IMPROVING THE EFFICIENCY OF DC GLOBAL OPTIMIZATION METHODS BY IMPROVING THE DC REPRESENTATION OF THE OBJECTIVE FUNCTION

Albert Ferrer[*] and Juan Enrique Martínez-Legaz[†]

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.

Key words dc representation, dc program, outer approximation, branch and bound, semi-infinite program.

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

1 Introduction

There are infinitely many ways of representing a d.c. function as a difference of convex functions. In the special case of polynomials, the problem of finding d.c. representations was addressed in [4].

In this paper we consider the following questions: When is a given representation better than another one from a computational point of view? How does computational efficiency depend on the d.c. representation of the objective function? Given a d.c. representation, is it possible to find another one that improves computational efficiency? Which is the best d.c. representation of a d.c. polynomial from a computational point of view, and how can it be obtained?

We introduce some necessary concepts to answer the above questions. Our analysis relies upon the notion of minimal d.c. representation in the space of polynomials. In Section 2 we state *the minimum norm problem* by using the concept of *least deviation decomposition* (*LDD*) of [7]. In Section 3 we give a semi-infinite programming formulation of the minimum norm problem and show that the solution to this problem provides a d.c. representation for an arbitrary polynomial. We find that this dc representation improves the computational efficiency of a global optimization algorithm by reducing the number of iterations needed to find a global optimal solution. In section 5 we report some numerical experiments, which show the efficiency of our minimal dc representations from a computational viewpoint. The conclusions of our analysis are summarized in section 6.

[*] 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 MCYT, DPI 2005-09117-C02-01

[†] Departament d'Economia i d'Història Econòmica, Universitat Autonoma de Barcelona, email: juanenrique.martinez@uab.es. The research of this author has been partially supported by the Ministerio de Ciencia y Tecnología, Project MTM2005-08572-C03-03, and by the Barcelona Economics Program of XREA.

2 The minimum norm problem

Let $\mathbb{R}_m [x_1, ..., x_n]$ and $H_k [x_1, ..., x_n]$, $k = 0, 1, \dots, m$ 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 nonempty 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 2.1 *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 such that $t^{k+1} + t^k + \alpha t^2$ is convex, and therefore $t^{k+1} + t^k$ is a d.c. function.*

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 2.2 *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 2.3 *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 [7] in the abstract setting of normed spaces; 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

$$y_1 = \frac{z+v}{2} \text{ and } y_2 = \frac{v-z}{2}, \tag{1}$$

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*:

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

Hence, using (1), the optimal solution v^* of problem (2) yields an optimal representation $z = y_1^* - y_2^*$, where

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

According to the terminology of [7], 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

$$\mathbb{R}_m [x_1, \ldots, x_n] = \bigoplus_{k=0}^{m} H_k [x_1, \ldots, x_n], \tag{3}$$

allows us to obtain an alternative dc representation of the polynomial z by using *LDD*s 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 (2) 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 (2), 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, that is,

$$\lambda^t \nabla^2(v \pm z)(x)\lambda \geq 0, \quad \forall \lambda \in S^n, \ \forall x \in C, \tag{4}$$

where $S^n = \{x \in \mathbb{R}^n : \|x\| = 1\}$. Thus problem (2) can be equivalently formulated as the semi-infinite quadratic programming problem

$$\left\{ \begin{array}{ll} \text{minimize} & \|v\|^2 = \sum_{f \in B^m} v_f^2 \\ \text{subject to:} & \lambda^t \sum_{f \in B^m}(v_f \pm a_f)\nabla^2 f(x)\lambda \geq 0, \quad \forall \lambda \in S^n \quad, \forall x \in C, \end{array} \right. \tag{5}$$

whose constraint set is indexed by the parameters x and λ.

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

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

and

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

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

$$\nabla^2(v \pm z)(x,y) = \begin{pmatrix} 6v_4x + 2(v_5 \pm 3)y + 2v_1 & 2(v_5 \pm 3)x + 2v_6y + (v_2 \pm 1) \\ 2(v_5 \pm 3)x + 2v_6y + (v_2 \pm 1) & 2v_6x + 6v_7y + 2v_3 \end{pmatrix}.$$

Imposing the positive-semidefinitiness condition to these matrices we get

$$\begin{gathered} 2\lambda_1^2v_1 + 2\lambda_1\lambda_2v_2 + 2\lambda_2^2v_3 + 6x\lambda_1^2v_4 + 2(y\lambda_1^2 + 2x\lambda_1\lambda_2)v_5 + \\ +2(x\lambda_2^2 + 2y\lambda_1\lambda_2)v_6 + (6y\lambda_2^2)v_7 \pm (6y\lambda_1^2 + 12x\lambda_1\lambda_2 + 2\lambda_1\lambda_2) \geq 0 \end{gathered}$$

