SOLVING DC PROGRAMS USING THE CUTTING ANGLE METHOD

Albert Ferrer,[*] Adil Bagirov[†] and Gleb Beliakov[‡]

Abstract: In this paper, we propose a new algorithm for global minimization of functions represented as a difference of two convex functions. The proposed method is a derivative free method and it is designed by adapting the extended cutting angle method. We present preliminary results of numerical experiments using test problems with difference of convex objective functions and box-constraints. We also compare the proposed algorithm with a classical one that uses prismatical subdivisions.

Key words: DC programming, Lipschitz programming, Cutting Angle method.

Mathematics Subject Classification (2000): 90C26, 90C30.

1 Introduction

Difference of convex (DC) programming is an important class of problems in global optimization. Global optimization techniques are substantially different from local ones and, among others, employ combinatorial tools such as cutting-plane, branch and bound, branch and cut and so on. Many optimization problems of potential interest can be expressed into the form of a DC program such as production-transportation planning, location planning, engineering design, cluster analysis, multilevel programming and multi-objective programming. DC optimization algorithms have been proved to be particularly successful for analyzing and solving a variety of highly structured problems (see [16, 24]).

Over the last three decades different methods have been designed to solve DC programming problems. Horst and Tuy [17], Konno, Thach and Tuy [20], Tuy [27], Strekalovsky and Tsevendorj [26], An and Tao [22] and Ferrer [14] among others have put forward deterministic algorithms for solving DC programming problems whose nonconvexity is only due to having reverse convex constraints. All the mentioned algorithms combine conical or prismatical subdivision processes with polyhedral outer approximation in such a way that a finite number of linear programs have to be solved at each iteration.

The problem of global optimization of DC functions is NP-hard. The number of local optima can grow exponentially with the dimension while we are only interested in the best of these optima. Since the volume of the search domain grows exponentially with the dimension, it makes DC optimization

[*]Departament de Matemàtica Aplicada I, Universitat Politècnica de Catalunya, email: alberto.ferrer@upc.edu. The research of this author has been partially supported by the Ministerio de Ciencia y Tecnología, Project MTM2011-29064-C03-01

[†]School of Science, Information Technology and Engineering, Ballarat University, email: a.bagirov@ballarat.edu.au.

[‡]School of Engineering and Information Technology, Deakin University, Melbourne, Victoria, Australia. email: gleb@deakin.edu.au

computationally very expensive even for relatively small problems (about 10 variables). Then, efficiency of the algorithms is paramount.

In this paper, we develop an algorithm for solving DC programs with box-constraints and this algorithm is designed by adapting the Extended Cutting Angle Method (ECAM), which is a version of the Generalized Cutting Plane Method (GCPM) for various sets of min-type elementary functions. The adaptation of the ECAM is based on a new order for the enumeration of the local minima of the piecewise linear underestimates of the first component of the DC representation of the objective function. The new order depends on the underestimate function at each iteration and of the second component of the DC representation of the objective function.

The paper is structured as follows. In Section 2 we present some definitions and results on DC programming. Section 3 provides a brief description of the prismatical algorithm for solving DC programming problems. A short description of ECAM is given in Section 4. In Section 5 we describe the new algorithm, named DCECAM. The convergence of this algorithm is proved in Section 6. Section 7 presents the implementation of DCECAM. In Section 8 we report results of numerical experiments using DCECAM and compare it with one similar DC programming solver. Conclusions are summarized in Section 9.

2 Preliminaries

In this section we give some preliminary results on DC programming. We start with the definition of DC functions.

Definition 2.1 *Let Ω be a convex subset of $\mathbb{R}^n$. A function $\varphi : \Omega \to \mathbb{R}$ is said to be a DC function (difference of convex functions or δ-convex function) on Ω if there exists a pair of continuous convex functions $\varphi_1, \varphi_2 : \Omega \to \mathbb{R}$ such that*

$$\varphi(x) = \varphi_1(x) - \varphi_2(x),\ x \in \Omega.$$

Additionally, a function φ is called locally DC on Ω, if for each $x \in \Omega$ there exist an open ball U centered at x such that φ is DC on $U \cap \Omega$. We denote by $DC(\Omega)$ the class of DC functions defined on Ω.

In general, DC programming problems can be formulated as follows:

$$\begin{array}{rl} \text{minimize} & \varphi_0(x) \\ \text{subject to:} & \varphi_i(x) \leq 0,\ i = 1, \ldots, m, \\ & \varphi_{m+j}(x) \leq 0,\ j = 1, \ldots, r, \end{array} \tag{1}$$

where all functions $\varphi_0,\ \varphi_i,\ i = 1 \ldots m$ are DC functions on $\boldsymbol{R}^n$ and $\varphi_{m+j},\ j = 1 \ldots r$ are convex functions on $\boldsymbol{R}^n$. If a DC representation of these functions is known, then (1) can be transformed into an equivalent canonical DC program,

$$\begin{array}{rl} \text{minimize} & c^T x \\ \text{subject to:} & H(x) \leq 0, \\ & G(x) \geq 0 \end{array} \tag{2}$$

with $c \in \boldsymbol{R}^n$ and both H and G are convex functions on $\boldsymbol{R}^n$ (Horst and Tuy [17]).

Definition 2.2 *A function $f : \mathbb{R}^n \to \mathbb{R}$ is called Lipschitz continuous on the set $A \subseteq \mathbb{R}^n$ if there exists some finite constant $L > 0$, called a Lipschitz constant, satisfying*

$$|f(x) - f(y)| \leq L\|x - y\|$$

for every $x, y \in A$. Additionally, a function $f : \mathbb{R}^n \to \mathbb{R}$ is called locally Lipschitz on the set $A \subseteq \mathbb{R}^n$, if for each $x \in A$ there exists an open ball U, centered at x, such that f is Lipschitz continuous on the set $U \cap A$.

The class of DC functions, $DC(A)$, where A is an open convex set of $\mathbb{R}^n$, is a remarkable subclass of locally Lipstchitz functions that is of interest both analysis and optimization. It appears very naturally as the smallest vector space containing all continuous convex functions on a given set. Moreover, the class $DC(A)$ is dense in the class of continuous functions, $C^0(A)$. This property is a consequence of the Stone-Weierstrass theorem, which states that continuous functions on compact sets are uniform limits of sequences of polynomials. Since polynomials are functions of the class $C^2(A)$ one has that polynomials are DC functions on A. It turns out that any problem of minimizing a continuous real function over any compact convex set $D \subset A$ can be approximated as closely as desired by a problem of minimizing a DC function on D. DC functions were initially considered by Alexandrov [1] and Landis [21]. Later Hartman [15] proved that every locally DC function on $A \subset \boldsymbol{R}^n$, where A is an open or closed convex set, is DC on A. In [10] Bougeard proved that if $\varphi \in C^2(A)$ then there exists a DC representation (φ_1, φ_2) of φ in which $\varphi_1 \in C^2(A)$ and $\varphi_2 \in C^\infty(A)$. Penot and Bougeard [23] establish a similar result with more global assumptions. Indeed, let A be an open convex set of a finite dimensional normed vector space, then any lower$-C^2$ function φ on A, in particular any $\varphi \in C^2(A)$, can be written as $\varphi = \varphi_1 - \varphi_2$ with φ_1 and φ_2 convex and $\varphi_2 \in C^\infty(A)$. Moreover, every lower$-C^2$ function can be characterized by its (locally) decomposability as a sum of a convex continuous and a concave quadratic function (see [28]).

