Computers and Operations Research 82 (2017) 95–101

Contents lists available at ScienceDirect

Computers and Operations Research

journal homepage: www.elsevier.com/locate/cor

A continuous DC programming approach for resource allocation in OFDMA/TDD wireless networks

Nguyen Canh Nam[1,*], Pham Thi Hoai

School of Applied Mathematics and Informatics, Hanoi University of Science and Technology, No1 Dai Co Viet Road, Ha Noi, Viet Nam

ARTICLE INFO

Article history:
Received 30 September 2015
Revised 2 January 2017
Accepted 19 January 2017
Available online 23 January 2017

Keywords:
DC programming
DCA
Pure 0-1 linear programming
Exact penalty techniques
Branch-and-Bound
WiMAX

ABSTRACT

The next generation broadband wireless networks deploys OFDM/OFDMA as the enabling technologies for broadband data transmission with QoS capabilities. Many optimization problems have arisen in the conception of such a network. This article studies an optimization problem in resource allocation. By using mathematical modeling technique we formulate the considered problem as a pure integer linear program. This problem is reformulated as a DC (Difference of Convex functions) program via an exact penalty technique. We then propose a continuous approach for its resolution. Our approach is based on DC programming and DCA (DC Algorithm). It works in a continuous domain, but provides integer solutions. To check globality of computed solutions, a global method combining DCA with a well adapted Branch-and-Bound (B&B) algorithm is investigated. Preliminary numerical results are reported to show the efficiency of the proposed method with respect to the standard Branch-and-Bound algorithm.

1. Introduction

Orthogonal Frequency Division Multiple Access (OFDMA) transmission scheme is becoming popular as it is the key technology for the fourth generation (4G) broadband wireless networks such as WiMAX. In OFDMA scheme data can be transmitted in both *time domain* and *frequency domain* simultaneously. From the network operator point of view, it is very important to utilize the channel resources effectively as the available radio resources become more and more scarce while revenue should be maintained or increased. In general, there has been a tremendous opportunity to improve the spectral efficiency while providing fairness and meeting Quality of Service (QoS) requirements of all users at the same time [1,2].

Thus, one functionality should be deployed in the MAC (Media Access Control) layer of the OFDMA wireless systems in order to ensure spectral efficiency, fairness and QoS, namely the resource allocation function. The resource allocation problem is to allocate time slots on a subset of the subcarriers available (frequency resource) to meet client demands and maximize the system throughput. Difficulties of such an allocation come from the limit of available resources while we need to serve different users as well as technology requirements should be satisfied. Different users should be assigned to different sub-channels at a slot of time. In addition the standard specifies that a data burst in the downlink subframe needs to be mapped into a time and frequency domain with a rectangular shape.

Until now there are several efforts in research community attempting to maximize the efficiency in an OFDMA/TDD (Time Division Duplexing) frame [3–6]. However, in OFDMA/TDD schemes, we find that the research on these previous mentioned issues is not sufficient. We then propose an efficient continuous approach to solve this problem.

In this work we develop a deterministic continuous approach based on DC programming and DCA. The contributions of the paper are 3-fold:

Firstly, we propose a classical mathematical model for the considered problem. By introducing the binary variables (the allocation variables) we formulate the resource allocation problem as a pure 0-1 linear program. Due to the large dimension and huge number of constraints of this problem in practice, the standard methods in combinatorial optimization such as Branch-and-Bound, Branch-and-Cut or cutting plane cannot give a globally optimal solution or even a solution *closed* to a globally optimal solution. Attempting to develop robust numerical solution approaches, we try to reformulate the problem as a DC program.

Secondly, we investigate an exact penalty technique to reformulate the pure 0-1 linear program into a continuous optimization problem that is in fact a DC program.

* Corresponding author.
E-mail addresses: nam.nguyencanh@hust.edu.vn (N.C. Nam), hoai.phamthi@hust.edu.vn (P.T. Hoai).

[1] This research is funded by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 101.01-2017.16.

http://dx.doi.org/10.1016/j.cor.2017.01.011
0305-0548/

Thirdly, we investigate DC programming and DCA for solving a related DC program.

DC programming and DCA were introduced by Pham Dinh Tao in 1985, as an extension of his earlier subgradient algorithms for concave programming, and extensively developed by Le Thi Hoai An and Pham Dinh Tao since 1994. The DCA has been successfully applied to real world nonconvex programs in different fields of Applied Sciences, to which it quite often gave global solutions and proved to be more robust and more efficient than related standard methods, especially in large scale settings. It is worth noting that DCA is one of the rare efficient algorithms for nonsmooth nonconvex programming which allows solving large-scale DC programs (see [7–13]).

DC programing and DCA ([9,12,13] and references therein) aim to solve a general DC program of the form

$$\alpha = \inf\left\{f(x) := g(x) - h(x) : x \in \mathbb{R}^p\right\} \tag{1.1}$$

where g, h are lower semicontinuous proper convex functions on $\mathbb{R}^p$. Such a function f is called a DC function, and $g - h$, a DC decomposition of f while g and h are the DC components of f. The construction of DCA involves convex DC components g and h but not the function f itself : each iteration k of DCA consists of computing

$$y^k \in \partial h(x^k), \quad x^{k+1} \in \arg\min\{g(x) - h(x^k) - \langle x - x^k, y^k\rangle : x \in \mathbb{R}^p\}. \tag{1.2}$$

Hence, for a DC program, each DC decomposition corresponds to a different version of DCA. Since a DC function f has infinitely many DC decompositions which have crucial impacts on the qualities (speed of convergence, robustness, efficiency, globality of computed solutions, etc.) of DCA, the search for a "relevant" DC decomposition is important from algorithmic point of view. Finding a "good" initial point is then also an important stage of DCA. How to develop an efficient algorithm generic DCA scheme for a practical problem is thus a judicious question to be studied, and the answer depends on the specific structure of the problem being considered. In the current work, using an appropriate DC decomposition we propose a DCA scheme which is very inexpensive, in term of CPU time, thanks to the rapidity of the algorithm for solving subproblem (1.2).