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

$$\begin{gathered} \lambda_1^2a + \lambda_1\lambda_2b + \lambda_2^2c + 3x\lambda_1^2d + (y\lambda_1^2 + 2x\lambda_1\lambda_2)e + \\ +(x\lambda_2^2 + 2y\lambda_1\lambda_2)f + 3y\lambda_2^2g - \mid 3y\lambda_1^2 + 6x\lambda_1\lambda_2 + \lambda_1\lambda_2 \mid \geq 0. \end{gathered}$$

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 $\lambda_1^2 + \lambda_2^2 + \ldots + \lambda_n^2 = 1$, $\lambda_1 \geq 0$ can be generated by using spherical coordinates:*

$$\left\{ \begin{array}{lcr} \lambda_1 & = & \sin\omega_1 \sin\omega_2 \ldots \sin\omega_{n-2} \sin\omega_{n-1}, \\ \lambda_2 & = & \cos\omega_1 \sin\omega_2 \ldots \sin\omega_{n-2} \sin\omega_{n-1}, \\ \ldots & \ldots & \ldots\ldots\ldots\ldots \quad , \\ \lambda_{n-2} & = & \cos\omega_{n-3} \sin\omega_{n-2} \sin\omega_{n-1}, \\ \lambda_{n-1} & = & \cos\omega_{n-2} \sin\omega_{n-1}, \\ \lambda_n & = & \cos\omega_{n-1}, \end{array} \right. \tag{6}$$

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

The algorithm we will use to solve the semi-infinite quadratic programming problem (5) is an adaptation of the semi-infinite linear programming methods described in [6] and [8], 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 (5) is convex and compact. Without loss of generality (by applying a translation if necessary), we will further assume that $C \subset I\!R^p_{++}$.

Consider a nonhomogeneous polynomial $z = \sum_{i=1}^k z_i$, with $z_i \in H_{n_i}[x_1, \ldots, x_p]$, $2 \leq n_1 < \ldots < n_k$. 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

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

$C(n,\ p)$ 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 $\alpha_1 + \ldots + \alpha_p = n$. 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 (5). Finally, by taking $v = w + q$, with $q(x) = x_1^2 + \ldots + x_p^2$, we get a feasible point at which all the constraints are satisfied strictly.

5 Results of numerical experiments

In this section we report the numerical experiments we have made on some test problems by using our optimal d.c. representation together with the deterministic global optimization algorithm for solving reverse convex programming problems described in [4]. This algorithm combines a prismatical subdivision process with polyhedral outer approximation, in such a way that only linear programs have to be solved. We present the results obtained on some test problems, which illustrate the efect of using optimal d.c. representations on solving d.c. polynomial programming problems; some of the test problems, taken from [3], correspond to a hydroelectric generation model.

5.1 Some test problems

We first consider three simple two variables optimization problems: $HPBr1$, $HOM3r2$ and $POL3r2$. The results are shown in Table 2, which displays the results of minimizing the functions using different d.c. representations and precision $\epsilon = 0.001$; the running CPU time is reported. *Iter* found is the number of iterations of the program needed to confirm the global minimum. DC represents the kind of d.c. representation of the objective function: the optimal d.c. representation is indicated by *opt* and the non optimal representations are noted (1) and (2). *Norm* indicates the norm of the corresponding d.c. representation.

The instance $HPBr1$ is

$$\begin{array}{rl} \text{minimize} & xy = \frac{1}{4}(x+y)^2 - \frac{1}{4}(x-y)^2 \\ \text{subject to:} & \quad x - y \leq 5.7, \\ & \quad -2 \leq x \leq 3, \\ & \quad -3 \leq y \leq 4, \end{array} \tag{7}$$

and the indicated d.c. representation of xy in (7) is the optimal one. Alternative d.c. representations of xy, which are not optimal, are

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

and

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

The objective function $f(x,\ y)$ of instances $HOM3r2$ and $POL3r2$ are $3x^2y$ and $xy + 3x^2y$, respectively, and both problems have the same constraints:

$$\begin{array}{rc} \text{minimize} & f(x,\ y) \\ \text{subject to:} & x - y \leq 1, \\ & -x - y \leq 2.5, \\ & 0.5 \leq x \leq 2, \\ & 2 \leq y \leq 4. \end{array}$$