Some authors provide interesting theoretical DC representation results but no practical means to get them. While it is not too difficult to prove theoretically that a given function is DC, it is often very problematical to obtain an effective DC representation of a DC function. Although a DC representation is often not explicitly given, except for specific functions such as polynomials (see [13]), the richness of the space of DC functions suggest that DC structure is present in many optimization problems.

3 The prismatical algorithm for solving DC programs

This section is a summary of the prismatical algorithm for solving DC programming problems. For details and the general case with convex constraints we refer to [14].

By introducing an additional variable t, a DC program

$$\begin{array}{rc} \text{minimize} & f(x) - h(x) \\ \text{subject to:} & Ax \leq b, \\ & x \in S, \end{array} \tag{3}$$

where A is a real $m \times n$ matrix, $b \in \mathbb{R}^m$ and f, h are convex functions on $\mathbb{R}^n$ and S is a $(n+1)$-simplex in $\mathbb{R}^n$, can be transformed into a equivalent reverse convex program in the form:

$$\begin{array}{rc} \text{minimize} & f(x) - t \\ \text{subject to:} & h(x) - t \geq 0, \\ & Ax \leq b, \\ & x \in S. \end{array} \tag{4}$$

We will use the following notation

$$\begin{array}{lcl} D & := & \{(x,t) \in \mathbb{R}^n \times \mathbb{R} : \ Ax \leq b\}, \\ C & := & \{(x,t) \in \mathbb{R}^n \times \mathbb{R} : \ h(x) - t \leq 0\}, \\ D_\alpha & := & \{(x,t) \in D : \ f(x) - t \leq \alpha\}, \ \alpha \in \mathbb{R}, \end{array} \tag{5}$$

where D, C and D_α are closed convex sets.

Algorithms for solving deterministic reverse convex programs are described, for example, in [17] and [27]. All these algorithms begin by obtaining an initial point by solving a convex program. Without this point these algorithms cannot work because the vertex of a conical subdivision process is needed. In this article we will use the prismatical algorithm (DCPA) described by Ferrer in [14], for comparing with DCECAM. DCPA though designed in a similar spirit to algorithms used for solving reverse convex programs, has several differences and advantages. Among others, DCPA uses prismatical subdivisions in place of conical ones so that it will not be necessary to solve an initial convex program. The algorithm combines a prismatical subdivision process with polyhedral outer approximation in such a way that only linear programs have to be solved.

Theorem 3.1 *With suitable conditions of regularity we have,*

$$(x^*, t^*) \textit{is a global optimal solution of the program (4)} \Leftrightarrow D_{\alpha^*} \subset C, \alpha^* = f(x^*) - t^*.$$

For the sake of clarity we next give a brief summary of the algorithm in the case of linear constraints.

Definition 3.1 *Let Z be an n-simplex in $\mathbb{R}^n$. The set*

$$T(Z) := \{(x,\ t) \in \mathbb{R}^n \times \mathbb{R} :\ x \in Z\}$$

is referred to as a simplicial prism of base Z. All simplicial prisms have $n+1$ edges that are parallel lines to the t-axis. Each such edge passes through one of the $n+1$ vertices of Z. Then, each radial subdivision $Z^1, \ldots, Z^r$, of the simplex Z via a point $z \in Z$ induces a prismatical subdivision of the prism $T(Z)$ in subprisms, $T(Z^1), \ldots, T(Z^r)$, via the line through z parallel to the t-axis.

Consider $T_0 := T(Z_0)$ an initial prism, with $Z_0 := [v_0^1, \ldots, v_0^{n+1}]$ an n-simplex of $\mathbb{R}^n$ which contains the polytope $\{x \in \mathbb{R}^n : Ax \leq b\}$. Let P_0 be an initial convex polyhedron

$$P_0 := \{(x,t) : Ax \leq b, l_i(x,t) \leq 0, i = 1, \ldots, n+1\},$$

with

$$l_i(x,t) := (x - v_0^i)^T p_i - t + c_i, i = 1, \ldots, n+1$$

and p_i a subgradient of the function f at the vertex $v_0^i \in Z$ and $c_i = h(v_0^i)$ (in practice we will actually take $c_i = h(v_0^i) + \epsilon$ to STOP the algorithm with precision $\epsilon > 0$). At each iteration the procedure involves some basic operations as follows:

- *Branching:* a selected prism $T(Z)$ is divided into a finite number of subprisms by using a simplicial partition of Z.
- *Outer approximation:* a new polyhedron P_k is obtained by using a cutting plane to cut off a part of P_{k-1} in such a way that a sequence of convex polyhedra $P_0, P_1, \ldots$ is constructed satisfying
$$P_0 \supset P_1 \supset \cdots \supset P_k \supset \cdots \supset D_{\alpha^*} \supset D_{\alpha^*} \backslash \mathrm{int}\ C.$$
- *Delete rule:* prisms containing feasible solutions worse than the best one obtained so far are deleted.

The basic operations used in the algorithm are related to the optimum $\mu(T)$ and the optimizer $(x(T), t(T))$ of the linear program

$$\begin{array}{rl} \mu(T) = \text{maximize} & a^t x - t - \rho \\ \text{subject to:} & (x,t) \in T \cap P, \end{array} \tag{6}$$

where $T := T(Z)$ is a prism with $Z = [v^1, \dots, v^{n+1}]$, P is the current convex polyhedral and $a^t x - t - \rho$ is the unique hyperplane passing through the points $(v^i, h(v^i)), i = 1, \dots, n+1$. The optimizer of (6) is the point of the polyhedral $T \cap P$ with the greatest distance to the hyperplane $a^t x - t - \rho$. When $\mu(T) \leq 0$ then $T(Z)$ is deleted because $T \cap P \subset C$. On the other hand, if a prism T is selected for division, which is associated with the largest value $\max\{\mu(T) : \mu(T) > 0\}$, a refined partition of T is constructed and a new cut is added to the polyhedral P to obtain a new convex polyhedral. From the solution $(x(T), t(T))$ the new point $(\bar{x}, \bar{t})$ is obtained with $\bar{x} := x(T)$ and $\bar{t} := \max\{h(x(T)), g(x(T))\}$. If $h(x(T)) - t(T) > 0$, a new cut $l(x,t)$ is added through the point $(\bar{x}, \bar{t})$ to obtain a new convex polyhedral

$$P \cap \{(x,t) : l(x,t) \leq 0\}.$$

The procedure continues until all generated prisms have been deleted. At this stage, we have

i) a partition $\{T_i, i = 1, 2, \dots\}$ of the initial prism $T_0 = \cup_{i=1}^{\infty} T_i$ with $\mu(T_i) < 0, i = 1, 2, \dots$

ii) a sequence of convex polyhedrons $P_0, P_1, \dots$ such that

$$P_0 \supset P_1 \supset \cdots \supset D_{\alpha^*} \supset D_{\alpha^*} \backslash \text{int } C,$$

where each polyhedral $P_j, j = 1, 2 \dots$ is obtained by using a cutting plane to cut off a part of $P_{j-1}, j = 1, 2 \dots,$

iii) a bounded sequence of points $\{(\bar{x}_k, \bar{t}_k), k = 0, 1, \dots\}$ so that there exists a subsequence $\{k_i\}$ such that $\{(\bar{x}_{k_i}, \bar{t}_{k_i})\} \to (x^*, t^*)$.

