European Journal of Operational Research 255 (2016) 151–162

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Decision Support

A branch-and-Benders-cut method for nonlinear power design in green wireless local area networks

Bernard Gendron[a,*], Maria Grazia Scutellà[b], Rosario G. Garroppo[c], Gianfranco Nencioni[c], Luca Tavanti[c]

[a] CIRRELT and DIRO, Université de Montréal, Montréal, Canada
[b] Dipartimento di Informatica, Università di Pisa, Pisa, Italy
[c] Dipartimento di Ingegneria dell'Informazione, Università di Pisa, Pisa, Italy

ARTICLE INFO

Article history:
Received 6 July 2015
Accepted 29 April 2016
Available online 4 May 2016

Keywords:
Integer programming
Benders decomposition
Branch-and-cut
Green wireless local area network
Network design

ABSTRACT

We consider a problem arising in the design of green wireless local area networks. Decisions on powering-on a set of access points (APs), via the assignment of one power level (PL) to each opened AP, and decisions on the assignment of the user terminals (UTs) to the opened APs, have to be taken simultaneously. The PL assigned to an AP affects, in a nonlinear way, the capacity of the connections between the AP and the UTs that are assigned to it. The objective is to minimize the overall power consumption of the APs, which has two components: location/capacity dimensioning costs of the APs; assignment costs that depend on the total demands assigned to the APs. We develop a branch-and-Benders-cut (BBC) method where, in a non-standard fashion, the master problem includes the variables of the Benders subproblem, but relaxes their integrality. The BBC method has been tested on a large set of instances, and compared to a Benders decomposition algorithm on a subset of instances without assignment costs, where the two approaches can be compared. The computational results show the superiority of BBC in terms of solution quality, scalability and robustness.

1. Introduction

We address an optimization problem arising in the design of green (or energy-saving) wireless local area networks (WLANs). We focus on the design of efficient reconfiguration algorithms to reduce the power consumption of the WLAN infrastructure when the load is scarce. Most of the currently deployed enterprise WLANs are continuously operated at full power, i.e., all access points are always turned on with the transmission power set to the maximum. This produces a considerable waste of energy, because the same power is employed at the peak hours and during the off peak periods. We address this issue by proposing an optimization model that is used to take two kinds of decisions: (i) associate each user with one of the available access points and (ii) set the transmission power level of each access point.

The area of wireless network design requires the development of optimization models and methods (see Kennington, Olinick, & Rajan, 2010 for an overview of the main challenges in this field). Recent contributions include the development of a nested Benders decomposition method (Naoum-Sawaya & Elhedhli, 2010) that combines classical Benders decomposition (Benders, 1962) with combinatorial Benders decomposition (Codato & Fischetti, 2006); and the derivation of pure 0–1 programming formulations, tightened with strong valid inequalities (D'Andreagiovanni, Mannino, & Sassano, 2013) (based on the Ph.D. Thesis of the first author D'Andreagiovanni, 2012).

The problem we consider is defined on a bipartite network structure, with a set of access points (APs) that must be assigned to user terminals (UTs) in order to satisfy the user demands, without exceeding the capacity of the connections between the APs and the UTs. Each UT must be assigned to exactly one powered-on AP. Several different power levels (PLs) are available for powering on each AP. If an AP is powered-on, then exactly one PL must be associated with it.

A key issue arises concerning the capacity of the connections between the APs and the UTs: the specific PL assigned to a (powered-on) AP affects, in a nonlinear way, the capacity of the connections between the AP and the UTs assigned to it. The only assumption is that the transmission capacity between a UT and an AP is a nonnegative nondecreasing function of the radiated power at the AP, which will be formally defined in Section 2. As a result, the optimization model is an integer

* Corresponding author. Tel.: +1 5143437093.
E-mail addresses: gendron@iro.umontreal.ca, Bernard.Gendron@cirrelt.ca (B. Gendron), scut@di.unipi.it (M.G. Scutellà), Rosario.Garroppo@iet.unipi.it (R.G. Garroppo), Gianfranco.Nencioni@iet.unipi.it (G. Nencioni), Luca.Tavanti@iet.unipi.it (L. Tavanti).

http://dx.doi.org/10.1016/j.ejor.2016.04.058
0377-2217/

nonlinear program, a class of notoriously difficult mathematical programs.

The objective is to minimize the overall power consumption of the APs, which has two components. The first component includes the location and capacity dimensioning costs of the APs, i.e., the costs associated with powering-on APs and assigning a PL to each of them. The second component concerns the assignment costs between UTs and APs, which are given by a linear dependency between the power consumed by the APs and the total demands assigned to the APs. In an earlier contribution by the same authors (Gendron, Garroppo, Nencioni, Scutellà, & Tavanti, 2013), it was assumed that the power consumed by an AP does not depend on the demands assigned to that AP and, therefore, only the first component of the objective function was considered. The presence of the second component, called *UT assignment costs*, yields a more realistic problem formulation that also complicates the development of a solution method, as discussed below.

We propose to address the intrinsic difficulties of the problem, i.e., nonlinear capacity constraints and complex objective function, by developing an exact algorithm inspired by Benders decomposition (Benders, 1962). Since the Benders subproblem in our approach is a 0–1 program and not a linear program (LP), we use *canonical cuts* (Balas & Jeroslow, 1972), as in logic-based Benders decomposition (Hooker & Ottosson, 2003) and combinatorial Benders decomposition (Codato & Fischetti, 2006), instead of the classical LP duality-based Benders cuts. The resulting Benders cuts are improved by simple arguments (also used in (Gendron et al., 2013)) based on the assumption that the transmission capacity functions are nondecreasing.

In a non-standard fashion, our master problem includes the variables of the Benders subproblem, but relaxes their integrality. This feature provides a simple, yet efficient, way to consider the UT assignment costs in the master problem to ensure that effective lower bounds are computed. Linear approximations of the nonlinear transmission capacity functions are also included in the formulation of the master problem. As a result, the master problem is a mixed-integer linear programming (MILP) relaxation, which we solve with a state-of-the-art MILP software tool (a similar approach that solves a nonlinear integer program through the addition of Benders cuts in a MILP relaxation can be found in (Zhang, Romero, Beck, & Amon, 2013)). Instead of solving one MILP master problem at every iteration, we use a *branch-and-Benders-cut (BBC) method*, also called Benders-based branch-and-cut method, where a single branch-and-cut (B&C) tree is constructed and the Benders cuts are added during the exploration of the B&C tree.

In the constraint programming (CP) literature, this algorithmic scheme is known as *branch-and-check* (Thorsteinsson, 2001) and has been the object of empirical work comparing it to a more traditional iterative Benders decomposition approach (Beck, 2010). In these CP references, the Benders subproblem is solved with CP and the Benders cuts are based on the notion of inference dual, introduced in logic-based Benders decomposition (Hooker & Ottosson, 2003). In the operations research (OR) literature, BBC has attracted the attention of many researchers recently, as it makes better use of the reoptimization capabilities of the MILP solvers than the classical Benders decomposition approach. This is discussed, for instance, in (Naoum-Sawaya & Elhedhli, 2013), which uses an interior-point method to solve a Benders reformulation in a BBC framework, applying it to facility location and network design problems. Other recent implementations of the BBC method include: (Fortz & Poss, 2009), which compares BBC to classical Benders decomposition for a multi-layer network design problem, showing significant speedups on average; (de Camargo, de Miranda Jr., & Ferreira, 2011), which combines the generation of outer approximation and Benders cuts in a BBC method for the single allocation hub location problem under congestion; (Botton, Fortz, Gouveia, & Poss, 2013), where a BBC method is used to solve a hop-constrained survivable network design problem; (Adulyasak, Cordeau, & Jans, 2013), which uses BBC algorithms for solving production routing problems under demand uncertainty. In all these OR references, the Benders subproblem is an LP and the Benders cuts are based on LP duality, as in the approach originally proposed by Benders (1962). As mentioned above, our Benders subproblem is a 0–1 program and we make use instead of canonical cuts (Balas & Jeroslow, 1972). Canonical cuts in a wireless network design problem (different from ours) have also been used in (Naoum-Sawaya & Elhedhli, 2010). A major difference between existing contributions and our paper is that our master problem includes the variables of the Benders subproblem, but relaxes their integrality. In the above references, a traditional partitioning of the variables into master problem variables and subproblem variables is used. Note that this traditional partitioning has been questioned recently in the context of stochastic programming (Crainic, Hewitt, & Rei, 2014), where a new approach called partial decomposition was proposed, in which a subset of scenario subproblems are kept in the master problem.

This paper is a follow-up on an earlier contribution by the same authors (Gendron et al., 2013) on the special case without UT assignment costs, for which a Benders decomposition method has been proposed. This method corresponds to a cutting-plane approach where feasibility cuts are iteratively added to the master problem, thanks to the information provided when solving the Benders subproblem. The latter is a feasibility problem, because of the absence of UT assignment costs. This is in contrast with the Benders subproblem defined in the present paper, which is an optimization problem, given the inclusion of UT assignment costs. This is a major difference, as the presence of such additional assignment costs prevents a straightforward extension of the Benders decomposition approach used in (Gendron et al., 2013), as we clarify in Section 3.5. Another notable difference is that the master problem in (Gendron et al., 2013) does not include the variables of the Benders subproblem. The decomposition adopted in (Gendron et al., 2013) thus follows a traditional variable partitioning approach as in the original Benders method (Benders, 1962), where the variables of the master problem and those of the subproblem do not overlap. In Section 4, we compare the performance of the two methods on the special case without UT assignment costs addressed in (Gendron et al., 2013). The computational results show the superiority of the proposed BBC approach in terms of solution quality, scalability and robustness.

The paper is organized as follows. In Section 2, we describe the problem, which we denote as GWLANP, and we present the integer nonlinear programming model we propose for the GWLANP. The BBC method is described in Section 3. Computational results from experiments on randomly generated realistic instances are reported in Section 4. The conclusion summarizes our findings and identifies promising research directions.

2. Problem description and formulation

In order to state the GWLANP in a formal way, we need to characterize the energy consumed by the powered-on APs and the capacity of the connections between the APs and the UTs. First, let us denote by $\mathcal{I}$, $\mathcal{J}$ and $\mathcal{K}$ the sets of UTs, APs and PLs, respectively.

