Applied Soft Computing 33 (2015) 150–169

Contents lists available at ScienceDirect

Applied Soft Computing

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

A biased random-key genetic algorithm for wireless backhaul network design

Carlos E. Andrade[a,*], Mauricio G.C. Resende[b,c], Weiyi Zhang[b], Rakesh K. Sinha[b], Kenneth C. Reichmann[b], Robert D. Doverspike[b], Flávio K. Miyazawa[a]

[a] Institute of Computing, University of Campinas, Avenida Albert Einstein 1251, Campinas, SP 13083-852, Brazil
[b] Network Evolution Research Department, AT&T Labs Research, 200 S. Laurel Avenue, Middletown, NJ 07748, USA
[c] Amazon.com, Inc., Mathematical Optimization and Planning Group, 333 Boren Avenue North, Seattle, WA 98109, USA

ARTICLE INFO

Article history:
Received 27 November 2014
Received in revised form 27 March 2015
Accepted 2 April 2015
Available online 16 April 2015

Keywords:
Wireless backhaul network design
Small cells
Genetic algorithm
Mixed integer programming model

ABSTRACT

This paper describes a biased random-key genetic algorithm for a real-world wireless backhaul network design problem. This is a novel problem, closely related to variants of the Steiner tree problem and the facility location problem. Given a parameter h, we want to build a forest where each tree has at most h hops from the demand nodes, where traffic originates, to the root nodes where each tree is rooted. Candidate Steiner nodes do not have any demand but represent locations where we can install cellsites to cover the traffic and equipment to backhaul the traffic to the cellular core network. Each Steiner node can cover demand nodes within a given distance, subject to a capacity constraint. The aggregate set of constraints may make it impossible to cover or backhaul all demands. A revenue function computes the revenue associated with the total amount of traffic covered and backhauled to the root nodes. The objective of the problem is to build a forest that maximizes the difference between the total revenue and the cost associated with the installed equipment. Although we will have a forest when we consider only the backhaul links and root nodes, the addition of demand vertices can induce undirected cycles, resulting in a directed acyclic graph. We consider instances of this problem with several additional constraints that are motivated by the requirements of real-world telecommunication networks.

1. Introduction

There has been a surge in the popularity of mobile devices (smart-phones and tablets). For example, in the United States, mobile devices now account for more than 50% of Internet usage [1]. This, coupled with the popularity of high bandwidth services such as video, has pushed mobile data usage. Cisco VNI Global IP Traffic Forecast [2] predicts 57% annual growth rate for mobile data, resulting in an 10-fold increase from 2014 to 2019. Service providers need to keep up with this growth in mobile data usage by providing better coverage and higher rates to their customers.

The associated network design problem needs to decide on the optimal concentration of cellular and Wi-Fi equipment to provide good service to users. Note that one can use Wi-Fi and LTE seamlessly for majority of the traffic without having to break and restart the session [3]. We also need to find the right backhaul strategy to route this traffic to the core network. A naive solution may be to run fiber to all sites, but it may be prohibitively expensive. Instead, we judiciously use the existing fiber infra-structure to pick the right backhaul. First of all, for installing Wi-Fi or cellular equipment, we may prefer a site that already has fiber. For other sites, we may be able to use a wireless backhaul to aggregate their traffic to a fibered site. Finally there are sites where running new fiber may be the best option.

In this paper, we propose the *wireless backhaul network design problem* (WBNDP) with practical constraints, and present a model to deal with real networks. The motivation of the proposed problem is to model wireless backhaul networks that operate over technologies such as Wi-Fi, LTE (4G technology), and HSPA+ (3G technology). In such networks, we must collect data traffic in a given geographic region and route it to the core network. Their structure is determined by equipment and service quality constraints. Usually, one wants to build a tree spanning high-capacity nodes over a sparse graph with capacity constraints. Although the WBNDP resembles variants of the Steiner tree and facility location problems, its revenue and cost structures distinguish it from these two problems.

* Corresponding author. Tel.: +55 19 3521 5882.
E-mail addresses: andrade@ic.unicamp.br (C.E. Andrade), resendem@amazon.com (M.G.C. Resende), maxzhang@research.att.com (W. Zhang), sinha@research.att.com (R.K. Sinha), kcr@research.att.com (K.C. Reichmann), rdd@research.att.com (R.D. Doverspike), fkm@ic.unicamp.br (F.K. Miyazawa).

http://dx.doi.org/10.1016/j.asoc.2015.04.016
1568-4946/

To solve the WBNDP, we propose a biased random-key genetic algorithm (BRKGA) with a sophisticated decoding procedure. The algorithm has several phases, such as equipment deployment, construction of routing trees, flow computation, and pruning of unused equipment. The choice of BRKGA is grounded on its recent success in solving large combinatorial optimization problems [4]. In order to evaluate the quality of results obtained by the BRKGA, we also formulate and solve a mixed integer linear programming model for computing the optimal solution.

The structure of the paper is as follows. Section 2 reviews the related literature. Section 3 presents the WBNDP in detail and Section 4, a formal definition as well as a mixed integer linear programming model are presented. Section 5 describes a biased random-key genetic algorithm to solve the WBNDP and Section 6 discusses the maximum flow problem that arises in WBNDP. Section 7 describes some instances derived from real-world problems and pre- and post-processing phases. Section 8 presents experimental results. Concluding remarks are made in Section 9.

2. Related literature

The wireless backhaul network design problem (WBNDP) is closely related to variants of the Steiner tree and the facility location problems. Steiner tree problems have been widely studied and among their many applications, network design may be the most expressive. Given a graph with a set of *terminal nodes*, a set of intermediate nodes called *Steiner nodes*, and edge costs, the Steiner Tree Problem (STP) consists in creating a minimum cost tree connecting all terminal nodes. The general version is $\mathcal{NP}$-hard (Garey and Johnson [5]) and a survey of the STP is presented in Voß [6].

A variant with many applications in industry is the Prize-Collecting Steiner Tree Problem (PCSP). In this problem, each vertex has a penalty value and each edge has a cost. The objective is to build a tree minimizing the sum of the costs of used edges plus the sum of the penalties of the vertices not spanned by the tree. This is a well-studied problem originated in Goemans and Williamson [7,8]. Goemans and Williamson [8] presented a 2-approximation algorithm for the PCSP for which practical results were discussed in Johnson et al. [9] and Canuto et al. [10]. Lucena and Resende [11] presented lower bounds using linear programming and cutting planes and their results were improved by Ljubić et al. [12]. Klau et al. [13] proposed a hybrid heuristic where the final population from a memetic algorithm is used to build reduced instances to be solved by an exact algorithm. In da Cunha et al. [14] a Lagrangian non-delayed relax-and-cut algorithm is proposed to generate primal and dual bounds for the problem.

The first problem to deal with limited number of hops was proposed by Gouveia [15] and is called Hop-Constrained Minimum Spanning Tree Problem (HMSTP). In this problem, we want to obtain a minimum spanning tree such that the path between the root node and a leaf node has no more than H hops (edges). In Gouveia [15], a formulation based on subcycle elimination inequalities was presented as well as several lifting procedures and bounds based on Lagrangian relaxation. This approach was refined in several papers described in Dahl et al. [16]. Recently, Gouveia et al. [17] presented several local search neighborhoods resulting in good solutions. In Gouveia et al. [18], HMSTP was presented as a directed Steiner tree model over layered graphs. Several cutting planes from other Steiner problems were applied. A branch-and-cut algorithm was also proposed and it was able to obtain the best results so far described in the literature. Furthermore, Gouveia et al. [18] reduced the Diameter-Constrained Minimum Spanning Tree Problem (DMSTP) to a Steiner tree problem using the same technique. In the DMSTP, the hop constraint is applied to a path between any pair of vertices in the tree.

Another related problem is the Steiner Tree Problem with Revenues, Budget and Hop Constraints (STPRBH). In this problem, the objective is to build a tree that maximizes the collected profit respecting an upper bound on network costs and maximum number of hops from the root node to other any node in the network. Differing from the HMSTP, in the STPRBH it is not mandatory to include all vertices and one has a limited budget to spend building the network. This problem was proposed by Costa et al. [19] in which a two-phase greedy algorithm was developed: one phase consists in a simple local search that destroys part of a solution and rebuilds it greedily, and a tabu search using two simple search neighborhoods. Later, Costa et al. [20] presented several integer linear programming models for the STPRBH and branch-and-cut algorithms were developed for each formulation. Layeb et al. [21] presented a compact formulation based on Miller–Tucker–Zemlin constraints which resulted in similar solutions to those found previously in the literature. Recently, Fu and Hao [22] proposed a new heuristic for the STPRBH which was able to find optimum solutions for instance with known optima and improved solutions for instances with unknown optima.

A similar problem is the Connected Facility Location Problem (ConFL) introduced in Karger and Minko [23]. This problem consists in assigning each client to exactly one opened facility and connecting the opened facilities using a Steiner tree. Ljubić and Gollowitzer [24] introduced a modification in the ConFL by limiting the number of hops between the facilities and a root node. This problem is known as the Hop Constrained Connected Facility Location Problem (HCConFL). Ljubić and Gollowitzer [24] used the same technique presented in Gouveia et al. [18] where the problem is modeled on a layered graph. Branch-and-cut algorithms were developed.

One can note that the problem addressed in this paper (WBNDP) has characteristics that are similar to those of the problems reviewed above. This is especially true with respect to the maximum number of hops. However, there are two main characteristics that distinguish the WBNDP from other problems. The first is the possibility that each demand node be served by more than one Steiner node or root node. In this sense, we do not have a tree like in the other problems. The second and most important characteristic is how the revenue is computed. In previous papers, the revenue considers the "full value" of a vertex if it is in the solution i.e., the backhaul network is capable of routing all the traffic. In the WBNDP, due to natural constraints, this is not always true. In fact we consider that the network has a limited routing capacity and the revenue is a function of the maximum flow in this network. This way, there is a strict relation between the network structure and the revenue, once both topology and link capacity directly influence the flow. Another small difference is that the degree constraints are applied only to Steiner nodes and macrocells (only to wireless links and not to fiber links).

3. Problem description

Consider a geographical region with each point in the region identified by its coordinates (latitude and longitude). Consider the graph $G = (V, A)$, where V and A are, respectively, the sets of vertices and arcs of G. The set $V^d \subset V$ is the subset of vertices that correspond to demand points or city blocks in this region. These demand points consist of mobile devices, such as mobile phones, tablets, laptops, and other devices that make use of wireless communication. The set $V^s \subset V$ is the set of vertices that correspond to equipment used to collect and route traffic from the demand points. This equipment is installed on utility poles distributed across the streets and highways of the region. We consider *small cells* network design, meaning that cellsites have relatively small coverage radii. Small cells have the advantage of using spectrum

more efficiently. We consider three types of equipment: Wi-Fi and LTE for access traffic (demand collection), and retransmitter for routing/backhauling. The set $V^r \subset V$ represents the *Fibered Access Points* (FAPs) where we have existing fiber connected to the core network. VRADs (Video-Ready Access Devices) are one example of FAPs. Installing equipment close to a VRAD will enable us to route the traffic on this fiber with no or little additional cost. We also have many existing *macro cellsites* where we can add additional radio equipment (eNB) to carry traffic. If this macro has a fiber backhaul, we can technically consider it a FAP but because macros also cover access traffic directly, we make a distinction between macro cells and other FAPs for notational convenience. Finally certain macro cellsites are connected to the core with very high speed wireless links. In our formulation, they serve the same purpose as macros with a fiber backhaul and we treat them identically.

The objective is to create routing trees such that the trade-off between the revenue derived from the routed traffic and the network cost be the best possible. In these networks, there are a large number of peculiarities that differ them from Steiner trees. We call this problem the *wireless backhaul network design problem* (WBNDP) and in the following we describe each one of these particularities so that later on we can present a formal definition.

3.1. Demand splitting and routing trees

In general, it is difficult to estimate the demand of each user due to the user's mobility in the region where the network will be built. One way to approximately model this scenario is to concentrate, for each city block, its total demand at the center of the block. Although this appears to be oversimplified, the errors are diluted by the mobility of the users. Therefore, it is reasonable to assume that a certain block may be served by several cellsites and its demand split among them. Although the demand may be split among several pieces of access equipment, this equipment and the routing equipment must be connected through trees rooted at the FAPs or the macrocells. Thus, one can note that a solution S for a backhaul network may contain an undirected cycle originated at a demand. Therefore, although we have a forest when we consider only the Steiner vertices and root nodes of S, the addition of demand vertices can induce these undirected cycles, resulting in a directed acyclic graph (DAG).