Hence, we have that for all $i = 1, 2, \dots$ there exists an index k such that $T_i \cap P_k \subset C$. Since $T_i \cap D_{\alpha^*} \subset T_i \cap P_k \subset C$ for all $i = 1, 2, \dots$ we can write

$$C \supset \cup_{i=1}^{\infty} (T_i \cap D_{\alpha^*}) = \cup_{i=1}^{\infty} (T_i) \cap D_{\alpha^*} = T_0 \cap D_{\alpha^*} = D_{\alpha^*}. \tag{7}$$

On the other hand, the point (x^*, t^*) as defined in (iii) satisfies

$$(x^*, t^*) \in D_{\alpha^*} \backslash \text{int } C \text{ and } \alpha^* = f(x^*) - t^*. \tag{8}$$

From the Theorem 3.1 we can deduce that the point (x^*, t^*) is a global minimizer.

4 A short description of ECAM

We start this section by recalling the Cutting Plane method by Kelley [19] and Cheney and Goldstein [11] to solve convex programs. It is known that a lower semicontinous convex function f is the upper envelope of the set of all its affine minorants:

$$f(x) = \sup \{h(x) \mid h \text{ affine function}, h \leq f\}. \tag{9}$$

The Cutting Plane method proceeds by constructing a piecewise affine underestimate of the objective function f as a pointwise maximum of elements of the subdifferential of f. At each iteration a new support hyperplane is built at the point of the global minimum of the current underestimate, which is found by methods of linear programming. Then the underestimate is modified by adding this new support hyperplane so it becomes a more accurate approximation to f, and the method iterates. The generalized cutting plane method (of which the ECAM is a special case) follows the same script, except that the underestimate is built using the so-called $H-$subgradients (see [25]). Consequently, minimization of an underestimate is no longer a convex problem. For special cases of abstract convex functions, specialized efficient numerical methods for minimizing the underestimates exist [2, 3, 4, 8, 25].

4.1 Solution to the relaxed problem for ECAM

This subsection reproduces details of the Cutting Angle Method from [3, 4, 6, 7, 8] to solve the following problem:

$$\begin{array}{rl} \text{minimize} & f(x) \\ \text{subject to:} & x \in S, \end{array} \tag{10}$$

with S the unit simplex and $f : \mathbb{R}^n \to \mathbb{R}$ a Lipschitz function. These details are essential for understanding how the DCECAM works. Let us define the *support vectors* $l^k,\ k = 1, 2, \ldots$:

$$l_i^k := \frac{f(x^k)}{C} - x_i^k, i = 1, \ldots, n+1. \tag{11}$$

As the support functions, we use

$$h^k(x) := \min_{i=1,\ldots,n+1} (f(x^k) - C(x_i^k - x_i)) = \min_{i=1,\ldots,n+1} C(l_i^k + x_i). \tag{12}$$

We are interested in locating all local minima of the function

$$H^K(x) := \max_{k=1,\ldots,K} h^k(x) \tag{13}$$

in S' (feasible domain in the plane $\{x \in \mathbb{R}^{n+1} | \sum x_i = 1\}$), which after sorting will yield the global minimum of H^K.

Theorem 4.1 *[7, 8] A necessary and sufficient condition for a point $x^* \in ri\, S'$ to be a local minimizer of $H^K(x)$ given by (12),(13) is that there exist an index set $J = \{k_1, k_2, \ldots, k_{n+1}\}$ of cardinality $n+1$, such that*

$$d = H^K(x^*) = C(l_1^{k_1} + x_1^*) = C(l_2^{k_2} + x_2^*) = \ldots = C(l_n^{k_{n+1}} + x_{n+1}^*),$$

and $\forall i \in \{1, \ldots, n+1\}$,

$$(l_i^{k_i} + x_i^*) < (l_j^{k_i} + x_j^*), j \neq i.$$

Let x^* be a local minimizer of $H^K(x)$, which corresponds to some index set J satisfying conditions of Theorem 4.1. Form the ordered combination of the support vectors $L = \{l^{k_1}, l^{k_2}, \ldots, l^{k_{n+1}}\}$ that corresponds to J. It is helpful to represent this combination with a matrix L whose rows are the support vectors l^{k_i}:

$$L := \begin{pmatrix} l_1^{k_1} & l_2^{k_1} & \cdots & l_{n+1}^{k_1} \\ l_1^{k_2} & l_2^{k_2} & \cdots & l_{n+1}^{k_2} \\ \vdots & \vdots & \ddots & \vdots \\ l_1^{k_{n+1}} & l_2^{k_{n+1}} & \cdots & l_{n+1}^{k_{n+1}} \end{pmatrix}, \tag{14}$$

so that its components are given by $L_{ij} = \frac{f(x^{k_i})}{C} - x_j^{k_i}$.

Theorem 4.2 *[8] Let the support vectors $l^k, k = 1, \ldots, K$ be defined using (11). Let x^* denote a local minimizer of $H^K(x)$ in $ri\, S'$ and $d = H^K(x^*)$. Then matrix (14) corresponding to x^* enjoys the following properties:*

1) $\forall i, j \in \{1, \ldots, n+1\}, i \neq j : l_i^{k_j} > l_i^{k_i}$;

2) $\forall r \notin \{k_1, k_2, \ldots, k_{n+1}\}\ \exists i \in \{1, \ldots, n+1\} : L_{ii} = l_i^{k_i} \geq l_i^r$;

3) $d = \frac{Trace(L)+1}{C^-}$, where $C^- = \sum\limits_{i \in \{1,\ldots,n+1\}} \frac{1}{C} = \frac{n+1}{C}$ and

4) $x_i^* = \frac{d}{C} - l_i^{k_i}$, $i = 1, \ldots, n+1$.

Table 1: Extended Cutting Angle Method framework.

Procedure: ECAM

```
Initialization: Set $k := 0$, stop:=1 and choose $x^0 \in X$;
begin
  while (stop) do
      Calculate a support function $h_k(x)$;
      Let $H_k(x) = \max_{i=0,\dots,k} h_i(x)$ for all $x \in X$; (saw-tooth understimate)
      Find a global minimum $y^*$ of the problem $\min_{x \in X} H_k(x)$;
      Set $k := k+1$, $x^k := y^*$;
      stop=evaluateStop();
  end while
  return bestSolution;
end
```

Furthermore, combinations L determine a partition of S' into subsets $A(L)$, such that in the interior of each subset the local minimum d of H^K is unique. The following system of inequalities determines a subset $A(L)$

$$\forall i, j \in \{1, \dots, n+1\}, i \neq j : C(x_j - x_j^{k_j}) \leq C(x_i - x_i^{k_j}), \tag{15}$$

and on each $A(L)$ function H^K is given by

$$H^K(x) = \max_{i=1,\dots,n+1} C(x_i + l_i^{k_i}). \tag{16}$$

Condition 1) of the Theorem 4.2 reads that the diagonal elements of matrix L are *dominated* by their respective columns, and condition 2) reads that no support vector l^r (which is not part of L) strictly dominates the diagonal of L. The approach taken in [4, 5] is to enumerate all combinations L with the properties 1)-2), which will give the positions of local minima x^* and their values d by using 3)-4).