Concerning the energy consumed by the powered-on APs, the power consumed by $j \in \mathcal{J}$ is composed of a fixed component and of two variable components. The fixed component, denoted p_0, is bound to the mere fact that the device is powered-on, and therefore, it encompasses AC/DC conversion, basic circuitry powering, dispersion, etc. The first variable power component associated with $j \in \mathcal{J}$ is given by its radiated power π_j, which depends on the PL assigned to $j \in \mathcal{J}$. More precisely, if $k \in \mathcal{K}$ is assigned

to $j \in \mathcal{J}$, then we have $\pi_j = p_k$, where p_k denotes the power provided by $k \in \mathcal{K}$. Regarding the second variable power component, it linearly depends on the total demands assigned to $j \in \mathcal{J}$, denoted T_j. Therefore, the energy consumed by a powered-on AP $j \in \mathcal{J}$ is given by $p_0 + \pi_j + \mu_j T_j$, where μ_j is a proportionality coefficient.

With regard to the capacity of the connections between the APs and the UTs, a key issue is that the specific PL assigned to a powered-on AP affects, in a nonlinear way, the capacity of the connections between the AP and the UTs assigned to it. In general, the capacity function can be determined by means of two steps (as reported, for example, in (Garroppo, Gendron, Nencioni, & Tavanti, 2014)). First, the received power is estimated starting from the transmitted power and the path loss model, which takes into account the propagation properties of the considered network scenario. Examples of path loss models can be found in (European Commission, 1999). Then, the capacity can be extracted from the computed signal-to-noise ratio (SNR) and other parameters, such as the specific modulation and coding schemes, as well as the medium access overhead. Examples of curves reporting the relation between the capacity and the SNR can be found in (Li, Fan, & Kaleshi, 2012; Wang, Zhang, Wu, Zhang, & Ni, 2014). In addition, an experimental study where the "capacity vs. transmitted power" curves are estimated for different system configurations is presented in (Huehn & Sengul, 2012).

Here, the only assumption we make is that the transmission capacity between $i \in \mathcal{I}$ and $j \in \mathcal{J}$, denoted $r_{ij}(\pi_j)$, is a nonnegative nondecreasing function of the radiated power π_j. This assumption is widely supported by both theoretical and experimental studies, such as the ones cited above. In practice, the transmission capacity function satisfies the following conditions:

- There exists a threshold $\gamma_{ij} > 0$ such that $r_{ij}(\pi_j) = 0$ if $\pi_j \leq \gamma_{ij}$ and $r_{ij}(\pi_j) > 0$ whenever $\pi_j > \gamma_{ij}$. Thus, $j \in \mathcal{J}$ can only be assigned to $i \in \mathcal{I}$ when its radiated power π_j remains above γ_{ij}.
- $r_{ij}(\pi_j) \leq r_{max}$ for any π_j, where r_{max} is the maximum rate achievable by any physical connection.

In all instances used in our computational experiments (see Section 4), we use the following piecewise linear transmission capacity function:

$$r_{ij}(\pi_j) = \begin{cases} 0, & \text{if } \pi_j \leq \gamma_{ij}, \\ \min\{\alpha_{ij}\pi_j, r_{max}\}, & \text{otherwise}, \end{cases} \tag{1}$$

where α_{ij} denotes a transmission loss factor between $j \in \mathcal{J}$ and $i \in \mathcal{I}$. It is important to note, however, that our BBC method does not depend on this particular function and can be generalized to any nonnegative nondecreasing transmission capacity function. The BBC method only requires an upper linear approximation $r^u_{ij}(\pi_j)$ to $r_{ij}(\pi_j)$. In the case of the function used in our instances, we simply use $r^u_{ij}(\pi_j) = \alpha_{ij}\pi_j$.

In the GWLANP, the decisions to be taken are what APs to power-on, how to assign a PL to each powered-on AP and how to assign exactly one powered-on AP to each UT. Such decisions must be taken in such a way as to satisfy the demand w_i for each $i \in \mathcal{I}$, by respecting the nonlinear transmission capacities between APs and UTs. As indicated above, the objective is to minimize the overall power consumption of the powered-on APs. The problem can be seen as a discrete location problem, where the capacity to assign to each location (which is the power level in this context) also has to be decided. In other words, the GWLANP is a particular case of a broader class of location-design problems, where both location and capacity dimensioning decisions must be taken.

To model the GWLANP, we define the following sets of binary variables:

- $x_{ij} = 1$, if AP $j \in \mathcal{J}$ is assigned to UT $i \in \mathcal{I}$; 0, otherwise; (*UT assignment variables*)
- $y_{jk} = 1$, if PL $k \in \mathcal{K}$ is assigned to AP $j \in \mathcal{J}$; 0, otherwise. (*PL assignment variables*)

Given the definitions of these variables, we derive the following relationships for the radiated power of $j \in \mathcal{J}$ and for the total demands assigned to $j \in \mathcal{J}$, respectively: $\pi_j = \sum_{k\in\mathcal{K}} p_k y_{jk}$ and $T_j = \sum_{i\in\mathcal{I}} w_i x_{ij}$. The model can then be written as follows:

$$z(GWLANP) = \min \sum_{j\in\mathcal{J}} \left\{ \sum_{k\in\mathcal{K}} (p_0 + p_k) y_{jk} + \sum_{i\in\mathcal{I}} \mu_j w_i x_{ij} \right\} \tag{2}$$

$$\sum_{j\in\mathcal{J}} x_{ij} = 1, \quad i \in \mathcal{I}, \tag{3}$$

$$\sum_{k\in\mathcal{K}} y_{jk} \leq 1, \quad j \in \mathcal{J}, \tag{4}$$

$$x_{ij} \leq \sum_{k\in\mathcal{K}} y_{jk}, \quad i \in \mathcal{I}, j \in \mathcal{J}, \tag{5}$$

$$\sum_{i\in\mathcal{I}|r_{ij}(\pi_j)>0} \frac{w_i x_{ij}}{r_{ij}(\pi_j)} \leq 1, \quad j \in \mathcal{J}, \tag{6}$$

$$x_{ij} \in \{0,1\}, \quad i \in \mathcal{I}, j \in \mathcal{J}, \tag{7}$$

$$y_{jk} \in \{0,1\}, \quad j \in \mathcal{J}, k \in \mathcal{K}. \tag{8}$$

The objective (2) is to minimize the total power consumption, which depends on the powering-on decisions, on the power levels assigned to the powered-on APs, and on the total demands assigned to the powered-on APs. Eq. (3) are the single assignment constraints that impose that exactly one AP must be assigned to each UT. Inequalities (4) impose that at most one PL can be selected for each AP. Inequalities (5) ensure that an AP cannot be assigned to any UT if the AP is powered-off. Inequalities (6) are the capacity constraints for each AP. Relations (7) and (8) define the integrality of the variables. Note that, given that at most one PL can be chosen for each AP, it is not necessary to associate further binary variables with the APs in order to state the powering-on decisions, since such decisions are captured by the terms $\sum_{k\in\mathcal{K}} y_{jk}$. This is why the fixed power cost p_0 is part of the cost associated with the y_{jk} variables in the objective function.

Note that the problem considered in (Gendron et al., 2013) can be seen as a special case of the GWLANP where, for each $j \in \mathcal{J}$, $\mu_j = \mu \geq 0$, a proportionality coefficient that is constant over all APs. In that case, the UT assignment costs can be removed from the objective function, since

$$\sum_{j\in\mathcal{J}} \sum_{i\in\mathcal{I}} \mu w_i x_{ij} = \mu \sum_{i\in\mathcal{I}} w_i \left(\sum_{j\in\mathcal{J}} x_{ij} \right) = \mu \sum_{i\in\mathcal{I}} w_i,$$

i.e., the UT assignment costs are the same, irrespective of the solution. We call this special case the *GWLANP without UT assignment costs*.

3. The branch-and-Benders-cut method

In this section, we present the BBC method for solving the GWLANP. Sections 3.1 and 3.2 describe the master problem and the Benders subproblem, respectively. The different types of Benders cuts added during the course of the algorithm are introduced in Section 3.3. Section 3.4 gives a formal statement of the BBC algorithm, as well as a proof of convergence. Finally, Section 3.5 is dedicated to an extensive comparison between the BBC method and the Benders decomposition algorithm proposed in (Gendron et al., 2013) for the GWLANP without UT assignment costs.

3.1. Master problem

In order to solve the nonlinear model (2)–(8), we propose a BBC method. As in classical Benders decomposition, the approach consists in solving a master problem, which is a relaxation of model (2)–(8), to which we gradually add Benders cuts. In a non-standard way, our master problem involves both the PL assignment variables y_{jk} and the UT assignment variables x_{ij}. The master problem is denoted M_{xy} and is initially defined as:

$$z(M_{xy}) = \min \sum_{j\in\mathcal{J}} \left\{ \sum_{k\in\mathcal{K}} (p_0 + p_k) y_{jk} + \sum_{i\in\mathcal{I}} \mu_j w_i x_{ij} \right\} \tag{9}$$

subject to (3), (4), (5), (8), and

$$\sum_{i\in\mathcal{I}|r^u_{ij}(\pi_j)>0} \frac{w_i x_{ij}}{r^u_{ij}(\pi_j)} \leq 1, \quad j \in \mathcal{J}, \tag{10}$$

$$x_{ij} \in [0,1], \quad i \in \mathcal{I}, \quad j \in \mathcal{J}. \tag{11}$$

Constraints (10) define a relaxation of the nonlinear capacity constraints (6) obtained by replacing functions $r_{ij}(\pi_j)$ by upper linear approximations $r^u_{ij}(\pi_j)$. With the transmission capacity function given in Eq. (1) and its linear upper approximation $r^u_{ij}(\pi_j) = \alpha_{ij}\pi_j$, constraints (10) would take the following form:

$$\sum_{i\in\mathcal{I}} \frac{w_i x_{ij}}{\alpha_{ij}} \leq \sum_{k\in\mathcal{K}} p_k y_{jk}, \quad j \in \mathcal{J}.$$

Constraints (11) define the UT assignment variables as continuous between 0 and 1. Together, these two sets of constraints, along with constraints (3), (4), (5), (8), define a MILP relaxation of model (2)–(8). During the course of the BBC algorithm, Benders cuts are gradually added to the master problem, as we see below.

