J Glob Optim (2009) 43:513–531
DOI 10.1007/s10898-008-9343-5

Improving the efficiency of DC global optimization methods by improving the DC representation of the objective function

Albert Ferrer · Juan Enrique Martínez-Legaz

Received: 19 June 2008 / Accepted: 12 August 2008 / Published online: 5 September 2008

Abstract There are infinitely many ways of representing a d.c. function as a difference of convex functions. In this paper we analyze how the computational efficiency of a d.c.optimization algorithm depends on the representation we choose for the objective function, and we address the problem of characterizing and obtaining a computationally optimal representation. We introduce some theoretical concepts which are necessary for this analysis and report some numerical experiments.

Keywords Dc representation · Dc program · Outer approximation · Branch and bound · Semi-infinite program

Mathematics Subject Classification (2000) 90C26 · 90C30

1 Introduction

Consider a programming problem with d.c. objective function and linear and convex constraints:

$$\begin{array}{ll} \text{minimize} & f(x) - h(x) \\ \text{subject to} & Ax \leq b, \\ & \varphi(x) \leq 0, \end{array} \tag{1}$$

where A is a real $m \times n$ matrix, $b \in \mathbb{R}^m$ and f, h and φ are proper convex functions on $\mathbb{R}^n$. By introducing an additional variable t the program (1) can be transformed into the equivalent convex minimization problem subject to an additional reverse convex constraint

A. Ferrer
Departament de Matemàtica Aplicada I, Universitat Politècnica de Catalunya, Barcelona, Spain
e-mail: alberto.ferrer@upc.edu

J. E. Martínez-Legaz (✉)
Departament d'Economia i d'Història Econòmica, Universitat Autonoma de Barcelona, Barcelona, Spain
e-mail: juanenrique.martinez@uab.es

$$\begin{array}{ll} \text{minimize} & f(x) - t \\ \text{subject to} & h(x) - t \geq 0, \\ & \varphi(x) \leq 0, \\ & Ax \leq b. \end{array} \tag{2}$$

Algorithms for solving deterministic reverse convex programs are described, for example, in [7] and [13]. 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 contrast, our algorithm, though designed in a similar spirit to algorithms used for solving reverse convex programs, has several differences and advantages. Among others, we will use 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 (see [4]).

Definition 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.

For the sake of clarity we next give a brief summary of the algorithm in the case of linear constraints (see [4] for the general case). Consider $T_0 := T(Z_0)$ an initial prism, with $Z_0 := [v_0^1, \ldots, v_0^{n+1}]$ a 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

$$P_0 \supset P_1 \supset \cdots \supset P_k \supset \ldots.$$

- *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 optimal value $\mu(T)$ and the optimal solution $(x(T), t(T))$ of the linear program

$$\begin{array}{ll} \text{maximize} & a^t x - t - \rho \\ \text{subject to} & (x, t) \in T \cap P, \end{array} \tag{3}$$

Table 1 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

where $T := T(Z)$ is a prism with $Z = [v^1, \ldots, v^{n+1}]$, P is the current convex polyhedron and $a^t x - t - \rho$ is the unique hyperplane passing through the points $(v^i, h(v^i))$, $i = 1, \ldots, n+1$. The optimal solution of (3) is the point of the polyhedron $T \cap P$ with the greatest distance to the hyperplane $a^t x - t - \rho$. When $\mu(T) \leq 0$ then $T(Z)$ is deleted. On the other hand, if a prism T with largest value $\mu(T)$ is selected for division,a refined partition of T is constructed and a new cut is added to the polyhedron P to obtain a new polyhedron. From the solution $(x(T), t(T))$ the new point $(\bar{x}, \bar{t})$ is obtained by setting $\bar{x} := x(T)$ 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 polyhedron

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

The cut $l(x, t) := (x - \bar{x})^T p - t + c$ is defined by means of a subgradient p of the function f at the point $x(T)$ and $c = h(x(T)) + \epsilon$. The procedure goes on until all generated prisms have been deleted (for details see [4]).

On the other hand, a d.c. function admits infinitely many representations as a difference of convex functions. Indeed, if a d.c. representation $f - h$ is available then by adding $\sigma(x) = k\|x\|^2$, $k > 0$, to both components we obtain a new d.c. representation, $(f + \sigma) - (h + \sigma)$. From a theoretical point of view σ adds more structure to the d.c. representation of the function, which is important in the context of duality (see [6]). Nevertheless, from a computational point of view adding σ is a drawback. For example, we can see in Tables 2–4 that when k increases then the number of linear subproblems to be solved usually increase as well as the number of iterations and the CPU time. The drawback comes from the solution of the linear subproblem (3). Using a representation $(f + \sigma) - (h + \sigma)$, the distances $\mu(T)$ are generally bigger than distances obtained by using the initial representation $f - g$, so the algorithm must perform more iterations and subdivisions. From a computational point of view, what should it mean that a d.c. representation is "better" that another one? The answer to this question is clear: that the given d.c. representation needs less CPU time (so it is more efficient) than the other one for finding an optimal value.

Some questions must be considered. Given a d.c. representation, is it possible to find another one that improves computational efficiency? Which is the best d.c. representation from a computational point of view, and how can it be obtained? Could these questions be connected with some suitable theoretical concept?

To compare two representations of a d.c. function f defined on a closed convex set $C \subset \mathbb{R}^n$, we need some ordering on the set $DC(f)$ of all d.c. representations of f. The following natural orderings can be considered: For $(g_1, h_1), (g_2, h_2) \in DC(f)$,

1) (g_1, h_1) is better than (g_2, h_2) if $g_2 - g_1 = h_2 - h_1 \geq 0$.
2) (g_1, h_1) is better than (g_2, h_2) if $g_2 - g_1 = h_2 - h_1$ is convex.

The question arises whether these orderings have computational implications, i.e., whether they yield d.c. representations that are better from a computational point of view.

Table 2 Computational results for the instance *HPTn2m*10

$Case$	ϵ	$D.c.(K)$	$Iter$	$Msdv$	$Tsdv$	$Mdepth$	$Obj.val.$	$Time$
1	ϵ_1	0.5	312	55	645	23	−2.1423	2.39
2	ϵ_2	0.5	354	55	732	28	−2.1423	3.17
3	ϵ_3	0.5	403	55	832	31	−2.1423	3.85
4	ϵ_1	5	371	55	662	26	−2.1424	3.34
5	ϵ_2	5	408	55	840	31	−2.1424	3.91
6	ϵ_3	5	449	55	925	33	−2.1424	4.50
7	ϵ_1	50	335	55	694	26	−2.1411	3.82
8	ϵ_2	50	342	55	708	26	−2.1411	3.93
9	ϵ_3	50	451	55	936	37	−2.1411	4.84