To globalize the local DCA, its combination with a customized B&B is investigated. The combined algorithm allows checking globality of solutions computed by DCA and restarting it if necessary, and, consequently, accelerates the B&B approach. This combination has been applied successfully to solve different classes of problems in various domains such that binary quadratic programming, supply chain design problem [14–16]. These successes motivated us to investigate it for solving problem in resource allocation in wireless network.

The remainder of the paper is organized as follows. In Section 2, we report the problem description and the mathematical formulation of the considered problem, which is a pure 0-1 program. Section 3 describes how to reformulate the problem in the continuous form via an exact penalty technique. Section 4 is devoted to the DC programming and DCA for solving the penalty equivalent. The combined DCA-B&B algorithm is presented in Section 5, while the computational results are reported in Section 6.

2. Problem description and mathematical formulation

2.1. Problem description

Consider an OFDMA/TDD frame which contains K users, M sub-channels and N time-slots. Each user will attain his maximum efficiency if he is allocated suitable resources, i.e., with right sub-channel and in right time. But there are, of course, many conflicts within users, see Fig. 1.

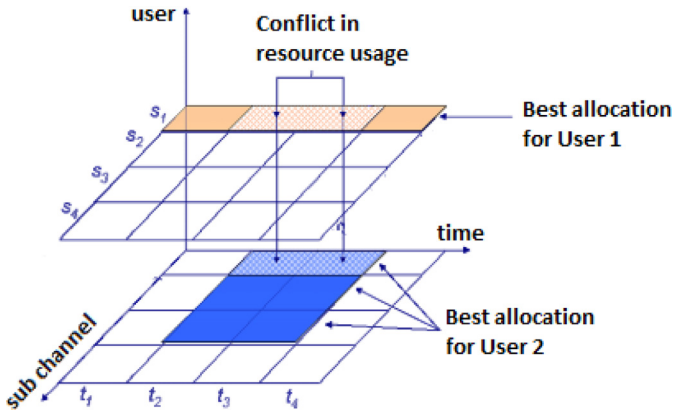

Fig. 1. OFDMA/TDD frame in which resource conflict occurs so often.

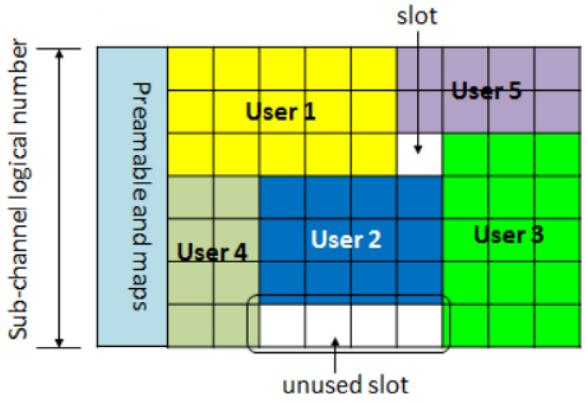

Fig. 2. Rectangular design in a frame.

Let b_{ijk}, $1 \le i \le M$, $1 \le j \le N$, $1 \le k \le K$, be the number of bit data that user k can send if he is provided i sub-channel and time-slot j, our efficiency measure. Our problem is to allocate the resources in a frame such that maximize the total efficiency. Nevertheless, such an allocation should satisfy the following conditions:

- At a slot of time and for a specific sub-channel there is at most one user (to avoid conflicts).
- All resource allocation for users should be done with shape of rectangle (IEEE802.16e standard on WiMAX network), see Fig. 2.

2.2. Pure 0-1 formulation

In the following we will formulate the above problem as a pure 0-1 linear programming.

If we denote

$$x_{ijk} \in \{0, 1\} \text{ for } 1 \le i \le M, 1 \le j \le N, 1 \le k \le K \tag{2.1}$$

with convention

$$x_{ijk} = \begin{cases} 1 & \text{if user } k \text{ is provided sub-channel } i \text{ at time-slot } j \\ 0 & \text{otherwise} \end{cases},$$

then the total data that can be transfered is

$$f(x) := \sum_{k=1}^{K}\sum_{i=1}^{N}\sum_{j=1}^{M} b_{ijk}x_{ijk}. \tag{2.2}$$

In order to avoid conflicts, at a time-slot j and in one sub-channel i, there is at most one user. Hence we introduce the

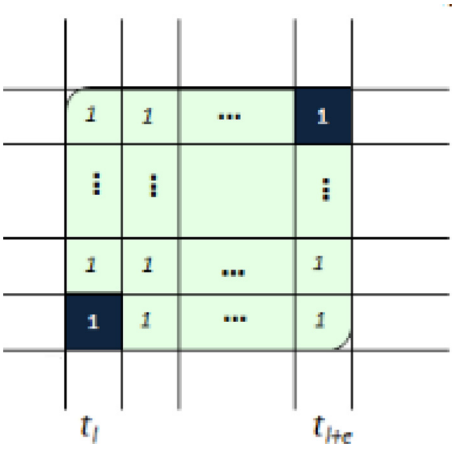

Fig. 3. Allocation for one user if he is provided twos nodes in the frame.

following constraint

$$\sum_{k=1}^{K} x_{ijk} \le 1 \quad \forall i = \overline{1,M}, \quad \forall j = \overline{1,N}. \tag{2.3}$$

In order to get rectangular allocation we use a technique similar to the well-known one in optimization called "Big-M" [17]. Suppose that we have $T+2$ binary variables y, z and $x_1, x_2, \ldots, x_T$. Then the following constraint ensures, when $y = z = 1$, $x_1 = x_2 = \ldots = x_T = 1$,

$$T(y+z-1) \le \sum_{i=1}^{T} x_i \Leftrightarrow T(y+z-1) - \sum_{i=1}^{T} x_i \le 0$$

Clearly, if y and z are not concurrently equal to one then this constraint does not force any value for x_i, $i = 1, 2, \ldots, T$.

