Journal of Global Optimization (2022) 83:187–200
https://doi.org/10.1007/s10898-021-01102-2

A revision of the rectangular algorithm for a class of DC optimization problems

Takahito Kuno[1]

Received: 13 December 2020 / Accepted: 11 October 2021 / Published online: 27 October 2021

Abstract
Every continuously differentiable function can be represented as a difference between a convex function and an additively separable convex function. We show that a DC function with this structure can be optimized using the rectangular algorithm for separable nonconvex optimization, and develop a revision to this algorithm for practical use. We also report some numerical results which indicate the effectiveness of the revision.

Keywords Global optimization · DC optimization · Branch-and-bound · Rectangular algorithm · ω-subdivision

1 Introduction

The DC optimization includes most optimization problems arising in a variety of applications because every continuously twice differentiable function is a DC function, represented as a Difference of two Convex functions on any compact convex set. In fact, for such a class of function $f : \mathbb{R}^n \to \mathbb{R}$ and a convex set $\Omega \subset \mathbb{R}^n$, it is shown in [6,14] that $g(\mathbf{x}) = f(\mathbf{x}) + \rho\|\mathbf{x}\|^2$ is convex if $\rho \geq -(1/2)\min\left\{\mathbf{y}^{\mathsf{T}}\nabla^2 f(\mathbf{x})\mathbf{y} \mid \mathbf{x} \in \Omega,\ \|\mathbf{y}\| = 1\right\}$, and therefore f is decomposable into a DC function:

$$f(\mathbf{x}) = g(\mathbf{x}) - \rho\|\mathbf{x}\|^2. \tag{1}$$

To solve this optimization problem of high generality, Tuy proposed a conical algorithm for global optimization in 1987 [13], and Tao and Hoai-An published in 1997 the so-called DCA (DC Algorithm) based on local optimization [12]. Since the 2000s, various studies with DC functions have been conducted mainly in the field of machine learning, and accordingly local optimization algorithms such as DCA have also been widely used there (see e.g., [1,7,16]). In contrast to this, any of global optimization algorithms, including the conical algorithm,

The author was partially supported by a Grant-in-Aid for Scientific Research (C) 19K11837 from Japan Society for the Promotion of Sciences.

✉ Takahito Kuno
takahito@cs.tsukuba.ac.jp

[1] Graduate School of Systems and Information Engineering, University of Tsukuba, Tsukuba, Ibaraki 305-8573, Japan

have not yet reached the stage of practical application, though discussed in detail in textbooks [9,10,15].

In general, it is not easy to detect a DC representation suitable for each algorithm, such as the conical algorithm and DCA, from a given continuous function. However, even a simple DC representation like (1) has the noteworthy feature that the subtrahend is additively separable. In this paper, we show that a DC function with the same structure can be globally optimized on a bounded polyhedron using the rectangular algorithm [3], a branch-and-bound algorithm for separable nonconvex optimization problems. Since the algorithm requires solving a sequence of convex minimization problems, we revise it so that the convex minimization procedure can be warm-started for efficient processing.

The organization of this paper is as follows. In Section 2, after formalizing the problem setting, we illustrate how the rectangular algorithm behaves on this problem. In Section 3, in order to warm-start the convex minimization procedure, we further relax the convex relaxation solved in the bounding process. We also show that the branching process can be carried out with ω-subdivision using the optimal solution of the modified convex relaxation. In Section 4, we summarize the algorithm and analyze its convergence. Finally, in Section 5, we report the numerical results, which indicate that the revised rectangular algorithm is promising for practical application in some fields.

2 DC optimization and the rectangular algorithm

Let g be a convex function from $\mathbb{R}^n$ to $\mathbb{R}$, and $h_1, \ldots, h_n$ strictly convex functions from $\mathbb{R}$ to $\mathbb{R}$. The problem considered in this paper is a DC optimization problem of the form:

$$\left| \begin{array}{l} \text{minimize } f(\mathbf{x}) = g(\mathbf{x}) - \displaystyle\sum_{j=1}^{n} h_j(x_j) \\ \text{subject to } \mathbf{A}\mathbf{x} \leq \mathbf{b}, \end{array} \right. \tag{2}$$

where $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{b} \in \mathbb{R}^m$. Let us denote the feasible set by

$$D = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{A}\mathbf{x} \leq \mathbf{b}\}.$$

We assume that D is a bounded polyhedron with a nonempty interior and contained in the effective domain of g. Let

$$s_j^1 = \min\{x_j \mid \mathbf{x} \in D\}, \quad t_j^1 = \max\{x_j \mid \mathbf{x} \in D\}, \quad j = 1, \ldots, n.$$

For each j, we have $s_j^1 < t_j^1$, and assume that the interval $[s_j^1, t_j^1]$ lies in the effective domain of h_j. We can locate a globally optimal solution of (2) by almost directly applying the rectangular algorithm designed for separable nonconvex optimization [3], as is illustrated below.

2.1 Behavior of the rectangular algorithm on (2)

Let $M^1 = [\mathbf{s}^1, \mathbf{t}^1] = \prod_{j=1}^n [s_j^1, t_j^1]$. In the rectangular algorithm, M^1 is subdivided into a number of subrectangles M^i, $i \in I$, such that

$$M^1 = \bigcap_{i \in I} M^i, \quad \text{int}(M^i) \bigcap \text{int}(M^\ell) = \emptyset \text{ if } i \neq \ell,$$

where int(·) represents the set of interior points. Let $M = [\mathbf{s}, \mathbf{t}] = M^i$ for an arbitrary $i \in I$. If $D \cap M = \emptyset$, then M contains no optimal solution of (2) and is discarded from consideration. Otherwise, *bounding* is performed to see if an optimal solution of (2) is given by a subproblem

$$\text{(P)} \left| \begin{array}{l} \text{minimize } f(\mathbf{x}) \\ \text{subject to } \mathbf{x} \in D \cap M. \end{array} \right.$$

Since this is essentially the same problem as (2), we cannot solve (P) directly. Instead, (P) is approximated into a convex minimization problem by replacing h_j's involved in the objective function f with linear functions:

$$c_j^M(x) = \frac{h_j(t_j) - h_j(s_j)}{t_j - s_j}(x - s_j) + h_j(s_j), \quad j = 1, \ldots, n. \tag{3}$$

Thus a convex relaxation of (P) is constructed as follows:

$$\text{(R)} \left| \begin{array}{l} \text{minimize } \overline{f}^M(\mathbf{x}) = g(\mathbf{x}) - \displaystyle\sum_{j=1}^{n} c_j^M(x_j) \\ \\ \text{subject to } \mathbf{x} \in D \cap M. \end{array} \right.$$

We can apply any one of the standard optimization algorithms to (R) (see e.g., [2]) and obtain an optimal solution, denoted $\boldsymbol{\omega}^M$.