Table 3 Computational results for the instance *Tn2r*4

$Case$	ϵ	$D.c.(K)$	$Iter$	$Msdv$	$Tsdv$	$Mdepth$	$Obj.val.$	$Time$
1	ϵ_1	7.5	706	247	1547	12	−8.3882	9
2	ϵ_2	7.5	5304	1999	11395	16	−8.3882	650
3	ϵ_3	7.5	20000	14020	43560	19	−8.3882	16099
4	ϵ_1	8.0	867	312	1901	11	−8.3882	16
5	ϵ_2	8.0	6305	2424	13518	15	−8.3882	970
6	ϵ_3	8.0	20000	16384	43504	18	−8.3882	15855
7	ϵ_1	8.5	989	355	2175	12	−8.3882	22
8	ϵ_2	8.5	7332	2736	15708	15	−8.3882	1371
9	ϵ_3	8.5	20000	17653	43343	18	−8.3882	15711

Table 4 Computational results for the instance *COSr*0

$Case$	ϵ	$d.c.(K)$	$Iter$	$Msdv$	$Tsdv$	$Mdepth$	$Obj.val.$	$Time$
1	ϵ_1	0.5	532	180	1144	12	−0.9920	7
2	ϵ_2	0.5	1163	312	2478	15	−0.9999	29
3	ϵ_3	0.5	1948	344	4157	18	−0.9999	86
4	ϵ_1	1.0	1335	481	2905	13	−0.9981	41
5	ϵ_2	1.0	3579	1040	7678	16	−0.9998	362
6	ϵ_3	1.0	6230	1330	13369	19	−0.9999	1297
7	ϵ_1	1.5	2184	804	4747	13	−0.9964	122
8	ϵ_2	1.5	6858	2120	14702	17	−0.9997	1692
9	ϵ_3	1.5	12675	2745	27289	20	−0.9999	6535

Each of the above orderings induces a notion of efficiency in the usual way. We say that $(f_1, f_2) \in DC(f)$ is effiicient if for any better $(g_1, g_2) \in DC(f)$ it also holds that (f_1, f_2) is better than (g_1, g_2) (that is, both representations are actually equivalent from the point of view of the ordering in question). In the case of the first ordering this amounts to saying that $(f_1, f_2) = (g_1, g_2)$, while in the case of the second one this means that the function $g_1 - f_1 = g_2 - f_2$ is affine. This second notion of efficiency was studied in [11] in an abstract framework. It is less restrictive than the notion of efficiency based on the first ordering. Indeed, assume that $(f_1, f_2) \in DC(f)$ is efficient in the sense of the first ordering and let $(g_1, g_2) \in DC(f)$ be better than (f_1, f_2) in the sense of the second ordering. Then the function $f_1 - g_1$ is convex, so that it has an affine minorant h. Since $(g_1 + h, g_2 + h) \in DC(f)$, from the nonnegativity of $f_1 - (g_1 + h)$ and the efficiency of (f_1, f_2) we deduce that $f_1 = g_1 + h$, so that $g_1 - f_1 = -h$ is affine, which shows that (f_1, f_2) and (g_1, g_2) are equivalent in the sense of the second ordering.

Some drawbacks of these orderings are to be reported. First, given a d.c. representation we do not know how to obtain a better d.c. representation. Secondly, apart from some particular cases (e.g., when one only considers polyhedral functions or one variable functions), minimal d.c. representation in the sense of 1) or 2) do not necessarily exist (see [6]). In particular, minimal representations in the sense of 1) do not exist for $C \neq \mathbb{R}^n$, since by the separation theorem for closed convex sets there is an affine function p that takes strictly positive values on C, and hence for every $(f_1, f_2) \in DC(f)$ the decomposition $(f_1 - p, f_2 - p) \in DC(f)$ is strictly better than (f_1, f_2).

Another atempt to define suitable decompositions of d.c. functions was made in [6], where the concept of normalized decomposition was introduced. One says that $(g, h) \in DC(f)$ is normalized if the infimum of h is equal to 0. However, as shown in that paper, even if we restrict ourselves to the class of normalized decompositions, optimal decompositions in the sense of 1) do not necessarily exist.

Notice that, as mentioned in [6], optimal d.c. representations should prefereably be searched for in restricted classes of functions depending on the algorithm used to solve the d.c. programming problem. Moreover, the question whether or not the optimal d.c. representation obtained is also the best d.c. representation from a computational viewpoint cannot be completely answered because of the infinity of representations of a d.c. function. Nevertheless, the concepts and procedures we will describe will give us a guide of how to obtain better d.c. representations in some cases.

In this paper we mainly consider polynomials and the algorithm described above; for other algorithmic approaches to d.c. optimization we refer to the survey [12]. We propose a new concept of decomposition, which is a particular case of the notion of *least deviation decomposition* (LDD) of [10] with respect to a pair of convex sets. Our proposed decomposition is optimal in a sense different from the ones considered above.

The problem of finding d.c. representations of polynomials was addressed in [4]. An additional motivation for using polynomials comes from the Stone-Weierstrass Theorem, which states that continuous functions on compact sets are uniform limits of sequences of polynomials.

We will introduce some necessary concepts to answer the above questions. In Sect. 2 we will state *the minimum norm problem,* which provides an abstract formulation of the problem of finding the least deviation decomposition of a polynomial. In Sect. 3 we will give a semi-infinite programming formulation of the minimum norm problem to obtain an optimal d.c. representation for an arbitrary polynomial. We will show how this d.c. representation improves the computational efficiency of the global optimization algorithm by reducing the number of iterations needed to find a global optimal solution. Section 4 addresses the issue of obtaining an initial feasible point. In Sect. 5 we report some numerical experiments, which show the efficiency of our minimal d.c. representations from a computational viewpoint. The conclusions of our analysis are summarized in Sect. 6.

2 The minimum norm problem

Let $\mathbb{R}_m[x_1, ..., x_n]$ and $H_k[x_1, ..., x_n]$, $k = 0, 1, \ldots, m$ be the vector spaces of real polynomials of degree less than or equal to m and of homogeneous polynomials of degree k, respectively, in the variables $x = (x_1, ..., x_n)$. Let B^m and BH^k be the usual bases of monomials in $\mathbb{R}_m[x_1, ..., x_n]$ and $H_k[x_1, ..., x_n]$, respectively. Each polynomial $z \in \mathbb{R}_m[x_1, ..., x_n]$ can be written in the form $z = \sum_{f \in B^m} a_f f$. We will endow these vector spaces with the Euclidean norm $\|z\| := (\sum_{f \in B^m} a_f^2)^{1/2}$.

Let $C \subset \mathbb{R}^n$ be a closed convex set and let $K^m(C)$ and $KH^k(C)$ be the closed convex cones of polynomials in $\mathbb{R}_m[x_1, ..., x_n]$ and $H_k[x_1, ..., x_n]$, respectively, that are convex on C. In the next proposition we consider the case when $C = \mathbb{R}^n$; we exclude the trivial case $m = 1$.

Proposition 2.1 *Let $m \geq 2$. The cone $K^m(\mathbb{R}^n)$ is reproducing, that is,*

$$\mathbb{R}_m[x_1, ..., x_n] = K^m(\mathbb{R}^n) - K^m(\mathbb{R}^n),$$

if and only if m is even.