Because resource allocated for a user should be in a rectangle form, if two nodes in the frame are assigned for a user k then all the nodes in the rectangle made by those two nodes will be assigned for this user too, see Fig. 3.

Hence we have some conditions like if $x_{i_1 j_1 k^*} = x_{i_2 j_2 k^*} = 1$, $i_1, i_2 \in \{1, \ldots, M\}$; $j_1, j_2 \in \{1, \ldots, N\}$; $(i_1, j_1) \ne (i_2, j_2)$, for some $k^* \in \{1, 2, \ldots, K\}$ then

$$x_{ijk^*} = 1, \quad \forall \min\{i_1, i_2\} \le i \le \max\{i_1, i_2\},$$
$$\min\{j_1, j_2\} \le j \le \max\{j_1, j_2\}.$$

Following the above technique, we introduce the constraint

$$(|i_1 - i_2| + 1)(|j_1 - j_2| + 1)(x_{i_1 j_1 k} + x_{i_2 j_2 k} - 1) - \sum_{i \in I_{i_1 i_2}} \sum_{j \in J_{j_1 j_2}} x_{ijk} \le 0, \tag{2.4}$$

where

$$I_{i_1 i_2} = \{i : \min\{i_1, i_2\} \le i \le \max\{i_1, i_2\}\},$$
$$J_{j_1 j_2} = \{j : \min\{j_1, j_2\} \le j \le \max\{j_1, j_2\}\}.$$

This process will be run repeatedly over all the rectangle parts of the frame and for all users.

Finally, we arrive in the optimization problem with the objective function (2.2) and the constraints (2.3), the system of constraints (2.4) and (2.1).

This problem is a pure 0-1 problem and hence a nonconvex program. Its difficulty is from the fact that the number of variables and the number of constraints increase so fast with respect to the number of users, the number of sub-channels and the number of time-slots. The number of binary variables and the number of constraints are calculated by the following formulas.

The number of binary variables is

$$M \cdot N \cdot K.$$

The number of constraints is

$$M \cdot N + K \cdot \frac{MN(MN-1)}{2}.$$

We then deal with a problem in the form of a large-scale pure 0-1 linear program.

3. Concave minimization reformulation

In this section, using the well-known results concerning the exact penalty [18,19], we will formulate the considered pure 0-1 problem in the form of concave minimization programming.

Denote D as the set of feasible points x determined by the system of constraints {(2.3), (2.4)}. It is clear that D is a nonempty, bounded polyhedral convex set in IR^n with $n = K \cdot M \cdot N$. The considered problem can be expressed in the form

$$\alpha = \min \left\{ c^T x : x \in D, x \in \{0,1\}^n \right\}. \tag{3.1}$$

Let us consider function p defined by $p(x) = \sum_{i=1}^{n} p_i(x_i) = \sum_{i=1}^{n} \min\{x_i, 1 - x_i\}$.

Set $S := \{x \in D: x \in [0, 1]^n\}$. It is clear that p is concave and finite on S, $p(x) \ge 0$ for all $x \in S$, and

$$\left\{ x \in D : x \in \{0,1\}^n \right\} = \{x \in S : p(x) \le 0\}.$$

Hence Problem (3.1) can be rewritten as

$$\alpha = \min \left\{ c^T x : x \in S, p(x) \le 0 \right\}. \tag{3.2}$$

From Theorem 3.1 below we get, for a sufficiently large number τ ($\tau > \tau_0$), the equivalent concave minimization problem to (3.1):

$$\min \{c^T x + \tau p(x) : x \in S\}. \tag{3.3}$$

Theorem 3.1. *(Theorem 2, [18]) Let S be a nonempty bounded polyhedral convex set, f be a finite concave function on S and p be a finite nonnegative concave function on S. Then there exists $\tau_0 \ge 0$ such that, for all $\tau > \tau_0$, the following problems have the same solution set and the same optimal value :*

$$(P_\tau) \qquad \alpha(\tau) = \inf\{f(x) + \tau p(x) : x \in S\}$$
$$(P) \qquad \alpha = \inf\{f(x) : x \in S, p(x) \le 0\}.$$

More precisely if the vertex set of S, denoted by $V(S)$, is contained in $\{x \in S, p(x) \le 0\}$, then $\tau_0 = 0$, otherwise $\tau_0 = \min\{\frac{f(x)-\alpha(0)}{s} : x \in S, p(x) \le 0\}$, where $S := \min\{p(x): x \in V(S), p(x) > 0\} > 0$.