A d.c. representation of the objective function of program $HOM3r2$ is

$$3x^2y = 0.5(2x+y)^3 + 0.5y^3 - (3x^3 + (x+y)^3),$$

and the optimal d.c. representation, obtained as mentioned above, is

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

with

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

From the d.c. representations of the functions xy and $3x^2y$ we obtain a d.c. representation of $xy + 3x^2y$:

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

with

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

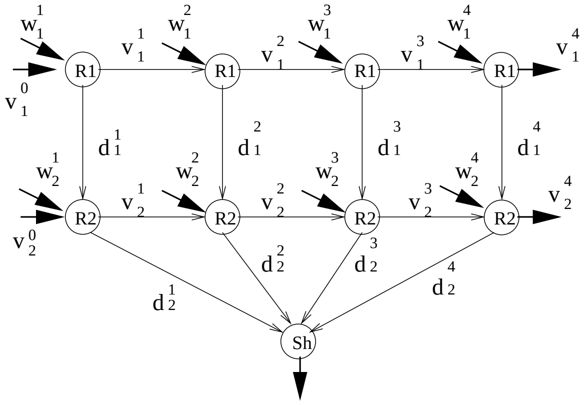

Figure 1: Four intervals and two reservoirs replicated hydronetwork

5.2 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. Figure 1 shows a network with only two reservoirs and a time period subdivided into 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 i^{th} interval and the volume stored v_j^i in reservoir j at the end of the i^{th} 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]):

$$\begin{aligned} h_j^i(v_j^{i-1}, v_j^i, d_j^i) \quad = \quad & \mathsf{k}_j^i d_j^i [s_{vd} + \tfrac{s_{vl}}{2}(v_j^{i-1} + v_j^i) + \tfrac{s_{vq}}{3}(v_j^i - v_j^{i-1})^2 + \\ & + s_{vq} v_j^{i-1} v_j^i + \tfrac{s_{vc}}{4}((v_j^{i-1})^2 + (v_j^i)^2)(v_j^{i-1} + v_j^i) - \\ & - s_{dl} d_j^i - s_{dq}(d_j^i)^2], \end{aligned} \tag{8}$$

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:

$$f(\ldots, v_j^{i-1}, v_j^i, d_j^i, \ldots) = \sum_{i=1}^{N_t} c_i \left(l^i - \sum_{j=1}^{N_e} h_j^i(v_j^{i-1}, v_j^i, d_j^i) \right). \tag{9}$$

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:

$$\underline{g} \leq l^i - \sum_{j=1}^{N_e} h_j^i(v_j^{i-1}, v_j^i, d_j^i) \leq \overline{g} \quad i = 1, ..., N_t. \tag{10}$$

There are positive bounds on all variables:

$$\underline{d_j} \leq d_j^i \leq \overline{d_j} \qquad j = 1, ..., N_e, \quad i = 1, ..., N_t$$

$$\underline{v_j} \leq v_j^i \leq \overline{v_j} \quad j = 1, ..., N_e,\ i = 1, ..., N_t - 1.$$

Thus our problem is

$$\boxed{\begin{array}{rll}
\text{minimize} & \sum_{i=1}^{N_t} c_i \left(l^i - \sum_{j=1}^{N_e} h_j^i(v_j^{i-1}, v_j^i, d_j^i) \right) & \\
\text{subject to} & \underline{g} \leq l^i - \sum_{j=1}^{N_e} h_j^i(v_j^{i-1}, v_j^i, d_j^i) \leq \overline{g}, & i = 1, ..., N_t, \\
& v_j^i - v_j^{i-1} - d_{j-1}^i + d_j^i = w_j^i, & j = 1, ..., N_e, \\
& & i = 1, ..., N_t, \\
& \underline{d_j} \leq d_j^i \leq \overline{d_j}, & j = 1, ..., N_e, \\
& & i = 1, ..., N_t, \\
& \underline{v_j} \leq v_j^i \leq \overline{v_j} & j = 1, ..., N_e, \\
& & i = 1, ..., N_t - 1.
\end{array}} \tag{11}$$

This model has the following useful properties:

1. It is easy to generate problems of different sizes (Table 1) and with different degrees of non-convexity, 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.