Proof If m is odd, no polynomial of degree m is convex (as it does not admit any affine minorant) and hence it is impossible to decompose it as a difference of two polynomials in $\mathbb{R}_m[x_1, ..., x_n]$. Assume now that m is even. Since the set of polynomials in $\mathbb{R}_m[x_1, ..., x_n]$ that are powers of linear functions spans the whole of $\mathbb{R}_m[x_1, ..., x_n]$ (see [2]) and such powers are convex when the exponent is even, we only need to prove the decomposability of a polynomial of the type l^k, with l linear and k odd, as a diference of two convex polynomials of degree $k+1$. Given that compositions of convex functions with linear mappings are convex, it will be enough to consider one variable monomials. Since the second derivative of the one variable function $t^{k+1} + t^k$ is bounded from below, there is a quadratic form αt^2, with $\alpha > 0$, such that $t^{k+1} + t^k + \alpha t^2$ is convex, and therefore $t^k = \left(t^{k+1} + t^k + \alpha t^2\right) - \left(t^{k+1} + \alpha t^2\right)$ is the difference of two convex polynomials of degree $k+1$. □

Corollary 2.1 *Let $C \subset \mathbb{R}^n$ be a closed convex set and $m \geq 2$ be an even number. Then $K^m(C)$ is reproducing.*

Proof This is an immediate consequence of Prop. 2.1, since $K^m(C) \supseteq K^m(\mathbb{R}^n)$. We next consider the case when C is bounded. □

Proposition 2.2 *Let $C \subset \mathbb{R}^n$ be a compact convex set. Then, for every $m \geq 2$, the convex cone $K^m(C)$ is reproducing.*

Proof Using the same arguments as in the proof of Prop. 2.1 we see that it is enough to consider compositions of one variable powers t^k with linear functions and that such compositions are d.c. on C (since the second derivative of t^k is bounded below on the image of C under any linear function, as this image is a closed interval). □

Assuming that $K^m(C)$ is reproducing, there are infinitely many representations of a given polynomial $z \in \mathbb{R}_m[x_1, ..., x_n]$ as a difference of two convex polynomials on C. Indeed, if $z = y_1 - y_2$, with $y_1, y_2 \in K^m(C)$, then we can also write, for instance, $z = (y_1 + d) - (y_2 + d)$ for an arbitrary $d \in K^m(C)$ and of course we have $y_1 + d, y_2 + d \in K^m(C)$. Thus the question arises how to find, among the infinitely many representations of z, one which is optimal from a computational point of view (that is, for its use in a d.c. optimization algorithm). A theoretical approach to this optimal representation problem was proposed in [10] in the abstract setting of normed spaces, as an alternative to the decomposition theory based on efficiency mentioned in the Introduction; we next recall the basic ideas developed in that paper.

Let $(\boldsymbol{E}, \|\cdot\|)$ be a normed space, $K \subset \boldsymbol{E}$ a reproducing convex cone and (y_1, y_2), $(w_1, w_2) \in K \times K$ be two representations of z as differences of elements in K, that is, $y_1 - y_2 = z = w_1 - w_2$. We define

$$(y_1, y_2) \text{ is better than } (w_1, w_2) \text{ with respect to } \|.\| \Leftrightarrow \|y_1 + y_2\| \leq \|w_1 + w_2\|.$$

A representation of z is *minimal* if it is better that any other representation of z. Thus finding a minimal representation of z amounts to solving a *minimum norm problem*. Given $z \in \boldsymbol{E}$ and a representation $(y_1, y_2) \in K \times K$ of z, consider the vector $v := y_1 + y_2 \in K$. We can write

$$y_1 = \frac{z+v}{2} \quad \text{and} \quad y_2 = \frac{v-z}{2}, \tag{4}$$

so $v = -z + 2y_1 = z + 2y_2$ and we have $v \in (-z + 2K) \cap (z + 2K)$. Thus, the minimum norm problem can be expressed as follows: *given* $z \in \boldsymbol{E}$, *find* $v \in (-z + 2K) \cap (z + 2K)$ *with minimum norm*:

$$\text{minimize } \{\|v\| : v \in (-z + 2K) \cap (z + 2K)\}. \tag{5}$$

Hence, using (4), the optimal solution v^* of problem (5) yields an optimal representation $z = y_1^* - y_2^*$, where

$$y_1^* = \frac{z+v^*}{2} \quad \text{and} \quad y_2^* = \frac{v^*-z}{2}.$$

According to the terminology of [10], the pair (y_1^*, y_2^*) is called a *least deviation decomposition (LDD)* of z.

When a LDD of a polynomial z is difficult to obtain, the equality

$$\mathbb{R}_m [x_1, \ldots, x_n] = \bigoplus_{k=0}^{m} H_k [x_1, \ldots, x_n], \tag{6}$$

allows us to obtain an alternative dc representation of the polynomial z by using LDDs of its homogeneous summands. This new dc representation of z will not generally be optimal but will often improve upon an initially given dc representation.

To solve the minimum norm problem in the case of the Euclidean norm, in the next section we will transform it into an equivalent semi-infinite quadratic programming problem with linear constraints.

3 A semi-infinite formulation of the minimum norm problem

A peculiarity of the minimum norm problem (5) in the case of the Euclidean norm is that it can be transformed into an equivalent semi-infinite quadratic programming problem with linear constraints. The feasible set of the problem (5), with $z = \sum_{f \in B^m} a_f f$, is the set of polynomials $v = \sum_{f \in B^m} v_f f$ such that $v \pm z \in K(C)$, i.e. such that $v \pm z$ are convex on C. Assuming that C has a nonempty interior, this amounts to imposing the Hessian matrices $\nabla^2(v \pm z)(x) = \sum_{f \in B^m} (v_f \pm a_f) \nabla^2 f(x)$ to be positive semidefinite for $x \in C$, that is,

$$\lambda^t \nabla^2(v \pm z)(x)\lambda \geq 0, \;\; \forall \lambda \in S^n, \; \forall x \in C, \tag{7}$$

where $S^n = \{x \in \mathbb{R}^n : \|x\| = 1\}$. Thus problem (5) can be equivalently formulated as the semi-infinite quadratic programming problem

$$\begin{cases} \text{minimize} & \|v\|^2 = \sum_{f \in B^m} v_f^2 \\ \text{subject to} & \lambda^t \sum_{f \in B^m} (v_f \pm a_f) \nabla^2 f(x) \lambda \geq 0, \;\; \forall \lambda \in S^n \;\; , \forall x \in C, \end{cases} \tag{8}$$

whose constraint set is parameterized by $(x, \lambda) \in C \times S^n$.

In practical applications the set C will usually be of the form $C = \prod_{i=1}^n [r_i, t_i]$, with $r_i < t_i$.