Remark 3.1 (Practical choice of the penalty parameter. $\tau > \tau_0$ and its use in our combined DCA and BB)

In general, it is difficult to compute explicitly any upper bound of τ_0 in Problem (3.3). In practice, we take τ sufficiently large in order for Problem (3.3) to be equivalent to Problem (3.1). To check the equivalence of these problems, we use the exact penalty results in Theorem 3.1: $\alpha(\tau) \le \alpha$ for every $\tau \ge 0$ and if an optimal solution to Problem (3.3) with given $\overline{\tau} \ge 0$ is feasible to Problem (3.1) then it is also an optimal solution of the latter one and $\overline{\tau} \ge \tau_0$.

4. DCA for solving Problem (3.3)

4.1. Outline of DC programming and DCA

Pham Dinh Tao in 1985 introduced DC Algorithm for DC programming which constitutes the backbone of (smooth or nonsmooth) nonconvex programming and global optimization. These theoretical and algorithmic tools have been extensively developed, since 1994, by Le Thi Hoai An and Pham Dinh Tao to become now classic and increasingly popular (see [9,12,13,20], and references therein). DC programming addresses the problem of minimizing a

function f which is a difference of convex functions on the whole space $\mathbb{R}^p$ or on a convex set $C \subset \mathbb{R}^p$. Generally speaking, a DC program takes the form

$$\alpha := \inf\left\{ f(x) := g(x) - h(x) \mid x \in \mathbb{R}^p \right\}, \tag{4.1}$$

with g, h being lower semicontinuous proper convex functions on $\mathbb{R}^p$. Such a function f is called DC function, and $g - h$, DC decomposition of f while g and h are DC components of f. Note that the closed convex constraint $x \in C$ can be included in the objective function of (4.1) by using the indicator function on C denoted by χ_C which is defined by $\chi_C = 0$ if $x \in C$, and $+\infty$ otherwise :

$$\inf\{f(x) := g(x) - h(x) \mid x \in C\} = \inf\{\chi_C + g(x) - h(x) \mid x \in \mathbb{R}^p\}.$$

A DC program (4.1) is called a polyhedral DC program when either g or h is a polyhedral convex function (i.e., the sum of the indicator function of a polyhedral convex set and the supremum of a finite collection of affine functions).

Recall that, for a convex function φ, the subdifferential of φ at $x' \in \operatorname{dom}\varphi := \{x \in \mathbb{R}^p \mid \varphi(x) < +\infty\}$, denoted by $\partial\varphi(x')$, is defined by

$$\partial\varphi(x') := \{y \in \mathbb{R}^p \mid \varphi(x) \geq \varphi(x') + \langle x - x', y\rangle, \forall x \in \mathbb{R}^p\}$$

The subdifferential $\partial\varphi(x')$ generalizes the derivative in the sense that φ is differentiable at x' if and only if $\partial\varphi(x') = \left\{\nabla_x\varphi(x')\right\}$.

A point x^* is said to be *a local minimizer* of $g - h$ if $g(x^*) - h(x^*)$ is finite and there exists a neighbourhood $\mathcal{B}$ of x^* such that

$$g(x^*) - h(x^*) \leq g(x) - h(x), \quad \forall x \in \mathcal{B}. \tag{4.2}$$

The necessary local optimality condition for (primal) DC program (4.1) is given by

$$\emptyset \neq \partial h(x^*) \subset \partial g(x^*). \tag{4.3}$$

The condition (4.3) is also sufficient, for local optimality, for polyhedral DC program with h being polyhedral convex [9].

The DCA (see [9,12,21] and reference therein) is based on local optimality conditions and duality in DC programming (by reason of simplicity we omit here the presentation of DC duality). The main idea of DCA is simple: each iteration k of DCA approximates the concave part $-h$ by its affine majorization and minimizes the obtaining convex function. Precisely, each iteration k of a general DCA consist of computing

$$y^k \in \partial h(x^k), \quad x^{k+1} \in \arg\min\{g(x) - h(x^k) - \langle x - x^k, y^k\rangle : x \in \mathbb{R}^p\}. \tag{4.4}$$

In fact, there is not only one DCA, but infinitely many DCAs for a considered DC program. DCA's properties rely upon the fact that DCA deals with the convex DC components g and h but not with the DC function f itself. The fact is crucial for nonconvex nonsmooth programs for which DCA is one of rare effective algorithms. The solution of a practical nonconvex program by DCA must have two stages : the search of an appropriate DC decomposition and the search of a good initial point. An appropriate DC decomposition, in our sense, is the one that corresponds to a DCA, which is not expensive and has interesting convergence properties.

In the next subsection, we will develop an instance of DCA to solve the Problem (3.3), and study the convergence properties of the proposed algorithm.

4.2. DCA for solving Problem (3.3).

Clearly that (3.3) is a DC program with the following DC decomposition

$$g(x) = \chi_S(x) \quad \text{and} \quad h(x) = -c^T x - \tau \sum_{i=1}^{n} \min\{x_i, 1 - x_i\} \tag{4.5}$$

where $\chi_S(x) = 0$ if $x \in S$, $+\infty$, otherwise (the indicator function on S).