In [4, 5] is showed that combinations L can be built incrementally, by taking initially the first $n+1$ support vectors (which yields the unique combination $L = \{l^1, l^2, \dots, l^{n+1}\}$), and then adding one new support vector at a time. Clearly,

$$H^K(x) = \max\{H^{k-1}(x), h^k(x)\}, k = n+2, \dots, K. \tag{17}$$

Suppose, we have already identified the local minima of $H^{k-1}(x)$, i.e., all the required combinations L. Let us denote this set by V^{k-1}. When we add another support vector l^k, we can "inherit" most of the local minima of $H^{k-1}(x)$ (a few will be lost since condition 2) of Theorem 4.2 may fail with l^k playing the role of l^r), and we only need to add a few new local minima, that are new combinations L necessarily involving l^k. In [4] is proved that these new combinations are simple modifications of those combinations that were discarded because they failed 2) with $l^r = l^k$. Furthermore, in [5, 8] is proved that all local minimizers of the functions $H^{n+1}, \dots, H^k, \dots H^K$ can be represented in a tree structure, in which the required minima of H^K are the leaves. The root of the tree is $V^{n+1} = \{(l^1, l^2, \dots, l^{n+1})\}$. It is possible to perform test of the condition 2) by starting the test from the root using depth-first search. This results in a highly efficient algorithm for enumeration of local minima of H^K [5, 6, 8]. Table 1 describes ECAM.

Algorithm 4.1 ***Extended Cutting Angle Algorithm***
$T^K, K \geq n+1$ denotes the tree of local minima of functions $H^{n+1}, \ldots, H^K$, V^K denotes the priority queue containing the leaves of the tree arranged in the order of increasing $d(L)$. $d(L)$ is computed by using property 3) in Theorem 4.2.

Step 0. (Initialisation)

- *0.1 Take the initial points $x^k, k = 1, \ldots, n+1$, and construct the support vectors $l^k, k = 1, \ldots, n+1$ according to (11).*
- *0.2 Set $K := n+1$, $L_{root} := \{l^1, l^2, \ldots, l^{n+1}\}$, $T^{n+1} = V^{n+1} := \{L_{root}\}$.*
- *0.3 $f_{best} := \min_{k=1,\ldots,n+1} f(x^k)$.*

Step 1. (Form a new support vector)

- *1.1 Choose $L^* := Head(V^K)$ (corresponds to the global minimum of H^K).*
- *1.2 Form $x^* := x^*(L^*)$ using condition 4) of Theorem 4.2.*
- *1.3 Evaluate $f^* := f(x^*)$. $f_{best} := \min\{f_{best}, f^*\}$.*
- *1.4 Set $K := K+1$.*
- *1.5 Form l^K using $l_i^K := \frac{f(x^*)}{C} - x_i^*$.*

Step 2. (Update T^K)

- *2.1 Call Algorithm 7.1 (T^{K-1}, l^K,V^{n+1},T^K,V^K).*
- *2.2 Delete from V^K elements L if $d_P(L) < f_{best}$.*

Step 3. (Stopping criterion)

- *3.1 If $K < K_{max}$ and $f_{best} - d_P(L^*) > \epsilon$ go to Step 1.*

5 The proposed algorithm DCECAM

We are interested in obtaining the global minimum of a DC function on a compact set $D \subset \mathbb{R}^n$.

$$\begin{array}{rc} \text{minimize} & f(x) - g(x) \\ \text{subject to:} & x \in D. \end{array} \tag{18}$$

where f and g are convex on $\boldsymbol{R}^n$. Without loss of generality we consider D the unit simplex. DC programs can be transformed into equivalent reverse convex programs by introducing an additional variable t.

$$\begin{array}{rc} \text{minimize} & f(x) - t \\ \text{subject to:} & g(x) - t \geq 0, \\ & x \in D. \end{array} \tag{19}$$

First, we present the following well-known result from convex analysis.

Proposition 5.1 *If $f : \mathbb{R}^n \to [-\infty, \ +\infty]$ is a proper convex function with $f(z)$ finite for some $z \in \mathbb{R}^n$ then it follows that the function f is continuous on ri(dom(f)) and Lipschitz continuous on every compact subset of ri(dom(f)).*

When the DC components f and g are both convex functions on $\mathbb{R}^n$ then the DC program (18) has a objective function that is Lipschitz on any compact D so algorithms for solving Lipschitz programs, such as the ECAM, can be directly applied for solving the program (18). Nevertheless, there exists DC functions on compact sets that are not Lipschitz on them, for example, the non Lipstchitz function $\phi(x) := \sqrt{x}$ on $[0, 1]$ is a DC function on $[0, 1]$, since it can be written in the form $\phi(x) = x^2 - (x^2 - \sqrt{x})$. Notice, that in the DC representation of $\sqrt{x}$ the first component, x^2, is a Lipschtiz function but not the second component, $x^2 - \sqrt{x}$. In this case Lipschitz programming cannot be used and specific DC algorithms are needed.

In this section we present a generalization of ECAM so additional DC programs can be solved. Indeed, when the first component of a DC representation of the DC objective function is a Lipschitz convex function then it is possible to design a new approach, named DCECAM, that generalizes ECAM for solving DC programs. We apply ECAM to the first component $f(x)$ of the DC representation of the DC objective function, which is considered as the new objective function. Then, at each iteration, k, a new increasing order is established on the local minimums of the saw-tooth underestimate, $H^k(x)$, of $f(x)$ by evaluating the function $H^k(x) - g(x)$ at every local minimum of the underestimate. The new approach performs a new Branch and Bound on ECAM that converges to a global minimum of the DC objective function $f(x) - g(x)$.

Following ECAM, at each iteration k, we consider support functions of type

$$h_k(x) = \min_{i=1,\dots,n} (f(x^k) - C_i(x_i^k - x_i)), \tag{20}$$

where coefficients C_i can be chosen to coincide with the Lipschitz constant of the objective function. Support functions can be rewritten in the form:

$$h_k(x) = \min_{i=1,\dots,n} (C_i x_i + b_i),\ C_i > 0,\ x \in \mathbb{R}^n,\ \sum_{i=1}^{n} x_i = 1, \tag{21}$$

where $b_i = f(x^k) - C_i x_i^k$. Explicit characterizations of the local minima of the underestimate provides a partition of the domain into smaller regions. Therefore this type of method for Lipschitz programming can be seen as a version of a branch-and-bound method. The underestimate provides lower bounds of the objective function on each region of the partition. When a new support function is added to the underestimate, the partition is refined (branching step), and new lower bounds are calculated (bounding step). Such relaxed problems arise at every iteration of the ECAM.

Our approach for solving the DC program (18) consists of the iterative process of building at each iteration a piecewise linear underestimate.

$$H^k(x) = \max\{H^{k-1}(x),\ h_k(x)\} = \max_{0 \le j \le k} h_j(x). \tag{22}$$

At this point we translate the relaxed problem to a combinatorial problem of enumerating all local minima by using the new order defined by the evaluation of the function $H^k(x) - g(x)$ at each local minima of $H^k(x)$. Then we consider the local minima, x^{k+1}, satisfying

$$H^k(x^{k+1}) - g(x^{k+1}) = \min\{H^k(z) - g(z) : z \text{ local minima } of H^k(x)\}. \tag{23}$$

Table 2 summarizes the DCECAM. We establish its convergence in Section 6.