3.2. Benders subproblem

The master problem is solved by a B&C method implemented in a state-of-the-art MILP solver (we use CPLEX, version 12.6.1). Each time an integer solution $\bar{y}$ is obtained during the exploration of the B&C tree, we solve the following Benders subproblem, denoted $S_x(\bar{y})$:

$$z(S_x(\bar{y})) = \min \sum_{j\in\bar{\mathcal{J}}} \left\{ \sum_{k\in\mathcal{K}} (p_0 + p_k) \bar{y}_{jk} + \sum_{i\in\mathcal{I}} \mu_j w_i x_{ij} \right\} \tag{12}$$

$$x_{ij} \in \{0,1\}, \quad i \in \mathcal{I}, j \in \bar{\mathcal{J}}, \tag{13}$$

$$\sum_{j\in\bar{\mathcal{J}}|\bar{r}_{ij}>0} x_{ij} = 1, \quad i \in \mathcal{I}, \tag{14}$$

$$\sum_{i\in\mathcal{I}|\bar{r}_{ij}>0} \frac{x_{ij} w_i}{\bar{r}_{ij}} \leq 1, \quad j \in \bar{\mathcal{J}} \tag{15}$$

$$x_{ij} = 0, \quad (i,j) \in \mathcal{I} \times \bar{\mathcal{J}} | \bar{r}_{ij} = 0, \tag{16}$$

where $\bar{\mathcal{J}} \subseteq \mathcal{J}$ is the set of APs that are powered-on according to $\bar{y}$ (i.e., $\sum_{k\in\mathcal{K}} \bar{y}_{jk} = 1$), while $\bar{r}_{ij} = r_{ij}(\sum_{k\in\mathcal{K}} p_k \bar{y}_{jk})$ is the capacity of the connection between $i \in \mathcal{I}$ and $j \in \bar{\mathcal{J}}$ induced by the power level assignment given by $\bar{y}$. Note that the integrality of the x_{ij} variables is now reimposed in the Benders subproblem and that the capacity constraints (15) are now linear. Also observe that inequalities (15) have, in general, the structure of knapsack constraints, which implies that $S_x(\bar{y})$ cannot be solved as an LP. In fact, this Benders subproblem has the structure of a generalized assignment problem, which can be solved by specialized algorithms (see, for instance, Pigatti, Poggi de Aragão, and Uchoa, 2005 and the references therein). In our implementation, we use the same state-of-the-art MILP software tool as when solving the master problem. Since $S_x(\bar{y})$ is not an LP, we cannot use LP duality-based Benders cuts, and we rely instead, as explained below, on the canonical cuts for the unit hypercube, studied in (Balas & Jeroslow, 1972), which are also used in logic-based Benders decomposition (Hooker & Ottosson, 2003) and combinatorial Benders decomposition (Codato & Fischetti, 2006).

3.3. Benders cuts

If $S_x(\bar{y})$ is feasible, and $\bar{x}(\bar{y})$ is the computed optimal solution, then a feasible solution $(\bar{x}(\bar{y}), \bar{y})$ to the original nonlinear formulation (2)–(8) has been determined. If the corresponding objective function value $z(S_x(\bar{y}))$ is better than the value of the current best feasible solution, denoted z^*_u, then both z^*_u and the best feasible solution are suitably updated. Note that z^*_u is not the B&C incumbent value managed by the MILP solver, since the latter corresponds to a feasible solution to M_{xy}, which is a relaxation of model (2)–(8), to which Benders cuts are added. In fact, to ensure the convergence of the BBC algorithm, it is necessary, as we see below in Theorem 3, that the value z^*_u is substituted to the incumbent value that would normally be stored by the MILP software tool when solving M_{xy} by B&C.

Furthermore, instead of fathoming the B&C node corresponding to the integer solution $\bar{y}$, the following canonical cut is added to M_{xy}:

$$\sum_{j\in\mathcal{J}} \sum_{k\in\mathcal{K}|\bar{y}_{jk}=0} y_{jk} + \sum_{j\in\mathcal{J}} \sum_{k\in\mathcal{K}|\bar{y}_{jk}=1} (1 - y_{jk}) \geq 1. \tag{17}$$

The B&C algorithm is then resumed at that node, i.e., the model corresponding to that node is now solved with the addition of cut (17), and the search in the B&C tree is continued. The rationale behind cut (17) is that, since the best solution $\bar{x}(\bar{y})$ for the given configuration $\bar{y}$ has been determined, we can cut all solutions of the form $(x, \bar{y})$. Note that cut (17) is not valid for the original formulation, but since it removes only the feasible solutions of the form $(x, \bar{y})$, for which we have already computed the best solution $(\bar{x}(\bar{y}), \bar{y})$, then no optimal solution can be missed. In fact, only the cut corresponding to an optimal solution of the GWLANP is not valid, since the canonical cuts associated with non-optimal solutions of the problem can be added without removing any optimal solution. Since the cut is not valid in general, M_{xy} is no more a relaxation of the original model, but rather a relaxation of the model representing the original set of feasible solutions with the exclusion of the solutions of the form $(x, \bar{y})$. To the best of our knowledge, such a simple cut generation strategy has never been used in a BBC or Benders decomposition approach. We further discuss this issue in Section 3.5, where we compare our BBC algorithm to the Benders decomposition method presented in (Gendron et al., 2013) for the GWLANP without UT assignment costs.

When $S_x(\bar{y})$ is infeasible, we could generate cut (17), which is now obviously valid for the original model, given that it removes only the solution $\bar{y}$, which cannot yield a feasible solution. However, we can strengthen this cut by using simple arguments (introduced in (Gendron et al., 2013)) based on the assumption that the transmission capacity functions $r_{ij}(\pi_j)$ are nondecreasing. We first define the *strengthened feasibility cut* as follows:

$$\sum_{j\in\mathcal{J}} \sum_{k\in\mathcal{K}|p_k>\bar{\pi}_j} y_{jk} \geq 1, \tag{18}$$

where $\bar{\pi}_j = \sum_{k\in\mathcal{K}} p_k \bar{y}_{jk}$.

Lemma 1. *If $S_x(\bar{y})$ is infeasible, then (18) is a valid inequality to (2)–(8).*

Proof. An infeasible $S_x(\bar{y})$ implies that the PLs assigned to the powered-on APs, according to $\bar{y}$, do not provide enough capacity to satisfy the demands of the UTs. Therefore, it is necessary to increase at least one of the values $\bar{r}_{ij}$, i.e., we must raise the PL of

at least one AP (which follows from the fact that the transmission capacity functions $r_{ij}(\pi_j)$ are nondecreasing functions of π_j, $j \in \mathcal{J}$). □

We can further strengthen the Benders cut in case $S_x(\bar{y})$ is infeasible by first solving an auxiliary Benders subproblem $S_x(\tilde{y})$, where $\tilde{y}$ is defined as follows:

$$\tilde{y}_{jk} = \begin{cases} 1, & \text{if} \quad k = k_{max} \quad \text{and} \quad \sum_{l \in \mathcal{K}} \bar{y}_{jl} = 1, \\ 0, & \text{otherwise}, \end{cases} \tag{19}$$

where k_{max} is the index of the PL providing the maximum radiated power.

Indeed, if both $S_x(\bar{y})$ and $S_x(\tilde{y})$ are infeasible, we define the *maximally strengthened feasibility cut* as:

$$\sum_{j \in \mathcal{J} | \bar{\pi}_j = 0} \sum_{k \in \mathcal{K}} y_{jk} \geq 1. \tag{20}$$

Lemma 2. *If $S_x(\bar{y})$ and $S_x(\tilde{y})$ are infeasible, then (20) is a valid inequality to (2)–(8).*

Proof. According to $\tilde{y}$, the maximum possible radiated power is associated with the group of APs that are powered-on in solution $\bar{y}$. An infeasible $S_x(\tilde{y})$ thus implies that no feasible solution exists that uses only such a subset of APs. Therefore, at least one AP that is powered-off in $\bar{y}$ (i.e., an AP j such that $\bar{\pi}_j = 0$) must be powered-on. □

Note that cut (20) coincides with cut (18), if we consider $\tilde{y}$ in place of $\bar{y}$. It is, in general, a stronger cut, since $\sum_{j \in \mathcal{J}} \sum_{k \in \mathcal{K} | p_k > \bar{\pi}_j} y_{jk} = \sum_{j \in \mathcal{J} | \bar{\pi}_j = 0} \sum_{k \in \mathcal{K}} y_{jk} + \sum_{j \in \mathcal{J} | \bar{\pi}_j > 0} \sum_{k \in \mathcal{K} | p_k > \bar{\pi}_j} y_{jk} \geq \sum_{j \in \mathcal{J} | \bar{\pi}_j = 0} \sum_{k \in \mathcal{K}} y_{jk} \geq 1$.

Whether $S_x(\bar{y})$ is feasible or not, the B&C node corresponding to $\bar{y}$ should not be fathomed, as its descendants or itself might contain another feasible solution to the GWLANP with a better objective function value. That is why, in both cases, the cut is added and the B&C algorithm is resumed at the current node. This point is made more precise in the following section.

3.4. Convergence of the algorithm

The BBC algorithm is summarized in Algorithm 1, where z_u^* and z_l^* denote, respectively, the best known upper bound on $z(GWLANP)$, and the best known lower bound on $z(M_{xy})$. The upper bound z_u^* is updated by the BBC algorithm and provided as incumbent value each time the B&C search is invoked. The lower bound z_l^* is updated by the B&C search and provided to the BBC algorithm to test the stopping criterion $z_u^* \leq z_l^*$.

Algorithm 1 Algorithm BBC (Input: M_{xy}, Output: z_u^*, $(\bar{x}(\bar{y}), \bar{y})$).