From the definition of c_j^M, it is easy to see that $c_j^M(x) = h_j(x)$ if $x \in \{s_j, t_j\}$. Furthermore, from the strict convexity of h_j, we have the following.

Proposition 2.1 *For each j, it holds that*

$$c_j^M(x) > h_j(x) \quad \textit{if and only if} \quad x \in (s_j, t_j).$$

Proof The 'if' part follows immediately from the definition of strict convexity of h_j. To show the 'only if' part, assume that $c_j^M(u) > h_j(u)$ for some $u \leq s_j$. Choosing $v \in (s_j, t_j)$ arbitrarily, let

$$d(x) = \frac{h_j(v) - h_j(u)}{v - u}(x - u) + h_j(u).$$

Then we have $d(u) = h_j(u) < c_j^M(u)$, and $d(v) = h_j(v) < c_j^M(v)$. Since both c_j^M and d are linear, $d(x) < c_j^M(x)$ holds for any $x \in (u, v)$. As noticed above, u cannot agree with s_j, and hence s_j lies in (u, v). Nevertheless, we have $d(s_j) < c_j^M(s_j) = h_j(s_j)$, which contradicts the convexity of h_j. Therefore, $c_j^M(x) \leq h_j(x)$ holds if $x \leq s_j$. Similarly, we have $c_j^M(x) \leq h_j(x)$ for any $x \geq t_j$. □

We see from Proposition 2.1 that

$$\overline{f}^M(\mathbf{x}) \leq f(\mathbf{x}) \quad \text{if } \mathbf{x} \in M,$$

where the equality holds if $\mathbf{x}$ lies on a vertex of M, while

$$\overline{f}^M(\mathbf{x}) \geq f(\mathbf{x}) \quad \text{if } \mathbf{x} \notin \text{int}(M).$$

Therefore, the optimal value $\overline{f}^M(\boldsymbol{\omega}^M)$ of (R) is a lower bound for the subproblem (P). If $\overline{f}^M(\boldsymbol{\omega}^M) \geq f(\mathbf{x}^*)$ holds for the incumbent $\mathbf{x}^*$, the best known feasible solution of (2), then M contains no feasible solution better than $\mathbf{x}^*$ and is excluded from further consideration. Otherwise, *branching* is performed and M is subdivided for further examination.

How to subdivide M greatly affects the convergence of the algorithm. It is naturally guaranteed with an exhaustive subdivision rule such as bisection, but the algorithm also converges under the ω-subdivision rule given below.

ω-subdivision

1° Choose $k \in \arg\max\{c_j^M(\omega_j^M) - h_j(\omega_j^M) \mid j = 1, \ldots, n\}$.
2° Divide $M = [\mathbf{s}, \mathbf{t}]$ into $M_- = [\mathbf{s}, \mathbf{t}']$ and $M_+ = [\mathbf{s}', \mathbf{t}]$, where

$$s'_j = \begin{cases} \omega_k^M & \text{if } j = k \\ s_j & \text{otherwise,} \end{cases} \qquad t'_j = \begin{cases} \omega_k^M & \text{if } j = k \\ t_j & \text{otherwise.} \end{cases}$$

Whatever the subdivision rule is, if the algorithm does not terminate, it generates a sequence of nested rectangles:

$$M^1 \supset M^2 \supset \cdots \supset M^i \supset \cdots . \tag{4}$$

Let $\overline{f}^i$ and $\boldsymbol{\omega}^i$ denote the objective function and the optimal solution, respectively, of (R) defined for $M = M^i$. Under the ω-subdivision rule, we have the following result, the proof of which is almost the same as for separable concave minimization (see e.g., 7.1.6 in [15]).

Proposition 2.2 *If the sequence (4) is generated according to ω-subdivision, there exists a subsequence K of $\{1, 2, \ldots\}$ such that $f(\boldsymbol{\omega}^i) - \overline{f}^i(\boldsymbol{\omega}^i) \to 0$ as $i \in K \to \infty$.*

The rectangular algorithm repeats updating the incumbent $\mathbf{x}^*$ while alternating the above branching and bounding processes. If we select the rectangle M with the smallest $\overline{f}^M(\boldsymbol{\omega}^M)$ to subdivide in the branching process, Proposition 2.2 guarantees that every accumulation point of the sequence $\{\boldsymbol{\omega}^i\}$ is an optimal solution of (2). To terminate the algorithm in a finite amount of time, M is discarded usually if the following pruning criterion is satisfied for a prescribed tolerance $\epsilon \in (0, 1)$:

$$f(\mathbf{x}^*) - \overline{f}^M(\boldsymbol{\omega}^M) \le \epsilon.$$

3 Revision of the rectangular algorithm

The rectangular algorithm presented in the preceding section requires solving the convex relaxation (R) iteratively. To solve the target problem (2) with sufficient accuracy, we have to solve tens of thousands of convex minimization problems even for a small scale instance, as will be seen in Section 5. Solving them one by one from scratch would take a huge amount of time and ruin the potential of the rectangular algorithm. In this section, to avoid such a case, we discuss a 'warm-start' which allows a convex minimization algorithm to solve the current relaxation with the solution of the last solved relaxation as the initial solution.