Performing DCA for Problem (3.3) amounts to computing the two sequences $\{x^k\}$ and $\{y^k\}$ defined by

$$y^k \in \partial h(x^k), \quad x^{k+1} \in \arg\min\{g(x) - h(x^k) - \langle x - x^k, y^k\rangle : x \in \mathbb{R}^p\}.$$

As usually, ∂h is often explicitly computed with the help of known rules in convex analysis. We have

$$\partial h(x^k) = \partial(-c^T x^k) + \tau\,\partial(-p)(x^k) \tag{4.6}$$

where $\partial(-c^T x^k)$ and $\partial(-p)(x^k)$ can be calculated explicitly. Indeed,

$$\partial(-c^T x^k) = -c,$$

and since

$$-p(x^k) = \sum_{i=1}^{n} \max\{-x_i^k, x_i^k - 1\}$$

hence

$$\partial h(x^k) = -c + d \text{ where } d_i = \begin{cases} -\tau & \text{if } x_i^k < 0.5 \\ \tau & \text{otherwise} \end{cases} \text{ for } i = 1, 2, \ldots, n. \tag{4.7}$$

As for finding x^{k+1}, we need only to solve the following linear program

$$\min\left\{-\langle x, y^k\rangle : x \in S\right\}. \tag{4.8}$$

Remark 4.1. The sequence $\{x^k\}$ can be then included in the vertex set $V(S)$ of S. This property constitutes one of the interesting advantages of DC decomposition (4.5), knowing that $V(S)$ contains the feasible set of (3.1).

We are now in a position to describe the DCA applied to Problem (3.3))

Algorithm 4.1 (DCA applied to (3.3))**.**

Let $\epsilon > 0$ be small enough and x^0. Set $k \leftarrow 0$; $er \leftarrow 1$.
while $er > \epsilon$ **do**
  Calculate $y^k \in \partial h(x^k)$ via (4.7).
  Calculate x^{k+1} by solving the linear program (4.8)
  $er \leftarrow \min\left\{\dfrac{||x^{k+1} - x^k||}{\|x^k\| + 1}, \dfrac{|f(x^{k+1}) - f(x^k)|}{|f(x^k)| + 1}\right\}$
  $k \leftarrow k + 1$
endwhile
x^{k+1} is the computed solution.

Remark 4.2. Performing of DCA amounts to solve a series of linear programming problems.

The convergence of Algorithm 4.1 can be summarized in the next theorem whose proof is essentially based on the convergence theorem of a DC polyhedral program [9,12,21].

Theorem 4.1. *(Convergence properties of Algorithm 4.1)*

(i) *Algorithm 4.1 generates a sequence $\{x^k\}$ contained in $V(S)$ such that the sequence $c^T x + \tau\, p(x)$ is decreasing.*
(ii) *There exists a nonnegative number τ_1 such that for every $\tau > \tau_1$ the sequence $\{p(x^k)\}$ is decreasing. In particular, if, at iteration r, x^r is a feasible solution of (3.1), then x^k, for $k \geq r$, is feasible too.*
(iii) *The sequence $\{x^k\}$ converges to $x^* \in V(S)$ after a finite number of iterations. The point x^* is a critical point of Problem (3.3). Moreover, if $x_i^* \neq \frac{1}{2}$ for $i = 1, \ldots, n$, then x^* is a locally optimal solution to Problem (3.3).*

Remark 4.3. According to Theorem 4.1, let us emphasize the key features of DCA applied to (3.3): ($\{x^k\}$ being generated by DCA applied to (3.3) with $\tau > \max\{\tau_0, \tau_1\}$)

(i) Both sequences $\{c^T x^k + \tau p(x^k)\}$ and $\{p(x^k)\}$ are decreasing.
(ii) If x^r is feasible for (3.1) then x^k, for $k \geq r$, is feasible too. In this case, sequence $\{x^k\}$ moves in the feasible set of (3.1), ($x^k \in D$, $x^k \in \{0, 1\}^n$), while decreasing the objective function.

These nice properties have great impacts on the search of binary solutions.

5. A global optimization based on DC programming approach

We will globally solve the obtained optimization problem by a customized Branch-and-Bound (B&B) method. Linear relaxation is applied to compute lower bounds while the upper bounds are determined by applying DCA to (3.3). Our combined algorithm can be summarized as follows : starting with the rectangle $R_0 := [0, 1]^n$, we consider at each iteration $k \geq 0$ the rectangle R_k corresponding to the smallest lower bound β_k. The selected rectangle R_k is divided into two subrectangles $R_{k_i}, i = 0, 1$ and the lower bound is improved by solving the corresponding linear programs. The upper bound γ_k is determined by applying the DCA to (3.3). The procedure is determined when $\gamma_k - \beta_k \leq \epsilon$ and it provides an $\epsilon-$optimal solution of (3.1).

Algorithm 5.1 (The combined algorithm)**.** Let $R_0 = [0, 1]^n$. Set $\gamma_0 := +\infty, \beta_0 := -\infty, restart := true, \mathcal{R} := \{R_0\}$, and $k = 0$. Let ϵ be a sufficiently small positive number.

1. Let R_k be the rectangle such that $\beta_k = \beta(R_k) = \min\{\beta(R) : R \in \mathcal{R}\}$. Bisect R_k into two subrectangles R_{k_0} and R_{k_1} via the index j^*
$$R_{k_i} = \{x \in R_k : x_{j^*} = i\}, \quad i = 0, 1.$$
2. Compute the lower bound $\beta_{k_i} (i = 0, 1)$ by solving the linear relaxation problems corresponding to the set R_{k_i}.
3. If ($restart = true$) then update γ_k, the best upper bound of the optimal value of (3.1) by applying DCA to Problem (3.3) from a suitable starting point found in Step 2. Update
$$\mathcal{R} \leftarrow \mathcal{R} \backslash \{R_t : \beta(R_t) \geq \gamma_k - \epsilon, R_t \in \mathcal{R}\}.$$
4. If $\mathcal{R} = \emptyset$ (i.e., $\gamma_k - \beta_k \leq \epsilon$), then STOP, the $\epsilon-$optimal solution is x^k that verifies $c^T x^k = \gamma_k$, otherwise update
$$\mathcal{R} \leftarrow \mathcal{R} \cup \{R_{k_i} : \beta(R_{k_i}) < \gamma_k - \epsilon, i = 0, 1\} \backslash R_k$$
and go to step 1.