Example 3.1 (Determination of the constraint set) Let $z(x, y) = xy + 3x^2 y$ and $C = [5, 20] \times [5, 20]$. In order to determine the constraint set of the semi-infinite formulation of the minimum norm problem we consider the usual bases in $H_2[x, y]$ and $H_3[x, y]$,

$$B_2 := \{f_1, \; f_2, \; f_3\}, f_1(x, y) := x^2, \;\; f_2(x, y) := xy, \;\; f_3(x, y) := y^2,$$

and

$$B_3 := \{f_4, \; f_5, \; f_6, \; f_7\}, f_4(x, y) := x^3, \;\; f_5(x, y) := x^2 y, \;\; f_6(x, y) := xy^2, \\ f_7(x, y) := y^3.$$

Since z has no linear part and the subspaces $H_1[x, y]$, $H_2[x, y]$ and $H_3[x, y]$ are mutually orthogonal, if we use the Euclidean norm we do not need to consider polynomials of degree 1 in our formulation. The set $B_2 \cup B_3$ is a base for $H_2[x, y] \oplus H_3[x, y]$; the coordinates of the polynomial $z(x, y)$ in this basis are $(0, \; 1, \; 0, \; 0, \; 3, \; 0, \; 0)$. For a polynomial $v \in H_2[x, y] \oplus H_3[x, y]$ we write $v = \sum_{i=1}^7 v_i f_i$. The Hessian matrices of the functions $v \pm z$ are

$$\nabla^2(v \pm z)(x, y) = \begin{pmatrix} 6v_4 x + 2(v_5 \pm 3)y + 2v_1 & 2(v_5 \pm 3)x + 2v_6 y + (v_2 \pm 1) \\ 2(v_5 \pm 3)x + 2v_6 y + (v_2 \pm 1) & 2v_6 x + 6v_7 y + 2v_3 \end{pmatrix}.$$

Imposing the positive-semidefinitiness condition to these matrices we get

$$2\lambda_1^2 v_1 + 2\lambda_1\lambda_2 v_2 + 2\lambda_2^2 v_3 + 6x\lambda_1^2 v_4 + 2(y\lambda_1^2 + 2x\lambda_1\lambda_2)v_5 \\ +2(x\lambda_2^2 + 2y\lambda_1\lambda_2)v_6 + (6y\lambda_2^2)v_7 \pm (6y\lambda_1^2 + 12x\lambda_1\lambda_2 + 2\lambda_1\lambda_2) \geq 0$$

for λ_1 and λ_2 such that $\lambda_1^2 + \lambda_2^2 = 1$ and $\lambda_2 \geq 0$; we can equivalently write

$$\lambda_1^2 a + \lambda_1\lambda_2 b + \lambda_2^2 c + 3x\lambda_1^2 d + (y\lambda_1^2 + 2x\lambda_1\lambda_2)e \\ +(x\lambda_2^2 + 2y\lambda_1\lambda_2)f + 3y\lambda_2^2 g - \mid 3y\lambda_1^2 + 6x\lambda_1\lambda_2 + \lambda_1\lambda_2 \mid \geq 0.$$

The parameters λ_1 and λ_2 can be generated by considering the new parameter $\omega \in [0, \pi[$, setting $\lambda_1 = \cos\omega$ and $\lambda_2 = \sin\omega$.

Remark 3.1 In the general n variables case, the parameters $\lambda_1, \lambda_2, \ldots, \lambda_n$ satisfying $\lambda_1^2 + \lambda_2^2 + \cdots + \lambda_n^2 = 1$, $\lambda_1 \geq 0$ can be generated by using spherical coordinates:

$$\begin{cases} \lambda_1 & = & \sin\omega_1 \sin\omega_2 \ldots \sin\omega_{n-2} \sin\omega_{n-1}, \\ \lambda_2 & = & \cos\omega_1 \sin\omega_2 \ldots \sin\omega_{n-2} \sin\omega_{n-1}, \\ \ldots & \ldots & \ldots\ldots\ldots\ldots \qquad , \\ \lambda_{n-2} & = & \cos\omega_{n-3} \sin\omega_{n-2} \sin\omega_{n-1}, \\ \lambda_{n-1} & = & \cos\omega_{n-2} \sin\omega_{n-1}, \\ \lambda_n & = & \cos\omega_{n-1}, \end{cases} \tag{9}$$

with $\omega_i \in [0, \pi[$, $i = 1, ..., n-1$.

The algorithm we will use to solve the semi-infinite quadratic programming problem (8) is an adaptation of the semi-infinite linear programming methods described in [9] and [14], which use interior point techniques. The algorithm needs an initial feasible point at which all the constraints are satisfied as strict inequalities; in the next section we explain how such a point can be obtained.

4 Obtention of an initial feasible point

In this section we describe a method for obtaining an initial feasible point at which all the inequality constraints are satisfied strictly. In the following, we will assume that the set C in (8) is convex and compact. Without loss of generality (by applying a translation if necessary), we will further assume that $C \subset IR^p_{++}$.

Consider a nonhomogeneous polynomial $z = \sum_{i=1}^k z_i$, with $z_i \in H_{n_i}[x_1, \ldots, x_p]$, $2 \leq n_1 < \cdots < n_k$. To obtain an initial strictly interior feasible point, we first express each z_i in the basis $\mathcal{U}$ of $H_n[x_1, \ldots, x_p]$ consisting of the polynomials

$$p_\alpha(x) = (\alpha_1 x_1 + \cdots + \alpha_p x_p)^n, \quad \alpha = (\alpha_1, \ldots, \alpha_p) \in C(n,\ p),$$

$C(n,\ p)$ being the set of *p-compositions* of n, that is, the set of p-tuples $\alpha = (\alpha_1, \ldots, \alpha_p)$ of nonnegative integers α_i such that $\alpha_1 + \cdots + \alpha_p = n$. This family of polynomials is indeed a basis of $H_n[x_1, \ldots, x_p]$ (see Cor. 1.2 in [1]).

We thus write

$$z_i = \sum_{\alpha \in P_i} \lambda^i_\alpha p_\alpha - \sum_{\alpha \in N_i} (-\lambda^i_\alpha) p_\alpha,$$

with P_i and N_i being the sets of α such that $\lambda^i_\alpha > 0$ and $\lambda^i_\alpha < 0$, respectively. This is a d.c. representation of z_i, which can be computed by using the algorithm proposed in the Appendix of [3]. In the same reference this algorithm was implemented using the MAPLE Symbolic Calculator. From the above representation we define the convex polynomial

$$w_i := \sum_{\alpha \in P_i} \lambda^i_\alpha p_\alpha + \sum_{\alpha \in N_i} (-\lambda^i_\alpha) p_\alpha.$$

It is easy to see that the polynomial $w = \sum_{i=1}^k w_i$ is a feasible point for (8). Finally, by taking $v = w + q$, with $q(x) = x_1^2 + \cdots + x_p^2$, we get a feasible point at which all the constraints are satisfied strictly.

5 Test instances and numerical results

In this section we report the numerical experiments we have made on some test problems.