1: $z_u^* = \infty$
2: Perform B&C for solving M_{xy} until an integer solution $\bar{y}$ is found or the B&C search is completed (Input: z_u^*, incumbent value of B&C, Output: z_l^*, lower bound computed by B&C)
3: **if** B&C search is completed **then**
4: STOP
5: **end if**
6: Solve $S_x(\bar{y})$
7: **if** $S_x(\bar{y})$ is feasible **then**
8: **if** $z(S_x(\bar{y})) < z_u^*$ **then**
9: $z_u^* = z(S_x(\bar{y}))$, and store the optimal solution $(\bar{x}(\bar{y}), \bar{y})$ to $S_x(\bar{y})$
10: **end if**
11: **if** $z_u^* \leq z_l^*$ **then**
12: STOP
13: **end if**
14: Add the canonical cut (17)
15: **else**
16: Solve $S_x(\tilde{y})$
17: **if** $S_x(\tilde{y})$ is infeasible **then**
18: Add the maximally strengthened feasibility cut (20)
19: **else**
20: add the strengthened feasibility cut (18)
21: **end if**
22: **end if**
23: Go to line 2 (resuming B&C at the current node)

Theorem 3. *The BBC algorithm identifies an optimal solution to model (2)–(8), if there is one.*

Proof. Assume that model (2)–(8) is feasible. Note that at least one feasible solution to (2)–(8) is identified by the BBC algorithm. Indeed, the initial master problem M_{xy} is a relaxation of model (2)–(8) and, consequently, its set of feasible solutions includes all feasible solutions to (2)–(8). When the B&C algorithm for solving M_{xy} identifies an integer solution $\bar{y}$, either a feasibility cut is added or a feasible solution to (2)–(8) is identified. This last alternative will necessarily arise after adding a finite number of feasibility cuts, since model (2)–(8) is feasible and the y solutions to M_{xy} coincide with the ones to (2)–(8).

Lemmas 1 and 2 show that the feasibility cuts (18) and (20) are valid for (2)–(8), and therefore their addition to M_{xy} cannot eliminate any feasible solution to model (2)–(8). Concerning the canonical cuts (17), observe that they are added to M_{xy} when the optimal feasible solution $(\bar{x}(\bar{y}), \bar{y})$ corresponding to $\bar{y}$ has been determined. Since the objective function value $z(S_x(\bar{y}))$ of $(\bar{x}(\bar{y}), \bar{y})$ is used to improve the best known upper bound z_u^* on $z(GWLANP)$, no optimal solution to model (2)–(8) can be discarded by the addition of (17).

To conclude, observe that the number of feasible $\bar{y}$ configurations is finite. Therefore, after a finite number of cut additions, the BBC algorithm must end, due to either one of the following reasons:

(1) The B&C search in line 2 is completed. In this case, we have identified a feasible solution to (2)–(8) of objective function value z_u^*. Now, assume that this solution is not optimal. This implies that there is an optimal solution to (2)–(8), say (x^*, y^*) of objective function value $z(x^*, y^*) < z_u^*$, for which the corresponding configuration y^* has not been generated when solving M_{xy} by B&C. This, in turn, implies that there exists some node p that has been fathomed, but would have yielded configuration y^* after a finite number of branchings. Node p has been fathomed by the lower bound test, i.e., $z^l(p) \geq z_u^*$, where $z^l(p)$ is the lower bound associated with node p (recall that z_u^* is the incumbent value used in B&C). Furthermore, the fact that node p would have yielded configuration y^* after a finite number of branchings implies that $z(x^*, y^*) \geq z^l(p)$, node p being a relaxation of subproblem $S_x(y^*)$ for which an optimal solution is x^*. Collecting together these facts, we obtain: $z_u^* > z(x^*, y^*) \geq z^l(p) \geq z_u^*$, a contradiction. Hence, the best feasible solution identified at the end of the BBC algorithm is necessarily optimal. Note that this part of the proof relies on the fact that z_u^* is substituted to the incumbent value that would normally be used when performing the B&C method in line 2. Failure to perform this substitution would result into an algorithm that is not necessarily exact.

(2) The condition $z_u^* \leq z_l^*$ in line 11 is verified. This case implies that any feasible solution (x, y) to the GWLANP that could still be generated by performing the B&C method for solving M_{xy} has an objective function value $z(x, y) \geq z_l^* \geq z_u^*$, and therefore cannot improve upon z_u^*.

Hence, the BBC algorithm ends with an optimal solution to model (2)–(8). □

3.5. Comparison with Benders decomposition for a special case

In this section, we describe the Benders decomposition (BD) algorithm (Gendron et al., 2013) developed for the GWLANP without UT assignment costs. Our objective is to state the similarities and the differences between this BD algorithm and the BBC method described above.

One of the main differences between the two methods lies in the way the two algorithms perform B&C on the master problem: while the BBC method explores a single B&C tree, adding the Benders cuts during the exploration of that tree, the BD approach performs B&C at every iteration, adding the Benders cuts only after the B&C has completed its exploration. Hence, the BD algorithm explores several B&C trees, with the master problem being gradually augmented with Benders cuts.

Another main difference between the two approaches is the way they define master problems. In the BD algorithm, the master problem includes only the AP assignment variables y_{jk}. Since the UT assignment constraints (3) are then relaxed, the following valid inequalities are introduced in the master problem:

$$\sum_{j\in\mathcal{J}} r^u_{ij}(\pi_j) \geq w_i, \quad i \in \mathcal{I}. \tag{21}$$

The master problem in the BD method, denoted M_y^{BD}, can therefore be formulated as follows:

$$z(M_y^{BD}) = \min \sum_{j\in\mathcal{J}} \sum_{k\in\mathcal{K}} (p_0 + p_k) y_{jk} \tag{22}$$

subject to (4), (8), (21) and the Benders cuts added so far during the course of the algorithm. In order to generate a good set of initial Benders cuts, the BD method first solves the relaxation corresponding to M_{xy}, the BBC master problem, which is further strengthened by imposing the integrality of the UT assignment variables x_{ij}. This type of initialization strategy, involving the solution of a relaxation of the problem to generate a good set of initial Benders cuts, is well-known in the Benders decomposition literature (see, for instance, McDaniel and Devine, 1977 for an early contribution on this topic). All subsequent iterations solve the classical Benders master problem M_y^{BD}.

At every iteration of the BD algorithm, the master problem is solved until an optimal solution $\bar{y}^0$ is obtained. All other integer solutions, say $\bar{y}^1, \bar{y}^2, \ldots, \bar{y}^n$, found during the exploration of the B&C tree, are also collected. For each solution $\bar{y} = \bar{y}^q, q = 0, 1, \ldots, n$, the Benders subproblem $S_x(\bar{y})$ is solved, as in the BBC algorithm. Note, however, that $S_x(\bar{y})$ is no more an optimization problem, but is rather a feasibility problem, since there are no UT assignment costs.

If $S_x(\bar{y})$ is infeasible, a Benders feasibility cut (18) or (20) is generated (the BD algorithm also solves subproblem $S_x(\tilde{y})$, with $\tilde{y}$ defined as in (19)). If $S_x(\bar{y})$ is feasible, and $\bar{x}(\bar{y})$ is the computed feasible solution, then a feasible solution $(\bar{x}(\bar{y}), \bar{y})$ to the GWLANP is obtained. If the corresponding objective function value $z(S_x(\bar{y}))$ is better than the value of the current best feasible solution, denoted z_u^*, then both z_u^* and the best feasible solution are updated. Whenever $z_u^* \leq z_l^*$, where z_l^* is the optimal value of the master problem, we can conclude that an optimal solution to the GWLANP has been identified. This is the case when $S_x(\bar{y})$ is feasible and $\bar{y} = \bar{y}^0$, the optimal solution to the master problem: the optimality of $\bar{y}$ for the master problem and the feasibility of the Benders subproblem suffice to conclude to the optimality of any feasible solution to $S_x(\bar{y})$, because of the absence of UT assignment costs.

Such a conclusion cannot be derived for the general case of the GWLANP with UT assignment costs. This is why we rely on the addition of the canonical cuts (17) in the BBC algorithm, which then take the place of the usual Benders optimality cuts, i.e., they cut the solutions of the form $(x, \bar{y})$ when the Benders subproblem $S_x(\bar{y})$ is feasible. For the GWLANP without UT assignment costs, these cuts are not needed. In fact, they are simply replaced by updating the B&C incumbent value with z_u^*. This, in effect, cuts all the feasible solutions $(\bar{x}(\bar{y}), \bar{y})$ such that $z(S_x(\bar{y})) \geq z_u^*$.

It is worth noting that, at any iteration of the BD algorithm, the master problem always defines a relaxation of the GWLANP. Hence, the stopping condition $z_u^* \leq z_l^*$ can equivalently be replaced by $z_u^* = z_l^*$, since z_l^* is then necessarily a lower bound on $z(GWLANP)$. In contrast, the master problem in the BBC algorithm is also a relaxation, but not of the GWLANP, rather of a restriction of the problem obtained by adding the canonical cuts corresponding to the feasible Benders subproblems. Hence, for the BBC method, it might happen that $z_u^* < z_l^*$ at the conclusion of the algorithm.

To further highlight the similarities and the differences between the two algorithms, an outline of the BD algorithm is provided in Algorithm 2.

Algorithm 2 Algorithm BD (Input: M_y^{BD}, Output: z_u^*, $(\bar{x}(\bar{y}), \bar{y})$).

1: $z_u^* = \infty$
2: Perform B&C for solving M_y^{BD} (or, initially, M_{xy} with the integralityimposed on the x_{ij} variables) until an optimal solution $\bar{y}^0$ of value z_l^* is found(Input: z_u^*, incumbent value of B&C, Output: z_l^*, lower bound computed by B&C)
3: Let $\bar{y}^1, \bar{y}^2, \ldots, \bar{y}^n$ be the other integer solutions obtained during B&C
4: **for all** $\bar{y} = \bar{y}^q, q = 0, \ldots, n$ **do**
5: Solve $S_x(\bar{y})$
6: **if** $S_x(\bar{y})$ is feasible **then**
7: **if** $z(S_x(\bar{y})) < z_u^*$ **then**
8: $z_u^* = z(S_x(\bar{y}))$, and store the feasible solution $(\bar{x}(\bar{y}), \bar{y})$ to $S_x(\bar{y})$
9: **end if**
10: **if** $z_u^* \leq z_l^*$ **then**
11: STOP
12: **end if**
13: **else**
14: Solve $S_x(\bar{y})$
15: **if** $S_x(\bar{y})$ is infeasible **then**
16: Add the maximally strengthened feasibility cut (20)
17: **else**
18: add the strengthened feasibility cut (18)
19: **end if**
20: **end if**
21: **end for**
22: Go to line 2 (restarting B&C from scratch)

4. Computational results

Our computational experiments aim to assess the effectiveness and the efficiency of the BBC method, and to stress its robustness and scalability issues. Note that the GWLANP has never been addressed in the general form studied in this paper. Therefore, no comparison with approaches from the literature can be performed. However, since the BD algorithm proposed in (Gendron et al., 2013) addresses the special case without UT assignment costs, we compare the two approaches on instances of this type. The main objective of this computational comparison is to assess the competitiveness of the BBC method, especially for large-scale instances. The BBC and BD algorithms have been implemented in C++ using IBM ILOG CPLEX 12.6.1. The experiments have been performed on a PC with 2 CPUs Intel Xeon E5-2609 v2 (quad Core) @ 2.50 gigahertz (no hyperthreading), 128 gigabyte RAM.