The combined algorithm differs from the classical B&B scheme by Step 3 in which DCA is investigated. Here *restart* is a boolean variable which takes value *true* when we decide to restart DCA. We will discuss in detail this procedure in Section 5.2. As in several DC programs, DCA provides a global solution to (3.3) (and so is to (3.1)) from a good starting point (detail will be found in Section 5.1). We will see in the computational experiments that DCA provides a global solution after first two iterations of the B&B algorithm. Nevertheless, we must continue the branch and bound process to improve the lower bound until it is closed enough to the best upper bound. In fact, the B&B algorithm is presented to get a good starting point for DCA and to check the globality of DCA.

5.1. Finding a good starting point for DCA

Theorem 4.1 says that, starting with a feasible solution of the pure 0-1 linear program (3.1), DCA provides a better one, although it works on the continuous feasible set of (3.3). It is so important to find a good feasible point to (3.1) for restarting DCA.

During the Branch and Bound procedure, we can restart DCA from the best feasible solution to (3.1) that is discovered while computing lower bounds. This is motivated by a similar and efficient way introduced in the combined DCA-Branch and Bound algorithm for nonconvex quadratic programming [14].

A good feasible point can be found by applying beforehand DCA to the concave programming problem as in [22]

$$0 = \min\left\{\sum_{i=1}^{n} \min\{x_i, 1 - x_i\}\right\}. \tag{5.1}$$

Problem (5.1) is a polyhedral DC program with known optimal value and whose solution set is exactly the feasible set F of (3.1). Fortunately, as for linear complementarity problems [22,23], DCA, with starting point $\bar{x}$ not necessarily feasible but satisfy $p(\bar{x}) \leq 0$, converges, almost always in practice, to a global solution of (5.1).

5.2. When do we restart DCA?

During the branch and bound process, we restart DCA when a feasible solution to (3.1) improving the best current upper bound is pointed out. Usually, an upper bound is obtained when a binary solution is found, we call this upper bound (the value of f of this solution a *Score*). However, by using the exact penalty technique, $c^T x + \tau p(x)$, with $x \in K$, is also an upper bound and is denoted by UB_f. So, in our algorithm, we restart DCA when the following condition is satisfied

$$c^T \bar{x} + \tau p(\bar{x}) < \min\{\text{Score}, UB_f\}(1 + 10e^{-3}), \tag{5.2}$$

where $\bar{x}$ is an optimal solution of the current relaxed linear program (computing lower bound).

Remark 5.1. Since the value of τ is large, so *Score* is often smaller than UB_f .

5.3. Branching procedure using binary subdivision

Using exact penalty techniques, we obtain equivalent DC programs in the continuous framework which make possible the application of the local continuous approach DCA. In the branching procedure of our branch and bound, we turn the binary character (of the variable in the binary linear program (3.1)) to take advantage in replacing the bisection of a chosen kth edge [0, 1] with the binary one : $x_k = 0$ and $x_k = 1$. The procedure is detailed in the following:

Suppose that, at a node in our Branch and Bound scheme, we have to solve

$$\begin{aligned} &\min c^T x \\ \text{subject to} \quad &x_i \in D \\ &x_i = 0 \quad i \in I, x_i = 1 \quad i \in J \\ &x_i \in \{0, 1\} \quad i \in \{0, 1, \ldots, n\} \backslash (I \cup J). \end{aligned} \tag{5.3}$$

Let x^R be an optimal solution of the linear relaxation of (5.3).

If $x^R \in \{0, 1\}^n$, i.e., $p(x^R) = 0$ then x^R is also an optimal solution of (5.3). Otherwise, there exists some j^* such that $x^R_{j^*} \in (0, 1)$. Then we replace Problem (5.3) with two subproblems by setting $x^R_{j^*} = 0$ and $x^R_{j^*} = 1$, respectively.

The index j^* is chosen such that the gaps between $p_j(x^R_j) = \min\{x^R_j, 1 - x^R_j\}$ and its convex hull on [0, 1] - which is identical to zero - is maximum with respect to $j = 1, \ldots, n$, i.e.,

$$\max_j \{\min\{x^R_j, 1 - x^R_j\}\} = \min\{x^R_{j^*}, 1 - x^R_{j^*}\}. \tag{5.4}$$

We now describe the combined DCA-Branch and Bound algorithm for globally solving Problem (3.1).

5.4. Global algorithm (DCA-BB)

Let $R_0 := [0,1]^n$. Let ϵ be a sufficient small positive number.
Solve the linear relaxation problem of (3.1) to obtain an optimal solution x^{R_0} and the first lower bound $\beta_0 := \beta(R_0)$.
Solve (3.3) by DCA from the starting point x^{R_0} to obtain $x_\tau^{R_0}$.
If $x_\tau^{R_0}$ is feasible to (3.1) **then** set $\gamma_0 := c^T x_\tau^{R_0}$ and set $x^0 = x_\tau^{R_0}$ **else** set $\gamma_0 := +\infty$.
If $(\gamma_0 - \beta_0) \le \epsilon|\gamma_0|$ **then** x^0 is an $\epsilon-$optimal solution of (3.1) **else** set $\mathcal{R} := \{R_0\}$, $k := 0$.
While stop = false **do**
Select a rectangle R_k such that $\beta_k = \beta(R_k) = \min\{\beta(R) : R \in \mathcal{R}\}$.
Bisect R_k into two subrectangles R_{k_0} and R_{k_1} via the index j^*, chosen by (5.4).
$R_{k_i} = \{x \in R_k : x_{j^*} = i, i = 0, 1\}$.
Solve the subproblems (P_{k_i}) to obtain $\beta(R_{k_i})$ and $x^{R_{k_i}}$:

$$\beta(R_{k_i}) := \min\{c^T x : x \in K \cap R_{k_i}\} \quad (i = 0, 1) \qquad \text{(Pki)}$$