First, in Tables 2–4 we present the results obtained on problems $HPTnXmY$, $TnXrY$ and $COSr0$, which illustrate the drawback of adding a quadratic term $\sigma(x) = k\|x\|^2, k > 0$, to both terms of a d.c. representation. In these tables *Case* is the number of the case instance, ϵ is the precision, $d.c.(K)$ is a nonnegative real number k such that $f(x) + k\left(\sum_{j=1}^n x_j^2\right)$ is a convex function (f being the nonconvex objective function of the current instance), *Iter* indicates the number of iterations required, *Msdv* indicates the maximum number of subdivisions that have been simultaneously active, *Tsdv* indicates the total number of subdivisions performed by the algorithm, *Mdepth* indicates the maximum depth reached for the subdivision procedure, *Obj.Val* indicates the optimum obtained by the algorithm, and *Time* is the CPU time in seconds.

Then in Tables 6 and 7 we present the results obtained on some test problems by using the optimal, or the best obtained, d.c. representation of the objective function. In these tables the running CPU time is reported, *Iter* is the number of iterations of the program needed to confirm the global minimum, *Norm* indicates the norm of the corresponding d.c. representation, and *DC* represents the kind of d.c. representation of the objective function: the optimal, or the best obtained, d.c. representation is indicated by *opt*, and the non optimal representations are noted (1) and (2).

5.1 The instance $HPTnXmY$

The following class of test problems can be found in [8] and turns out to be rich enough to produce representative numerical results. We seek an ϵ-solution of

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

where $a^i \in \{x \in \mathbb{R}^n : 0 \le x_j \le 10, 1 \le j \le n\}$ and $c_i > 0$. The initial simplex S_0 for this class of test problems is

$$S_0 = \left\{ x \in \mathbb{R}^n_+ : \sum_{j=1}^{n} x_j \le 10n \right\}.$$

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 (10) 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 $x \in \mathbb{R}^n$. We can write

$$f(x) = \left(f(x) + k\sum_{j=1}^{n} x_j^2 \right) - \left(k\sum_{j=1}^{n} x_j^2 \right), \tag{11}$$

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 (10) are denoted by $HPTnXmY$, where X represents the dimension and Y means the number of local optimal solutions of the instance. In Table 2 we consider the instance $HPTn2m10$ with the parameters c_i, a^i_j, $i=1,\ldots,10$, $j=1,2$ from Table 1. Table 2 displays relevant results on the computational effort required to minimize the function f by means of our algorithm with different d.c. representations of the objective function, which are defined by the values $k=0.5$, $k=5$ and $k=50$ and the precisions $\epsilon_i = 10^{-i}$, $i=1,2,3$.

5.2 The instance $TnXrY$

Let $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$. A reduced version of the test problem

$$\begin{array}{ll} \text{minimize} & f(x) = \Pi_{i=1}^{n}(x_i^2 + c_i x_i) \\ \text{subject to} & Ax \leq b, \\ & d_i \leq x_i \leq e_i, i = 1, \ldots, n, \end{array} \tag{12}$$

where $A \in \mathbb{R}^{m*n}$ and $b \in \mathbb{R}^m$, can be found in [13]. The names of the different instances of the test problem (12) 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 same parameters $c_1 = 0.09$ and $c_2 = 0.1$ for the objective function in (12) and the feasible domain

$$\{(x_1, x_2) : Ax \leq b, -2 \leq x_i \leq 1, i = 1, 2\},$$

where

$$A = \begin{bmatrix} 1 & -1 \\ -1 & 1 \\ 1 & 1 \\ -1 & -1 \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} 1 \\ 2.5 \\ 1 \\ 3.5 \end{bmatrix}.$$

As before, by using the convex function $k\left(\sum_{j=1}^{n} x_j^2\right)$, k being a large enough positive number, many different d.c. representations of the objective function can be obtained by setting

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

Table 3 displays the numerical results for the instance $Tn2r4$ with different d.c. representations of the objective function, which are defined by the values $k = 7.5$, $k = 8$ and $k = 8.5$ and the precisions $\epsilon_i = 10^{-i}$, $i = 1, 2, 3$.

5.3 The instance $COSr0$

The instance

$$\begin{array}{ll} \text{minimize} & f(x, y) := 0.03(x^2 + y^2) - \cos x \cos y \\ \text{subject to} & -6 \leq x \leq 4, \\ & -5 \leq y \leq 2, \end{array} \tag{13}$$

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 as follows:

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

Table 4 displays the results of minimizing this function with different d.c. representations of the objective function, which are defined by the values $k = 0.5$, $k = 1$ and $k = 1.5$ and the precisions $\epsilon_i = 10^{-i}$, $i = 1, 2, 3$.

5.4 Polynomial instances

By using the mentioned semi-infinite algorithm we have obtained the optimal d.c. representations of the objective functions of instances *HPBr*1, *HOM*3*r*2 and *POL3r*2.

5.4.1 The instance HPBr1

The instance

$$\begin{array}{ll}\text{minimize} & xy\\ \text{subject to} & x-y\le 5.7,\\ & -2\le x\le 3,\\ & -3\le y\le 4\end{array}$$

is named $HPBr1$. An initial d.c. representation of the objective function is

$$xy=\frac{1}{2}(x+y)^2-\frac{1}{2}(x^2+y^2). \tag{14}$$

The obtained optimal solution of the minimum norm program is

$$v_1^*(x,y)=0.5001x^2+0.0xy+0.4999y^2,$$

with optimal value $\|v_1^*\|^2=0.5$ and optimal d.c. representation

$$xy=\frac{1}{4}(x+y)^2-\frac{1}{4}(x-y)^2.$$

5.4.2 The instance HOM3r2

The instance

$$\begin{array}{ll}\text{minimize} & 3x^2y\\ \text{subject to} & x-y\le 1,\\ & -x-y\le 2.5,\\ & 0.5\le x\le 2,\\ & 2\le y\le 4\end{array}$$

will be named $HOM3r2$. A d.c. representation of the objective function is

$$3x^2y=0.5(2x+y)^3+0.5y^3-(3x^3+(x+y)^3). \tag{15}$$

The obtained optimal solution of the minimum norm program is

$$v^*(x,y)=2.6247x^3+1.0377x^2y+0.6745xy^2+1.7551y^3,$$

with optimal value $\|v^*\|^2=11.534$ and optimal d.c. representation

$$3x^2y=y_1^*(x,y)-y_2^*(x,y),$$

in which

$$\begin{aligned}y_1^*(x,y)&=1.31235x^3+2.0188x^2y+.33725xy^2+.87755y^3,\\ y_2^*(x,y)&=1.31235x^3-.9811x^2y+.33725xy^2+.87755y^3.\end{aligned}$$