3.2. Capacity of access equipment

Each access equipment is limited in its handling of traffic. These constraints are access radius and access capacity. Values of these parameters vary across vendors and also depend on geographic terrain. We have used the following representative values in our simulations. For access radius, we have Wi-Fi: 100 m; LTE small cell: 400 m; LTE macro cell and HSPA: 3000 m. For access capacity, we have Wi-Fi: 100 Mbps; LTE small cell: 20 Mbps single band, 40 Mbps dual band; LTE macro cell and HSPA: 25 Mbps. Although, for the sake of simplification, we considered LTE macro cell and HSPA to have the same access radius and capacity, even though these two technologies can present different values. For more details about these limitations, see [25].

3.3. Capacity of retransmitter equipment

The capacity of retransmitter equipment is one of the most complex aspects of this type of network. In addition to having a maximum access radius (representative value of 1000 m), retransmitter equipment also have a physical restriction that limits the sum of the flow that it receives and sends to the other retransmitters. This quantity is limited to a value U_{bh} (e.g., 100 Mbps). Though this equipment also deals with access traffic from other equipment (Wi-Fi/LTE) that share the same utility pole, this limit is not applied to that traffic. Therefore, the limit U_{bh} does not account for all incoming traffic but shapes the outgoing traffic.

More precisely, let $v \in V^s$ be a retransmitter. Take $A_v^{+s} = \{(w, v) \in A : w \in V^s\}$ as the set of arcs outgoing from neighbors of v that send to it backhaul traffic. Let a_v^- be the outgoing arc of v in the solution (recall that we are looking for a backhaul forest). Let $A_v^{+d} = \{(w, v) \in A : w \in V^d\}$ be the set of arcs from demand vertices to v. Let $f : A \to \mathbb{R}^+$ be a function that defines the flow on the arcs. Therefore, we have

$$\sum_{e \in A_v^{+s}} f_e + f_{a_v^-} \leq U_{bh}, \qquad \forall v \in V^s \tag{1}$$

such that

$$\sum_{e \in A_v^{+s}} f_e + \sum_{e \in A_v^{+d}} f_e = f_{a_v^-}, \qquad \forall v \in V^s. \tag{2}$$

While Inequality (1) only restricts the backhaul flow to at most the capacity U_{bh}, Equality (2) is the classical flow conservation equation which ensures that v has no excess flow. Although these restrictions do not influence the structure of the backhaul forest, they have a direct impact on the maximum flow in the forest, used to compute the revenue. In this case, we cannot use a classical algorithm for maximum flow such as Edmonds–Karp or Goldberg–Tarjan directly (see [26] for a comprehensive list).

Another physical restriction on the retransmitters is that they support only a limited number of neighbors sending backhaul traffic to it (known as *fan-in*, equal to 5 in our simulation results). Thus, this restriction limits the incoming degree of each retransmitter vertex.

3.4. Sight and minimum distance

One can note that the subgraph induced by V^s may be sparse. The main reason for this is the existence of physical barriers, such as buildings and hills between pairs of utility poles. Moreover, the distance between them can be so large that the signal loses strength and impairs communication. Therefore, one must consider that the links (u, v) and (v, u) only exists when poles u and v are close enough (1000 m in our simulations) and they are in each other's *line of sight*, i.e. we can trace a straight line between them without obstruction.

Another important aspect is the interference among cellsites. Both LTE and HSPA use licensed spectrum and can interfere with each other. Therefore, a pair of LTE equipment cannot be installed too close together. In our simulations, we restrict a minimum distance of 300 m between two LTE small cellsites and a minimum distance of 500 m between LTE small cell and macro cellsites. Note that this concern does not exist with Wi-Fi equipment which, although can exhibit interference, occupies a non-licensed spectrum. Such spectrum is very hard to control due to external interferences.

3.5. Number of hops

The first hops are defined to be arcs that link root nodes to poles. In practice, these hops may be built using either wireless links or fiber links. The links between utility poles and FAPs must use fiber. Links between poles and macrocells may use fiber or be wireless. The other hops are built over wireless links. The restriction is that the number of wireless hops must be limited to an upper bound (in practice, two or three hops). This restriction aims to reduce the

latency of the wireless network and is commonly considered in network planning [16]. Note that only backhaul links are considered.

Example

Fig. 1a depicts an example of a base graph where the dashed arcs represent possible wireless links and the solid blue arc represents an optical fiber link. The dotted sinuous arcs represent the possible links between a demand block and an access equipment. This graph is based on geographic information of the backhaul region and takes into consideration maximum coverage radii as well as the lines of sight of pairs of equipment. Note that the macrocell (represented by a large yellow rhombus/diamond) contains two attached vertices. These are the backhaul traffic aggregator (BTA) and the macro cell traffic aggregator (MTA) (represented by small yellow diamonds). Note that these vertices are virtual and the links between them are internal to the devices. They are used to better model the traffic (more detail is given in Section 4). As with the macrocells, each utility pole is represented by a red square and has attached to it two virtual vertices representing Wi-Fi and LTE equipment and their respective aggregators (small orange diamonds for WTAs and purple diamonds for LTAs). The demands are represented by light blue circles. Note that each demand may be satisfied in three distinct ways: Wi-Fi, LTE, and HSPA. In this example, only demands 0 and 2 can be served by the macro cell because of their proximity to the macrocell. Demand 2 can split its traffic between the macrocell and utility poles UP0 and UP1 using LTE. The same situation occurs with demand 3. Demand 1 can use Wi-Fi or LTE on pole UP2, and LTE on pole UP0. Demand 4 is only close to pole UP4 and can use both Wi-fi and LTE. Note that utility poles UP0, UP1, and UP2 are close enough to each other and in each other's lines of sight, enabling the installation of retransmitters on them. Utility poles UP3 and UP4 are in another region and do not have communication links with UP0, UP1, or UP2.

Fig. 1b shows a valid solution. The solid arcs represent the links among the pieces of equipment. In this example we ignore capacities. Note that several pieces of equipment are not installed since they do not serve any demand. For example, in utility pole UP2 it suffices to install Wi-Fi. Note that if we consider only the black and blue arcs (straight lines), we have a forest. Note also that demand 2 splits its traffic between utility pole UP0 using LTE and the macrocell. Demand 3 is more interesting: it is split between two distinct subtrees. In this case, demand 3 uses LTE on UP3 and UP1 but nothing prevents it from using the Wi-Fi on UP1 and LTE on UP3.

4. Formal definition

Let V^d be the set of demand points or *demand nodes*. Let V^s be the set of utility poles where the access and retransmission equipment can be installed. We refer to V^s as *Steiner nodes*. Let V^r be the set of points which have fiber or a high capacity link which we call *root nodes*. Consider $V^m \subseteq V^r$ as the set of macrocells and $V^v \subseteq V^r$ as the set of FAPs. Let $V = V^d \cup V^s \cup V^r$ and note that $V^m \cup V^v = V^r$, and the sets V^d, V^s, V^m, and V^v are pairwise disjoint. Consider $V^{ld} \subseteq V^r$ to be the set of FAPs whose through traffic is leased.

To model the traffic in the different technologies, consider the following sets:

- *WTA*: set of vertices that represents units of Wi-Fi equipment to be installed on the poles. We call each $w \in WTA$ a *Wi-Fi Traffic Aggregator*. There is a one-to-one correspondence between Wi-Fi traffic aggregators and utility poles;
- *LTA*: set of *LTE Traffic Aggregators* whose description is similar to *WTA*, except that they are for LTE equipment;
- *MTA*: set of *Macrocell Traffic Aggregators* which have a one-to-one correspondence with each macrocell;
- *BTA*: set of *Backhaul Traffic Aggregators* which also have a one-to-one correspondence with each macrocell. The BTAs are restricted to wireless backhaul traffic.

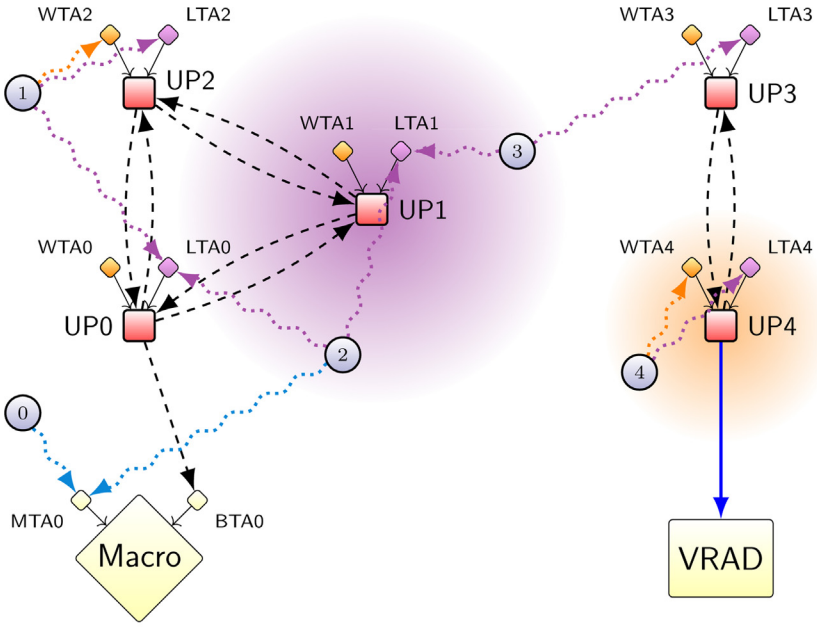
Let $v \in V^s$, $u \in WTA$. If u is assigned to pole v, then $wta(v) = u$ and $wta(u) = v$. Consider the same for *LTA*, *MTA*, and *BTA*. As in previous sections, consider a directed graph $G = (W, A)$ such that $W = V \cup WTA \cup LTA \cup MTA \cup BTA$. We will define the set of arcs A later. Let $d : V^d \to \mathbb{R}^+$ be the function that maps the maximum traffic (in Mbps) that originates at each demand point.

Consider the following constants:

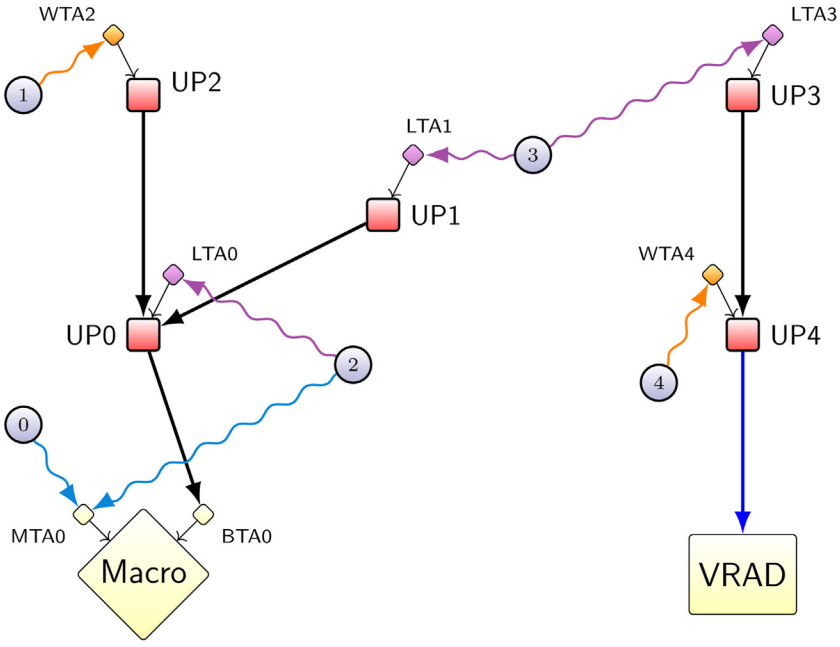
- Maximum number of wireless hops: H;
- Access Radii (in m):
  - Wi-Fi access radius: R_{wifi};
  - LTE access radius: R_{lte};
  - Macrocell access radius: R_{mc};
  - Retransmitter radius: R_{bh};
- Minimum distance (in m):
  - LTE to LTE (pole to pole): δ_{lte};
  - LTE (pole) to macrocell: δ_{macro};
- Capacities (in Mbps):
  - Wi-Fi: U_{wifi};
  - LTE: U_{lte};
  - Macrocell (HSPA and LTE access): U_{mc};
  - Retransmitter: U_{bh};
- Revenue factor (in some monetary unit): ϱ;
- Cost (in some monetary unit):
  - Equipment deployment on a pole: C_p;
  - Wi-Fi equipment: C_{wifi};
  - LTE equipment: C_{lte};
  - Retransmitter with one-element antenna (fan-in = 1): C_{fan1};
  - Retransmitter with two-or-more-element antenna (fan-in ≥ 2): C_{fan2};
  - Maintenance of equipment on a pole (per year): C_{man}. The maintenance cost is the sum of the pole leasing costs and small cell maintenance costs, per year. Note that, if the pole has no access equipment but has a retransmitter, the maintenance cost shall be paid;
  - Macrocell annual cost: C_{mc}. This costs is compounded by the annual site leasing cost and the annual maintenance cost;
  - Meter of deployed fiber: C_{fiber};
  - Leased traffic (in \$/Mbps): C_{ld};
- Maximum number of incoming backhaul neighbors: δ^+_{bh};
- Length of fibered link: ℓ_{uv} for arc (u, v);
- Maximum length of fibered links: R_{fiber}. Theoretically, this limit does not exist since we may spread fiber over the entire network. In practice, the cost to do this is very high and therefore we impose this limit. However, this is a weak restriction which may be violated if necessary.