3.1 Relaxation of the convex relaxation (R)

Let $\boldsymbol{\omega}'$ denote the solution to the last solved relaxation. After obtaining $\boldsymbol{\omega}'$, which rectangle to select next for bounding depends on the rule of searching the branching tree, and the rectangle M defining the current relaxation (R) is not guaranteed to contain $\boldsymbol{\omega}'$. In the numerical experiments reported in Section 5, the Frank-Wolfe algorithm [4] was used to solve (R). If $\boldsymbol{\omega}'$ is feasible for (R), the algorithm uses $\boldsymbol{\omega}'$ as the initial solution, but otherwise, it requires

preprocessing to find a feasible solution. In order to save this effort, we propose to relax (R) further into the following, so that $\boldsymbol{\omega}'$ is feasible for the current relaxation without fail:

$$(\overline{\mathrm{R}}) \quad \left| \begin{array}{l} \text{minimize } \overline{f}^M(\mathbf{x}) = g(\mathbf{x}) - \displaystyle\sum_{j=1}^{n} c_j^M(x_j) \\ \text{subject to } \mathbf{x} \in D. \end{array} \right.$$

Since the feasible set is the same as (2), this problem is assumed to have an optimal solution, denoted $\overline{\boldsymbol{\omega}}^M$. The lower bound $\overline{f}^M(\overline{\boldsymbol{\omega}}^M)$ provided by $(\overline{\mathrm{R}})$ is obviously somewhat inferior to $\overline{f}^M(\boldsymbol{\omega}^M)$. This weakness is, however, offset by the advantage that $\boldsymbol{\omega}'$ can be used as the initial solution for a convex minimization algorithm of the type generating a sequence of feasible solutions. In addition to the benefit of warm-start, when D has some favorable structure like a graph, $(\overline{\mathrm{R}})$ might be solved efficiently without damaging that structure. What seems to be problematic is that $\overline{\boldsymbol{\omega}}^M$ is not always a point in M. If $\overline{\boldsymbol{\omega}}^M \notin M$, the $\boldsymbol{\omega}$-subdivision rule described in the preceding section might fail to subdivide the rectangle M. As is shown below, such a concern is unnecessary.

Let us try to apply the ω-subdivision rule to M at $\overline{\boldsymbol{\omega}}^M$. In step 1°, an index k is chosen from $\arg\max\{c_j^M(\overline{\omega}_j^M) - h_j(\overline{\omega}_j^M) \mid j = 1, \ldots, n\}$. If $c_k^M(\overline{\omega}_k^M) - h_k(\overline{\omega}_k^M) > 0$, then we see from Proposition 2.1 that $\overline{\omega}_k^M$ lies in the relative interior of $[s_k, t_k]$. According to step 2°, the rectangle M is divided by cutting the interval $[s_k, t_k]$ at $\overline{\omega}_k^M$. However, if $c_k^M(\overline{\omega}_k^M) - h_k(\overline{\omega}_k^M) \leq 0$, then $[s_k, t_k]$ cannot be cut at $\overline{\omega}_k^M$ because $\overline{\omega}_k^M \notin (s_k, t_k)$. In that case, the rectangle M contains no feasible solution better than $\overline{\boldsymbol{\omega}}^M$. In fact, we have

$$c_j^M(\overline{\omega}_j^M) - h_j(\overline{\omega}_j^M) \leq c_k^M(\overline{\omega}_k^M) - h_k(\overline{\omega}_k^M) \leq 0, \quad j = 1, \ldots, n,$$

which implies that

$$f(\overline{\boldsymbol{\omega}}^M) = g(\overline{\boldsymbol{\omega}}^M) - \sum_{j=1}^{n} h_j(\overline{\omega}_j^M) \leq g(\overline{\boldsymbol{\omega}}^M) - \sum_{j=1}^{n} c_j^M(\overline{\omega}_j^M) = \overline{f}^M(\overline{\boldsymbol{\omega}}^M).$$

Even if $D \cap M \neq \emptyset$, the following chain of inequalities holds:

$$f(\overline{\boldsymbol{\omega}}^M) \leq \overline{f}^M(\overline{\boldsymbol{\omega}}^M) \leq \overline{f}^M(\boldsymbol{\omega}^M) \leq f(\mathbf{x}), \quad \forall \mathbf{x} \in D \cap M.$$

We can therefore remove M from further consideration. In either case, $\overline{\boldsymbol{\omega}}^M$ is a feasible solution of (2), and so the incumbent $\mathbf{x}^*$ can be updated with $\overline{\boldsymbol{\omega}}^M$ if $f(\overline{\boldsymbol{\omega}}^M) < f(\mathbf{x}^*)$.

3.2 Revised rectangular algorithm

The revised rectangular algorithm we propose can be summarized as follows.

algorithm revised_rectangular$(g, h_1, \ldots, h_n, D, \varepsilon)$

for $j \leftarrow 1, \ldots, n$ do

$s_j \leftarrow \min\{x_j \mid \mathbf{x} \in D\}$; $t_j \leftarrow \max\{x_j \mid \mathbf{x} \in D\}$;

end for

$\mathscr{L} \leftarrow \emptyset$; $\mathscr{T} \leftarrow \{[\mathbf{s}, \mathbf{t}]\}$; $\mathbf{x}^* \leftarrow nil$; $\alpha \leftarrow +\infty$; $stop \leftarrow false$;

while $stop = false$ do

bounding process

for each $M \in \mathscr{T}$ do

construct the relaxation $(\overline{\mathrm{R}})$ of the subproblem (P) associated with M;

compute an optimal solution $\overline{\boldsymbol{\omega}}^M$ of $(\overline{\mathrm{R}})$, and let $\beta^M \leftarrow \overline{f}^M(\overline{\boldsymbol{\omega}}^M)$;

if $f(\overline{\boldsymbol{\omega}}^M) < \alpha$ then

$\mathbf{x}^* \leftarrow \overline{\boldsymbol{\omega}}^M$; $\alpha \leftarrow f(\mathbf{x}^*)$;

end if

end for

$\mathscr{L} \leftarrow \{M \in \mathscr{L} \cup \mathscr{T} \mid \alpha - \beta^M > \varepsilon\}$;

if $\mathscr{L} = \emptyset$ then

$stop \leftarrow true$;

else

branching process

select $M = [\mathbf{s}, \mathbf{t}]$ from $\mathscr{L}$ according to the prescribed rule, and let $\mathscr{L} \leftarrow \mathscr{L} \setminus \{M\}$;

choose $k \in \arg\max\{c_j^M(\overline{\omega}_j^M) - h_j(\overline{\omega}_j^M) \mid j = 1, \ldots, n\}$;

if $c_k^M(\overline{\omega}_k^M) - h_k(\overline{\omega}_k^M) > 0$ then

$\mathbf{s}' \leftarrow \mathbf{s}$; $s'_k \leftarrow \overline{\omega}_k^M$; $\mathbf{t}' \leftarrow \mathbf{t}$; $t'_k \leftarrow \overline{\omega}_k^M$; $\mathscr{T} \leftarrow \{[\mathbf{s}, \mathbf{t}'], [\mathbf{s}', \mathbf{t}]\}$;

end if

end if

end while

return $\mathbf{x}^*$;

end.