5.4.3 The instance POL3r2

The instance

$$\begin{array}{ll} \text{minimize} & xy + 3x^2y \\ \text{subject to} & x - y \le 1, \\ & -x - y \le 2.5, \\ & 0.5 \le x \le 2, \\ & 2 \le y \le 4 \end{array}$$

will be named $POL3r2$. From (14) and (15) we obtain a d.c. representation of its objective function, and the optimal solution from the minimum norm program is

$$v^*(x, y) = .0932x^2 - .0110xy + .0611y^2 + 2.6291x^3 + 1.0286x^2y + .6860xy^2 \\ +1.7550y^3$$

with optimal value $\|v^*\|^2 = 11.534$ and optimal d.c. representation

$$xy + 3x^2y = y_1^*(x, y) - y_2^*(x, y),$$

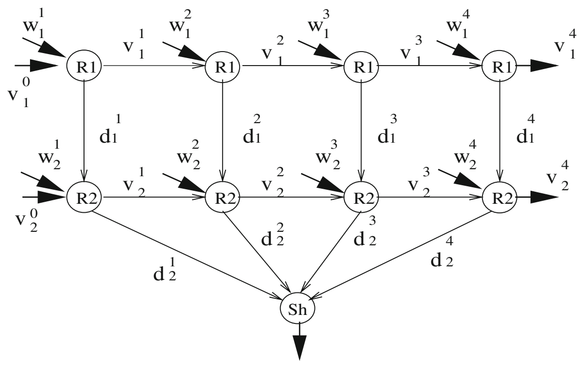
where

$$\begin{aligned} y_1^*(x, y) &= .0466x^2 + .4945xy + .03055y^2 + 1.31455x^3 + 2.0143x^2y + .3430xy^2 \\ &\quad +.8775y^3, \\ y_2^*(x, y) &= .0466x^2 - .5055xy + .03055y^2 + 1.31455x^3 - .9857x^2y + .3430xy^2 \\ &\quad +.8775y^3. \end{aligned}$$

5.4.4 The hydroelectric generation problem

Given a short-term time period, one wishes to find values for each time interval in the period so that the demand of electricity consumption for each time interval can be satisfied and the generation cost of thermal units is minimized subject to these and some other suitable constraints. The model we consider contains the replicated hydronetwork through which the temporary evolution of the reservoir system is represented (see [3] for additional information). Figure 1 shows a network with only two reservoirs and a time period subdivided into

Fig. 1 Four intervals and two reservoirs replicated hydronetwork



four intervals. We consider N_e reservoirs, $j = 1, \ldots, N_e$, and N_t time intervals, $i = 1 \ldots N_t$. Our model consists of the following ingredients:

- The variables are the water discharges d_j^i from reservoir j over the ith interval and the volume stored v_j^i in reservoir j at the end of the ith time interval.
- In each time interval i, the water discharge from reservoir R_1 to reservoir R_2 establishes a link between the reservoirs.
- The volume stored at the end of the time interval i and the volume stored at the beginning of the time interval $i+1$ are the same on each reservoir R_j; this establishes a link between the time intervals i and $i+1$ for each reservoir.
- The volumes stored at the beginning and at the end of the time period are known (they are not variables). Acceptable forecasts for electricity consumption l^i and for natural water inflow w_j^i into the reservoirs of the hydrogeneration system at each interval are available. An important assumption in our formulation is that the power hydrogeneration function h_j^i at the reservoir j over the i^{th} interval can be approximated by a polynomial function of degree 4 in the variables v_j^{i-1}, v_j^i and d_j^i (see [3]):

$$\begin{aligned} h_j^i(v_j^{i-1}, v_j^i, d_j^i) \;=\; & \mathsf{k}_j^i d_j^i \Big[s_{vd} + \tfrac{s_{vl}}{2}(v_j^{i-1} + v_j^i) + \tfrac{s_{vq}}{3}(v_j^i - v_j^{i-1})^2 \\ & + s_{vq} v_j^{i-1} v_j^i + \tfrac{s_{vc}}{4}((v_j^{i-1})^2 + (v_j^i)^2)(v_j^{i-1} + v_j^i) \\ & - s_{dl} d_j^i - s_{dq}(d_j^i)^2 \Big], \end{aligned} \tag{16}$$

where k_j^i (efficiency and unit conversion coefficient), s_{vd}, s_{vl}, s_{vq}, s_{vc}, s_{dl} and s_{dq} are technological coefficients, which depend on each reservoir. The objective function, to be minimized, is the generation cost of thermal units:

$$f(\ldots, v_j^{i-1}, v_j^i, d_j^i, \ldots) = \sum_{i=1}^{N_t} c_i \left(l^i - \sum_{j=1}^{N_e} h_j^i(v_j^{i-1}, v_j^i, d_j^i) \right). \tag{17}$$

The linear constraints are the flow balance equations at all nodes of the network:

$$v_j^i - v_j^{i-1} - d_{j-1}^i + d_j^i = w_j^i \quad j = 1, ..., N_e,\ i = 1, ..., N_t.$$

The nonlinear constraints are the thermal production with generation bounds:

$$\underline{g} \le l^i - \sum_{j=1}^{N_e} h_j^i(v_j^{i-1}, v_j^i, d_j^i) \le \overline{g} \quad i = 1, ..., N_t. \tag{18}$$

There are positive bounds on all variables:

$$\begin{aligned} &\underline{d_j} \le d_j^i \le \overline{d_j} \quad j = 1, ..., N_e, \ \ i = 1, ..., N_t \\ &\underline{v_j} \le v_j^i \le \overline{v_j} \quad j = 1, ..., N_e,\ i = 1, ..., N_t - 1. \end{aligned}$$

Table 5 Characteristics of generation systems

Problem	Nodes N_e	Intervals N_t	Dimension $N_e(2N_t-1)$	Linear const. N_eN_t	Nonlinear const. N_t
c2e02i	2	2	6	4	2
c2e03i	2	3	10	6	3
c4e03i	4	3	20	12	3

Thus our problem is

$$\begin{array}{lll}\text{minimize} & \sum_{i=1}^{N_t} c_i \left(l^i - \sum_{j=1}^{N_e} h_j^i(v_j^{i-1}, v_j^i, d_j^i)\right) & \\ \text{subject to} & \underline{g} \leq l^i - \sum_{j=1}^{N_e} h_j^i(v_j^{i-1}, v_j^i, d_j^i) \leq \overline{g}, & i = 1, ..., N_t, \\ & v_j^i - v_j^{i-1} - d_{j-1}^i + d_j^i = w_j^i, & j = 1, ..., N_e, \\ & & i = 1, ..., N_t, \\ & \underline{d_j} \leq d_j^i \leq \overline{d_j}, & j = 1, ..., N_e, \\ & & i = 1, ..., N_t, \\ & \underline{v_j} \leq v_j^i \leq \overline{v_j} & j = 1, ..., N_e, \\ & & i = 1, ..., N_t - 1. \end{array} \tag{19}$$

This model has the following useful properties:

1. It is easy to generate problems of different sizes (Table 5) and with different degrees of nonconvexity, depending on the efficiency and unit conversion coefficient, on whether the thermal units can satisfy all the demand of electricity during every time interval and on the water inflows.
2. The objective function and the nonlinear constraints are polynomial functions.
3. The linear constraints are the flow balance equations at the nodes of a network.