6 Convergence of DCECAM

In this section we establish the convergence of the DCECAM for solving the DC programming problem 18, in which f and g are convex and in addition, f is Lipschit on D. We introduce

Table 2: DCECAM general framework

Procedure: DCECAM

```
Initialization: Set $k := 0$, stop:=1 and choose $x^0 \in D$;
begin
  while (stop) do
     Calculate $h_k(x) = \min_{i=0,\dots,n} f(x_k) - C_i(x_i^k - x_i)$;
     Let $H^k(x) = \max_{i=0,\dots,k} h_i(x)$ for all $x \in D$;
     Find a local optimum $y^*$ of the problem $\max_{x\in D} H^k(x)$
     with minimum value for the function $H^k(x) - g(x)$;
     Set $k := k + 1$, $x^k := y^*$;
     stop=evaluateStop();
 end while
 return bestSolution;
end
```

the following useful quantities: $\alpha_k := f_{k-1}(x^k) - g(x^k)$, $k = 1, 2, \dots$, and $\beta_k := f_k(x^k) - g(x^k)$, $k = 0, 1, \dots$. Next lemmas point out some properties of the functions f_k and the numbers α_k and β_k.

Lemma 6.1 *For all $k = 0, 1, \dots$ we have $\beta_k = f(x^k) - g(x^k) = h_k(x^k) - g(x^k)$.*

Proof. Indeed,

$$\beta_k = f_k(x^k) - g(x^k) \le f(x^k) - g(x^k) = h_k(x^k) - g(x^k) \le \max_{0\le i\le k} h_i(x^k) - g(x^k) = f_k(x^k) - g(x^k) = \beta_k.$$

Lemma 6.2 *For all $k = 1, 2, \dots$, the following inequalities hold:*

$$\alpha_k \le \min_{x\in D}(f - g)(x) \le \beta_k.$$

Proof.

- For all $x \in D$ we have $f_{k-1}(x) - g(x) \le f(x) - g(x)$, then

$$\alpha_k = f_{k-1}(x^k) - g(x^k) = \min_{x\in D} f_{k-1}(x) - g(x) \le \min_{x\in D} f(x) - g(x).$$

- From lemma 6.1 we can write

$$\min_{x\in D}(f - g)(x) \le f(x^k) - g(x^k) = h_k(x^k) - g(x^k) = \beta_k.$$

Corollary 6.1 *The sequence $\{\alpha_k\}$ is nondecreasing and bounded, that implies the existence of the limit $\lim\limits_{k\to+\infty} \alpha_k = \alpha$.*

Proof. The algorithm has been designed in such a way that satisfies the inequality

$$\alpha_k = f_{k-1}(x^k) - g(x^k) \le f_k(x^{k+1}) - g(x^{k+1}) = \alpha_{k+1}.$$

On the other hand, from Lemma 6.2 we have that $\{\alpha_k\}$ is bounded for any β_k.

Theorem 6.1 *Each limit point x^* of the sequence $\{x^k\}$ produced by DCECAM is a global minimizer of the program (18).*

Proof. Let $x^* = \lim_{j\to+\infty} x_{k_j}$ then for each j, consider the function $h_i(x)$ with $i \leq k_j$ and find a supergradient $\varphi_i(x)$ of the concave function $h_i(x)$ at the point x^{k_j}, $\varphi_i(x) = h_i(x^{k_j}) + \langle a_i, x - x^{k_j}\rangle$. Then $\varphi_i(x) \geq h_i(x)$ for all $x \in D$ and $\varphi_i(x^{k_j}) = h_i(x^{k_j})$.

$$\begin{aligned}
\beta_{k_{j-1}} &= f_{k_{j-1}}(x^{k_{j-1}}) - g(x^{k_{j-1}}) \\
&= \max_{0\leq i\leq k_{j-1}} h_i(x^{k_{j-1}}) - g(x^{k_{j-1}}) \\
&\leq \max_{0\leq i\leq k_{j-1}} \varphi_i(x^{k_{j-1}}) - g(x^{k_{j-1}}) \\
&= \max_{0\leq i\leq k_{j-1}} (h_i(x^{k_j}) + \langle a_i, x - x^{k_j}\rangle) - g(x^{k_{j-1}}) \\
&\leq \max_{0\leq i\leq k_{j-1}} h_i(x^{k_j}) + \max_{0\leq i\leq k_{j-1}} \|a_i\|\|x - x^{k_j}\| - g(x^{k_{j-1}}) \\
&= f_{k_{j-1}}(x^{k_j}) + \max_{0\leq i\leq k_{j-1}} \|a_i\|\|x - x^{k_j}\| - g(x^{k_{j-1}}) \\
&= f_{k_{j-1}}(x^{k_j}) - g(x^{k_j}) + \max_{0\leq i\leq k_{j-1}} \|a_i\|\|x - x^{k_j}\| + g(x^{k_j}) - g(x^{k_{j-1}}) \\
&= \alpha_{k_j} + \max_{0\leq i\leq k_{j-1}} \|a_i\|\|x - x^{k_j}\| + g(x^{k_j}) - g(x^{k_{j-1}}).
\end{aligned}$$

Since $\|a_i\|$ with $i \leq k_j$ are bounded from above for all j (for the chosen constant $C \geq L$ with L being the Lipstchitz constant of the component f), $\|x - x^{k_j}\| \to 0$ and $g(x^{k_j}) - g(x^{k_{j-1}}) \to 0$ when $j \to +\infty$, we have that $\limsup \beta_{k_j} \leq \alpha$. On the other hand from Lemma 6.2 we deduce that $\liminf \beta_{k_j} \geq \alpha$. Finally, we obtain:

$$\min_{x\in D}(f-g)(x) = \alpha = \lim_{j\to+\infty} \beta_{k_j} = \lim_{j\to+\infty}(f_{k_j}(x^{k_{j-1}}) - g(x^{k_j})) = \lim_{j\to+\infty}(f(x^{k_j}) - g(x^{k_j})) = f(x^*) - g(x^*).$$

7 Implementation of algorithms

In this section we discuss the implementation of algorithms. We start with the Extended Cutting Angle Method.

7.1 Implementation of ECAM

This implementation consists of two parts. The inner recursive part updates the tree of local minima T^{K-1} to T^K. The main Algorithm 4.1 calls the Algorithm 7.1 to maintain the tree T^K. A special data structure was designed to hold the information about the local minimizers of H^K [9]. It consists of an $n+1$-ary tree and a priority queue (binary heap) to hold the references to the leaves of the tree. The use of the priority queue simplifies sorting out local minima of H^K, as locating the global minimum is $O(1)$ operation. The maintenance of the heap at every iteration takes $O(\log |V^K|)$ operations.