In Section 4.1, we describe the procedure used to generate a set of 340 realistic GWLANP instances. Section 4.2 presents the

measures used to assess the performance of the BBC algorithm and to compare it with the BD method. Section 4.3 focuses mostly on a computational comparison of the BBC and BD algorithms. We also briefly investigate the impact of UT assignment costs and strengthened cuts on the performance of the BBC algorithm.

4.1. Generation of instances

To generate realistic GWLANP instances, we use related features extracted from real-life measurement campaigns in corporate environments (Balazinska & Castro, 2003; Jardosh et al., 2009). We first specify the values of $|\mathcal{I}|$, $|\mathcal{J}|$ and $|\mathcal{K}|$. Then, the positions of the APs and of the UTs in each instance are randomly determined as follows. First, we divide the test field into a regular grid of $|\mathcal{J}|$ squares. Then, the APs are placed one per square, with their coordinates chosen randomly within the square. The set of UTs is also split into $|\mathcal{J}|$ subsets, and the elements of each subset are randomly spread over each square. This strategy ensures enough uniformity in the placement of the UTs and the APs, so as to mimic a corporate scenario and to avoid heavily unbalanced instances.

Other relevant instance characteristics are the transmission loss factors α_{ij}, which have been computed by using a simplified version of the COST-231 multi-wall path loss model for indoor, non-LOS environments (European Commission, 1999), and the maximum achievable rate r_{max}, set to 54 megabits per second according to the 802.11g standard. In addition, the addressed traffic demands w_i have an average value of 300 kilobits per second, and they have been randomly generated within a variation of ± 10 percent. To complete the parameter list, we set the sensitivity thresholds γ_{ij} and the power component figures p_0 and p_k according to (Cisco, 2014). Finally, the proportionality coefficients μ_j are selected based on the indications in (Garcia-Saavedra, Serrano, Banchs, & Bianchi, 2012).

By setting $\mu_j = 0$, $j \in \mathcal{J}$, we have generated 340 instances that are used to compare the two algorithms, BBC and BD. We divide these instances into three classes, according to the number of APs, which helps us differentiate the behavior of the algorithms, as explained below. The classes of *small* and *medium* instances have 10 and 15 APs, respectively. Each includes 6 sets of 20 instances each, for a total of 120 instances, obtained by combining $|\mathcal{I}| = \{100, 150, 200\}$ with $|\mathcal{K}| = \{3, 4\}$. The class of *large* instances is characterized by a number of APs equal to 20, 30 or 50. For each such value of $|\mathcal{J}|$, 3 sets of 10 instances each (for a total of 90 instances) are obtained by combining $|\mathcal{I}| = \{200, 300, 500\}$ with $|\mathcal{K}| = 4$.

4.2. Performance measures

To assess the performance of the algorithms, we use the following measures:

- The CPU time, in seconds, denoted *Time*. Note that the algorithms are given a time limit before they are stopped (3600 seconds on small/medium instances and 7200 seconds on large instances). In addition to *Time*, we also report the *time ratio* of algorithm A defined as

  $$\tau_A = Time(A)/\min\{Time(BBC), Time(BD)\},$$

  where *Time(A)* is the CPU time taken by algorithm A, which can be either BBC or BD.
- The final gap, in percentage, between the bounds, measured as

  $$Gap = \max\{0, 100 \times (z_u^* - z_l^*)/z_l^*\}.$$

  Note that, because it might happen that $z_u^* < z_l^*$ at the end of the BBC algorithm, it is necessary to modify the usual formula for computing the gap. Even if z_l^* is not necessarily a lower bound on $z(GWLANP)$, this modified gap measure is a fair approximation of the distance between the current best known upper bound z_u^* and the optimal value $z(GWLANP)$. Note that z_l^* is a lower bound on $z(GWLANP)$ for the BD method. Hence, the reported gaps are exact and not approximations, as it is the case for the BBC algorithm.
- To better compare the capacity of the algorithms to identify high-quality solutions when they reach the time limit, we report the *upper bound ratio* of algorithm A defined as

  $$\mu_A = z_u^*(A)/\min\{z_u^*(BD), z_u^*(BBC)\},$$

  where $z_u^*(A)$ is the best upper bound on $z(GWLANP)$ found by algorithm A, either BBC or BD.
- The total number of cuts (denoted *Cuts*), including strengthened and maximally strengthened feasibility cuts for both algorithms, as well as canonical cuts for the BBC algorithm.
- The total number of nodes generated by the BBC algorithm, denoted *Nodes*. For the BD algorithm, we could have reported the same measure, but we found the number of iterations, *Iter*, to be a more useful measure of the computational effort.

These different performance measures are computed for each instance. Then, we chose to summarize these detailed results in two ways. First, we compute the arithmetic average of each performance measure (except the time ratio and the upper bound ratio) over each set of instances, i.e., all the instances with the same size $|\mathcal{J}|$, $|\mathcal{I}|$, $|\mathcal{K}|$. Using this simple approach, we are able to compare the behavior of the algorithms with respect to problem size. In particular, we highlight the importance of the number of APs in explaining the relative performance of the algorithms. To perform a more detailed comparative analysis on each class of instances, small, medium and large, we use performance profiles with respect to either the time ratio (on small and medium instances, where most instances are solved to optimality) or the upper bound ratio (on large instances, where most instances cannot be solved to optimality within the time limit), as suggested in (Dolan & Moré, 2002). The performance profile of algorithm A with respect to metric $\tau_A(s)$ measured over each instance s in a set S is simply the graph of the cumulative distribution function, defined as

$$F_A(t) = |\{s \in S \mid \tau_A(s) \leq t\}|/|S|.$$

Such performance profiles provide useful information about the relative performance of algorithms, often hidden when we look only at average results. For example, the performance profiles of the algorithms with respect to the time ratio on a set of instances will tell us which algorithm is more often the fastest (by looking at the largest value between $F_{BBC}(1)$ and $F_{BD}(1)$) or the proportion of the instances in the set for which algorithm BD is more than two times slower than algorithm BBC (by computing $1 - F_{BD}(2)$). As we want to focus on the instances for which the performance of the two algorithms differ, we remove instances in the set for which the measure gives the same value. This way, an algorithm that achieves a value 1 on a particular ratio (time or upper bound) is the "winner" (in other words, there is no tie). In particular, when the two algorithms reach the time limit on a given instance, that instance is removed from the set of instances on which we compute the performance profiles with respect to the time ratio, i.e., there is no "winner" in that case. However, if only one of the two algorithms reaches the time limit, the other algorithm is the "winner" and the instance is kept when computing the performance profiles. Note that the time ratio of the "loser" for that instance underestimates the "true" time ratio that would have been obtained if the "loser" was allowed to run long enough to stop with a proof of optimality.

Table 1
Average results of BBC and BD; 240 small/medium instances (max. 3600 seconds) and 90 large instances (max. 7200 seconds); Gap (percent); Time (second); number of cuts; number of nodes (BBC); number of iterations (BD); solved/total number of instances.

	BBC					BD				
$\|\mathcal{J}\|$, $\|\mathcal{I}\|$, $\|\mathcal{K}\|$	*Gap*	*Time*	*Cuts*	*Nodes*	*Solved*	*Gap*	*Time*	*Cuts*	*Iter*	*Solved*
10, 100, 3	0	9	524	702	20/20	0	2	28	15	20/20
10, 100, 4	0	17	939	1299	20/20	0	4	47	23	20/20
10, 150, 3	0	49	1307	1793	20/20	0	18	117	70	20/20
10, 150, 4	0	158	4159	5611	20/20	0	117	441	246	20/20
10, 200, 3	0	706	4586	5971	20/20	0	190	798	381	20/20
10, 200, 4	0	631	6985	9556	20/20	0	530	1182	505	20/20
15, 100, 3	0	155	5289	7630	20/20	0	490	554	359	20/20
15, 100, 4	0	171	6147	9662	20/20	0.40	946	668	277	19/20
15, 150, 3	0	237	4293	6779	20/20	0	184	197	91	20/20
15, 150, 4	0	708	11343	17593	20/20	4.37	1627	857	313	11/20
15, 200, 3	1.49	1684	6704	10384	15/20	11.57	2531	1018	460	6/20
15, 200, 4	2.30	2029	10632	16618	15/20	20.71	2885	1212	444	4/20
20, 200, 4	1.00	5213	27466	46102	4/10	27.78	7200	1326	527	0/8[a]
20, 300, 4	5.37	7200	13467	23408	0/10	58.10	7200	1040	428	0/10
20, 500, 4	30.40	7200	1088	2454	0/10	157.54	7200	1180	419	0/7[a]
30, 200, 4	2.29	6594	37098	67120	1/10	42.19	7200	1337	372	0/8[a]
30, 300, 4	7.42	7200	15156	30780	0/10	81.17	7200	1195	309	0/10
30, 500, 4	16.94	7200	1318	3384	0/10	148.10	7200	867	209	0/8[a]
50, 200, 4	3.70	7200	24503	62913	0/10	74.62	7200	1567	337	0/9[a]
50, 300, 4	5.52	7200	9901	27949	0/10	98.59	7200	1317	260	0/7[a]
50, 500, 4	11.00	7200	2006	6080	0/10	145.38	7200	955	194	0/8[a]

[a] Some instances removed: the first iteration could not be completed within 7200 seconds

4.3. Analysis of the performance of the algorithms

Table 1 summarizes the results of the experiments on the two algorithms, BBC and BD, with time limits of 3600 seconds and 7200 seconds, respectively, for small/medium and large instances. We recall that: small instances are characterized by $|\mathcal{J}| = 10$ and thus represent the 6 sets of instances in the upper part of the table; medium instances have 15 APs and correspond to the 6 sets of instances in the middle part of the table; large instances, with more than 15 APs, are the 9 sets of instances shown in the lower part of the table. For the two algorithms, *Gap*, *Time* and *Cuts* are shown, while the number of nodes and the number of iterations are given, respectively, for BBC and BD. These performance measures are averaged over all instances in a given set. In addition, the table also shows, in column *Solved*, a fraction where the numerator is the number of instances solved to optimality within the time limit and the denominator is the number of instances in the corresponding set (20 for each small/medium set and 10 for each large set). Note that, for 15 of the 90 large instances, the BD algorithm could not complete the first iteration (i.e., solving by B&B the first master problem) within the limit of 7200 seconds. Since no gaps are obtained for these instances, we removed them when computing the average results for the BD algorithm. The denominator in column *Solved* then shows how many instances in each set were used to compute the average performance measures.