Let $dist : V \times V \to \mathbb{R}^+$ be the (geodesic) distance between two points. For $v \in V^s \cup V^m$, consider $LOS_v = \{w \in V^s \cup V^m : dist(v, w) \leq R_{bh}$ and there is direct line of sight between v and $w\}$. The set of arcs A is defined as the union of the following sets:

- $A^{dm} = \{(u, v) : u \in V^d, v \in MTA$ and $dist(u, mta(v)) \leq R_{mc}\}$: Set of arcs from demand blocks to macrocells (MTAs) whose blocks are inside the radius of action of a macrocell;
- $A^{dw} = \{(u, v) : u \in V^d, v \in WTA$ and $dist(u, wta(v)) \leq R_{wifi}\}$: Set of arcs from demand blocks to utility poles (WTAs) inside the radius of action of Wi-Fi equipment;



(a) Base graph.



(b) Valid solution.

Fig. 1. The figures above represent a base graph from where a valid solution is extracted. We omitted equipment capacities for legibility. (For interpretation of the references to color in the text, the reader is referred to the web version of this article.)

- $A^{dl} = \{(u, v) : u \in V^d, v \in LTA \text{ and } dist(u, lta(v)) \leq R_{lte}\}$: Set of arcs from demand blocks to utility poles (LTAs) inside the radius of action of LTE equipment;
- $A^{ss} = \{(u, v) : u, v \in V^s \text{ and } v \in LOS_u\}$: Set of arcs between utility poles. Note that both (u, v) and (v, u) belongs to A^{ss};
- $A^{smw} = \{(u, v) : u \in V^s, v \in BTA \text{ and } u \in LOS_{bta(v)}\}$: Set of arcs from utility poles to macrocells (BTAs). This set is restricted to wireless links;
- $A^{smf} = \{(u, v) : u \in V^s, v \in V^m \text{ and } dist(u, v) \leq R_{fiber} + \epsilon\}$: Set of arcs from utility poles to macrocells using fibered links (recall that this is a weak restriction which can be relaxed by adjusting the value of ϵ);
- $A^{sv} = \{(u, v) : u \in V^s, v \in V^v \text{ and } dist(u, v) \leq R_{fiber} + \epsilon\}$: Set of arcs from utility poles to FAPs (same as before);
- $A^{wta} = \{(wta(v), v) : v \in V^s\}$: Set of arcs from WTAs to utility poles;
- $A^{lta} = \{(lta(v), v) : v \in V^s\}$: Set of arcs from LTAs to utility poles;
- $A^{mta} = \{(mta(v), v) : v \in V^m\}$: Set of arcs from MTAs to macrocells;
- $A^{bta} = \{(bta(v), v) : v \in V^m\}$: Set of arcs from BTAs to macrocells.

Note that arc sets $A^{dm}, A^{dw}, A^{dl}, A^{ss}$, and A^{smw} correspond to wireless links. Arc sets A^{sv} and A^{smf} correspond to fibered links. Arc sets A^{wta}, A^{lta}, A^{mta}, and A^{bta} represent connections between several pieces of equipment in the same small cell or macrocell. We use these arcs to model equipment capacities.

4.1. Mixed integer linear programming model

To better describe the objective function and constraints of the WBNDP, we next model it as a mixed integer linear program (MIP). The constraints are based on the observations of Section 3 and on topological restrictions of a DAG, as well as maximum capacity and flow conservation limitations. One important definition to model the WBNDP, it is the notion of level. We define *level* as the number of wireless hops between a utility pole and a root node. Consider the following decision variables:

- $x^p_{uv} \in \{0, 1\}$ for $(u, v) \in A$, $u \in V^s$, $v \in V^s \cup V^r \cup MTA$: $x^p_{uv} = 1$ indicates that arc (u, v) is in level p of some tree, for $p = 0, \ldots, H$; $x^p_{uv} = 0$, otherwise. For $v \in V^r$, only variables x^0_{uv} are defined. For $v \in BTA$, only variables x^1_{uv} are defined;
- $y_v \in \{0, 1\}$ for $v \in V^s$: $y_v = 1$ indicates that a vertex/pole v is in the solution; $y_v = 0$, otherwise;
- $f_{uv} \in \mathbb{R}^+$ for $(u, v) \in A$: is the flow through arc (u, v);
- $t_v \in \{0, 1\}$ for $u \in V^s$: $t_v = 1$ indicates that pole v has installed Wi-Fi equipment; $t_v = 0$, otherwise;
- $z_v \in \{0, 1\}$ for $u \in V^s$: $z_v = 1$ indicates that pole v has installed LTE equipment; $z_v = 0$, otherwise;
- $a^1_v \in \{0, 1\}$ for $v \in V^s \cup BTA$: $a^1_v = 1$ indicates that the pole or the macrocell has a retransmitter with a one-element antenna (fan-in = 1); $a^1_v = 0$, otherwise;
- $a^2_v \in \{0, 1\}$ for $v \in V^s \cup BTA$: $a^2_v = 1$ indicates that the pole or the macrocell has a retransmitter with a two-or-more-element antenna (fan-in ≥ 2); $a^2_v = 0$, otherwise.

Variables x are used to model the backhaul trees in levels. If an arc (u, v) is in level 0 (i.e., $x^0_{uv} = 1$), then the arc is a fibered link of high capacity. If the same arc is in level 1 (i.e., $x^1_{uv} = 1$), then it is a wireless link. Note that the variables x may be not defined for all arcs (u, v) and all levels p while some arcs only appear in deep levels of some tree. For instance, arc (UP1, UP2) from the example of Fig. 1a can only appear in level 3. Such cases can be identified in a preprocessing phase, as we will describe in Section 7.2.

The following MIP models the WBNDP:

$$\max \quad \varrho \sum_{(u,v):v \in V^r} f_{uv} \tag{3a}$$

$$-\sum_{v \in V^s} (C_p + YC_{man}) y_v \tag{3b}$$

$$-\sum_{v \in V^s} C_{wifi} t_v - \sum_{v \in V^s} C_{lte} z_v \tag{3c}$$

$$\sum_{v \in V^s \cup V^m} (C_{fan1} a^1_v + (C_{fan2} - C_{fan1}) a^2_v) \tag{3d}$$

$$- \sum_{(u,v) \in A^{sv} \cup A^{smf}} C_{fiber} \ell_{uv} x^0_{uv} \tag{3e}$$

$$- \sum_{(u,v) \in A: v \in V^{ld}} C_{ld} f_{uv} + C_{mc} \tag{3f}$$

$$\text{s.t.} \quad \sum_{p=0}^{H} x^p_{uv} \leq 1 \qquad \forall (u, v) \in A : u \in V^s \tag{3g}$$

$$x^{p+1}_{uv} \leq \sum_{(v,w) \in A: w \neq u} x^p_{vw} \qquad \forall (u, v) \in A : u \in V^s, \; p = 0, \ldots, H-1 \tag{3h}$$

$$\sum_{(u,v) \in A} \sum_{p=1}^{H} x^p_{uv} \leq \delta^+_{bh} y_v \qquad \forall v \in V^s \cup BTA \tag{3i}$$

$$\sum_{(v,w) \in A} \sum_{p=0}^{H} x^p_{vw} \leq y_v \qquad \forall v \in V^s \tag{3j}$$

$$a^1_v \geq \frac{1}{\delta^+_{bh}} \left(\sum_{(u,v) \in A} \sum_{p=1}^{H} x^p_{uv} \right) \qquad \forall v \in V^s \cup BTA \tag{3k}$$

$$a^2_v \geq \frac{1}{\delta^+_{bh}} \left(\sum_{(u,v) \in A} \sum_{p=1}^{H} x^p_{uv} - 1 \right) \qquad \forall v \in V^s \cup BTA \tag{3l}$$

$$a^1_v \geq \sum_{(v,w) \in A} \sum_{p=1}^{H} x^p_{vw} \qquad \forall v \in V^s \tag{3m}$$

$$t_v \leq y_v \qquad \forall v \in V^s \tag{3n}$$

$$z_v \leq y_v \qquad \forall v \in V^s \tag{3o}$$

$$z_u + z_v \leq 1 \qquad \forall u, v \in V^s : dist(u, v) \leq \delta_{lte} \tag{3p}$$

$$z_u = 0 \qquad \forall u \in V^s, v \in V^m : dist(u, v) \leq \delta_{macro} \tag{3q}$$

$$\sum_{(v,w) \in A} f_{vw} \leq d_v \qquad \forall v \in V^d \tag{3r}$$

$$f_{uv} \leq U_{wifi} t_v \qquad \forall (u, v) \in A : u \in WTA \tag{3s}$$

$$f_{uv} \leq U_{lte} z_v \qquad \forall (u, v) \in A : u \in LTA \tag{3t}$$

$$f_{uv} \leq \sum_{p=1}^{H} U_{bh} x^p_{uv} \qquad \forall (u, v) \in A : u, v \in V^s \cup BTA \tag{3u}$$

$$f_{uv} \leq M x^0_{uv} \qquad \forall (u, v) \in A : u \in V^s, v \in V^r \tag{3v}$$

$$\sum_{(u,v) \in A: u \in V^s} f_{uv} + \sum_{(v,w) \in A} f_{vw} \leq U_{bh} \qquad \forall v \in V^s \tag{3w}$$

$$\sum_{(u,v) \in A} f_{uv} - \sum_{(v,w) \in A} f_{vw} = 0 \qquad \forall v \in V \setminus (V^r \cup V^d) \tag{3x}$$

$$f_{vw} \leq U_{bh} \qquad \forall v \in BTA \tag{3y}$$

$$f_{vw} \leq U_{mc} \qquad \forall v \in MTA. \tag{3z}$$

Terms (3a)–(3f) constitute the objective function and computes the net profit that the network generates in a time windows of $Y \in \mathbb{R}^+$ years. In practice, the revenue is computed by a complex function based on service packages offered to customers and, in general, is an estimate based on the experience of operators and on market fluctuations. For the particular scenario considered in this paper, the revenue is a function of the total traffic routed through the FAPs. We consider a linear function using the revenue factor ϱ as shown by Term (3a). The remaining terms add up to the total cost: Term (3b) is the cost of deployment and maintenance of poles

over Y years; Term (3c) is the cost of Wi-Fi and LTE equipment; Term (3d) is the cost of retransmitters; Term (3e) is the cost of trenching for fiber; and Term (3f) is the cost of leased traffic. The constant C_{mc} represents the macrocell cost as described in the beginning of this section.

The constraints can be partitioned into three blocks.

The first block ((3g)–(3j)) models the backhaul trees: Constraint (3g) forbids an arc to be in more than one level; Constraint (3h) requires that if an incoming arc into node v is in level $p+1$, v should have an outgoing arc in level p; Constraint (3i) limits the incoming degree for poles and BTAs according to Section 3.3 (note that the constraint only consider levels greater than or equal to one, since fibered arcs of level 0 are only allowed to be incoming arcs at FAPs and macrocells); and Constraint (3j) guarantees that each pole has at most one outgoing arc.

The second block of constraints ((3k)–(3p)) is tied to equipment deployment. Constraints (3k) and (3l) indicate, respectively, the presence of a retransmitter with a one-element antenna or a multiple-element antenna. Note that if, for some node v, $a_v^2 = 1$, then necessarily $a_v^1 = 1$. In this case, we do not consider a one-element antenna subtracting its cost from the objective function as shown in Term (3d). Constraint (3m) guarantees placement of a retransmitter in node v if there exists an arc outgoing from v. Constraints (3n) and (3o) permit the deployment of Wi-Fi and LTE equipment on pole v, respectively, only if pole v is used. Constraints (3p) and (3q) prohibit the deployment of two pieces of LTE equipment near each other or close to a macrocell, respectively. Note that Constraint (3q) may be redundant since closeby pole/macrocell pairs are discarded in a preprocessing phase.