Algorithm 7.1 ***Update of the tree*** T^{K-1}, $K > n+1$
Input: The tree T^{K-1} of local minima of $H^{n+1}, H^{n+2}, \dots, H^{K-1}$; the new support vector l^K; tested node L.
Output: The tree T^K; the set of leaves V^K.

```
Step 1. Test $L$ against condition 2), with $l^r = l^K$.
Step 2. If test succeeds, go to Step 5 (cut off this branch).

Step 3. If test fails, and $L$ is not a leaf, then
        call Algorithm 7.1 ($T^{K-1}$,$l^K$,child($L$), $T^K$,$V^K$) for all children of $L$.
        Go to Step 5.

Step 4  Otherwise (test failed, and $L$ is a leaf) add $n+1$ children to $L$.
        Each child node is a copy of $L$, with $l^{k_i}$ replaced with $l^K$ in the $i$-th child.
        Test condition 1) for each child. If test fails, delete this child node.

Step 5  If $L$ is $V^n$ (root), then $T^K$=$T^{K-1}$; $V^K$ = leaves($T^K$)
        (we need to check this only once, at the first level of recursion).
        Return.
```

7.2 Implementation of DCECAM

The problem of adapting the Extended Cutting Angle method for solving DC programs is important as far as the new approach will be more efficient from a computational point of view. Basically the new approach perform a new Branch and Bound on ECAM by using the reverse convex constraint $g(x) - t \geq 0$.

Algorithm 7.2 ***DCECAM***
T^K denotes the tree of local minima of functions $H^{n+1}, \dots, H^K$, V^K denotes the priority queue containing the leaves of the tree arranged in the INCREASING order determinated by evaluating $H^K(x) - g(x)$ at every local minimum of $H^K(x)$.

```
Step 0. (Initialisation)
    0.1 Take the initial points $x^k, k = 1, \dots, n+1$, and construct the support vectors $l^k, k = 1, \dots, n+1$
        according to (11).
    0.2 Set $K := n+1$, $L_{root} := \{l^1, l^2, \dots, l^{n+1}\}$, $T^{n+1} = V^{n+1} := \{L_{root}\}$.
    0.3 $h_{best} := \min_{k=1,\dots,n+1} f(x^k) - g(x^k)$.
Step 1. (Form a new support vector)
    1.1 Choose $L^* := Head(V^K)$ (corresponds to the maximum of $g(x) - t$ at the nodes of $H^K$).
    1.2 Form $x^* := x^*(L^*)$ using condition 4) of Theorem 4.2.
    1.3 Evaluate $h^* := f(x^*) - g(x^*)$, $h_{best} := \min\{h_{best}, h^*\}$.
    1.4 Set $K := K+1$.
    1.5 Form $l^K$ using $l_i^K := \frac{f(x^*)}{C} - x_i^*$.
Step 2. (Update $T^K$)
    2.1 Call Algorithm 7.1 ($T^{K-1}$, $l^K$,$V^{n+1}$,$T^K$,$V^K$).
Step 3. (Stopping criterion)
    3.1 If $K < K_{max}$ and $h_{best} + H^K(L^*) > \epsilon$ go to Step 1.
```

8 Numerical experiments

In this section we present results of numerical experiments using some DC programming test problems. We consider only test problems with box constraints. The description of test problems can be found in Section 10: Appendix. We used both the DCECAM and DCSOL in numerical experiments. These experiments were carried out on a PC with Processor Intel(R) Core(TM) i5-3470S CPU 2.90 GHz.

Results are presented in Tables 3 and 4 in which ε is the given precision. Note that ε is the accuracy with which both methods guarantees the globally optimal solution, i.e., the difference between the minimum found and its guaranteed underestimate, which is different to the accuracy compared to the known solution. In tables we include the number of function calls, the CPU time required by algorithms and the best function value found by these algorithms. For the DCECAM we report the number of DC objective function evaluations whereas for the DCSOL we report this number for each component of the DC representation. For the DCECAM algorithm we report CPU time and the number of function calls necessary to obtain the best reported value.

Table 3: Results for problems with non-Lipschitz objective functions and with $\varepsilon = 0.01$

Prob	Dim	DCECAM			DCSOL		
		Best value	Function calls	CPU time(sec.)	Best value	Function calls	CPU time(sec.)
10.1	2	0.999981	1264	2.61	−0.999998	5037/12717	8.19
10.2	2	0.758714	761	0.16	0.758588	18/46	0.00
10.3	2	−2.092702	529	0.31	−2.098612	44/111	0.00
10.4	3	−6.289743	1058	1.24	−6.295837	1291/3642	1.28
10.4	4	−2.183011	1846	1.53	−2.197225	858/2731	1.28
10.4	5	3.861705	2109	2.78	3.861668	955/3631	3.34
10.4	10	8.503067	3107	24.15	8.501569	1284/7441	26.25
10.4	15	12.64905	3461	34.71	12.643976	932/7105	41.84
10.4	20	18.918043	12095	86.14	19.015127	722/6905	71.20

9 Discussion and concluding remarks

The results are analyzed using the performance profile introduced in [12]. We compare the efficiency of the solvers both in terms of CPU time and the number of function evaluations. Results of the numerical experiments are described in Tables 3 and 4 and Figure 1. As we can see both solvers are robust and the comparative behavior of solvers are similar for CPU-time, in the 56% of the problems DCECAM is the fast against the 44% of DCSOL. Nevertheless, DCECAM is better than DCSOL in the number of function evaluations, 72% and 28% respectively, as it can be seen in Figure 1, which is due to the efficiency of the design of DCECAM, which developed a modified version of Extended Cutting Angle Method. The new algorithm takes into account a new order for the enumeration of the local minima of the piecewise linear underestimates of the first component of the DC representation of the objective function. The new order depends on the underestimate function at each iteration and of the second component of the DC representation of the objective function, the only function that is necessary to evaluate. This is one of the main advantages of using DCECAM from a computational point of view.

Table 4: Numerical results for Lipschitz DC problems with $\varepsilon = 0.01$

Prob	Dim	DCECAM			DCSOL		
		Best value	Function calls	CPU time(sec.)	Best value	Function calls	CPU time(sec.)
10.5	2	−2.140901	1108	0.25	−2.140748	297/747	0.17
10.6	2	−0.009604	953	1.47	−0.009212	2855/7217	2.22
10.7	2	−9.000000	671	0.05	−9.000000	28/76	0.00
10.8	2	−0.999453	1206	1.14	−0.999869	1109/2774	0.78
10.9	4	0.002348	3210	9.71	0.002209	2842/7652	12.09
10.10	2	0.000000	1201	0.21	0.000241	364/925	0.22
10.10	3	3.572103	4587	3.57	3.571803	2269/5775	4.63
10.10	4	0.553429	2395	2.74	0.545601	666/1843	0.78
10.10	5	1.500652	14382	345.12	1.500000	12679/35677	502.29

On the other hand, DCECAM's contribution is also important from a theoretical point of view because it allows generalizing ECAM (which requires that the objective function be a Lipschitz function) to non-Lipschitzian objective functions conveniently expressed as a difference of convex functions. In this way, ECAM, through of DCECAM, solves DC programming problems (in addition to solving Lipschitz programming problems) in a competitive way against specific DC solvers.