As for the rule of selecting M from the list $\mathscr{L}$, which is not specified in the branching process of this description, we can adopt any one of the rules commonly used in ordinary branch-and-bound algorithms, e.g., the best-bound rule that selects the M with the smallest β^M, the depth-first rule that selects the last generated M, and so forth. If the algorithm terminates, then $\mathbf{x}^*$ is output as an ϵ-optimal solution of the problem (2), which satisfies

$$f(\mathbf{x}^*) - f(\mathbf{x}) \leq \epsilon, \quad \forall \mathbf{x} \in D.$$

The idea of removing constraints to the rectangle M from the relaxation is also adopted in Soland [11] for separable concave minimization. If the convex function g is absent from the objective function, $(\overline{\mathrm{R}})$ is a linear program and some vertex of the feasible set D provides an optimal solution. Soland has exploited the finiteness of the vertices to show the finite convergence of his rectangular algorithm. However, once g exists as a nonlinear function, there is no guarantee that the optimal solution $\overline{\boldsymbol{\omega}}^M$ of $(\overline{\mathrm{R}})$ lies on a vertex of D. In general, when the tolerance ϵ is set to zero, the algorithm does not terminate and generates an infinite

sequence of nested rectangles in $\mathscr{L}$. Let us observe the behavior of this sequence in the next section, and analyze the convergence of revised_rectangular.

4 Convergence of the revised rectangular algorithm

Suppose that the algorithm revised_rectangular generates a sequence of nested rectangles:

$$M^1 \supset M^2 \supset \cdots \supset M^i \supset \cdots . \tag{5}$$

To simplify notation, we omit 'M' from the superscript 'M^i' such as $\overline{\omega}_j^{M^i}$. Since the sequence (5) is infinite, $\max\{c_j^i(\overline{\omega}_j^i) - h_j(\overline{\omega}_j^i) \mid j = 1, \ldots, n\}$ mast be positive for every i. The rectangle M^i is divided by cutting some interval $[s_k^i, t_k^i]$ at its interior point $\overline{\omega}_k^i$ into M_-^i and M_+^i, either of which is M^{i+1}. There are only n candidates for the index k, and so at least one of them, say 1 without loss of generality, is chosen infinitely many times in the course of generating (5). Let K be the subsequence of $\{1, 2, \ldots\}$ such that $1 \in \arg\max\{c_j^i(\overline{\omega}_j^i) - h_j(\overline{\omega}_j^i) \mid j = 1, \ldots, n\}$ for every $i \in K$.

Lemma 4.1 *The sequences $\{s_1^i \mid i \in K\}$ and $\{t_1^i \mid i \in K\}$ converge to some s_1^0 and t_1^0, respectively, either of which is the limit of each convergent subsequence of $\{\overline{\omega}_1^i \mid i \in K\}$.*

Proof Since $\{s_1^i \mid i \in K\}$ is monotonically nondecreasing and bounded from above by t_1^1, it is a convergent sequence and has the limit s_1^0. Similarly, $\{t_1^i \mid i \in K\}$ has the limit t_1^0. For simplicity, renumber the indices in K so that $i = 1, 2, \ldots$. Then $\overline{\omega}_1^i$ coincides with either s_1^{i+1} or t_1^{i+1}. Since $s_1^{i+1} \to s_1^0$ and $t_1^{i+1} \to t_1^0$ as $i \to \infty$, each convergent subsequence of $\{\overline{\omega}_1^i \mid i \in K\}$ has the limit in $\{s_1^0, t_1^0\}$. □

Under the ω-subdivision rule, while $s_1^i < t_1^i$ holds for every i, we might have $s_1^0 = t_1^0$. In that case, $\{\overline{\omega}_1^i \mid i \in K\}$ also converges to $\overline{\omega}_1^0 = s_1^0 = t_1^0$, and $y = c_1^i(x)$ tends to a tangent line of the graph $\{(x, y) \mid y = h_1(x)\}$ at $\left(\overline{\omega}_1^0, h_1(\overline{\omega}_1^0)\right)$. To see this, let $\lambda^i = \left(\overline{\omega}_1^0 - s_1^i\right) / \left(t_1^i - s_1^i\right)$ for each $i \in K$. Then we have

$$c_1^i(x) = \left((1-\lambda^i)\frac{h_1(t_1^i) - h_1(\overline{\omega}_1^0)}{t_1^i - \overline{\omega}_1^0} + \lambda^i \frac{h_1(\overline{\omega}_1^0) - h_1(s_1^i)}{\overline{\omega}_1^0 - s_1^i}\right)(x - s_1^i) + h_1(s_1^i).$$

Since $\lambda^i \in (0, 1)$ for every $i \in K$, we can assume $\lambda^i \to \lambda^0 \in [0, 1]$ as $i \in K \to \infty$, by passing to a subsequence if necessary. Therefore, the sequence $\{c_1^i \mid i \in K\}$ converges pointwise on $\mathbb{R}$ to

$$c_1^0(x) = \left((1-\lambda^0)h'_+(\overline{\omega}_1^0) + \lambda^0 h'_-(\overline{\omega}_1^0)\right)(x - \overline{\omega}_1^0) + h_1(\overline{\omega}_1^0),$$

where h'_+ and h'_- are the right and left derivative functions of h_1. The subdifferential of h_1 at $\overline{\omega}_1^0$ is given by $[h'_-(\overline{\omega}_1^0), h'_+(\overline{\omega}_1^0)]$, to which the coefficient $\left((1-\lambda^0)h'_-(\overline{\omega}_1^0) + \lambda^0 h'_+(\overline{\omega}_1^0)\right)$ of c_1^0 belongs.