The last block of constraints ((3r)–(3z)) is related to flow. Constraint (3r) limits the flow outgoing from demand blocks. Constraints (3s) and (3t) ensure that Wi-Fi and LTE capacities are respected. Constraint (3u) limits the capacity of wireless arcs to the retransmitter capacity. Constraint (3v) enables unlimited flow on fibered links when $M \geq \sum_{v \in V^d} d_v$. Constraint (3w) limits the retransmitter capacity as discussed in Section 3.3 (note that only backhaul flow is considered in this constraint). Constraint (3x) is the classical flow conservation constraint. Finally, Constraint (3y) applies the retransmitter capacity to BTAs, while Constraint (3z) limits macro cellsite traffic. All integrality requirements are omitted since they are described in the beginning of this section.

5. Solution procedure using BRKGA

Because of the large-scale nature of practical instances of MIP (3), it can be difficult to solve the WBNDP using an exact approach, such as a branch-and-cut algorithm. Typically, these instances are situated in regions of approximately 80 km^2 with about 15 macrocells, 130 VRADs, 3200 utility poles, and 16,000 demand blocks. Although the underlying graph is relatively sparse, the number of valid solutions can be huge. To deal with this situation, we propose a Biased Random-Key Genetic Algorithm (BRKGA) to solve the WBNDP. See Gonçalves and Resende [3] for an introduction to BRKGA. The main reason for opting for a BRKGA is that this metaheuristic has been shown to be adaptable to a wide variety of combinatorial optimization problems, such as packing [27], routing [28], clustering [29], and winner determination [30].

From an implementation point of view, most of the effort in building a BRKGA focuses on devising a decoder, which takes as input a vector of random keys and outputs a valid solution for the problem. The decoding phase for the WBNDP consists in iteratively building acyclic directed graphs and the computation of maximum flows, costs, and revenues. This procedure builds the solution in a *bottom-up* fashion, starting at the root nodes and ending at the demand nodes. Since the WBNDP has a large number of peculiarities, the design of a decoder is not trivial and consists of a number of intermediate steps. In the next subsections, we describe each of these steps.

5.1. Representation

Given an instance of the problem, without loss of generality we assume that the utility poles are listed in an arbitrary but fixed order $V^s = (\overline{p}_1, \ldots, \overline{p}_n)$ where n is the number of poles, i.e., $n = |V^s|$. We also assume that the root nodes are listed in an arbitrary but fixed order beginning with macrocells and ending with the FAPs, i.e, $V^r = (\overline{m}_1, \ldots, \overline{m}_\alpha, \overline{f}_1, \ldots, \overline{f}_\beta)$ where α is the number of macrocells and β is the number of FAPs.

A chromosome is a vector of real numbers $\mathbf{v} \in [0, 1]^{5n}$. This vector is partitioned into five sections whose values are used to build the backhaul network. Each value is associated with a utility pole through an index given by V^s. As we explain below, each pole is associated with five values, or keys, of the chromosome, one in each partition.

The first section consists of values $\mathbf{v}_1, \ldots, \mathbf{v}_n$ that define which poles are present in the solution and their deployment order. The deployment order defines the sequence of LTE equipment installation. We refer to this first group as $\kappa^A = (\mathbf{v}_1, \ldots, \mathbf{v}_n)$.

The second section consists of the values $\mathbf{v}_{n+1}, \ldots, \mathbf{v}_{2n}$ which are used as activation parameters. They determine whether:

- an LTE equipment is deployed on the utility pole (L or NL);
- the utility pole is connected via fiber (F or NF);
- the utility pole connects directly to a FAP or a macrocell (D or ND).

Using this notation we have:

- $v_i \in [0.000, 0.125)$, then: NL, NF, ND;
- $v_i \in [0.125, 0.250)$, then: NL, NF, D;
- $v_i \in [0.250, 0.375)$, then: NL, F, ND;
- $v_i \in [0.375, 0.500)$, then: NL, F, D;
- $v_i \in [0.500, 0.625)$, then: L, NF, ND;
- $v_i \in [0.625, 0.750)$, then: L, NF, D;
- $v_i \in [0.750, 0.875)$, then: L, F, ND;
- $v_i \in [0.875, 1.000)$, then: L, F, D.

The first parameter controls how to distribute LTE equipment and, as a consequence, how the demands are distributed amongst them. Furthermore, it may be valuable not to install LTE on a given utility pole even if there is nearby demand, since the cost/benefit ratio may be too low or negative. The second parameter dictates if a pole can be connected by fiber. It is used in cases where the utility pole is connected to a macrocell by fiber or by a wireless link. Again, this parameter controls the cost/benefit ratio. Lastly, the third parameter dictates the minimum tree level in which a pole appears in the backhaul network. We refer to these keys as $\kappa^P = (\mathbf{v}_{n+1}, \ldots, \mathbf{v}_{2n})$.

The third section $\kappa^O = (\mathbf{v}_{2n+1}, \ldots, \mathbf{v}_{3n})$ defines the evaluation order of the utility poles as the network is grown. This order is used to build the next level of nodes in the forest. Suppose, for example, that a utility pole is already connected to the network and that it can only support one additional backhaul connection (i.e., Constraint (3i) is nearly saturated). If there are two other poles to be connected to this one, the order induced by κ^O will define which pole will be connected. Another order induced by $\kappa^{O'}$ can potentially generate a different network.

The fourth section $\kappa^N = (\mathbf{v}_{3n+1}, \ldots, \mathbf{v}_{4n})$ defines the evaluation order of a pole neighborhood. Suppose that a pole is about to be included in the network and that it has two or more previously deployed neighbor nodes with utility poles to which it can

connect. The order induced by κ^N will determine to which neighbor the connection will be made.

Lastly, the fifth section $\kappa^L = (\mathbf{v}_{4n+1}, \ldots, \mathbf{v}_{5n})$ defines the minimum tree level on which a utility pole can be placed. The root nodes and fibered utility poles are considered to be at level zero. Thus, the remaining poles are distributed among levels 1 through H. The minimum level of utility pole i is given by $\lfloor (H+1)\kappa_i^L \rfloor$. If the minimum level of pole i is zero, it can be placed in any tree level. If it is one, pole i can only be placed in level 1 or above. In this case, the pole cannot be connected by fiber.

Although at first glance, there is a superposition of functionalities among the several keys, we see below that each key plays a very particular role in the process of network construction.

5.2. Decoder

Algorithm 1 shows the basic steps to decode a chromosome into a valid solution. It consists of several procedures described in Algorithms 2–5. In addition to the above discussion, the algorithms also make use of the following definitions:

- *parent*(p): indicates which utility pole, macrocell, or FAP that pole p is linked to, i.e., arc (p, *parent*(p)) exists in the backhaul network;
- *children*(p): set of poles linked to p, i.e., there exist links (w, p) for $w \in$ *children*(p);
- *level*(p): indicates the level in which pole p is placed in the tree;
- *fibered*(p): indicates if pole p is linked by fiber to a root node;
- $\deg^+(p)$: number of arcs leaving p (i.e., outgoing degree of p);
- *CP*: set of the most external poles in the current stage of network construction, i.e., the leaves of current forest;
- *TK*: set of all utility poles in current forest. Note that $CP \subseteq TK$;
- *NL*: set of utility poles to be considered for connection in the next forest levels.

Algorithm 1. Decoder.

1 Define the activation order and install (LTE) equipment in the poles;
2 Build the backhaul graph;
3 Remove non-used pieces of equipment and poles (1st phase);
4 Compute the maximum flow. Let f_{max} be the value of maximum flow;
5 Remove non-used pieces of equipment and poles (2nd phase);
6 Compute the cost over the time window of Y years. Let C_{total} be the total cost;
7 Compute the revenue over the time window of Y years. Let R_{total} be the revenue obtained;
8 **return** $R_{total} - C_{total}$

The first step is to choose which poles can be added to the solution. Line 1 of Algorithm 2 chooses the poles whose keys have a value greater than or equal to 0.5, and activates these poles in an order defined by their key values. Therefore, for each activated pole, we install Wi-Fi and LTE equipment in the given order when there is demand in the access radii of the pole. In the case of LTE, we need to check whether the pole is sufficiently far from any macrocell or other poles with previously installed LTE equipment.

Algorithm 2. Equipment activation and installation.

1 Let L be a pole list in non-increasing order of keys κ^A such that $p \in L$ iff $\kappa_p^A \geq 0.5$. Each $p \in L$ is said *active* and each $p' \notin L$ is said *inactive*;
2 **foreach** $p \in L$ *in the given order* **do**
3 **if** $\exists v \in V^d : d(p,v) \leq R_{wifi}$ **then**
4 Install a Wi-Fi equipment on p;
5 **if** $(\kappa_p^P \geq 0.5)$ **and** $(\exists v \in V^d : d(p,v) \leq R_{lte})$ **and** $(\nexists v \in V^m : d(p,v) \leq \delta_{macro})$ **and** $(\nexists v \in V^s : lte(v) = 1 \text{ and } d(p,v) \leq \delta_{lte})$ **then**
6 Install a LTE equipment on p;

The construction of the backhaul forest is an iterative bottom-up process. First, we create the first level connecting poles to the root nodes using Algorithm 3. Sets *CP*, *TK*, and *NL* are initially empty. We initialize pole levels to indicate that level assignment has not yet been made. Likewise, we initialize no connectivity by fiber to all poles (lines 1 and 2). We add to *CP*, the set of current poles, all active neighbors of each root node (lines 3 and 5) sorted in non-increasing order of their corresponding keys in the chromosome (line 6). Using this permutation, we try to connect each pole to a FAP or a macrocell obeying the minimum level and activation parameters (note that the function `extractparameters()` returns a triple according to the description of Section 5.1). If a connection by fiber is allowed, we choose the closest FAP or macrocell under the maximum distance constraint (line 12) and connect the pole to the chosen root node, setting it as "parent" of this pole, and placing the pole in the set of children of the root node. As it is a fibered connection, we consider that the pole is in level zero. If it is not possible to connect by fiber, we try to create a wireless link. To do this, we create a list of neighbor root nodes of the pole which will be visited circularly from the point i determined by the corresponding key in the chromosome. This is done until we obtain a connection (lines 20–33). Note that, in this case, the pole will be in level 1 and its parent will have its incoming degree incremented by one (lines 27 and 28). In the last case, when it is impossible to make a connection we remove the pole from the set *CP* of current poles and place it in the set *NL* of poles for consideration in the next level.

Algorithm 3. Building of backhaul forest (level 0).

1 $CP \leftarrow \varnothing$; $TK \leftarrow \varnothing$; $NL \leftarrow \varnothing$;
2 **foreach** $p \in V^s$ **do** $level(p) \leftarrow -1$; $fibered(p) \leftarrow 0$;
3 **foreach** $r \in V^r$ **do**
4 Let $N_r^- = \{p \in V^s : (p,r) \in E \text{ and } p \text{ is active}\}$;
5 $CP \leftarrow CP \cup N_r^-$;
6 Let Π to be a permutation of CP induced by the non-increasing order of correspondent keys in κ^O;
7 **foreach** $p \in \Pi$ *in given order* **do**
8 $minimum\_level \leftarrow \lfloor (H+1)\kappa_p^L \rfloor$;
9 $(\hat{\ell}, \hat{f}, \hat{d}) \leftarrow$ `extractParameters`(κ_p^P);
10 **if** $\hat{d} =$ `'D'` **and** $minimum\_level \leq 1$ **then**
11 **if** $\hat{f} =$ `'F'` **then**
12 Choose $r = \text{argmin}_{r' \in V^r}(dist(p,r) - R_{fiber}) \leq 0$;
13 **if** r *exists* **then**
14 $parent(p) \leftarrow r$;
15 $children(r) \leftarrow children(r) \cup \{p\}$;
16 $level(p) \leftarrow 0$;
17 $fibered(p) \leftarrow 1$;
18 $TK \leftarrow TK \cup \{p\}$;
19 **if** $level(p) = -1$ **then**
20 Let $N_p^+ = \{m \in V^m : (p, bta(m)) \in A^{smw}\}$ and consider that N_p^+ is in the order induced by V^r;
21 $i \leftarrow \lfloor |N_p^+| \kappa_p^N \rfloor$;
22 $begin \leftarrow i$; $repeated \leftarrow$ false;
23 **while** $level(p) = -1$ **and not** $repeated$ **do**
24 **if** $\deg^+(N_p^+[i]) < \delta_{bh}^+$ **then**
25 $parent(p) \leftarrow N_p^+[i]$;
26 $children(N_p^+[i]) \leftarrow children(N_p^+[i]) \cup \{p\}$;
27 $level(p) \leftarrow 1$;
28 $\deg^+(N_p^+[i]) \leftarrow \deg^+(N_p^+[i]) + 1$;
29 $TK \leftarrow TK \cup \{p\}$;
30 **else**
31 i++;
32 **if** $i = |N_p^+|$ **then** $i \leftarrow 1$;
33 $repeated \leftarrow (i = begin)$;
34 **if** $level(p) = -1$ **then**
35 $CP \leftarrow CP \setminus \{p\}$;
36 $NL \leftarrow NL \cup \{p\}$;

After the first level of the backhaul forest is created, set *CP* consists in the poles that are forest leaves. From them, Algorithm 4 tries to augment the reach of the network. First, we add to list *NL* the neighbor poles of *CP* that are active, in the line of sight with some pole in *CP*, and are not yet part of the forest. We only consider neighbors from poles whose incoming degree is not saturated (line 3). These operations are described in lines 2–8. After constructing these lists, we consider the poles in *NL* as current poles and take a permutation according to the order induced by the chromosome keys. The remainder of the algorithm (lines 12–28) is similar to Algorithm 3 with respect to wireless connections. Note that at the end of this procedure, set *NL* may have poles that are not connected to any tree. These poles will be eliminated from the solution in a pruning phase.