5.4.5 A d.c. formulation of the hydroelectric generation program

Let

$$h_j^i(v_j^{i-1}, v_j^i, d_j^i) = f_j^i(v_j^{i-1}, v_j^i, d_j^i) - g_j^i(v_j^{i1}, v_j^i, d_j^i),$$

be a d.c. representation of the power hydrogeneration function, where $f_j^i(v_j^{i-1}, v_j^i, d_j^i)$ and $g_j^i(v_j^{i1}, v_j^i, d_j^i)$ are convex functions defined on a convex set which contains the feasible domain of program (19). Then, by defining for every $i = 1, ..., N_t$ the convex functions

$$F^i(\ldots, v_j^{i-1}, v_j^i, d_j^i, \ldots) = c^i \left(l^i + \sum_{j=1}^{N_e} g_j^i(v_j^{i-1}, v_j^i, d_j^i)\right) \tag{20}$$

and

$$G^i(\ldots, v_j^{i-1}, v_j^i, d_j^i, \ldots) = c^i \sum_{j=1}^{N_e} f_j^i(v_j^{i-1}, v_j^i, d_j^i), \tag{21}$$

and using these expressions to define

$$F(\ldots, v_j^{i-1}, v_j^i, d_j^i, \ldots) = \sum_{i=1}^{N_t} F^i(\ldots, v_j^{i-1}, v_j^i, d_j^i, \ldots) \tag{22}$$

and

$$G(\ldots, v_j^{i-1}, v_j^i, d_j^i, \ldots) = \sum_{i=1}^{N_t} G^i(\ldots, v_j^{i-1}, v_j^i, d_j^i, \ldots), \tag{23}$$

d.c. representations of all functions in (19) are obtained. We further define $n = N_e(2N_t - 1)$, $m = N_e N_t$ and $x = (\ldots, v_j^{i-1}, v_j^i, d_j^i, \ldots) \in \mathbb{R}^n$; thus, by expressing the linear constraints in the form $Ax = b$, with $A \in \mathbb{R}^{m*n}$ and $b \in \mathbb{R}^m$, program (19) has the structure

$$\begin{array}{ll} \text{minimize} & F(x) - G(x) \\ \text{subject to} & (G^i(x) + c^i \underline{g}) - F^i(x) \le 0 \quad i = 1, ..., N_t, \\ & F^i(x) - (G^i(x) + c^i \overline{g}) \le 0 \quad i = 1, ..., N_t, \\ & Ax = b, \\ & \underline{x} \le x \le \overline{x}, \end{array} \tag{24}$$

where $\underline{x}$, $\overline{x} \in \mathbb{R}^n$. After renumbering the variables if necessary, the matrix A in (24) can be written as $A = [B, N]$, where B is a non singular square matrix (see [5] and references therein). Let y and z be the variables corresponding to the matrices B and N, respectively. Then, the solutions of $Ax = b$ are those $x = (y, z)$ with $y = B^{-1}(b - Nz)$, so that it is possible to reduce the size of the d.c. program (24) by defining the functions $\varphi_1(z) = F(B^{-1}(b - Nz), z)$, $\varphi_2(z) = G(B^{-1}(b - Nz), z)$, $\varphi_1^i(z) = F^i(B^{-1}(b - Nz), z)$ and $\varphi_2^i(z) = G^i(B^{-1}(b - Nz), z)$. By using these functions in (24) we obtain the following equivalent d.c. program of reduced size

$$\begin{array}{ll} \text{minimize} & \varphi_1(z) - \varphi_2(z) \\ \text{subject to} & (\varphi_2^i(z) + c^i \underline{g}) - \varphi_1^i(z) \le 0 \quad i = 1, ..., N_t, \\ & \varphi_1^i(z) - (\varphi_2^i(z) + c^i \overline{g}) \le 0 \quad i = 1, ..., N_t, \\ & \underline{b} \le Mz \le \overline{b}, \\ & \underline{z} \le z \le \overline{z}, \end{array} \tag{25}$$

where $M = B^{-1}N$ and $\underline{b}$, $\overline{b}$, $\underline{z}$ and $\overline{z}$ are defined by $\left(B^{-1}b - \overline{b}, \underline{z}\right) = \underline{x}$ and $\left(B^{-1}b - \underline{b}, \overline{z}\right) = \overline{x}$.

By defining the closed convex sets

$$\Omega = \left\{ (z, t) : \begin{array}{ll} \varphi_1(z) + \varphi_2(z) + (\varphi_2^i(z) - \varphi_1^i(z) + c^i \underline{g}) - t \le 0, & i = 1, ..., N_t, \\ \varphi_1(z) + \varphi_2(z) + (\varphi_1^i(z) - \varphi_2^i(z) - c^i \overline{g}) - t \le 0, & i = 1, ..., N_t, \\ \underline{b} \le Mz \le \overline{b}, & \\ \underline{z} \le z \le \overline{z} & \end{array} \right\}$$

and

$$\Delta = \{(z, t) : \ \varphi_1(z) + \varphi_2(z) - t \le 0\},$$

Table 6 Results and CPU requirements of problems solved ($\epsilon = 0.001$)

Instance	*DC*	*Norm*	*Iter*	*Obj.Val*	*CPU time*
$HOM2r1$	*opt*	0.707	32	−8.1224	0.07
	(1)	1.732	54	−8.1224	0.12
	(2)	1.732	163	−8.1224	0.77
$HOM3r2$	*opt*	3.339	150	1.4999	3.41
	(1)	13.601	601	1.5000	69.78
$POL3r2$	*opt*	3.396	99	2.5000	1.44
	(1)	13.711	287	2.5000	12.75

we reformulate (25) as the following equivalent reverse convex program:

$$\begin{array}{ll} \text{minimize} & 2\varphi_1(z) - t \\ \text{subject to} & (z,t) \in \Omega \setminus \text{int}\Delta. \end{array} \tag{26}$$

By using a prismatical subdivision process, this formulation allows for an advantageous adaptation of the *combined outer approximation cone splitting conical algorithm for canonical d.c. programming* as described in [4].

5.4.6 Characteristics of generation systems and computational results

The characteristics of the generation systems we have considered are described in Table 5. The names of the problems in Table 7 are of the form **c***n***e***m***i** and **c***n***e***m***iXYZ,** respectively, where **n, m, X, Y** and **Z** have the following meanings: **n** (one digit) is the number of nodes, **m** (two digits) is the number of time intervals, $\mathbf{X} = v$ when k^i_j in (16) depends on water discharges, and $\mathbf{X} = k$ if it is a constant, $\mathbf{Y} = 1$ when the thermal units satisfy the entire demand for electricity in every time interval, and $\mathbf{Y} = 0$ if this is not possible and $\mathbf{Z} = b$ when we solve the problem instance using the optimal d.c. representation of the power hydrogeneration functions, and $Z = a$ otherwise. The maximum number of iterations allowed in the global optimization algorithm is 5000, and the precision used is $\epsilon = 0.005$. In Table 7 *Iter* indicates the number of iterations required, *Obj.Val* is the optimal value obtained by the global optimization algorithm, and *CPU time* is the CPU time in seconds. To solve all problems we have used a computer SUN ULTRA 2 with 256 Mb of main memory and 2 CPUs of 200 MHz, SPCint95 7.88 and SPCfp95 14.70. Moreover, to compare the performances on different computers, problems number 17 and 18 in Table 7 have been solved with a computer Compaq AlphaServer HPC320: 8 nodes ES40 (4 EV68, 833 MHz, 64 KB/8 MB), 20 GB of main memory, 1.128 GB on disk and top speed of 53,31 Gflop/s, connected with Memory Channel II of 100 MB/s.