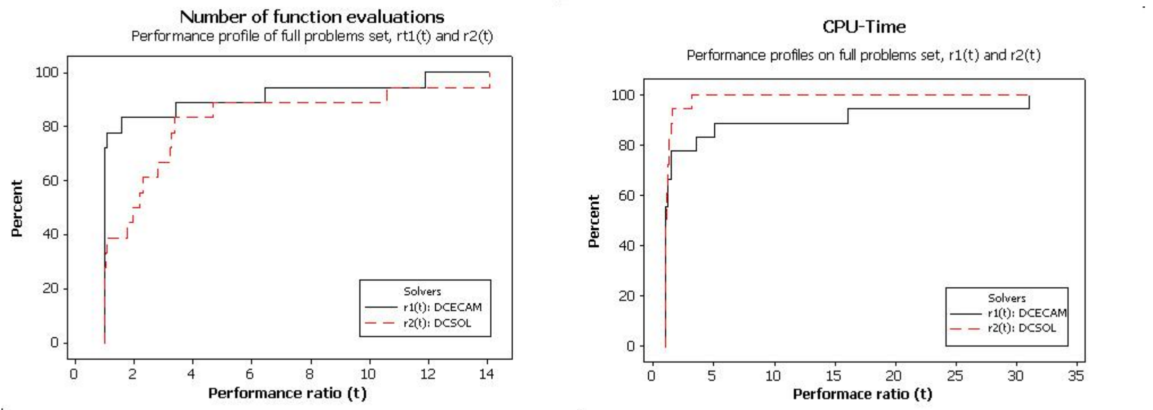

Figure 1: Performance profiles of number of function evaluations and CPU-time

Notice that mathematical and computational modeling in several areas requires solving DC programs such as production-transportation planning, location planning, engineering design, cluster analysis, multilevel programming and multi-objective programming. DC optimization algorithms have been proved to be particularly successful for analyzing and solving a variety of highly structured problems (see [16, 24]).

References

[1] A.D. Alexandrov. On surfaces which may be represented by a difference of convex functions (in russian). *Izvestia Akademii Nauk Kazakhskoj SSR, Seria Fiziko-Matematicheskikh Nauk*, 3:3–20, 1949.

[2] A.M. Bagirov and A.M. Rubinov.

[3] A.M. Bagirov and A.M. Rubinov. Modified versions of the cutting angle method. In N. Hadjisavvas and P.M. Pardalos, editors, *Convex analysis and global optimization*, volume 54 of *Nonconvex optimization and its applications*, pages 245–268. Kluwer, Dordrecht, 2001.

[4] L.M. Batten and G. Beliakov. Fast algorithm for the cutting angle method of global optimization. *Journal of Global Optimization*, 24:149–161, 2002.

[5] G. Beliakov. Geometry and combinatorics of the cutting angle method. *Optimization*, 52(4-5):379–394, 2003.

[6] G. Beliakov. The cutting angle method - a tool for constrained global optimization. *Optimization Methods and Software*, 19:137–151, 2004.

[7] G. Beliakov. A review of applications of the cutting angle methods. In A. Rubinov and V. Jeyakumar, editors, *Continuous Optimization*, pages 209–248. Springer, New York, 2005.

[8] G. Beliakov. Extended cutting angle method of global optimization. *Pacific Journal of Optimization*, 4(1):153–175, 2008.

[9] G. Beliakov, K. M. Ting, M. Murshed, A.M. Rubinov, and M. Bertoli. Efficient serial and parallel implementations of the cutting angle method. In G. Di Pillo, editor, *High Performance Algorithms and Software for Nonlinear Optimization*, pages 57–74. Kluwer Academic Publishers, 2003.

[10] Bougeard, M. *Contribution à la théorie de Morse en dimension finie.* PhD thesis, Université de Paris IX, Paris, 1978.

[11] E.W. Cheney and A.A. Goldstein. Newton's method for convex programming and Tchebycheff approximation. *Numer. Math.*, 1:253–268, 1959.

[12] Dolan E.D. and Moré J.J. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91:201–213, 2002.

[13] Ferrer, A. Representation of a polynomial function as a difference of convex polynomials, with an application. *Lectures Notes in Economics and Mathematical Systems*, 502:189–207, 2001.

[14] Ferrer, A. Applying global optimization to a problem in short-term hydrotermal scheduling. *Nonconvex Optimization and its Applications*, 77:263–285, 2005.

[15] Hartman, P. On functions representable as a difference of convex functions. *Pacific Journal of Mathematics*, 9:707–713, 1959.

[16] Holmberg, K. and Tuy, H. A production-transportation problem with stochastic demand and concave production costs. *Mathematical Programming*, 85:157–179, 1999.

[17] Horst, R. and Tuy, H. *Global optimization. Deterministic Approaches.* Springer-Verlag, Heilderberg, first edition, 1990.

[18] Horst, R., Pardalos, P.M. and Thoai, Ng.V. *Introduction to gobal optimization.* Kluwer Academic Publishers, Dordrescht, first edition, 1995.

[19] J.E. Kelley, Jr. The cutting-plane method for solving convex programs. *J. Soc. Indust. Appl. Math.*, 8:703–712, 1960.

[20] Konno, H., Thach, P.T. and Tuy, H. *Optimization on low rank nonconvex structures.* Kluwer Academic Publishers, New York, 1997.

[21] Landis, E.M. On functions representable as the difference of two convex functions. *Dokl. Akad. Nauk SSSR*, 80:9–11, 1951.

[22] Le Thi Hoai An and Pham Dinh Tao. The DC (difference of convex functions) Programming and DCA revisited with DC models of real world nonconvex optimization problems. *Annals of Operations Research*, 133:23–46, 2005.

[23] Penot, J.P. and Bougeard, M.L. Approximation and decomposition properties of some classes of locally d.c. functions. *Mathematical Programming*, 41:195–227, 1988.

[24] Pey-Chun Chen, Hansen, P., Jaumard, B. and Tuy, H. Solution of the multisource weber and conditional weber problems by d.c. programming. *Operations Research*, 46(4):548–562, 1998.

[25] A.M. Rubinov. *Abstract convexity and global optimization*, volume 44 of *Nonconvex Optimization and its Applications.* Kluwer Academic Publishers, Dordrecht, 2000.

[26] Strekalovsky, A. and Tsevendorj, I. Testing the $\mathcal{R}$-strategy for a reverse convex problem. *Journal of Global Optimization*, 13:61–74, 1998.

[27] Tuy, H. *Convex analysis and global optimization.* Kluwer Academic Publishers, Dordrescht, first edition edition, 1998.

[28] Vial, J.-Ph. Strong and weak convexity of sets and functions. *Math. Oper.Res.*, 8:231–259, 1983.

10 Appendix

In Appendix we describe test problems used in the numerical experiments.

10.1 Test problems with non-Lipschitz objective functions

Problem 10.1

$$\begin{array}{rl} \text{minimize} & f(x,y) := -\sin(\sqrt{3x+2y+|x-y|}), \\ \text{subject to:} & 0 \le x \le 5, \\ & 0 \le y \le 5. \end{array} \tag{24}$$

The function $5(x^2+y^2)$, allows us to obtain a d.c. representations of the objective function f so DCECAM can be applied:

$$f(x,y) = 5(x^2+y^2) - (-f(x,y) + 5(x^2+y^2)).$$

The best solution found is $(0.29658\ldots, 0.62279\ldots)$ with $f^* = -0.99999825\ldots$.