Algorithm 4. Building of backhaul forest (level ≥ 1).

1 **while** $CP \neq \varnothing$ **do**
2 **foreach** $p \in CP$ **do**
3 **if** $\deg^+(p) < \delta^+_{bh}$ **then**
4 Let $N_p^- = \{p' \in V^s : (p', p) \in A^{ss},\ p' \text{ is active and } p' \notin TK\}$;
5 $NL \leftarrow NL \cup N_p^-$;
6 **foreach** $p \in NL$ **do**
7 $N_p^+ \leftarrow \{p' \in CP : (p, p') \in A^{ss}\}$;
8 $\forall p \in V^s$, sort N_p^+ in a non-increasing order of $dist(p, q)$ such that $q \in N_p^+$;
9 $CP \leftarrow NL$;
10 $NL \leftarrow \varnothing$;
11 Let Π to be a permutation of CP induced by the non-increasing order of correspondent keys in κ^O;
12 **foreach** $p \in \Pi$ **do**
13 $minimum\_level \leftarrow \lfloor (H+1)\kappa_p^L \rfloor$;
14 $i \leftarrow \lfloor |N_p^+| \kappa_p^N \rfloor$;
15 $begin \leftarrow i$; $repeated \leftarrow$ false;
16 **while** $level(p) = -1$ **and not** $repeated$ **do**
17 **if** $\deg^+(N_p^+[i]) < \delta^+_{bh}$ **then**
18 $parent(p) \leftarrow N_p^+[i]$;
19 $level(p) \leftarrow level(N_p^+[i]) + 1$;
20 $\deg^+(N_p^+[i]) \leftarrow \deg^+(N_p^+[i]) + 1$;
21 $TK \leftarrow TK \cup \{p\}$;
22 **else**
23 i++;
24 **if** $i = |N_p^+|$ **then** $i \leftarrow 1$;
25 $repeated \leftarrow (i = begin)$;
26 **if** $level(p) = -1$ **then**
27 $CP \leftarrow CP \setminus \{p\}$;
28 $NL \leftarrow NL \cup \{p\}$;

After the complete construction of the backhaul forest, it is possible that there exist active poles not used in the forest, or yet poles present in the forest but not serving any demand. Algorithm 5 removes these poles. We have two pruning phases. The first occurs after the construction of the backhaul forest and the second after the maximum flow computation. In the first phase, we generate a "virtual" flow in each arc only present for the convenience of the algorithm. Thus, a recursive pruning procedure is applied in each root node (`pruneSubtree()`). This procedure traverses each tree using the depth-first strategy until it reaches a leaf node. This way it verifies if demand is served by this leaf node using either Wi-Fi or LTE (lines 5–8). In case demand is present, the pole is kept. If there is no demand but the pole has chidren nodes, then we can deduce that the pole is being used only as a retransmitter and we keep the pole in the forest. Otherwise, the pole is marked for later removal from set *TK*. Note that when the recursion returns, line 3 removes all marked children.

Algorithm 5. Equipment and poles pruning.

1 **if** *first phase* **then**
2 **foreach** $e \in E$ **do**
3 $f_e \leftarrow 1$;
4 **foreach** $r \in V^r$ **do**
5 `pruneSubtree` (r);
6 Mark as inactive all $p \notin TK$ that is marked as active;

Procedure pruneSubtree(r).

1 **foreach** $p \in children(r)$ **do**
2 `pruneSubtree` (p);
3 Remove from $children(r)$, all p marked to remotion;
4 **if** $r \in V^s$ **then**
5 **if** $\deg^+(wta(r)) = 0$ **or** $f_{(wta(r),r)} = 0$ **then**
6 $wifi(r) \leftarrow 0$;
7 **if** $\deg^+(lta(r)) = 0$ **or** $f_{(lta(r),r)} = 0$ **then**
8 $lte(r) \leftarrow 0$;
9 **if** $|children(r)| = 0$ **and** $wifi(r) = lte(r) = 0$ **then**
10 Mark r to remotion;
11 $TK \leftarrow TK \setminus \{r\}$;

After the forest construction and first pruning phases, it is necessary to create a graph induced by this forest to compute the maximum flow from the demand nodes to the root nodes. As pointed out in Section 3.3, the maximum flow problem to be solved is neither classical nor straightforward. To solve this problem, we proposed two solutions which are described in Section 6.

Computed the maximum flow, a second pruning phase is applied to the forest and all non-used pieces of equipment are removed as described earlier (Algorithm 5). Lastly, the revenue and the costs are computed. As aforementioned, these calculations may use different approaches depending on the objective of the study. In this paper, the revenue is derived from the maximum flow directly as shown in Term (3a) of the objective function of MIP (3). The cost is computed using the remaining Terms ((3b)–(3f)). The total cost depends on the deployed equipment, deployed fiber, deployment and maintenance costs, and leased traffic. Computed the revenue and costs, the profit is returned as the solution value and fitness of the current decoded chromosome.

6. Maximum backhaul flow problem

6.1. Bounds

As pointed out in Section 3.3, wireless backhaul equipment has very specific constraints with respect to the reception and retransmission of backhaul traffic. These constraints are mainly related to the physical proprieties of the wave spectrum used. This way, the total capacity of reception and retransmission is limited to a certain constant U_{bh}. At the same time, this equipment also collects local traffic sent by other equipment, such as Wi-Fi and LTE receptors.

For a given retransmitter v, we can assume three distinct components. The first component is the *incoming backhaul traffic*, denoted by F_b^i, such that $F_b^i = \sum_{u \in V^s:(u,v) \in A^{ss}} f_{uv}$, i.e., the sum of backhaul traffic sent to v from neighbors. The second component is the *access traffic*, denoted by F_a, which is the sum of the Wi-Fi and LTE traffic in v, i.e., $F_a = f_{wta(v),v} + f_{lta(v),v}$.[1] The third component is the *outgoing backhaul traffic*, denoted by F_b^o, such that $F_b^o = f_{v,parent(v)} = F_b^i + F_a$. The relationship among these components is given by Inequality

[1] Notation detail: $f_{u,v} = f_{uv}$

(1), that restricts the backhaul flow capacity, and by Eq. (2) that ensures flow conservation. The difficulty is that classical maximum flow algorithms do not deal with these restrictions at the same time and, as far as we know, there is no reduction from the maximum backhaul flow problem to any classical flow problem.

One way to bypass this problem is to consider the access traffic as incoming backhaul traffic. Thus, note that

$$F_b^i + F_a + F_b^o \leq U_{bh}. \tag{4}$$

However $F_b^i + F_a = F_b^o$, which leads us to

$$F_b^o \leq \frac{U_{bh}}{2} \qquad \forall v \in V^s. \tag{5}$$

In this case, as both incoming flows are of the same kind, it suffices to bound either the incoming or outgoing flow to half of the original capacity. The major drawback of this approach is the large flow loss that may result. Suppose, for example, a backhaul capacity of $U_{bh} = 100$, backhaul incoming flow of $F_b^i = 30$, and an access flow of $F_a = 40$. Using the previous technique, we will have the nominal capacity of $U'_{bh} = 50$ in the outgoing arc, which limits the maximum flow to the same value. In this case, 20 units of traffic, either demand or backhaul traffic, cannot be backhauled. But note that using the original constraints, all traffic can be routed since the outgoing traffic would $F_b^o = F_a + F_b^i = 70$ that respects the capacity constraint ($F_b^i + F_b^o = 30 + 70 = 100 = U_{bh}$). In fact, we can provide a bound on this loss using simple algebra. Note that the most constraining factor is the incoming backhaul traffic. As it tends to zero, it enables the increase of the capacity of the outgoing traffic, thus allowing more access traffic be routed (see Lemma 1 below). Consider Inequality (1) in terms of access traffic:

$$\begin{aligned} F_b^i + F_b^o &\leq U_{bh} \\ F_b^i + F_b^i + F_a &\leq U_{bh} \\ F_a &\leq U_{bh} - 2F_b^i. \end{aligned} \tag{6}$$

Now consider Inequality (4) in terms of access traffic:

$$\begin{aligned} F_b^i + F_a + F_b^o &\leq U_{bh} \\ 2F_b^i + 2F_a &\leq U_{bh} \\ F_a &\leq \frac{U_{bh} - 2F_b^i}{2}. \end{aligned} \tag{7}$$

Taking the limit of the proportion between Inequalities (6) and (7) when the incoming backhaul traffic tends to zero, we have:

$$\lim_{F_b^i \to 0} \frac{\frac{U_{bh} - 2F_b^i}{2}}{U_{bh} - 2F_b^i} = \frac{1}{2}. \tag{8}$$

Therefore, the proposed simplification may cause a loss of up to 50% in access traffic (and, consequently, total traffic) that the network can transport. Although the theoretical bound is not very good, this approach leads to reasonable results in practice, as shown in Section 8.2. The following lemmata also give us bounds on the flows in the forest.

Lemma 1. *Consider a vertex $v \in V^s$ with backhaul capacity U_{bh}, F_b^o be the outgoing backhaul traffic from v, and F_a be the value of the access traffic incoming in v. Then, F_b^o is maximum only if F_a is maximum.*

Proof 1. The proof is simple by inspection of the maximality. First, note that we want to maximize $F_b^o = F_a + F_b^i$. But, by constraint capacity (1), we have that $F_b^i + F_b^o \leq U_{bh}$ which means that $F_a + 2F_b^i \leq U_{bh}$. Let $0 \leq \hat{F}_a < F_a$ and $0 \leq F_b^i < \hat{F}_b^i$ and suppose that $\hat{F}_a$ and $\hat{F}_b^i$ yield the maximum flow $\hat{F}_b^o$. Suppose that $\hat{F}_b^i$ is maximum which means that $\hat{F}_b^i = U_{bh}/2$ enforcing $\hat{F}_a = 0$. Therefore $\hat{F}_b^o = U_{bh}/2$. But choosing $F_b^i = \hat{F}_b^i - \varepsilon$, we have that $F_a + 2(U_{bh}/2 - \varepsilon) \leq U_{bh}$ which is $F_a \leq 2\varepsilon$. Therefore $\max F_b^o = 2\varepsilon + U_{bh}/2 - \varepsilon = \varepsilon + U_{bh}/2 > \hat{F}_b^o$ contradicting the maximality of $\hat{F}_b^o$.

Lemma 2. *Let $v \in V^s$ such that v is in level 1 or greater in the backhaul forest. Let $T(v)$ be a subtree of Steiner vertices rooted at v. For all vertices $x, y \in T(v)$ such that arc $(x, y) \in A$, $f_{xy} \leq U_{bh}/2$.*

Proof 2. Let $N^b(v)$ be the neighbor vertices of v that send to it backhaul traffic. Also consider F_a and F_b^o as defined before.

Let $x \in T(v)$ such that $(x, v) \in A$. Therefore:

$$\begin{aligned} \sum_{u \in N^b(v)} f_{uv} + F_b^o &\leq U_{bh} \\ f_{xv} + \sum_{u \neq x \in N^b(v)} f_{uv} + F_b^o &\leq U_{bh} \\ f_{xv} &\leq U_{bh} - F_b^o - \sum_{u \neq x \in N^b(v)} f_{uv} \end{aligned}$$

which leads us to

$$\begin{aligned} f_{xv} &\leq U_{bh} - F_b^o - \sum_{u \neq x \in N^b(v)} f_{uv} \\ &= U_{bh} - \left(\sum_{u \in N^b(v)} f_{uv} + F_a \right) - \sum_{u \neq x \in N^b(v)} f_{uv} \\ &= U_{bh} - f_{xv} - 2 \sum_{u \neq x \in N^b(v)} f_{uv} - F_a \\ &= \frac{U_{bh} - 2\sum_{u \neq x \in N^b(v)} f_{uv} - F_a}{2} \\ &\leq U_{bh}/2. \end{aligned}$$

As $T(v)$ is a tree, all descendent arcs of v have their capacities bounded by $U_{bh}/2$ since v is the unique output vertex in $T(v)$.