If we analyze first the performance of each algorithm independently, we observe that the BBC algorithm solves to optimality most instances in the small and medium classes. Indeed, only 10 of the 240 instances could not be solved within the time limit of 3600 seconds. Moreover, all instances with $|\mathcal{I}| = 100$ and 150 are solved to optimality, while high-quality results are obtained for instances with $|\mathcal{I}| = 200$. This last observation is also true for the large instances, where 5 instances out of 30 with $|\mathcal{I}| = 200$ are solved to optimality, with average gaps below 4 percent. The algorithm shows relatively good performance on large instances with $|\mathcal{I}| = 300$, with average gaps below 8 percent, but it is struggling on the largest instances with $|\mathcal{I}| = 500$, with average gaps between 11 percent and 30 percent.

We also looked at the results obtained by solving the same 340 instances, but by keeping the UT assignment costs. The results are similar, except that the average CPU times are about twice larger for the instances with UT assignment costs. This is not surprising, given that the Benders subproblems for these instances are optimization problems that are more difficult to solve than the feasibility problems for the case without UT assignment costs. Another issue we considered when evaluating the performance of the BBC algorithm is the impact of strengthened and maximally strengthened cuts. Specifically, we disabled the option of generating these cuts and examined the results obtained by comparing them with the BBC method using these cuts. We show the results on the class of small instances, since it is the one with the largest number of instances solved to optimality by the two approaches (117 out of 120). The performance profiles of BBC and *BBC with no strengthened cuts (BBC-NS)* with respect to the time ratio are shown in Fig. 1. We see that, with strengthened cuts, only 36 percent of the instances are solved faster, but less than 2 percent of the instances are solved more than two times slower, with a maximum time ratio around 4. Without strengthened cuts, 10 percent of the instances are solved more than two times slower and the maximum time ratio exceeds 5 (in fact, it is 27). By looking at the detailed results, we observe that BBC-NS is faster on “easy” instances (solved within 60 seconds), but in general slower, sometimes considerably so, for more difficult instances. Note that 3 instances out of the 120 small instances could not be solved to optimality by BBC-NS, with one of these instances having a time ratio of 16.

Turning to the performance of the BD algorithm, Table 1 shows that it solves to optimality all small instances and a majority of the medium instances (80 out 120). However, on medium instances, its performance deteriorates sharply as the size increases: only 10 of the 40 instances with $|\mathcal{I}| = 200$ are solved to optimality, displaying average gaps above 10 percent. On large instances, the situation gets worse. As mentioned above, 15 of the 90 large instances could not go beyond the first iteration. None of the remaining 75 instances could be solved to optimality, while the average gaps are always above 25 percent and could go higher than 150 percent. We also note that, irrespective of the class of instances, the performance deteriorates significantly with an increase in the number of PL assignment variables (i.e., the y_{jk} variables), which can be seen, in particular, by examining the values of *Time* and *Iter*. Indeed, on small instances with a fixed value of $|\mathcal{I}|$, both the time

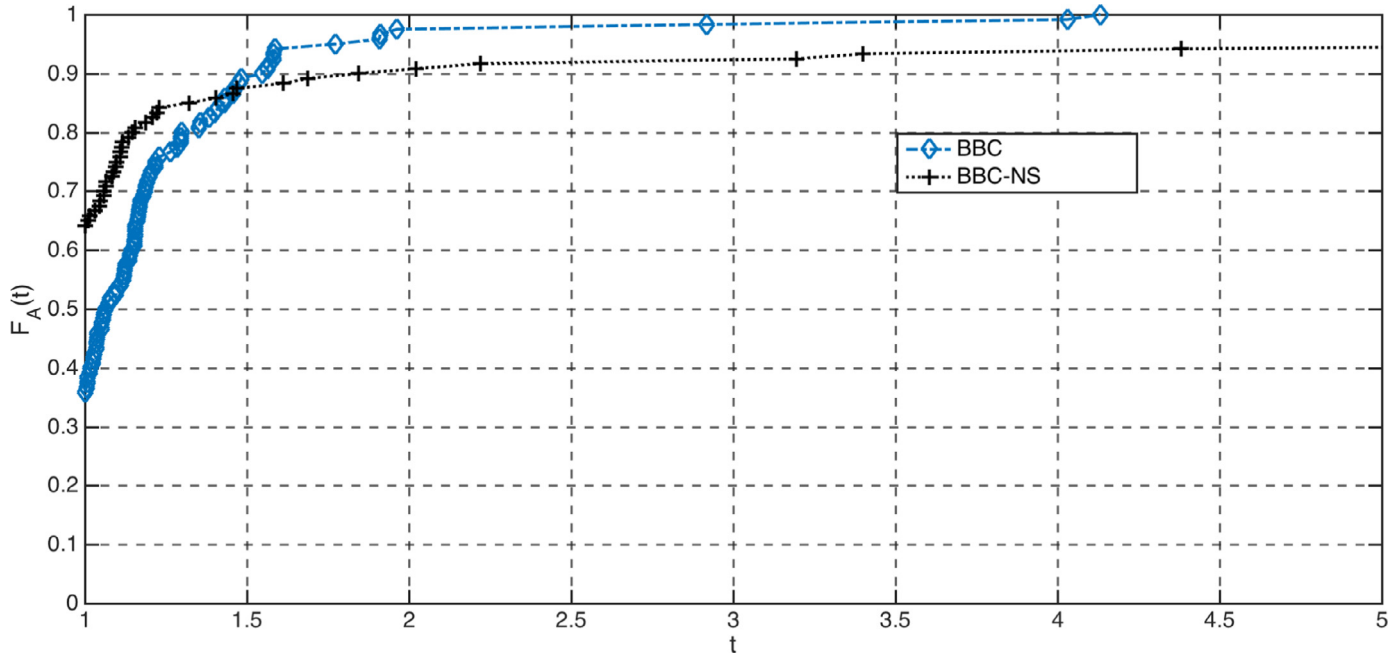

Fig. 1. Small instances: performance profiles of BBC and BBC-NS w.r.t. time ratio in [1,5].

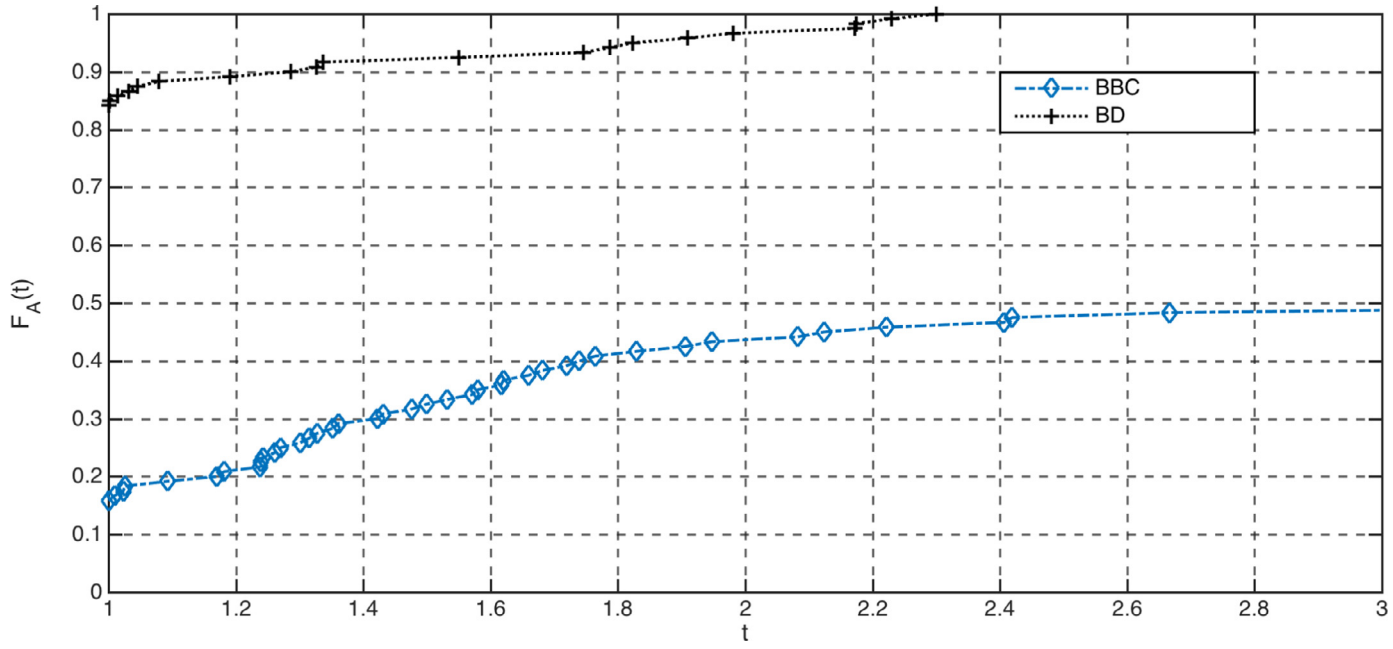

Fig. 2. Small instances: performance profiles of BBC and BD w.r.t. time ratio in [1,3].

and the number of iterations increase when $|\mathcal{K}|$ increases from 3 to 4, but more importantly, the ratio *Time*/*Iter* also increases, which indicates that the time per BD iteration (that depends mostly on the time to solve the master problem by B&B) increases. The same observation applies to medium instances with a fixed value of $|\mathcal{I}|$, where the ratio *Time*/*Iter* increases with an increase of $|\mathcal{K}|$. On large instances with a fixed value of $|\mathcal{I}|$, we make a similar observation: *Time* being constant, we see a significant decrease in the number of iterations when $|\mathcal{J}|$ increases, showing again that the time per BD iteration increases significantly with an increase in the number of PL assignment variables.