If $x^{R_{k_i}}$ is the best feasible solution to (3.1) **then** update γ_k and the best feasible solution $x^k$by applying DCA to (3.3) from $x^{R_{k_i}}$.
End if
If the condition of restarting DCA (5.2) is satisfied **then**
Apply DCA to (5.1) from $x^{R_{k_i}}$ to obtain $x_\tau^{R_{k_i}}$
If $x_\tau^{R_{k_i}}$ is feasible to (3.1) and $c^T x_\tau^{R_{k_i}} < \gamma_k$ **then**
Update upper bound $\gamma_k = c^T x_\tau^{R_{k_i}}$ and solution $x^k = x_\tau^{R_{k_i}}$
End if
End if
Update the list of rectangles $\mathcal{R} \leftarrow \mathcal{R} \cup \{R_{k_i} : \beta(R_{k_i}) < \gamma_k - \epsilon, i = 0, 1\} \setminus \left\{R_k \cup \{R_j : \beta(R_j) \ge \gamma - \epsilon, R_j \in \mathcal{R}\}\right\}$
If $\mathcal{R} = \emptyset$ **then** STOP, x^k is an ϵ-optimal solution **else** $k \leftarrow k + 1$
End if
End while

The correctness and the convergence of the algorithm are stated in the following result whose proof is fairly standard from the branching procedure and the bounding one [24,25]. Its finiteness is due to the binary subdivision used in the algorithm.

Proposition 5.1. *Algorithm DCA-BB terminates after finitely many iterations and yielding an $\epsilon-$optimal solution of Problem (3.1).*

6. Computational experiments

The algorithms were implemented on a PC Intel Core i3, CPU 2.2 GHz, 4G RAM, in C++. To solve the linear programs in relaxation procedure and in DCA, we used CLP solver, a very famous open source solver from COIN-OR (www.coin-or.org).

In order to evaluate the performance of the proposed algorithms as well as the efficiency of DCA, we randomly generated 10 examples in which the number of binary variables is increased by changing the number of users, number of sub-channels and number of slots of time. In this experiment, we compare the efficiency of our DCA-BB algorithm with the classical Brand and Bound algorithm. For all test problems, we always try to find out an ϵ-optimal solution, with $\epsilon \le 5.10^{-2}$. We limit the algorithms in number of iteration to 10^5, i.e, after 10^5 iterations the problem will be considered as failed in finding an $\epsilon-$optimal solution. The results of our algorithm in each case are summarized in Table 1.

Some notations used in Table 1:

#Var :	Number of variables
#Constr :	Number of constraints
#Iter :	Number of iterations
UB :	Upper bound
LB :	Lower bound
N°F :	the 'th'-restarting DCA in DCA-BB at which DCA provides the first feasible solution to (3.1)

Comments on the numerical simulations : From the results in the tables above we observe that

- The DCA with restarting provides the first feasible solution to (3.1) very rapidly, at most after 2 restarting times.
- DCA is inexpensive and can so handle problems with large number of binary variables. The superiority of DCA-Branch and Bound relative to the Branch and Bound algorithm increases when the number of binary variables increases. DCA-Branch and Bound is fast for large-scale problems while Branch and Bound algorithm is quite slow or it cannot solve some problems in reasonable time.

7. Conclusion

We considered an important problem in wireless network conception, the resource allocation problem. By using modeling techniques we have presented the considered problem in the form of pure 0-1 linear program. The binary character of the variable in this problem and exact penalty in DC programming allow reformulating the obtained problem into an appropriate equivalent polyhedral DC program. That leads to a very simple DCA, consisting of simply solving a finite number of linear programs with the same constraint set. This is the first time a deterministic optimization approach is investigated in the literature for solving this problem. This constitutes an interesting contribution of the paper.

Preliminary numerical simulations show that the combination DCA-Branch and Bound is efficient. Its success is due to DCA, which is featured by the fact: Although being a continuous approach, DCA generates a finite sequence of binary solutions with decreasing objective values. DCA gives $\epsilon-$optimal solutions in almost cases.

Table 1
Numerical results.

No.	M	N	K	#Var	#Constr	B&B				DCA - BB				
						#Iter	UB	LB	Time(s)	#Iter	UB	LB	Time(s)	N°F
1	3	3	4	36	280	5	4981	4981	1.4	2	4981	4981	0.9	1
2	4	4	3	48	1215	11	2267	2267	0.5	2	2267	2267	0.3	1
3	4	4	4	64	1615	5	16,528	16,500	1.4	2	16,543	16,500	1	1
4	3	3	6	54	416	39	40	40	8.8	17	40	40	1.4	1
5	4	4	5	80	2015	797	876,753	876,624	25.4	95	876,654	876,534	23.4	2
6	4	4	6	96	2415	49	5962	5950	65.6	38	5962	5871	35.4	2
7	5	5	4	100	6024	123	9564	9514	792.4	37	9587	9432	76.8	2
8	6	6	3	108	13,055	137	3757	3743	1023.2	33	3789	3641	151.4	1
9	5	4	6	120	4699	–	–	–	–	399	12,367	12,356	1500	2
10	5	5	5	125	7524	–	–	–	–	278	98,794	98,765	1809	2

– : Failed in finding optimal solution.

