[6], [7],p525). The same characteristics of the Algorithms 1, 2, 3 are that they have infinite convergence (although an approximate solution is achieved by a tolerance criterion), and that the linear constraints grow in size as the number of iterations increases. Algorithm 2 is much less efficient than the other two since not only it needs more iterations to solve the problems but also more linear constraints are generated in computation. Compared to the large amount of time saved due to the less iterations required in parallel processing for Algorithms 1, 2, 3, the communication overhead is unlike to have a significant effect. In conclusion, the Algorithms presented here can be expected to produce promising results in parallel processing.

References

[1] T. Pham Dinh and S. El Bernoussi, "Numerical Methods for Solving a Class of Global Nonconvex Optimization Problems," New Methods in Optimization and Their Industrial Uses, Ed. J.P. Penot, Birkhäuser Verlag, pp. 97-132, 1989.

[2] K.L. Hoffman, "A Method for Globally Minimizing Concave Functions over Convex Sets," Mathematical Programming 20, pp. 22-32, 1981.

[3] J.E. Falk and K.L. Hoffman, "A Successive Underestimation Method for Concave Minimization Problems," Mathematics of Operations Research 1, pp. 251-259, 1975.

[4] R. Horst, "An Algorithm for Nonconvex Programming Problems," Mathematical Programming 10, pp.312-321, 1976.

[5] R. Horst, N.V. Thoai and J. de Vries, "On Finding New Vertices and Redundant Constraints in Cutting Plane Algorithms for Global Optimization ," Operations Research Letters 7, pp. 85-90, 1988.

[6] R. Horst, T.Q. Phong, Ng. V. Thoai and J. de Vries, "On Solving a D.C. Programming Problem by a Sequence of Linear Programs," Journal of Global Optimization 1, pp. 183-203, 1991.

[7] R. Horst and H. Tuy, Global Optimization, 2nd Edition, Springer-Verlag, Berlin, 1993.

[8] R. Horst and N.V. Thoai, "Modification, Implementation and comparison of Three Algorithms for Globally Solving Linearly Constrained Concave Minimization Problems," Computing, 42, pp. 271-289, 1989.

[9] E.A. Jonckheere, C.Y. Cheng and C.K. Chu, " Robust Stability, Hex Game and Sphere Packing," submitted for publication, 1994.

[10] Shih-Mim Liu and G. P. Papavassilopoulos, "A Parallel Method for Globally Minimizing Concave Functions Over a Convex Polyhedron," in Proceedings of the 2nd IEEE Mediterranean Symposium on New Directions in Control & Automation, 1994.

[11] L. D. Muu and W. Oettli, "Method for Minimizing a Convex-Concave Function over a Convex Set," Journal of Optimization Theory and Applications, Vol. 70, pp. 377-384, 1991.

[12] P.M. Pardalos and J.B. Rosen, "Methods for Global Concave Minimization: A Bibliographic Survey," SIAM Review 28, pp. 367-379, 1986.

[13] P.M. Pardalos, J.H. Glick and J.B. Rosen, "Global Minimization of Indefinite Quadratic Problems," Computing 39, pp281-291, 1987.

[14] H. Tuy, "A General Deterministic Approach to Global Optimization via D.C. Programming," J.B. Hiriart-Urruty (editor), FERMAT Days 85: Mathematics for Optimization, pp. 273-303.

[15] Timothy Doup, Simplicial Algorithms on the Simplotope, Springer-Verlag, 1988.