When we compare the two algorithms, we first note that the BBC method is also sensitive to an increase in the number of PL assignment variables, as can be observed from the increase in the average number of generated nodes (column *Nodes* in Table 1) on small/medium instances when $|\mathcal{K}|$ increases for fixed values of $|\mathcal{I}|$ and $|\mathcal{J}|$. BBC is, however, much less sensitive than BD, as can be easily observed by looking at the relative increase in *Time* when $|\mathcal{K}|$ increases from 3 to 4 for fixed $|\mathcal{I}|$, $|\mathcal{J}|$: the increase is always sharper for BD (with the exception of the instances with $|\mathcal{I}| = 200$ and $|\mathcal{J}| = 15$, which can be discarded because very few instances are solved to optimality by BD). We even observe a decrease with BBC in one case: when $|\mathcal{I}| = 200$ and $|\mathcal{J}| = 10$, *Time* decreases from 706 to 631 with the increase of $|\mathcal{K}|$.

The different ways in which the algorithms perform with an increase in the number of AP assignment variables also explain

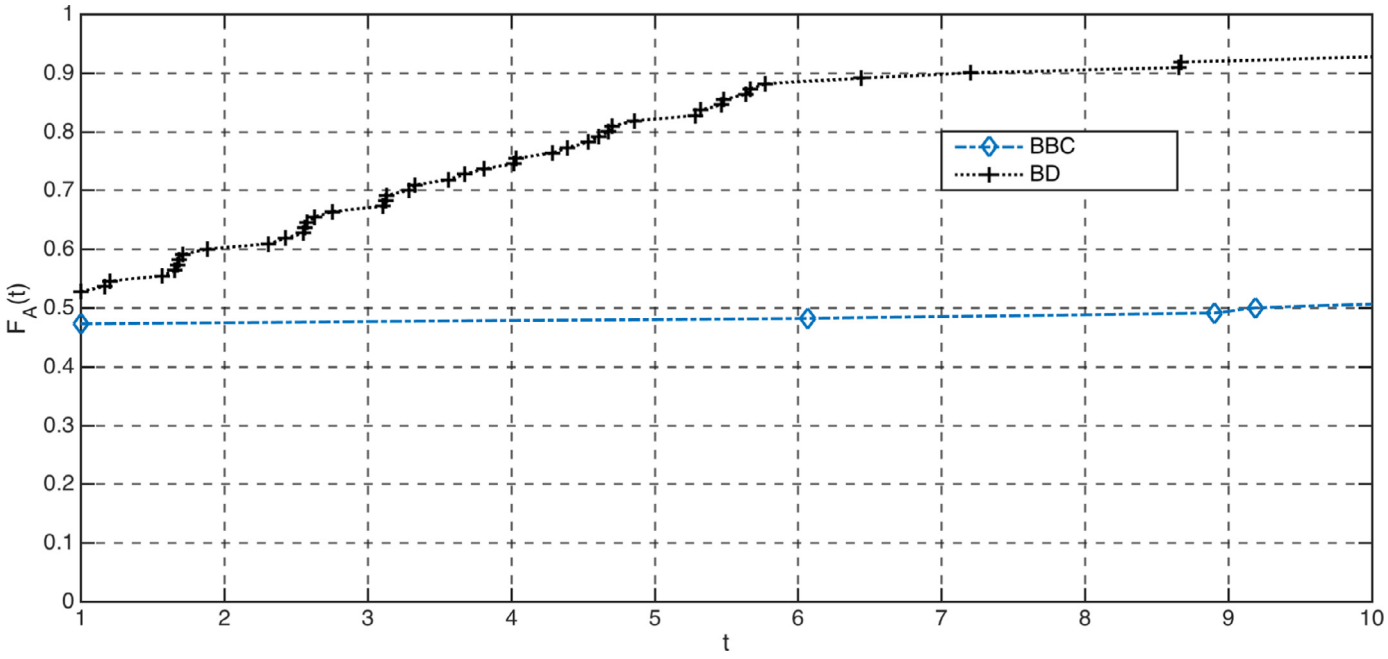

Fig. 3. Medium instances: performance profiles of BBC and BD w.r.t. time ratio in [1,10].

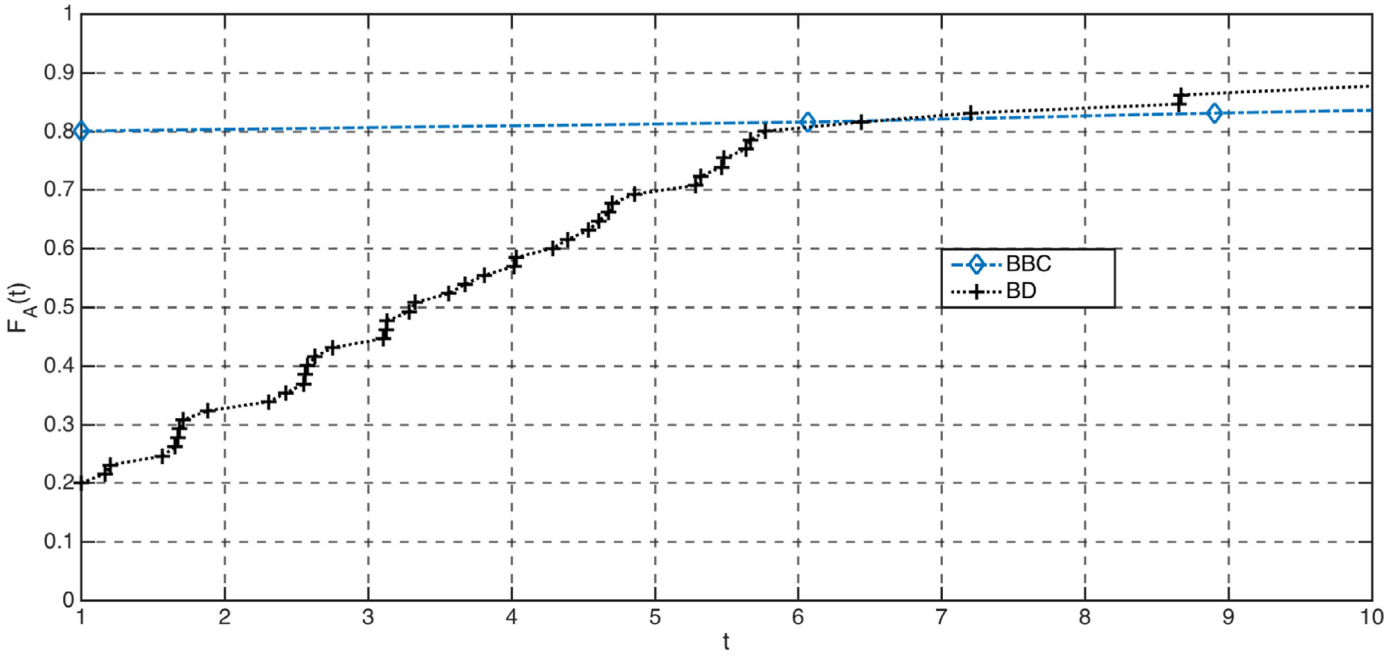

Fig. 4. "Difficult" medium instances: performance profiles of BBC and BD w.r.t. time ratio in [1,10].

the major differences observed when we look at the three classes of instances. On small instances, the BD algorithm is significantly faster than the BBC algorithm, as can be seen by the average values of *Time* in Table 1. This observation is confirmed by the performance profiles of the two algorithms with respect to the time ratio (Fig. 2). This figure shows that 84 percent of the small instances are solved faster by BD and that the BBC algorithm is more than 3 times slower than BD on about 50 percent of the small instances. A plausible explanation for the relatively poor performance of the BBC algorithm on small instances might be the large number of cuts it generates compared to the BD method.

On medium instances, Table 1 shows a completely different situation: on 5 of the 6 sets of instances, the average performance of BBC is better than that of BD and significantly so on the 60 largest instances. Particularly on these instances, the BD algorithm spends a lot of time solving hundreds of master problems, each of which become significantly heavier to solve as the size increases, especially the number of PL assignment variables. The performance profiles shown in Fig. 3 seem to tell a different story: we see that 52 percent of the medium instances are solved faster by BD and that the BBC algorithm is more than 10 times slower than BD on about 50 percent of the medium instances. This apparent contradiction is easily explained by two facts: first, a significant number of medium instances are solved very quickly by the BD algorithm, while the BBC algorithm is typically more than 10 times slower on these instances; second, the BD algorithm is stopped prematurely

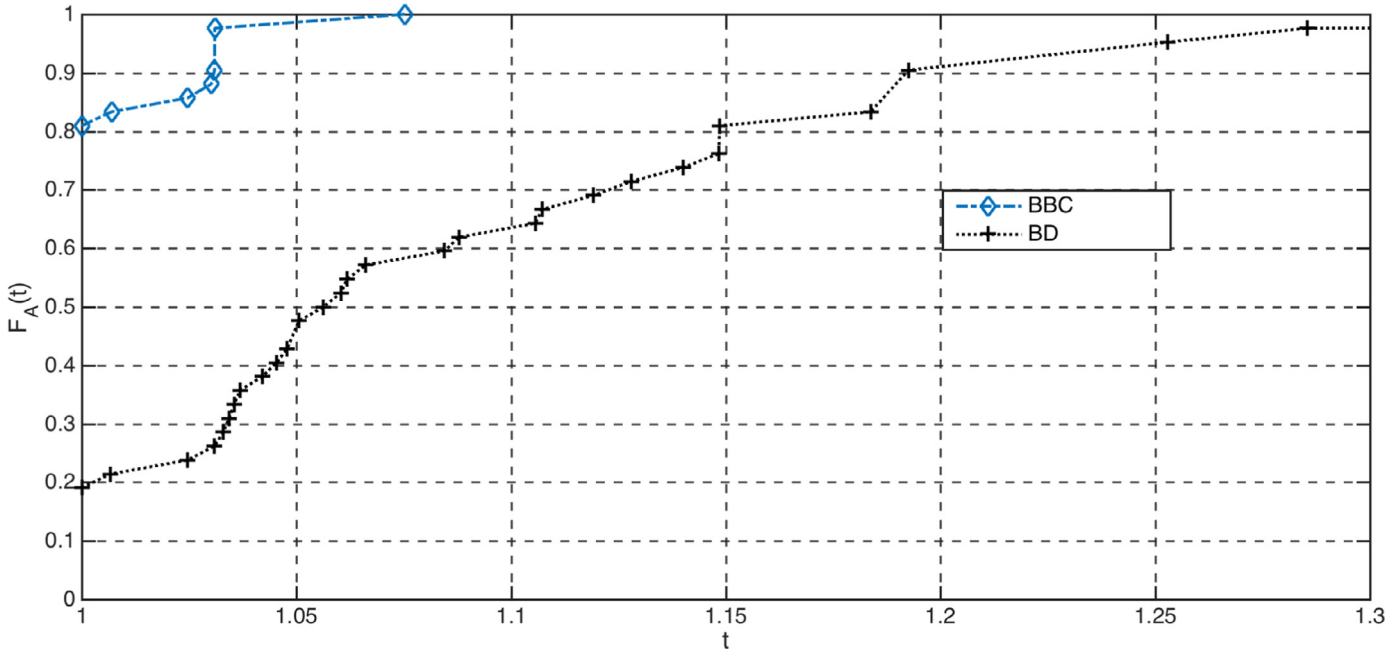

Fig. 5. Large instances: performance profiles of BBC and BD w.r.t. upper bound ratio in [1,1.3].