Lemma 1 shows that it is worthwhile to route the maximum access traffic available at a pole. Although this lemma is valid for all poles, it may be enforced in nodes at level 1 of the forest, since nodes at level 0 have a fiber connection allowing us to route all access traffic and backhaul traffic subject to the processing constraint. For poles at level 2 or above, we may drop Lemma 1 due to Lemma 2. Note that by Lemma 2, the capacity Constraint (1) does not play a role at poles at levels 2 or above since all traffic in those poles will respect this constraint.

6.2. Solution approach

The maximum backhaul flow problem can be solved to optimality using a linear programming formulation derived from Constraints (1) and (2). The major problem with this approach is that, computationally, it is too slow to be used within the decoder. In Section 8.2, experimental results illustrate this problem. Another approach is to map this flow problem into a classical maximum flow problem [26]. One way to implement backhaul capacity constraint (1) in a classical maximum flow problem, is to set capacities on arcs instead of node equipment. To guarantee feasibility, one sets the capacities of all arcs connecting pairs of poles and arcs connecting pairs of poles/BTAs to half of the backhaul capacity, i.e, $U_{bh}/2$. Restricting capacity this way enables the utilization of classical flow algorithms at the expense, however, of potentially producing suboptimal flows.

Consider a forest generated with Algorithms 2–5. In particular, consider the set TK of poles determined to be in the backhaul network. The maximum flow is computed over the graph induced by TK. For this, we take all vertices in TK and create subsets WTA' and LTA' restricted to poles in TK. This means that $WTA' \subseteq WTA$ and $LTA' \subseteq LTA$ since not all poles are in the forest and, for some poles, LTE equipment are forbidden. We also create the set BTA' with vertices that aggregate wireless backhaul traffic in the macrocells. Note that a BTA exists in a macrocell only if it has children connected to it by wireless links. If all children are connected by fiber, then neither backhaul equipment nor a BTA is needed. Vertices in V^d and V^r also are considered when they are part of the backhaul forest. We add the vertex s to be the source node and vertex t to be the sink. We consider all arcs induced by the chosen vertices and create arcs from s to all demands and from all root nodes to t. In the following, define $cap : E \to \mathbb{R}^+$ be the capacity of an arc:

- For arc a incident to $v \in V^d$, let $cap(a) = d_v$;
- For arc a, outgoing from vertex:
  - $v \in WTA$, let $cap(a) = U_{wifi}$;
  - $v \in LTA$, let $cap(a) = U_{lte}$;
  - $v \in MTA$, let $cap(a) = U_{mc}$;
  - $v \in BTA$, let $cap(a) = U_{bh}$;
  - $v \in V^r$, let $cap(a) = \infty$;
- For each arc $a \in A^{ss} \cup A^{smw}$, let $cap(a) = U_{bh}/2$;
- For each arc $a \in A^{sv} \cup A^{smf}$, let $cap(a) = U_{bh}$.

Note that, although the fibered links in set $A^{sv} \cup A^{smf}$ are considered to have unlimited capacity, we set their capacities to the capacity of retransmitter, modeling the incoming wireless backhaul traffic. In such case, we may lose access traffic if the pole with the fibered link has Wi-Fi and/or LTE traffic. To overcome this, we do the following. Let v be a pole with a fibered link to some root node w. We remove the arcs $(wta(v), v)$ and $(lta(v), v)$ and add the arcs $(wta(v), w)$ and $(lta(v), w)$ with the same respective capacities. Such change allows the maximum access traffic to by-pass pole v and only limits the incoming backhaul traffic. Since we remove capacity Constraint (1), we may use any classical maximum flow algorithm to solve the maximum backhaul flow problem.

As noted above, our approach may generate a suboptimal flow. To improve this, we propose a *pumping algorithm* to augment the generated flow. This algorithm is inspired on the push-relabel algorithm of [26] using Lemmata 1 and 2. The general idea is to push residual flow from the root vertices to the demand vertices observing Lemmata 1 and 2 and the capacity constraints. For each vertex v, let $excess(v)$ be the excess flow in v that must be pushed away. Algorithm 6 considers each root vertex and pumps flow through its subtrees. Lines 3–7 treat the fibered connections. For each child pole, the maximum flow increment is computed and passed to it as excess traffic. Then a procedure applied only to poles in level zero is called, and upon return, the flow is accumulated. In lines 10–19, the wireless connections are considered. In this case, the maximum flow of U_{bh} must be shared with all wirelessly connected children. This is done by computing the maximum flow from the remaining capacity and excess. Since wireless children are considered be in level two or greater, a special procedure is called to treat this case. Again, the totals are accumulated.

Algorithm 6. Pumping root.

```
1  Let r be a FAP or macrocell;
2  f_rt ← 0;
3  foreach p ∈ children(m) such that fibered(p) = 1 do
4      excess(p) ← C_bh − f_pr;
5      f_pr ← C_bh;
6      pumpPoleLevelZero(p);
7      f_rt ← f_rt + f_pr;
8  if r is not a macrocell then
9      return;
10 excess(r) ← C_bh − f_bta(r),r;
11 f_bta(r),r ← 0;
12 foreach p ∈ children(r) such that fibered(p) = 0 do
13     maxflow ← min(excess(r), C_bh − f_p,bta(r));
14     excess(r) ← excess(r) − maxflow;
15     excess(p) ← excess(p) + maxflow;
16     f_p,bta(r) ← f_p,bta(r) + maxflow;
17     pumpPoleLevel1neorMore(p);
18     f_bta(r),r ← f_bta(r),r + f_p,bta(r);
19     f_rt ← f_rt + f_p,bta(r);
```

Algorithm 7. pumpPoleLevelZero(p).

```
1  f_p,parent(p) ← f_wta(p),p + f_lta(p),p;
2  foreach c ∈ children(p) do
3      maxflow ← min(excess(p), C_bh − f_cp);
4      excess(p) ← excess(p) − maxflow;
5      excess(c) ← excess(c) + maxflow;
6      f_cp ← f_cp + maxflow;
7      pumpPoleLevel1neorMore(c);
8      f_p,parent(p) ← f_p,parent(p) + f_cp;
9  excess(parent(p)) ← excess(parent(p)) + excess(p);
10 excess(p) ← 0;
```

Algorithm 7 deals with poles at level zero. Since the above described by-pass guarantees that access traffic is maximum, one can limit their attention to only the incoming backhaul traffic. The algorithm just accumulated the access traffic (line 1) and pumped the maximum allowed flow to the children poles. In the end, if the pole has excess flow, it is pumped back to the parent vertex (line 9).

Algorithm 8 deals with poles at level 1 or greater. The basic idea is the same of previous algorithms except that one must pay attention to the backhaul capacity and Lemmata 1 and 2. Considering Lemma 1, lines 2–16 aim to first maximize the access traffic. This block is considered twice: once for Wi-Fi and once for LTE. Because of this, we rename some terms to reduce the algorithm (lines 2 and 15). After the maximization of the access traffic, lines 22–35 try to push the remaining flow to the children nodes using a recursive call. As in other pumping algorithms, in the last two lines the excess is pumped back to the parent vertex. The key of this algorithm are lines 4–7 and 18–21. If the pole is in level 1, the access flow is given by Eq. (7) using simple substitution (the same occurs for the backhaul traffic in line 18). If the pole is in level two or greater, Lemma 2 comes into scene limiting the traffic to at most $C_{bh}/2$. In this case, we can consider that all traffic flows are of the same type and calculate the maximum local flow from the residue flow of all types. This ensures that the flow through that pole will respect the capacity constraint.

Note that the proposed pumping heuristic has no relabel phase as in the push-relabel algorithm. The pumping algorithm ends after no more pushing is possible in the recursive calls and, therefore, its run-time complexity is $O(|V|)$. Furthermore, the resulting flow is sensitive to the order that the poles are visited. Although the optimum flow is not guaranteed to be found, the pumping heuristic can improve the flow considerably (see Section 8.2 for more details).

Algorithm 8. pumpPoleLevelOneorMore(p).

1 $bf \leftarrow \sum_{c \in children(p)} f_{cp}$;
2 Let ω be $wta(p)$, ϕ be $f_{wta(p),p}$, Φ be $f_{lta(p),p}$, and Γ be C_{wifi};
3 $maxflow \leftarrow \min(excess(p), \Gamma - f_{\omega p})$;
4 **if** $level(p) = 1$ **then**
5 $\quad maxflow \leftarrow \min(maxflow, C_{bh} - 2bf - \Phi)$;
6 **else**
7 $\quad maxflow \leftarrow \min(maxflow, C_{bh}/2 - (bf + \phi + \Phi))$;
8 **foreach** $b \in V^d$ *such that* $(b, \omega) \in E$ **do**
9 $\quad maxinc \leftarrow \min(maxflow, f_{sb} - f_{b\omega})$;
10 $\quad f_{b\omega} \leftarrow f_{b\omega} + maxinc$;
11 $\quad \phi \leftarrow \phi + maxinc$;
12 $\quad maxflow \leftarrow maxflow - maxinc$;
13 $\quad excess(p) \leftarrow excess(p) - maxinc$;
14 **if** $\omega = wta(p)$ **then**
15 $\quad$ Let ω be $lta(p)$, ϕ be $f_{lta(p),p}$, Φ be $f_{wta(p),p}$, and Γ be C_{lte};
16 $\quad$ Go to line 3;
17 $f_{p,parent(p)} \leftarrow f_{wta(p),p} + f_{lta(p),p}$;
18 **if** $level(p) = 1$ **then**
19 $\quad residue \leftarrow (C_{bh} - f_{p,parent(p)})/2$;
20 **else**
21 $\quad residue \leftarrow (C_{bh}/2) - f_{p,parent(p)}$;
22 $pushable \leftarrow \min(residue - bf, excess(p))$;
23 **foreach** $c \in children(p)$ **do**
24 $\quad maxflow \leftarrow \min(pushable, C_{bh} - f_{cp})$;
25 $\quad$ **if** $maxflow \leq 0$ **then**
26 $\quad\quad f_{p,parent(p)} \leftarrow f_{p,parent(p)} + f_{cp}$;
27 $\quad\quad$ Go to line 23;
28 $\quad t \leftarrow excess(p)$;
29 $\quad excess(p) \leftarrow excess(p) - maxflow$;
30 $\quad excess(c) \leftarrow excess(c) + maxflow$;
31 $\quad f_{cp} \leftarrow f_{cp} + maxflow$;
32 $\quad$ `pumpPoleLevelOneorMore`(c);
33 $\quad f_{p,parent(p)} \leftarrow f_{p,parent(p)} + f_{cp}$;
34 $\quad bf \leftarrow bf - excess(p) + t$;
35 $\quad pushable \leftarrow \min(residue - bf, excess(p))$;
36 $excess(parent(p)) \leftarrow excess(parent(p)) + excess(p)$;
37 $excess(p) \leftarrow 0$;

7. Experimental setup

7.1. Instances and scenario descriptions

In this section, we describe the setup of the computational experiments performed to analyze the algorithms presented in this paper. The experiments were conducted using 30 instances derived from real-world scenarios.[2] These instances are taken from neighborhoods of a large city in the United States. Each instance consists of a set of macrocells, VRADs, utility poles, and demand blocks. For each location, longitude and latitude coordinates are given. For each macrocell and utility pole, a list of street segments is given. We assume that if two locations share a segment, they are in the line of sight of each other. For each block, a traffic demand is given. For each macrocell and VRAD, there is an indication of whether traffic through them is leased or not. We classify the instances as small, medium, and large according the number of poles. Each class has ten instances. Table 1 shows a summary and Table A.1 (in Appendix A) brings a complete description. The areas of the regions were computed for illustrative purposes only. The calculation of each area was based on the convex hull considering all locations in the region and their geodesic characteristics. Instance `re01` is the smallest in terms of number of poles with 454 poles while instance `re30` is the largest with 8740 poles. In terms of area, the smallest instance is `re19` with 4.82 km^2 and the largest is `re30` with 411.71 km^2.

While all locations are real, the demand values are based on estimates of the actual demand and are scaled in an arbitrary range. The access radii, minimum distances, capacities, and backhaul constraints are real-life constraints and are displayed in Table 2. We also consider a scenario where the backhaul trees have restrictions neither in depth nor in breadth, and therefore the maximum number of hops H and the maximum number of incoming backhaul links δ^+_{bh} are unlimited (in practice, they are the number of poles in the instance). We call this scenario *unrestricted* in opposition to the *restricted* real-life scenario. The revenue factor and the costs are based on actual values but are also scaled in an arbitrary range. It is worthwhile to mention that the revenue factor, costs, and demands were scaled similarly so as to mimic real world values.[3] The size of the fibered hop, ℓ_{uv}, is defined by the geodesic distance, in meters, between locations u and v. We consider a 3-year planning horizon, i.e. $Y = 3$.