Problem 10.2

$$\begin{array}{ll} \text{minimize} & \varphi(x) := a\sqrt{|1-x|} + |2-x|^3, \\ \text{subject to:} & 1 \le x \le 3, \end{array} \tag{25}$$

with $a = 0,9$ and $a = 1.5$. Then we have the convex functions

$$\begin{aligned} \varphi_1(x) &= |2-x|^3, \\ \varphi_2(x) &= -a\sqrt{|1-x|}. \end{aligned}$$

that satisfy

$$\varphi(x) = \varphi_1(x) - \varphi_2(x).$$

Problem 10.3

$$\begin{array}{ll} \text{minimize} & \phi(x) := -\log(x) + \min\{\sqrt{|1-x|}, (2-x)^3, \sqrt{|3-x|}\} \\ \text{subject to:} & 1 \le x \le 3, \end{array} \tag{26}$$

Then we have the convex functions

$$\phi_1(x) = 6x^2 - 12x + 8 + \max\{0, -x^3\} - \log(x),$$

and

$$\phi_2(x) = \max \left\{ \begin{array}{l} -\sqrt{|3-x|} + 6x^2 - 12x + 8 + \max\{0, -x^3\}, \\ -\sqrt{|1-x|} + 6x^2 - 12x + 8 + \max\{0, -x^3\}, \\ \max\{0, x^3\}) \end{array} \right\}$$

that satisfy

$$\phi(x) = \phi_1(x) - \phi_2(x).$$

Problem 10.4 By combining the above-mentioned functions we can obtain new function with n variables.

$$\begin{array}{ll} \text{minimize} & f(x) := \sum_{i=1}^n \varphi(x_i) \\ \text{subject to:} & 1 \le x_i \le 3, i = 1, \ldots, n \end{array} \tag{27}$$

and

$$\begin{array}{ll} \text{minimize} & f(x) := \sum_{i=1}^n \phi(x_i) \\ \text{subject to:} & 1 \le x_i \le 3, i = 1, \ldots, n. \end{array} \tag{28}$$

10.2 Test problems with Lipschitz objective functions

By using the convex function $g(x) := K\|x\|^2$ with $K > 0$, we can obtain a DC representation of the objective functions of the test problems as follows.

$$f(x) = (f(x) + g(x)) - g(x), \tag{29}$$

with K being a real number such that $f(x) + k\sum_{j=1}^n x_j^2$ is a convex function.

Problem 10.5 The class of test problems $HPTnXmY$.
The following class of test problems can be found in [18]:

$$\begin{array}{rl} \text{minimize} & -\sum_{i=1}^{m} 1/\left(\|x-a^i\|^2+c_i\right) \\ \text{subject to} & x \in \mathbb{R}^n, 0 \le x_j \le 10, j=1,\ldots,n \end{array} \tag{30}$$

where $a^i \in \{x \in \mathbb{R}^n : 0 \le x_j \le 10, 1 \le j \le n\}$ and $c_i > 0$. By using the convex function $k\left(\sum_{j=1}^{n} x_j^2\right)$ with $k > 0$, we can obtain a d.c. representation of the objective function in (30) as follows. Consider $f(x) = \sum_{i=1}^{m} f_i(x)$, with $f_i(x) := 1/\left(\|x-a^i\|^2+c_i\right)$ and $x \in \mathbb{R}^n$. Hence, we can write

$$f(x) = \left(f(x) + k\sum_{j=1}^{n} x_j^2\right) - \left(k\sum_{j=1}^{n} x_j^2\right), \tag{31}$$

with k a real number such that $f(x) + k\sum_{j=1}^{n} x_j^2$ is a convex function. The different instances of the test problem (30) are denoted by $HPTnXmY$ where X represents the dimension and Y means the number of local optimal solutions of the instance. Parameters are given in Table 5.

Table 5: Parameters for the test problem $HPTnXmY$

i	1	2	3	4	5	6	7	8	9	10
c_i	0.70	0.73	0.76	0.79	0.82	0.85	0.88	0.91	0.94	0.97
a_1^i	4.0	2.5	7.5	8.0	2.0	2.0	4.5	8.0	9.5	5.0

Problem 10.6 The class of test problems $TnXrY$.
Let $x \in \mathbb{R}^n$ be $x = (x_1, \ldots, x_n)$. A reduced version of the test problem

$$\begin{array}{rl} \text{minimize} & f(x) = \Pi_{i=1}^{n}(x_i^2 + c_i x_i) \\ \text{subject to} & -2 \le x_i \le 1, i = 1, \ldots, n, \end{array} \tag{32}$$

where $A \in \mathbb{R}^{m*n}$ and $b \in \mathbb{R}^m$, can be found in [27]. The names of the different instances of the test problem (32) are denoted by $TnXrY$, where X is the dimension and Y means the number of linear constraints of the instance. For numerical tests, we have chosen the instance $Tn2r4$ with the parameters $c_1 = 0.09$ and $c_2 = 0.1$. As before, by using the convex function $k\left(\sum_{j=1}^{n} x_j^2\right)$ different d.c. representations of the objective function can be obtained in the form:

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

We consider the values $k = 7.5$, $k = 8$ and $k = 8.5$.

Problem 10.7 The class of test problems $HPBr1$.
The problem

$$\begin{array}{rl} \text{minimize} & f(x,y) = \frac{1}{4}(x+y)^2 - \frac{1}{4}(x-y)^2 \\ \text{subject to:} & -2 \le x \le 3, \\ & -3 \le y \le 4, \end{array} \tag{33}$$

which will be denoted by $HPBr1$, is a nonconvex programming problem. The objective function of $HPBr1$ is an homogeneous polynomial of degree two with two variables (in this case it is a hyperbole). It is known that the d.c. representation of xy in (33) is the optimal. Alternative non-optimal d.c. representations of xy are:

(1) $xy = \frac{1}{2}(x+y)^2 - \frac{1}{2}(x^2+y^2)$, and

(2) $xy = \frac{1}{2}(x^2+y^2) - \frac{1}{2}(x-y)^2$.

Problem 10.8 The class of test problems $COSr0$
The problem

$$\begin{array}{rl} \text{minimize} & f(x,y) := 0.03(x^2+y^2) - cos(x)cos(y) \\ \text{subject to:} & -6 \le x \le 4, \\ & -5 \le y \le 2, \end{array} \tag{34}$$

which will be denoted by $COSr0$, is a multiextremal programming problem with minimizer $(0,0)$ and minimum -1. The function $k(x^2+y^2)$, $k > 0$ allows us to obtain many different d.c. representations of the objective function f:

$$f(x,y) = (f(x,y) + k(x^2+y^2)) - k(x^2+y^2).$$

Problem 10.9

$$\begin{array}{rl} \text{minimize} & f(x) := f_1(x) - f_2(x),\ x \in \mathbb{R}^4 \\ \text{subject to:} & -10 \le x_i \le 10,\ i = 1,2,3,4. \end{array} \tag{35}$$

Here

$$f_1(x) = |x_1 - 1| + 200\max\{0, |x_1| - x_2\} + 180\max\{0, |x_3| - x_4\} + |x_3 - 1|$$
$$+10.1(|x_2 - 1| + |x_4 - 1|) + 4.95|x_2 + x_4 - 2|,$$
$$f_2(x) = 100(|x_1| - x_2) + 90(|x_3| - x_4) + 4.95|x_2 - x_4|.$$

Problem 10.10

$$\begin{array}{rl} \text{minimize} & f(x) := f_1(x) - f_2(x),\ x \in \mathbb{R}^n \\ \text{subject to:} & -10 \le x_i \le 10,\ i = 1,\dots,n. \end{array} \tag{36}$$

Here

$$f_1(x) = |x_1 - 1| + 200\sum_{i=2}^{n} \max\{0, |x_{i-1}| - x_i\},$$

$$f_2(x) = 100\sum_{i=2}^{n}(|x_{i-1}| - x_i).$$