6 Conclusions

The superior computational efficiency of the method when we use the optimal d.c. representation is clearly shown in Tables 6 and 7. In all instances where we used it, the algorithm obtained better CPU times and needed fewer iterations than in problem instances where it was not used.

Table 7 Results and CPU requirements of problems solved ($\epsilon = 0.005$)

Num	*Instance*	*Iter*	*Obj.Val*	*CPU time*
1	c2e02ik0a	13	114.348	1.00
2	c2e02ik0b	11	114.347	0.77
3	c2e02iv0a	277	93.036	89.66
4	c2e02iv0b	21	93.266	3.44
5	c2e02ik1a	12	114.348	0.88
6	c2e02ik1b	10	114.347	0.67
7	c2e02iv1a	14	93.093	1.11
8	c2e02iv1b	13	93.092	0.95
9	c2e03ik0a	1062	262.795	2037.04
10	c2e03ik0b	1030	262.296	1409.44
11	c2e03iv0a	1077	250.016	1951.55
12	c2e03iv0b	904	262.498	1433.19
13	c2e03ik1a	1223	262.796	1972.71
14	c2e03ik1b	1117	262.796	1810.76
15	c2e03iv1a	1079	250.015	1454.01
16	c2e03iv1b	1056	250.015	1420.34
17	c4e03ik0a	5000	374.068	115359.00
				23064.80
18	c4e03ik0b	5000	374.067	70771.50
				13105.80

In the hydroelectric generation problem, on the instances with constant coefficient of efficiency and unit conversion, our algorithms appear to work well as they succeed to find a good solution. However, instances with a variable coefficient of efficiency yield less accurate optimal values, but all solutions are very near to the optimal. The superior computational efficiency reveals particularly high when we compare problem instances number 3 and 4. We also observe from instances number 17 and 18 that the CPU time can be reduced to one fifth by using the Compaq AlphaServer HPC320 computer. Note that the optimal d.c. representations of the power hydrogeneration functions give us more efficient d.c. representations of the functions in (19), but the latter are not necessarily the optimal d.c. representation of these functions, whose calculation would require the solution of a harder semi-infinite programming problem.

On the other hand, instance number 12 presents worse optimal values (but better CPU time) than instance number 11. During the execution of the algorithm we can arrive at an iteration in which the current ϵ -optimal solution obtained is an isolated point because it does not belong to any active subdivision. Moreover, no active subdivision contains a feasible point better than the incumbent, which is already an ϵ -optimal solution. In which iteration this happens depends on the d.c. representation of the objective function in (1) and on the linear subproblems (3) to be solved. Hence, we can deduce that the algorithm cannot improve the incumbent from the current iteration anymore. Of course, this is not an ideal situation, but it is not as bad as we might suppose. Actually, both ϵ-optimal solutions are very near to the optimal solution.

As the sizes of the problems increase, the problems become more and more difficult to solve. The size of the problem instances is a very serious limitation.

From our numerical experiments we conclude that by using optimal d.c. representations of the objective polynomial functions more efficient implementations for nonconvex optimization problems is obtained, both in the case of problems having a specific structure, as in the generation problem, and in the case when no such a structure exists.

As mentioned in the introduction, the question whether or not the given optimal d.c. representation obtained is the best d.c. representation from a computational viewpoint cannot be completely answered because of the infinity of representations of a d.c. function. Nevertheless, the concepts and procedures we have described give us a guide of how to obtain better d.c. representations in some cases.

Acknowledgment We are grateful to an anonymous referee for helpful remarks on an earlier version of this paper. The research of Albert Ferrer has been partially supported by the Ministerio de Ciencia y Tecnología, Project MCYT, DPI 2005-09117-C02-01. The research of Juan Enrique Martínez-Legaz has been partially supported by the Ministerio de Ciencia y Tecnología, Project MTM2005-08572-C03-03, and by the Barcelona Economics Program of XREA.

References

1. Brunat, J.M., Montes, A.: The power-composition determinant and its application to global optimization. SIAM J. Matrix Anal.Appl. **23**(2), 459–471 (2001)
2. Chambadal, L., Ovaert, J.L.: Algèbre Linéaire et Algèbre Tensorielle. Dunod Université, Dunod, Paris (1968)
3. Ferrer, A.: Representation of a polynomial function as a difference of convex polynomials, with an application. Lect. Notes Econ. Math. Syst. **502**, 189–207 (2001)
4. Ferrer, A.: Applying global optimization to a problem in short-term hydrotermal scheduling. Nonconvex Optim. Appl. **77**, 263–285 (2005)
5. Heredia, F.J., Nabona, N.: Optimum short-term hydrothermal scheduling with spinning reverse through network flows. IEEE Trans. Power Syst. **10**(3), 1642–1651 (1995)
6. Hiriart-Urruty, J.B.: Generalized differentiability, duality and optimization for problems dealing with differences of convex functions. Lect. Notes Econ. Math. Syst. **256**, 37–69 (1985)
7. Horst, R., Tuy, H.: Global optimization. Deterministic Approaches. Springer-Verlag (1990)
8. Horst, R., Pardalos, P.M., Thoai, N.V.: Introduction to global optimization. Kluwer Academic Publishers, Dordrecht (1995)
9. Kaliski, J., Haglin, D., Roos, C., Terlaky, T.: Logarithmic barrier decomposition methods for semi-infinite programming. Int. Trans. Oper. Res. **4**(4), 285–303 (1997)
10. Luc, D.T., Martínez-Legaz, J.E., Seeger, A.: Least deviation decomposition with respect to a pair of convex sets. J. Convex Anal. **6**(1), 115–140 (1998)
11. Martínez-Legaz, J.E., Seeger, A.: A general cone decomposition theory based on efficiency. Math. Program. **65**(1), 1–20 (1994)
12. Tuy, H.D.C.: Optimization: theory, methods and algorithms. Handbook of global optimization. Noncon-Vex Optim. Appl. **2**, 149–216 (1995)
13. Tuy, H.: Convex analysis and global optimization. Kluwer Academic Publishers, Dordrecht (1998)
14. Zhi-Quan Luo, Roos, C., Terlaky, T.: Complexity analysis of logarithmic barrier decomposition methods for semi-infinite linear programming. Appl. Numer. Math. **29**, 379–394 (1999)