They confirm the practical observations concerning DCA: DCA is inexpensive and can be applied to large-scale DC programs to which it gives quite often optimal solutions, while starting from a good initial point. Finally, the combined DCA-Branch and Bound will be able to handle large-scale instances if its number of iterations remains in certain reasonable limits. Otherwise, thanks to its inexpensiveness, DCA still works well to find good local binary solutions, but we do not have any more means of checking their globality. We hope that the DCA will be useful for people having to solve real-life problems. Their computational results will make it possible to strongly appreciate the robustness and effectiveness of the DCA.

Acknowledgments

The authors would like to thank two referees for their useful comments and suggestions which have considerably improved the presentation of the revised paper.

References

[1] Rodrigues EB, Casadevall F. Control of the trade-off between resource efficiency and user fairness in wireless networks using utility-based adaptive resource allocation. IEEE Commun Mag 2011;49:90–8.
[2] Ali-Yahiya T, Beylot A-L, Pujolle G. Downlink resource allocation strategies for ofdma based mobile wimax. Telecommun Syst 2010;44:29–37.
[3] Bacioccola A, Cicconett C, Lenzini L, Mingozzi E, Erta A. A downlink data region allocation algorithm for ieee 802.16e ofdma. In: Proc. 6th int. conf. information, communications and signal processing; 2007. p. 1–5.
[4] Wand T, Feng H, Hu B. Two-dimensional resource allocation for ofdma system. In: Proc. IEEE int. conf. communications workshop. Beijing, China; 2008. p. 1–5.
[5] Rodrigues E, Casadevall F, Sroka P, Moretti M, Dainelli G. Resource allocation and packet scheduling in ofdma-based cellular networks. In: Proc. 4th international conference on cognitive radio oriented wireless networks and communications; 2009. p. 1–6.
[6] Cicconetti C, Lenzini L, Lodi A, Martello S, Mingozzi E, Monaci M. Efficient two-dimensional data allocation in ieee 802.16 ofdma. IEEE/ACM Trans Networking 2014;22(5):1645–58.
[7] An LTH, Tao PD. Solving a class of linearly constrained indefinite quadratic problems by dc algorithms. J Global Optim 1997;11:253–85.
[8] An LTH, Tao PD. Large scale molecular optimization from distance matrices by a dc optimization approach. SIAM J Optim 2003;14(1):77–117.
[9] An LTH, Tao PD. The dc (difference of convex functions) programming and dca revisited with dc models of real world nonconvex optimization problem. Ann Oper Res 2005;133:23–46.
[10] An LTH, Tao PD, Nam NC. Local and global approaches based on dc programming, branch-and-bound and sdp techniques for nonconvex quadratic programming. Tech. Rep.. National Institute for Applied Sciences, Rouen; 2005.
[11] Nam NC. Approches locales et globales basées sur la programmation dc & dca et les techniques b&b avec relaxation sdp pour certaines classes des programmes non convexes. simulations numériques et codes à l'usage industriel. National Institute for Applied Sciences, Rouen, France; 2007.
[12] Tao PD, An LTH. Convex analysis approach to dc programming: theory, algorithms and applications. Acta Mathematica Vietnamica 1997;22:289–355. dedicated to Professor Hoang Tuy on the occasion of his 70th birthday
[13] Tao PD, An LTH. Dc optimization algorithms for solving the trust region subproblem. SIAM J Optim 1998;8(2):476–505.
[14] An LTH, Tao PD. A continuous approach for globally solving linearly constrained quadratic zero-one programming problems. Optimization 2001;50:93–120.
[15] An LTH, Phuc NT, Tao PD. A continuous dc programming approach to the strategic supply chain design problem from qualified partner set. Eur J Oper Res 2007;183(3):1001–12.
[16] Tao PD, Nam NC, An LTH. An efficient combined dca and b& b using dc/sdp relaxation for globally solving binary quadratic programs. J Global Optim 2010;48:595–632.
[17] Griva I, Nash SG, Sofer A. Linear and nonlinear optimization. 2nd ed. Society for Industrial Mathematics; 2009. ISBN 978-0-89871-661-0.
[18] An LTH, Tao PD, Muu LD. Exact penalty in dc programming. Vietnam J Math 1999;27:1216–31.
[19] An LTH, Tao PD, Ngai HV. Exact penalty and error bounds in dc programming. J Global Optim 2012;52:509–35.
[20] An LTH, Tao PD. Large scale molecular optimization from distances matrices by a dc optimization approach. SIAM J Optim 2003;14(1):77–116.
[21] An LTH, Tao PD. Dc programming: theory, algorithms and applications. the state of the art. In: Proceedings (containing refereed contributed papers) of the first international workshop on global constrained Optimization and Constraint Satisfaction (Cocos' 02). Valbonne-Sophia Antipolis, France; 2002. 26 pages.
[22] An LTH, Tao PD. Dc programming approaches and dca for globally solving linear complementarity problems. Tech. Rep.. National Institute for Applied Sciences, Rouen; 2004.
[23] Tao PD, An LTH, Francois A. Combining dca and interior point techniques for large-scale nonconvex quadratic programming. Optim Methods Software 2008;23(4):609–29.
[24] Phong TQ, An LTH, Tao PD. Decomposition branch and bound method for globally solving linearly constrained indefinite quadratic minimization problems. Oper Res Lett 1995;17:215–22.
[25] Phong TQ, An LTH, Tao PD. On the global solution of linearly constrained indefinite quadratic minimization problems by decomposition branch and bound method. RAIRO, Recherche Operationelle 1996;30(1):31–49.