Lemma 4.2 *The difference $c_1^i(\overline{\omega}_1^i) - h_1(\overline{\omega}_1^i)$ vanishes as $i \in K \to \infty$.*

Proof We may assume $\overline{\omega}_1^i - s_1^i \to 0$ as $i \in K \to \infty$. Even assuming $\overline{\omega}_1^i - t_1^i \to 0$, the result is the same. If $s_1^0 < t_1^0$, then we have

$$c_1^i(\overline{\omega}_1^i) - h_1(\overline{\omega}_1^i) = \frac{h_1(t_1^i) - h_1(s_1^i)}{t_1^i - s_1^i}(\overline{\omega}_1^i - s_1^i) + \left(h_1(s_1^i) - h_1(\overline{\omega}_1^i)\right) \to 0, \quad \text{as } i \in K \to \infty.$$

If $s_1^0 = t_1^0$, then $\{c_1^i \mid i \in K\}$ converges to c_1^0 pointwise, as seen above, and again we have $c_1^i(\overline{\omega}_1^i) - h_1(\overline{\omega}_1^i) \to 0$ as $i \in K \to \infty$. □

Now, we are ready to prove the convergence of revised_rectangular.

Theorem 4.3 *Concerning the convergence of the algorithm* revised_rectangular*, the following results hold:*

(a) If $\epsilon > 0$, then revised_rectangular *terminates in a finite number of iterations and returns an ϵ-optimal solution $\boldsymbol{x}^*$ of (2).*
(b) If $\epsilon = 0$ and revised_rectangular *terminates, then $\boldsymbol{x}^*$ is an optimal solution of (2).*
(c) Even if revised_rectangular *does not terminate, if the best-bound rule is applied to the branching process, then every accumulation point of the sequence $\{\overline{\boldsymbol{\omega}}^i\}$ is an optimal solution of (2).*

Proof Since (b) is obvious, let us prove (a) and (c).
(a) Assume that revised_rectangular does not terminate and generates the sequence of nested rectangles (5). By passing to a subsequence of K, we can assume that $\{\overline{\boldsymbol{\omega}}^i \mid i \in K\}$ converges to some $\overline{\boldsymbol{\omega}}^0 \in D$, and so $c_j^i(\overline{\omega}_j^i) - h_j(\overline{\omega}_j^i) \to \delta_j \in \mathbb{R}$ for every j, as $i \in K \to \infty$. Then we have

$$\alpha - \beta^i \leq f(\overline{\boldsymbol{\omega}}^i) - \overline{f}^i(\overline{\boldsymbol{\omega}}^i) = \sum_{j=1}^n \left(c_j^i(\overline{\omega}_j^i) - h_j(\overline{\omega}_j^i) \right) \to \sum_{j=1}^n \delta_j, \quad \text{as } i \in K \to \infty.$$

However, for each $i \in K$, we have

$$c_1^i(\overline{\omega}_1^i) - h_1(\overline{\omega}_1^i) \geq c_j^i(\overline{\omega}_j^i) - h_j(\overline{\omega}_j^i), \quad j = 1, \ldots, n,$$

which implies $\delta_j \leq 0$ for every j by Lemma 4.2, and hence $\sum_{j=1}^n \delta_j \leq 0$. This is a contradiction because the list $\mathscr{L}$ should be empty after finitely many iterations, and the algorithm should terminate.
(c) Let $\mathbf{x} \in D$ be any feasible solution of (2). Also let M denote the rectangle to which $\mathbf{x}$ belongs when M^i is selected in the branching process according to the best-bound rule. Then we have the following for each $i \in K$:

$$f(\mathbf{x}) \geq \beta^M \geq \beta^i = \overline{f}^i(\overline{\boldsymbol{\omega}}^i).$$

Since $\overline{f}^i(\overline{\boldsymbol{\omega}}^i) \to f(\overline{\boldsymbol{\omega}}^0) - \sum_{j=1}^n \delta_j$ as $i \in K \to \infty$, we have

$$f(\mathbf{x}) \geq f(\overline{\boldsymbol{\omega}}^0) - \sum_{j=1}^n \delta_j \geq f(\overline{\boldsymbol{\omega}}^0),$$

by noting $\sum_{j=1}^n \delta_j \leq 0$. Thus, $\overline{\boldsymbol{\omega}}^0$ is an optimal solution by the arbitrariness of $\mathbf{x} \in D$. □

5 Computational experiments

So far, we have assumed that the objective function f involves n univariate functions h_j's, but in fact some of them can be missing. For example, if $f(\mathbf{x}) = g(\mathbf{x}) - \sum_{j=1}^p h_j(x_j)$ for $p \leq n-1$, we only need to linearize p strictly convex functions, and obtain a lower bound

when $s_j \le x_j \le t_j$, $j = 1, \ldots, p$, from a convex minimization problem:

$$\left| \begin{array}{l} \text{minimize } \overline{f}^M(\mathbf{x}) = g(\mathbf{x}) - \displaystyle\sum_{j=1}^{p} c_j^M(x_j) \\ \text{subject to } \mathbf{x} \in D, \quad s_j \le x_j \le t_j, \quad j = 1, \ldots, p, \end{array} \right.$$

where c_j^M is defined in the same way as in (3). This relaxation corresponds to (R), and if $s_j \le x_j \le t_j$, $j = 1, \ldots, p$, are dropped from the constraint, it corresponds to our proposed relaxation ($\overline{\text{R}}$). In the branching process, we can subdivide the rectangle $M = \prod_{j=1}^{p}[s_j, t_j]$ in the p-dimensional space, instead of the n-dimensional whole space. Therefore, if p is small, even a large-scale problem might be solved in a realistic computational time.