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

Let

$$h^i_j(v^{i-1}_j, v^i_j, d^i_j) = f^i_j(v^{i-1}_j, v^i_j, d^i_j) - g^i_j(v^{i1}_j, v^i_j, d^i_j),$$

be a d.c. representation of the power hydrogeneration function, where $f^i_j(v^{i-1}_j, v^i_j, d^i_j)$ and $g^i_j(v^{i1}_j, v^i_j, d^i_j)$ are convex functions defined on a convex set which contains the feasible domain of program (11). Then, by defining for every $i = 1, ..., N_t$ the convex functions

$$F^i(\ldots, v^{i-1}_j, v^i_j, d^i_j, \ldots) = c^i \left(l^i + \sum_{j=1}^{N_e} g^i_j(v^{i-1}_j, v^i_j, d^i_j) \right) \tag{12}$$

and

$$G^i(\ldots, v^{i-1}_j, v^i_j, d^i_j, \ldots) = c^i \sum_{j=1}^{N_e} f^i_j(v^{i-1}_j, v^i_j, d^i_j), \tag{13}$$

and using these expressions to define

$$F(\ldots, v^{i-1}_j, v^i_j, d^i_j, \ldots) = \sum_{i=1}^{N_t} F^i(\ldots, v^{i-1}_j, v^i_j, d^i_j, \ldots) \tag{14}$$

and

$$G(\ldots, v^{i-1}_j, v^i_j, d^i_j, \ldots) = \sum_{i=1}^{N_t} G^i(\ldots, v^{i-1}_j, v^i_j, d^i_j, \ldots), \tag{15}$$

d.c. representations of all functions in (11) are obtained. We further define $n = N_e(2N_t - 1)$, $m = N_e N_t$ and $x = (\ldots, v^{i-1}_j, v^i_j, d^i_j, \ldots) \in \mathbb{R}^n$; 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 (11) has the structure

$$\begin{array}{rcl} \text{minimize} & F(x) - G(x) & \\ \text{subject to} & (G^i(x) + c^i\underline{g}) - F^i(x) \le 0 & i = 1, ..., N_t, \\ & F^i(x) - (G^i(x) + c^i\overline{g}) \le 0 & i = 1, ..., N_t, \\ & Ax = b, & \\ & \underline{x} \le x \le \overline{x}, & \end{array} \tag{16}$$

where $\underline{x},\ \overline{x} \in \mathbb{R}^n$. After renumbering the variables if necessary, the matrix A in (16) 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 (16) by defining the functions $\varphi_1(z) = F(B^{-1}(b - Nz), z)$, $\varphi_2(z) = G(B^{-1}(b - Nz), z)$, $\varphi^i_1(z) = F^i(B^{-1}(b - Nz), z)$ and $\varphi^i_2(z) = G^i(B^{-1}(b - Nz), z)$. By using these functions in (16) we obtain the following equivalent d.c. program of reduced size

$$\begin{array}{rcl} \text{minimize} & \varphi_1(z) - \varphi_2(z) & \\ \text{subject to} & (\varphi^i_2(z) + c^i\underline{g}) - \varphi^i_1(z) \le 0 & i = 1, ..., N_t, \\ & \varphi^i_1(z) - (\varphi^i_2(z) + c^i\overline{g}) \le 0 & i = 1, ..., N_t, \\ & \underline{b} \le Mz \le \overline{b}, & \\ & \underline{z} \le z \le \overline{z}, & \end{array} \tag{17}$$

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

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

and

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

we reformulate (17) as the following equivalent reverse convex program:

$$\boxed{\begin{array}{ll} \text{minimize} & 2\varphi_1(z) - t \\ \text{subject to} & (z,t) \in \Omega \setminus \text{int}\Delta. \end{array}} \tag{18}$$

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].

5.2.2 Characteristics of generation systems and computational results

The characteristics of the generation systems we have considered are described in Table 1. The names of the problems in Table 3 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_j^i in (8) 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 3 *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 3 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.

Table 1: Characteristics of generation systems

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

Table 2: 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

Table 3: 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*

6 Conclusions

The superior computational efficiency of the method when we use the optimal d.c. representation is clearly shown in Tables 2 and 3. 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.

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 (11), 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.

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.

References

[1] Brunat, J.M. and 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. and 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. *Lectures Notes in Economics and Mathematical Systems*, 502:189–207, 2001.

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

[5] Heredia, F.J. and Nabona, N. Optimum short-term hydrothermal scheduling with spinning reverse through network flows. *IEEE Trans. on Power Systems*, 10(3):1642–1651, 1995.

[6] Kaliski, J., Haglin, D., Roos, C. and Terlaky, T. Logarithmic barrier decomposition methods for semi-infinite programming. *Int.Trans.Oper.Res.*, 4(4):285–303, 1997.

[7] Luc, D.T., Martinez-Legaz, J.E. and Seeger, A. Least deviation decomposition with respect to a pair of convex sets. *Journal of Convex Analysis*, 6(1):115–140, 1998.

[8] Zhi-Quan Luo, Roos, C. and Terlaky, T. Complexity analysis of logarithmic barrier decomposition methods for semi-infinite linear programming. *Applied Numerical Mathematics*, 29:379–394, 1999.