Table 1
Summary of instance characteristics. The presented values are averages of the numbers of respective locations and are rounded to the next integer (except the demand and area).

Type	Poles	VRADs	Macros	Blocks	Demand (Mbps)	Area (km^2)
Small	718	63	10	3907	8210.70	35.92
Medium	2281	86	14	17,306	36,348.00	72.14
Large	6396	243	22	25,566	53,601.00	132.87

Table 2
Description of equipment capacities, design constraints, revenue factor, and costs. The values reflect usual assumptions made in practice.[a]

Short description	Symbol	Value	Unit
Wi-Fi radius	R_{wifi}	100	m
LTE radius	R_{lte}	400	
Macrocell radius	R_{mc}	3000	
Retransmitter radius	R_{bh}	1000	
Wi-Fi capacity	U_{wifi}	100	Mbps
LTE capacity	U_{lte}	20	
HSPA capacity	U_{mc}	25	
Retransmitter capacity	U_{bh}	100	
LTE to LTE min. distance	δ_{lte}	300	m
LTE to macro min. distance	δ_{macro}	500	
Max. fiber size	R_{fiber}	300	
Max. # of incoming links	δ^+_{bh}	5 / ∞	Units
Max. # of wireless hops	H	2 / ∞	
Revenue factor	ϱ	150.00	Monetary units
Equipment deployment	C_p	90.00	
Wi-Fi	C_{wifi}	12.00	
LTE	C_{lte}	70.00	
Backhaul equip. (fan-in = 1)	C_{fan1}	40.00	
Backhaul equip. (fan-in $\geq$ 2)	C_{fan2}	70.00	
Maintenance (annual)	C_{man}	120.00	
Macrocell (annual)	C_{mc}	1050.00	
Meter of deployed fiber	C_{fiber}	12.00	
Leased traffic (in $/Mbps)	C_{ld}	10.00	

[a] Disclaimer: these are generalized costs and revenues to assess performance of the algorithms and do not imply an actual business case for any carrier.

7.2. Instance preprocessing

The instance preprocessing aims to reduce the size of the instance and build the base graph that represents the potential wireless and fibered links, and the arcs representing the links

[2] These instances are available at http://www.loco.ic.unicamp.br/instances/wbndp.html.

[3] Disclaimer: these are generalized costs and revenues to assess performance of the algorithms and do not imply an actual business case for any carrier.

between the demands points and access equipment. This graph is built using the definitions of Section 4. Note that, due to the minimum distance constraint between a macrocell and an LTE equipment, a pole u has an LTA associated with it if and only if for each macrocell v, $dist(u, v) \geq \delta_{macro}$.

The first step is to prune poles that will never be used in feasible solutions. To do this, we calculate the shortest path from each root node to each utility pole annotating the size of the shortest path from any root node to that utility pole. We consider that wireless arcs have weight one and fibered arcs have weight zero. Such paths represent the minimum level that a pole can have in the forest. Let q be the length of the shortest path from pole u to its closest root node. All poles for which q is greater than the maximum number of hops allowed (i.e., $q > H$) are eliminated since they cannot be used in any valid solution. Note that the corresponding WTA and LTA vertices are also deleted.

The distances are also used to create the x variables of MIP (3). We only define the variables x^p_{uv}, for $p = q, \ldots, H$, and $(u, v) \in A$. Note that as u can be in level q or greater, the outgoing backhaul link (u, v) can only be in level q or greater. This preprocessing significantly reduces the size of the MIP and, consequently, the computational time needed to solve it.

Another important observation is that several demand blocks may be served by the same group of poles and macrocells. This is particularly true for residential buildings and commercial areas. In such cases, we group these blocks making a super block whose demand is the sum of the demands of the original blocks. However, at the conclusion of the optimization, it will be necessary to "ungroup" these super blocks and redistribute the access flow to the original blocks.

7.3. Post-optimization flow recomputation

At the conclusion of the BRKGA iterations, we obtain an optimal or near-optimal solution using the strategy described in Section 6.2. One may note that if we compute the exact flow using the forest structure of the best solution found so far, we may be able to improve its objective function value. Note that this is true since this best solution was obtained using the heuristic maximum flow algorithm, a lower bound of the actual maximum flow. In view of this fact, at the end of the BRKGA iterations, we recompute the maximum flow for the best solution found using the linear programming model with Constraints (1) and (2). As we compute the exact flow just once, the execution time of entire algorithm is not compromised.

The new linear programming based flow may traverse new paths. In some cases, some devices will no longer serve demands and can be disregarded. We can apply Algorithm 5 again to prune such unused equipment. Note that this post-processing can potentially further reduce the costs and improve the overall solution.

7.4. Computational environment and parameters

The experiments were conducted on identical machines with four-core Intel Xeon 2.4 GHz CPUs (two threads per core) and 50 GB of RAM running GNU/Linux. Running times reported are UNIX real wall-clock times in seconds, excluding the effort to read the instance. The algorithms are implemented in C++ and we use the GNU `g++` compiler version 4.8. Random numbers were generated by an implementation of the Mersenne Twister [31]. We used the Lemon library [32] to implement the graph structures and compute the maximum flow using its push-relabel algorithm implementation.

To tune the BRKGA parameters, we use the *iterated racing* procedure [33]. This method consists in sampling configurations from a particular distribution, evaluating them using either the Friedman test or the t-test, and refining the sampling distribution with repeated applications of F-Race. We use the `irace` package [34], implemented in R, for parameter tuning. For each heuristic, we use a budget of 1000 experiments in the tuning procedure, where each experiment was limited to 1 h. To tune the BRKGA parameters, we used the following ranges: population size $\in$[300, 2000], elite percentage $\in$[0.15, 0.30]; percentage of mutants introduced at each generation $\in$[0.10, 0.20]; probability of inheriting each allele from elite parent $\in$[0.5, 0.8]; number of independent populations $\in\{1, 2\}$; exchange interval $\in$[50, 200]; number of elite individuals in an exchange $\in$[1, 2]; and reset population $\in$[300, 700].

The following values were recommended by `irace`. The population size was set to $p = 500$, the elite size to $p_e = \lceil 0.30p \rceil$, and the number of mutants to $p_m = \lfloor 0.15p \rfloor$. The probability of inheriting each allele from the elite parent was $\rho_e = 0.70$. We used the island model [35] with two independent and concurrent populations where every 100 generations each population exports its best solution to the other populations. After 300 generations without improvement, all populations are reset to vectors of random keys. We use four simultaneous cores for decoding.

To solve the MIP, we used IBM ILOG CPLEX Optimizer version 12.6.0.0. We set CPLEX to use a maximum of 40 GB of memory, using at most 40 GB of disk memory when necessary. We allowed CPLEX use four threads in parallel. All other parameters were kept at their default values. We use a short run of the BRKGA to generate an incumbent solution for CPLEX. For this, we use BRKGA with the same parameters as above but limit its run to 100 iterations or 10% of maximum time, whichever comes first.

We also tested a multi-start algorithm that uses the decoder of Section 5.2. In each iteration, the multi-start algorithm generates a random vector and uses the decoding function to obtain a solution. It also keeps the best solution over all iterations. In the end, the post-processing is applied to the best solution.

Thirty independent runs were performed for the BRKGA and the multi-start algorithm. Since CPLEX is an implementation of an exact algorithm,[4] a single run for each instance was performed. We carried out two types of experiments, one limiting the running time of each algorithm to 1 h and another to 5 h. The limit of 1 h enables network designers to work with several models in a manageable time while the limit of 5 h enables a more thorough search. For both the BRKGA and the multi-start algorithm, we use an additional stopping criterion: 1000 generations (or iterations) without improvement of the best solution.

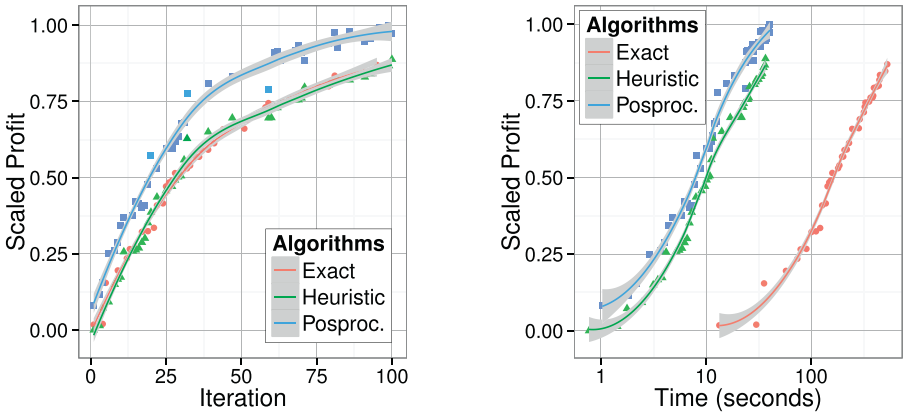
8. Experimental results and discussion

8.1. Instance preprocessing

As discussed in Section 7.2, it is important to preprocess the instance in order to reduce its size before optimization. With respect to the number of poles, the preprocessing phase in the restricted scenario achieved an average reduction of 13.00± 16.39 % (min = 0.59, 1st Qu. = 2.31, median = 6.06, 3rd Qu. = 17.97, max = 67.61). For the unrestricted scenarios, the reduction was 4.04± 7.68 % (min = 0.15, 1st Qu. = 0.58, median = 1.30, 3rd Qu. = 3.82, max = 39.40), since there is no restriction imposed on depth and breadth of the forest. As expected, this reduction has more impact on the MIP than it does on the BRKGA.

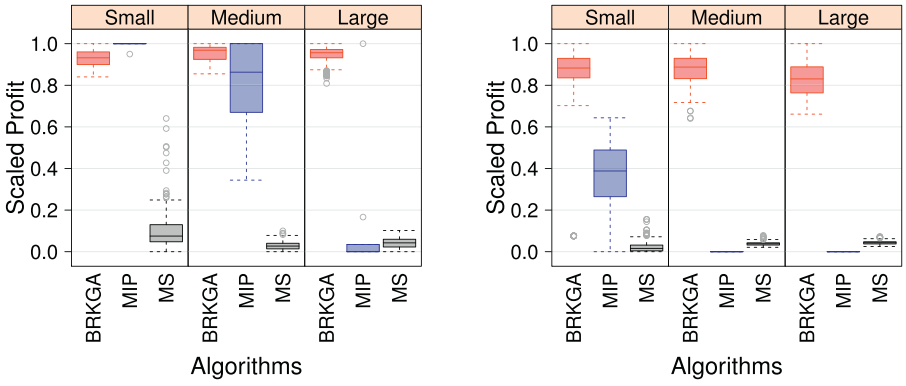
The reduction in the number of demand blocks was huge in both scenarios: 95.96± 1.88 % of the original number of blocks for the restricted scenarios (min = 88.26, 1st Qu. = 95.49, median = 96.25,

[4] According to its documentation, the IBM ILOG CPLEX Optimizer, version 12.6.0.0 is fully deterministic when used with its default parameters (as done in our experiments).

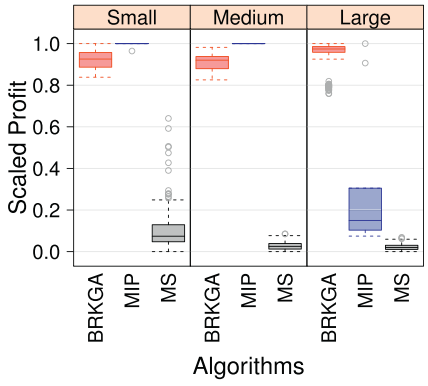


(a) Evolution over iterations. (b) Evolution over time.

Fig. 2. Evolution of the profit using different flow algorithms. The shadow around the curves represents the standard deviation. (For interpretation of the references to colo in the text, the reader is referred to the web version of this article.)



(a) Restricted scenarios (1h). (b) Unrestricted scenarios (1h).



(c) Restricted scenarios (5h).

Fig. 3. Dispersion of profit for each algorithm. Each section corresponds to an instance class.

3rd Qu. = 97.04, max = 98.05), and 95.65 ± 2.69 % of the original number of blocks for the unrestricted scenarios (min = 85.67, 1st Qu. = 95.45, median = 96.17, 3rd Qu. = 96.98, max = 98.05). This fact is mainly due to the concentration of demand blocks in residential buildings and commercial areas. On average, each super block corresponds to 30.97 ± 11.43 original blocks.

The instance graphs that result from preprocessing are sparse. In the restricted instances, the number of vertices varies from 717 to 25,690 and the number of arcs from 6917 to 314,229. The graph density, given by $2|A|/(|V|(|V|-1))$, has an average of 0.0056 ± 0.0063. For unrestricted instances, the number of vertices varies from 1290 to 27,322 and the number of arcs from 10,224 to 315,330. The graph density has an average of 0.0043 ± 0.0035. Detailed results can be found in Tables B.1 and B.2 in the supplementary material.