To confirm the above, we randomly generated quadratic optimization problems of the following form, and solved them using the algorithm revised_rectangular:

$$\left| \begin{array}{ll} \text{minimization} & f(\mathbf{x}) = \dfrac{1}{2}\mathbf{x}^{\mathsf{T}}\mathbf{Q}\mathbf{x} + \theta\mathbf{c}^{\mathsf{T}}\mathbf{x} \\ \text{subject to} & \mathbf{A}\mathbf{x} \le \mathbf{e}, \quad \mathbf{x} \ge \mathbf{0}, \end{array} \right. \tag{6}$$

where θ is a positive parameter, and $\mathbf{e} \in \mathbb{R}^m$ is the all-ones vector. All entries of the last row of $\mathbf{A} \in \mathbb{R}^{m \times n}$ were set to $1/n$ to make the feasible set bounded. The remaining components of $\mathbf{A}$ were randomly generated in the interval $[-0.5, 1.0]$ so that the percentages of zeros and negative numbers were about 20% and 10%, respectively. As regard to $\mathbf{Q} \in \mathbb{R}^{n \times n}$, the tridiagonal components of the pth principal submatrix were random numbers in $[-1.0, 1.0]$, and the other components were all set to zero. Also the components of $\mathbf{c} \in \mathbb{R}^n$ were random numbers in $[-1.0, 1.0]$. The DC representation of f was given as follows:

$$g(\mathbf{x}) = f(\mathbf{x}) + h(\mathbf{x}), \quad h(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{p}\sum_{j=1}^{p}|q_{ij}|x_i^2,$$

where q_{ij} is the (i, j)-component of $\mathbf{Q}$. We also solved the following concave minimization problem derived from (6):

$$\left| \begin{array}{ll} \text{minimization} & -h(\mathbf{x}) + \theta\mathbf{c}^{\mathsf{T}}\mathbf{x} \\ \text{subject to} & \mathbf{A}\mathbf{x} \le \mathbf{e}, \quad \mathbf{x} \ge \mathbf{0}. \end{array} \right. \tag{7}$$

With slight modification to solve (6) and (7), we coded revised_rectangular in GNU Octave 4.0.0 [5], a numerical computing environment, and tested it on one core of Intel Core i7 (3.70GHz). As the procedure for solving the convex relaxation, we used the Frank-Wolfe algorithm, which was also coded from scratch in Octave. In the branching process, we adopted the depth-first rule, and in the bounding process, we replaced the pruning operation $\mathscr{P} \leftarrow \{M \in \mathscr{P} \cup \mathscr{T} \mid \alpha - \beta^M > \epsilon\}$ with

$$\mathscr{P} \leftarrow \{M \in \mathscr{P} \cup \mathscr{T} \mid \alpha - \beta^M > \epsilon|\alpha|\},$$

where ϵ was set to 10^{-5}, in order to prevent the convergence from being affected by the magnitude of the optimal value. For the purpose of comparison, we also coded the usual rectangular algorithm, which uses (R) as the convex relaxation, and refer to this code as usual_rectangular. It should be noted that usual_rectangular behaves exactly the same way as the standard algorithm in textbooks [9,10,15] for separable concave minimization problems like (7). Using these two program codes, we solved ten instances of (6) and (7) for each set of parameters m, n, p, θ, and measured their average performances.