on a majority of the largest instances and, as a result, its time ratio is significantly underestimated. These two facts explain why the performance profiles make the BD algorithm "look so good" on the medium instances. To show a different picture, we removed from the comparison the instances solved in less than 10 seconds (an arbitrary threshold) by BD. There are 45 such instances; the time ratio of the BBC algorithm on these instances varies between 9 and 183. The performance profiles on the remaining "difficult" instances are shown in Fig. 4. We see that 80 percent of these "difficult" medium instances are solved faster by the BBC algorithm and that the BD algorithm is at least 10 times slower on 12 percent of these instances. Note that the situation would have been worse for BD if the time limit was increased, since the computing ratios for 40 instances are underestimated when the time limit is 3600 seconds.

On large instances, Table 1 shows the superiority of the BBC algorithm as the size increases. Overall, the BBC algorithm is able to construct master problems with significant information, both by including the UT assignment variables and by generating a large number of Benders cuts. Fig. 5 shows the performance profiles of the two algorithms with respect to the upper bound ratio. We recall that the instances where the two algorithms obtained the same upper bounds are removed from the comparison, leaving us with 42 instances. The performance profiles show that the BBC algorithm found a better upper bound for 81 percent of these instances and that the worst upper bound shows a ratio smaller than 1.08. By comparison, the BD algorithm shows an upper bound ratio smaller than 1.08 for less than 60 percent of these instances, while its worst upper bound ratio exceeds 1.3.

5. Conclusion

In this paper, we considered a location-design problem that arises from the development of network reconfiguration algorithms for reducing the power consumption of wireless local area networks (WLANs). The resulting optimization problem, called the green WLAN problem, or GWLANP, was formally described and modeled. While the GWLANP was introduced in (Gendron et al., 2013), we studied a non-trivial extension of the problem where the power consumed by each access point depends on the demands assigned to the access points. An exact solution method, based on the branch-and-Benders-cut framework, was developed. The results on a large set of realistic instances showed that the approach is effective and efficient, as it delivers high-quality solutions in limited computational effort. Furthermore, when comparing its performance on the special case solved by the algorithm proposed in (Gendron et al., 2013), we showed that the proposed algorithm is preferable in terms of solution quality, scalability and robustness.

This work opens up interesting research perspectives. In particular, it would be interesting to generalize the proposed branch-and-Benders-cut approach to other optimization problems. Several features of the algorithm seem to be generalizable, in particular, the inclusion of the Benders subproblem variables in the formulation of the master problem and the addition of cuts that exclude feasible solutions, but that are not based on the objective function value, as in classical Benders decomposition methods.

Acknowledgments

We thank three anonymous referees whose comments have helped us improve our paper. The work of Bernard Gendron was funded by the Natural Sciences and Engineering Council of Canada (NSERC) under grant 184122–2010. The work of Maria Grazia Scutellà was funded by project PRIN 2012, "Mixed-Integer Nonlinear Optimization: Approaches and Applications" (2012JXB3YF). The work of Rosario Garroppo, Gianfranco Nencioni and Luca Tavanti was partially supported by the Italian Ministry of University and Research under the "GreenNet (Greening the Networks)" FIRB project. This support is gratefully acknowledged.

References

Adulyasak, Y., Cordeau, J.-F., & Jans, R. (2013). Benders decomposition for production routing under demand uncertainty. *Technical Report G-2012-57*. GERAD, Montreal, Canada.

Balas, E., & Jeroslow, R. (1972). Canonical cuts on the unit hypercube. *SIAM Journal of Applied Mathematics, 23*(1), 61–69.

Balazinska, M., & Castro, P. (2003). Characterizing mobility and network usage in a corporate wireless local-area network. In *Proceedings of the 1st international conference on mobile systems, applications and services (MOBISYS)* (pp. 303–316). New York, USA: ACM Publisher.

Beck, J. C. (2010). Checking-up on branch-and-check. In D. Cohen (Ed.), *CP 2010-lecture notes on computer science 6308* (pp. 84–98). Springer.

Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik, 4*(1), 238–252.

Botton, Q., Fortz, B., Gouveia, L., & Poss, M. (2013). Benders decomposition for the hop-constrained survivable network design problem. *INFORMS Journal on Computing, 25*(1), 13–26.

de Camargo, R., de Miranda, G., Jr., & Ferreira, R. (2011). A hybrid outer-approximation/Benders decomposition algorithm for the single allocation hub location problem under congestion. *Operations Research Letters, 39*(5), 329–337.

Cisco (2016). Cisco Aironet 3600 Series, http://www.cisco.com/c/en/us/products/wireless/aironet-3600-series/index.html (accessed 16.05.16).

Codato, G., & Fischetti, M. (2006). Combinatorial Benders' cuts for mixed-integer linear programming. *Operations Research, 54*(4), 756–766.

Crainic, T. G., Hewitt, M., & Rei, W. (2014). Partial decomposition strategies for two-stage stochastic integer programs. *Technical Report*. CIRRELT-2014-13, CIRRELT, Montreal, Canada.

D'Andreagiovanni, F. (2012). Pure 0–1 programming approaches to wireless network design. *4OR, 10*(2), 211–212.

D'Andreagiovanni, F., Mannino, C., & Sassano, A. (2013). Gub covers and power-indexed formulations for wireless network design. *Management Science, 59*(1), 142–156.

Dolan, E. D., & Moré, J. J. (2002). Benchmarking optimization software with performance profiles. *Mathematical Programming A, 91*(2), 201–213.

European Commission (1999). COST 231 – Digital mobile radio towards future generations systems. Final Report. European Commission

Fortz, B., & Poss, M. (2009). An improved Benders decomposition applied to a multi-layer network design problem. *Operations Research Letters, 37*(5), 359–364.

Garcia-Saavedra, A., Serrano, P., Banchs, A., & Bianchi, G. (2012). Energy consumption anatomy of 802.11 devices and its implication on modeling and design. In *Proceedings of the 8th international conference on emerging networking experiments and technologies, conext '12* (pp. 169–180). New York, USA: ACM Publisher.

Garroppo, R. G., Gendron, B., Nencioni, G., & Tavanti, L. (2014). Energy efficiency and traffic offloading in wireless mesh networks with delay bounds. *International Journal of Communication Systems*, n/a–n/a. doi:10.1002/dac.2902.

Gendron, B., Garroppo, R., Nencioni, G., Scutellà, M., & Tavanti, L. (2013). Benders decomposition for a location-design problem in green wireless local area networks. *Electronic Notes in Discrete Mathematics, 41*, 367–374.

Hooker, J., & Ottosson, G. (2003). Logic-based Benders decomposition. *Mathematical Programming, 96*(1), 33–60.

Huehn, T., & Sengul, C. (2012). Practical power and rate control for wifi. In *Proceedings of the 21st international conference on computer communications and networks, ICCCN* (pp. 1–7).

Jardosh, A., Papagiannaki, K., Belding, E., Almeroth, K., Iannaccone, G., & Vinnakota, B. (2009). Green WLANs: on-demand WLAN infrastructures. *Mobile Networks and Applications, 14*(6), 798–814.

Kennington, J., Olinick, E., & Rajan, D. (2010). *Wireless network design: Optimization models and solution procedures*. Germany: Springer, Heidelberg.

Li, L., Fan, Z., & Kaleshi, D. (2012). Using multiple metrics for rate adaptation algorithms in ieee 802.11 wlans. In *Proceedings of the IEEE wireless communications and networking conference, WCNC* (pp. 2807–2812).

McDaniel, D., & Devine, M. (1977). A modified Benders partitioning algorithm for mixed integer programming. *Management Science, 24*(3), 312–319.

Naoum-Sawaya, J., & Elhedhli, S. (2010). A nested Benders decomposition approach for telecommunication network planning. *Naval Research Logistics, 57*(6), 519–539.

Naoum-Sawaya, J., & Elhedhli, S. (2013). An interior-point Benders based branch-and-cut algorithm for mixed integer programs. *Annals of Operations Research, 210*, 33–55.

Pigatti, A., Poggi de Aragão, M., & Uchoa, E. (2005). Stabilized branch-and–cut-and-price for the generalized assignment problem. *Electronic Notes in Discrete Mathematics, 19*, 389–395.

Thorsteinsson, E. S. (2001). Branch-and-check: A hybrid framework integrating mixed integer programming and constraint logic programming. In T. Walsh (Ed.), *CP -lecture notes on computer science 2239* (pp. 16–30). Springer.

Wang, G., Zhang, S., Wu, K., Zhang, Q., & Ni, L. (2014). Tim: Fine-grained rate adaptation in wlans. In *Proceedings of the international conference on distributed computing systems* (pp. 577–586).

Zhang, P. Y., Romero, D. A., Beck, J. C., & Amon, C. H. (2013). Solving wind farm layout optimization with mixed integer programming and constraint programming. In C. Gomes, & M. Sellmann (Eds.), *CPAIOR -lecture notes on computer science 7874* (pp. 284–299). Springer.