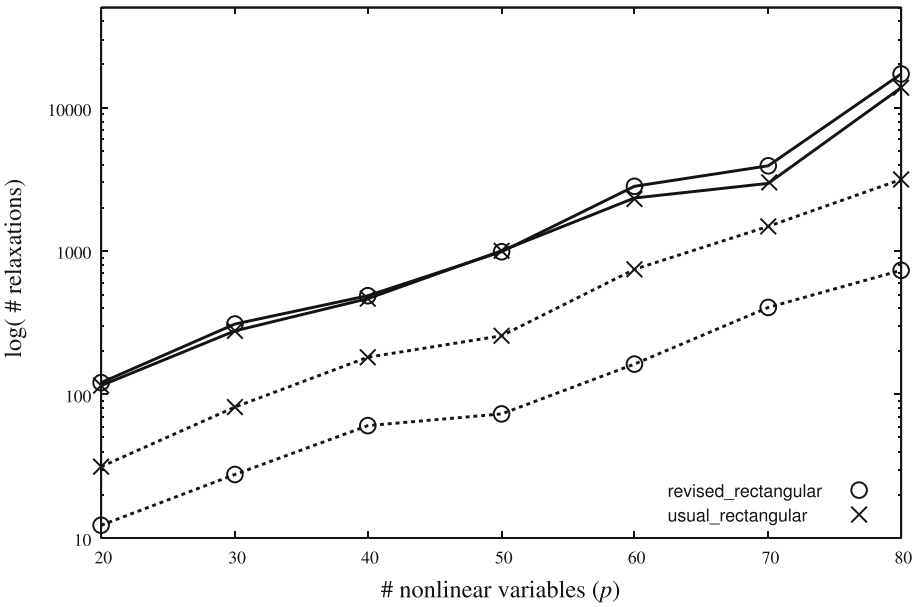

Fig. 1 Number of solved relaxations when $(m, n, \theta) = (20, 100, 5.0)$

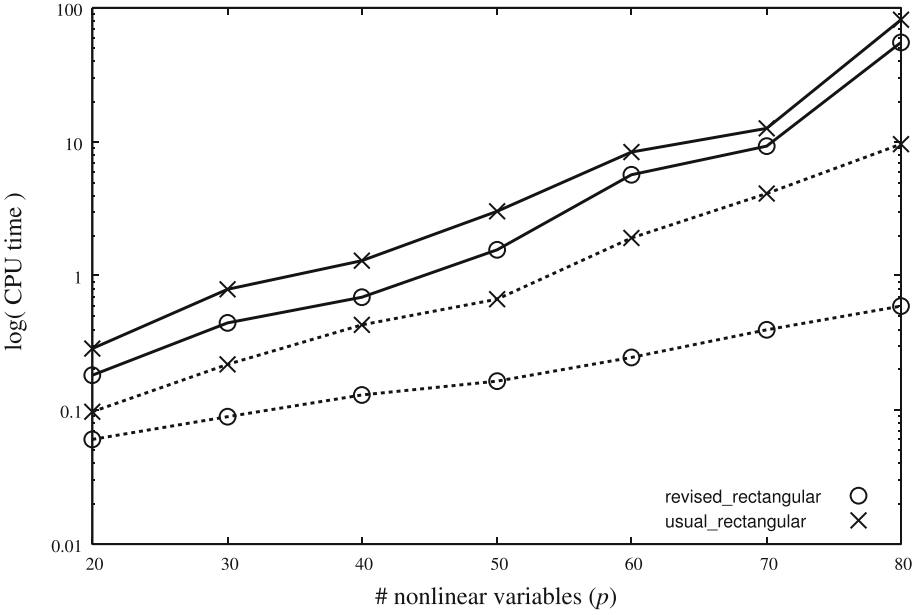

Fig. 2 CPU time (in seconds) when $(m, n, \theta) = (20, 100, 5.0)$

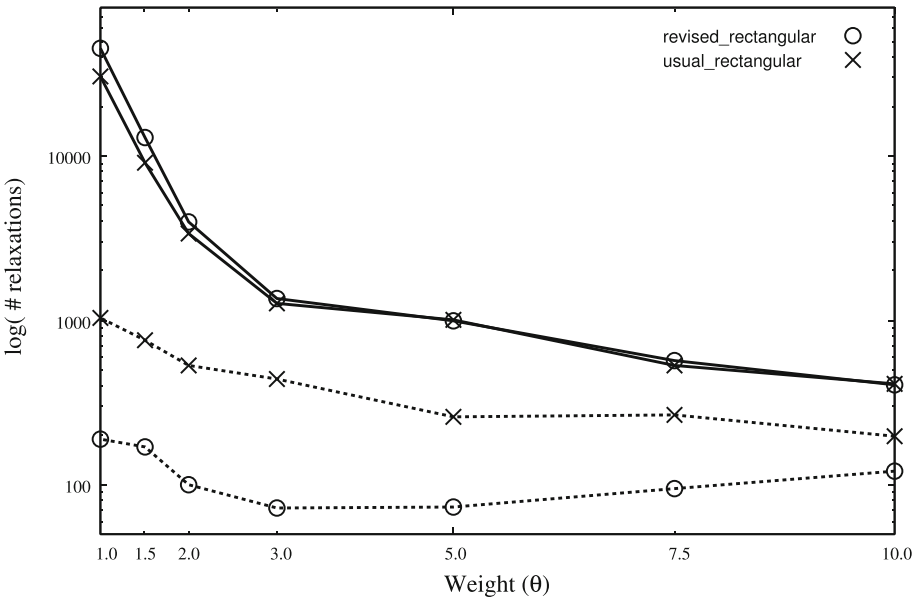

Fig. 3 Number of solved relaxations when $(m, n, p) = (20, 100, 50)$

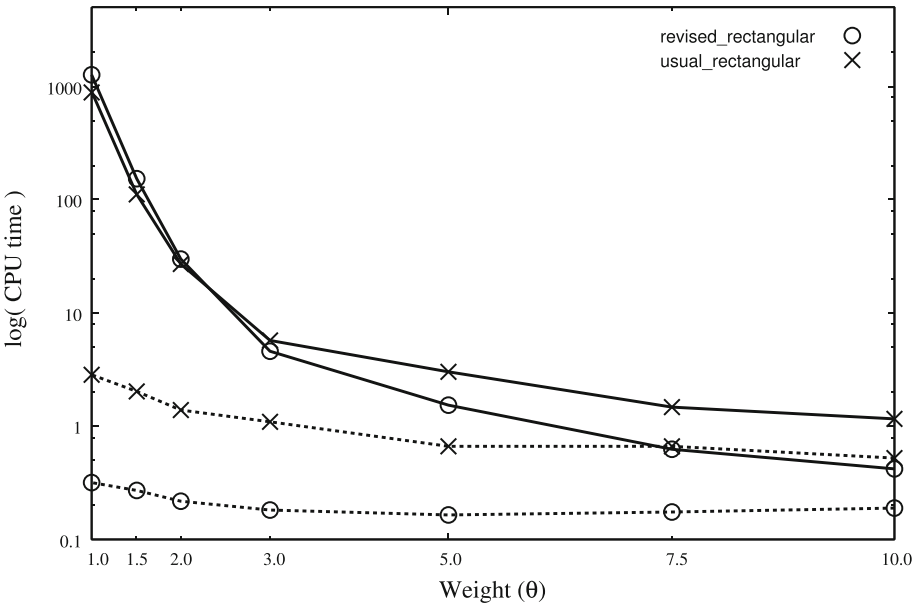

Fig. 4 CPU time (in seconds) when $(m, n, p) = (20, 100, 50)$

5.1 Numerical results

Figure 1 shows the change in the average number of convex relaxations solved in each code to process one instance when $(m, n, \theta) = (20, 100, 5.0)$, and the number p of nonlinear variables was increased by 10 from 20 to 80. The solid lines are the results for (6), and the dashed lines are the results for (7). There is little difference in the behavior between revised_rectangular and usual_rectangular for (6), and in both codes the number of solved relaxations increases exponentially with respect to p. Figure 2 shows the change in the average CPU time (in seconds) required by each code under the same conditions. We can see from this figure that revised_rectangular is fairly improved over usual_rectangular in terms of time efficiency. This indicates that the Frank-Wolfe algorithm was successfully warm-started as intended in Section 3.1. Also, comparing the solid and dashed lines, we see that (6) can be solved in two or three times the computational time taken to solve a separable concave minimization problem of the same scale using the standard algorithm. Figures 3 and 4 show the average behavior of both codes when p was fixed at 50 and the weight θ of the linear term in the objective function was increased from 1.0 to 10.0. Again, there is no difference in the number of relaxations solved in both codes for (6), but the CPU time tends to be less for revised_rectangular than for usual_rectangular, especially when θ is large.

The numerical results on larger-scale instances of (6) are summarized in Table 1, where the column labeled '#' lists the average number of solved relaxations and the column labeled '*time*' the average CPU time (in seconds) when (m, n, p) ranged up to $(60, 200, 100)$ with θ fixed at 5.0. Values in parentheses represent the standard deviation. As the problem size increases, the number of relaxations needed for revised_rectangular increases slightly faster than for usual_rectangular. However, for any size, revised_rectangular requires only about half the CPU time taken by usual_rectangular, and solves an instance of $(m, n, p) = (60, 200, 100)$ in less than ten minutes on average. Although this problem size might not be so large for DCA and heuristics, the output solution of revised_rectangular is assured with any desired accuracy, which would make it a promising algorithm for applications demanding rigorous accuracy.

5.2 Comparison with a general global optimization algorithm

Lastly, let us discuss a little about the comparison of revised_rectangular with other global optimization algorithms. To our knowledge, with the exception of revised_rectangular and usual_rectangular, no algorithm has yet been proposed aimed at globally optimizing DC functions with a separable subtrahend. For this reason, we tried to compare revised_rectangular with a global optimization algorithm for general DC optimization problems. Among such algorithms, typical ones that often appear in textbooks are the conical algorithm and the simplicial algorithm, the latter of which we chose to implement in Octave.

The simplicial algorithm first proposed in [8] is a kind of branch-and-bound algorithm, which performs branching by subdividing a simplex enclosing the feasible set D into subsimplices. In contrast to our approach, the minuend term g of the objective function f is linearized and the resulting concave underestimator is used to obtain a lower bound for f over each subsimplex. The convergence is guaranteed by any exhaustive subdivision rule, e.g., bisection which we adopted. In addition to relaxing the tolerance ϵ for pruning to 10^{-2}, we also set each subsimplex to be pruned if its longest edge is less than 10% of the initial simplex. Despite those loose settings, the performance of the simplicial algorithm for problem (6) was considerably poor, and it could only solve small instances involving up to four

Table 1 Comparison between revised_rectangular and usual_rectangular when $\theta = 5.0$

m × n		$p = 0.3n$		$p = 0.4n$		$p = 0.5n$	
		#	Time	#	Time	#	Time
40 × 100	Revised	329.6	0.500	1019	1.600	7854	14.27
		(192.9)	(0.500)	(1320)	(1.600)	(1.072×10^4)	(20.46)
	Usual	318.6	1.000	945.8	3.500	7169.0	33.93
		(202.2)	(0.633)	(1207)	(5.035)	(9802)	(54.56)
40 × 150	Revised	358.8	0.600	2001	2.900	1.323×10^4	36.40
		(259.6)	(0.490)	(1989)	(3.618)	(1.721×10^4)	(65.43)
	Usual	353.2	1.400	1689	8.200	9329	69.19
		(244.7)	(0.917)	(1480)	(7.972)	(1.086×10^4)	(98.00)
60 × 150	Revised	4317	12.50	5586	11.40	3.508×10^4	179.6
		(9570)	(32.52)	(5413)	(15.61)	(3.979×10^4)	(305.3)
	Usual	3891	29.50	4648	33.20	2.636×10^4	307.5
		(8965)	(72.99)	(4347)	(35.97)	(2.944×10^4)	(418.4)
60 × 200	Revised	1194	2.900	1.655×10^4	54.61	8.043×10^4	544.92
		(760.5)	(1.700)	(1.441×10^4)	(67.74)	(6.903×10^4)	(597.4)
	Usual	1279	9.600	1.249×10^4	153.8	4.917×10^4	863.9
		(821.1)	(5.783)	(1.208×10^4)	(194.2)	(4.053×10^4)	(827.9)

Table 2 Comparison with the simplicial algorithm when $\theta = 5.0$

rmm × n		$p = 2$		$p = 3$		$p = 4$	
		#	Time	#	Time	#	Time
	Revised	0.600	0.003	0.800	0.004	2.800	0.005
		(1.600)	(0.002)	(3.816)	(0.002)	(5.936)	(0.002)
10 × 25	Usual	0.600	0.003	1.400	0.004	2.800	0.006
		(1.281)	(0.002)	(3.105)	(0.002)	(3.816)	(0.003)
Simplicial		1640	0.783	5.593×10^4	25.27	1.583×10^6	704.5
		(1939)	(0.825)	(4.546×10^4)	(19.00)	(1.231×10^6)	(514.0)

nonlinear variables. As shown in Table 2, both revised_rectangular and usual_rectangular are incomparably superior to the simplicial algorithm in average performance, even though their settings were the same as before. However, the performance of the simplicial algorithm is expected to improve by strengthening the lower bound. In that case, the utility of the simplicial algorithm would be significantly enhanced because it does not require any extra structure for the objective function. Next time, we will report on a revision to the simplicial algorithm elsewhere.

Acknowledgements The author would like to thank the anonymous referees for their valuable comments, which significantly improved the quality of this article.

References

1. Argyriou, A., Hauser, R., Miccheli, C.A., Pontil, M.: "A DC-programming algorithm for kernel selection", *Proceedings of the 33rd International Conference on Machine Learning*, 41–48 (2006)
2. Bertsekas, D.P.: Nonlinear Programming, 3rd edn. Athena Scientific, Belmont (2016)
3. Falk, J.E., Soland, R.M.: An algorithm for separable nonconvex programming problems. Manag. Sci. **15**, 550–569 (1969)
4. Frank, M., Wolfe, P.: An algorithm for quadratic programming. Naval Res. Logistics Quarterly **3**, 95–110 (1956)
5. GNU Octave, http://www.gnu.org/software/octave/
6. Hiriart-Urruty, J.-B.: "Generalized differentiability, duality and optimization for problems dealing with differences of convex functions", *Lecture Notes in Economics and Mathematical Systems* **256**, Springer, Berlin, 37–69 (1985)
7. Hoai-An, L.T., Le, H.M., Nguyen, V.V., Tao, P.D.: A DC programming approach for feature selection in support vector machines learning. Adv. Data Anal. Classification **2**, 259–278 (2008)
8. Horst, R., Dien, L.V.: A solution concept for a very general class of decision problems. In: Optiz, O., Rauhat, B. (eds.) Ökonomie und Mathematik, pp. 139–149. Springer, Berlin (1987)
9. Horst, R., Pardalos, P.M., Thoai, N.V.: Introduction to Global Optimization. Kluwer Academic Publishers, Dordrecht (1995)
10. Horst, R., Tuy, H.: Global Optimization: Deterministic Approaches, 3rd edn. Springer, Berlin (1996)
11. Soland, R.M.: Optimal facility location with concave costs. Operations Research **22**, 373–382 (1974)
12. Tao, P.D., Hoai-An, L.T.: Convex analysis approach to DC programming: theory, algorithms and applications. Acta Mathematica Vietnamica **22**, 289–355 (1997)
13. Tuy, H.: Convex programs with an additional reverse convex constraint. J. Optimiz. Theory and Appl. **52**, 463–486 (1987)
14. Tuy, H.: D.C. optimization: theory, methods and applications. In: Horst, R., Pardalos, P.M. (eds.) Handbook of Global Optimization, pp. 149–216. Springer, Berlin (1995)
15. Tuy, H.: Convex Analysis and Global Optimization, 2nd edn. Springer, Berlin (2016)
16. Yang, S., Yuan, L., Lai, Y.C., Shen, X., Wonka, P.: "Feature grouping and selection over an undirected graph", *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 922–930 (2006)