8.2. Computing flow during the optimization

In Section 6, one can see that the maximum flow problem embedded in the WBNDP is not trivial and, as far as we know, a fast combinatorial algorithm to solve it does not exist in the literature. As commented in Section 6.2, one can solve this flow problem

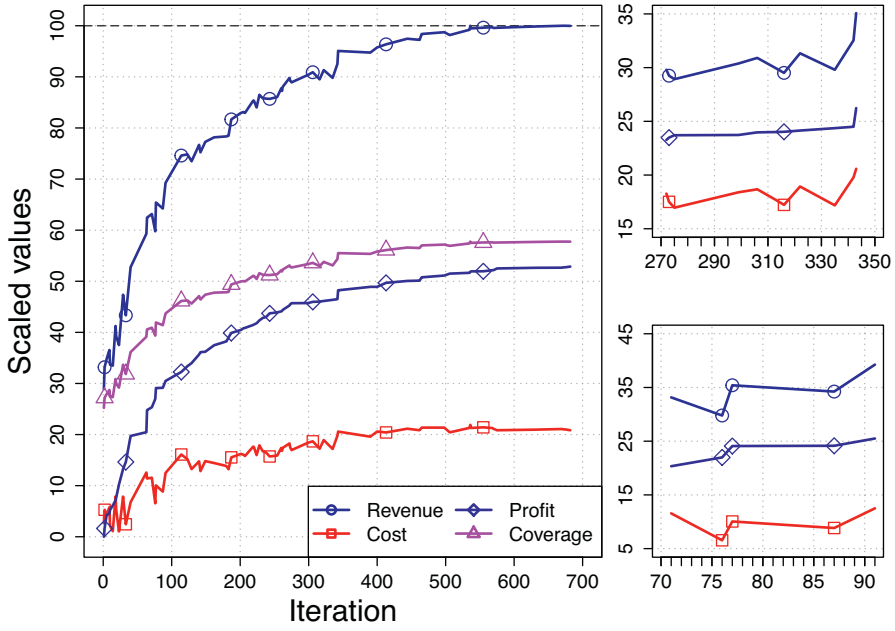


Fig. 4. Evolution of revenue, cost, profit, and coverage over the first iterations of the algorithm. All values are scaled in the range [0,100]. Disclaimer: these are generalized costs and revenues to assess performance of the algorithms and do not imply an actual business case for any carrier. (For interpretation of the references to color in the text, the reader is referred to the web version of this article.)

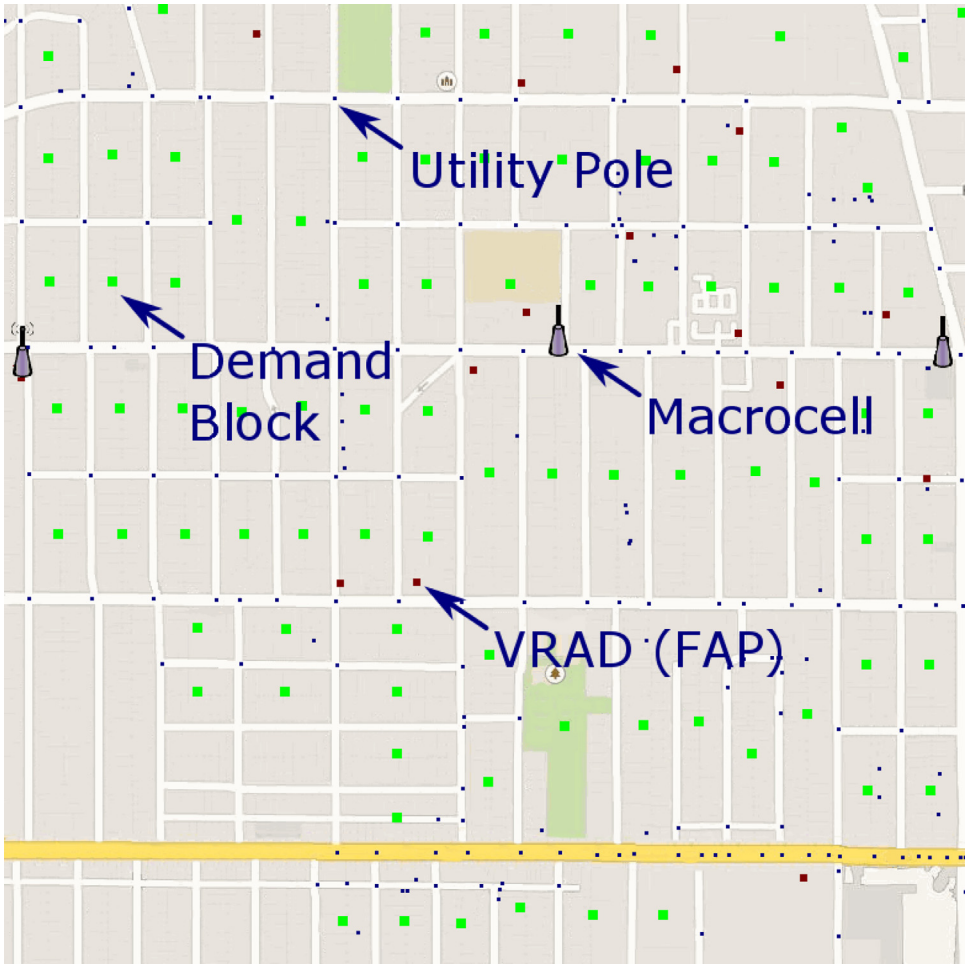


Fig. 5. Example of region. (For interpretation of the references to color in the text, the reader is referred to the web version of this article.)

with a linear programming formulation using Constraints (1) and (2), but this approach can be too slow to be used in the decoding procedure. Therefore, in the same section, we presented a fast heuristic, coupled with *pumping*, to compute the maximum flow. Given a forest, this section studies the effects of choosing one or the other strategy to compute the maximum flow.

The first experiment consists of 1200 independent runs of the decoder. For each run a random chromosome was generated and decoding was done twice: once using the heuristic flow algorithm and once using the exact flow algorithm. Then, the proportional difference in the flow values and solution times were computed. The heuristic flow algorithm was able to compute an average of 95.19± 2.68 % of the maximum flow computed by the exact flow algorithm (min = 79.66, 1st Qu. = 93.58, 3rd Qu. = 96.47, max = 100.00). We consider this performance very good even in light of Lemma 1. However, the computing times were extremely different. Since the decoding process is very fast, we chose to compare the number of CPU ticks used by each algorithm. The exact flow algorithm used an average of 2027± 2080.42 % more CPU ticks than did the heuristic flow algorithm (min = 200, 1st Qu. = 800, 3rd Qu. = 2450, max = 15, 100). This means that the exact flow algorithm is three orders of magnitude slower than the heuristic flow algorithm.

Fig. 2 shows the evolution of the profit as a function of number of iterations and CPU wall-clock time. For this, we used instance `re30` and let BRKGA evolve for 100 iterations. In Fig. 2a the *X* axis represents the number of iterations and, in Fig. 2b, the wall-clock time in s in log scale. The *Y*-axis represents the scaled profit in both figures (see the definition of scaled profit in the beginning of Section 8.3). The shaded area represents the standard deviation. The red line and dots represent the algorithm using exact flow computation, while the blue squares and line represent the algorithm using the heuristic flow. The green triangles and line represent the utilization of heuristic flow and post-processing. In the experiment with the heuristic flow computation and post-processing, the exact flow was recomputed (followed by pruning) in each iteration. One may note in Fig. 2a that both heuristic and exact flow are able to generate almost the same profit in a given iteration. The post-processing approach performs better than the others since it has a second phase pruning as described in Section 7.3. In terms of running time, one can note in Fig. 2b that the exact flow algorithm obtains about the same profit as the heuristic but using much more time.

8.3. Comparing the profit generated by the algorithms

To compare the algorithms with respect to profit, it is necessary to scale the results since instances can have very different profit values. For each instance $\mathcal{I}$, let $\chi_{\mathcal{I}}$ be the set of values of the solutions found for $\mathcal{I}$, and $D_{\mathcal{I}} = \max(\chi_{\mathcal{I}}) - \min(\chi_{\mathcal{I}})$. The scaling is done by the simple transformation

$$\chi'_{\mathcal{I}} = \begin{cases} (x - \min(\chi_{\mathcal{I}}))/D_{\mathcal{I}} & \forall x \in \chi_{\mathcal{I}} \text{ and } D_{\mathcal{I}} > 0, \\ 1 & \text{otherwise,} \end{cases}$$

where $\chi'_{\mathcal{I}}$ is the set of scaled values. Note that all values are scaled to the range [0, 1].

Using this scaling process, Fig. 3 shows the distribution of profits for each algorithm. The box plots show the location of the first quartile, the profit median, and the third quartile. The whiskers extend to the most extreme revenue no more than 1.5 times the length of the box. The dots are the outliers. In the bar labels, `MS` stands for the multi-start algorithm, `MIP` stands for the exact algorithm using CPLEX and Formulation (3), and `BRKGA` is the biased random-key genetic algorithm. In the restricted scenario, `MIP` was able to overcome `BRKGA` for most small instances on the 1-h experiments and on most small and medium instances on the 5-h experiments. `BRKGA` presented a small variation in its results, although close to those of the `MIP` in both cases. `MS` presented a large variation and its results were the worst. This is due to, mainly, the decoder has no local search procedures (unless the pumping algorithm). This fact just reinforces the importance of the learning mechanism of BRKGA. For medium size instances, `BRKGA` presented more solid results while `MIP` showed more variation for 1-h experiments. In the 5-h runs the results were similar to those for the small instances. For large instances, `BRKGA` was able to produce very good results when compared to `MS` and `MIP` in both cases. In fact, `MIP` does not produce any results besides the incumbent value generated by the `BRKGA` short run in large instances considering the limit of 1 h. For 5 h, only in two instances did `MIP` improve the incumbent. In the unrestricted scenario, `BRKGA` presented the best results. In fact, for all medium and large instances, `MIP` was not able to produce any solution (because of memory issues in building the model).

Table 3
Difference in median location for profit distributions for the restricted scenario using a confidence interval of 99%. The omitted *p*-values are less than 0.008. The diagonal elements in shaded-bold represent the medians, the upper-right elements represent the median difference, and the bottom-left elements represent the *p*-values.

Class		1h experiment			5h experiment		
		BRKGA	MIP	MS	BRKGA	MIP	MS
Small	BRKGA	**0.92**	-0.07	0.84	**0.91**	-0.08	0.83
	MIP		**1.00**	-0.92		**1.00**	0.92
	MS			**0.08**			**0.07**
Medium	BRKGA	**0.96**	0.07	0.92	**0.92**	-0.08	0.89
	MIP	$p > 0.33$	**0.88**	-0.85		**1.00**	0.97
	MS			**0.03**			**0.03**
Large	BRKGA	**0.94**	0.92	0.89	**0.97**	0.80	0.95
	MIP		**0.00**	0.03		**0.15**	0.13
	MS			**0.05**			**0.02**

To confirm the results presented in Fig. 3, we tested the normality of these distributions using the Shapiro–Wilk test and applied the Mann–Whitney–Wilcoxon *U* test, considered more effective than the *t*-test for distributions sufficiently far from normal and for sufficiently large sample sizes [36,37]. For all tests, we assume a confidence interval of 99%. For small, medium, and large instances the Shapiro–Wilk tests revealed that no profit distribution fits a normal distribution since the *p*-values for all tests are less than 0.01. Therefore, we applied the *U* test which assumes as null hypothesis that the location statistics are equal in both distributions. As several statistical tests were performed, we used a *p*-value correction procedure based on false discovery rate (FDR) to minimize the number of false positives (Type I error) as indicated by [38].

Table 3 shows *U* test results for each pair of algorithms and different instance sizes of the restricted scenario, at a 99% confidence level. The structure of this table is as follows: Each row and column is indexed by one algorithm. Each element in the diagonal (bold) is the median of the scaled profit of the corresponding algorithm. The upper-right diagonal elements are the differences in location statistics for each pair of algorithms. A positive difference indicates that the "row algorithm" has its location statistics higher (better) than the "column algorithm," and the negative difference is the opposite. The bottom-left diagonal elements are the *p*-values of each test. We omitted all $p < 0.01$ values, that indicate that the difference is statistically significant for those pairs. We also omitted confidence intervals since for all tests the values lie in these intervals and are very narrow. One can notice that almost all comparisons are statistically significant, confirming the box plot results. The exception is `BRKGA` and `MIP` for medium instances using 1-h runs for which the test was inconclusive since $p > 0.01$. `MS` is significantly worse than the other algorithms except for the 1-h `MIP` experiments on

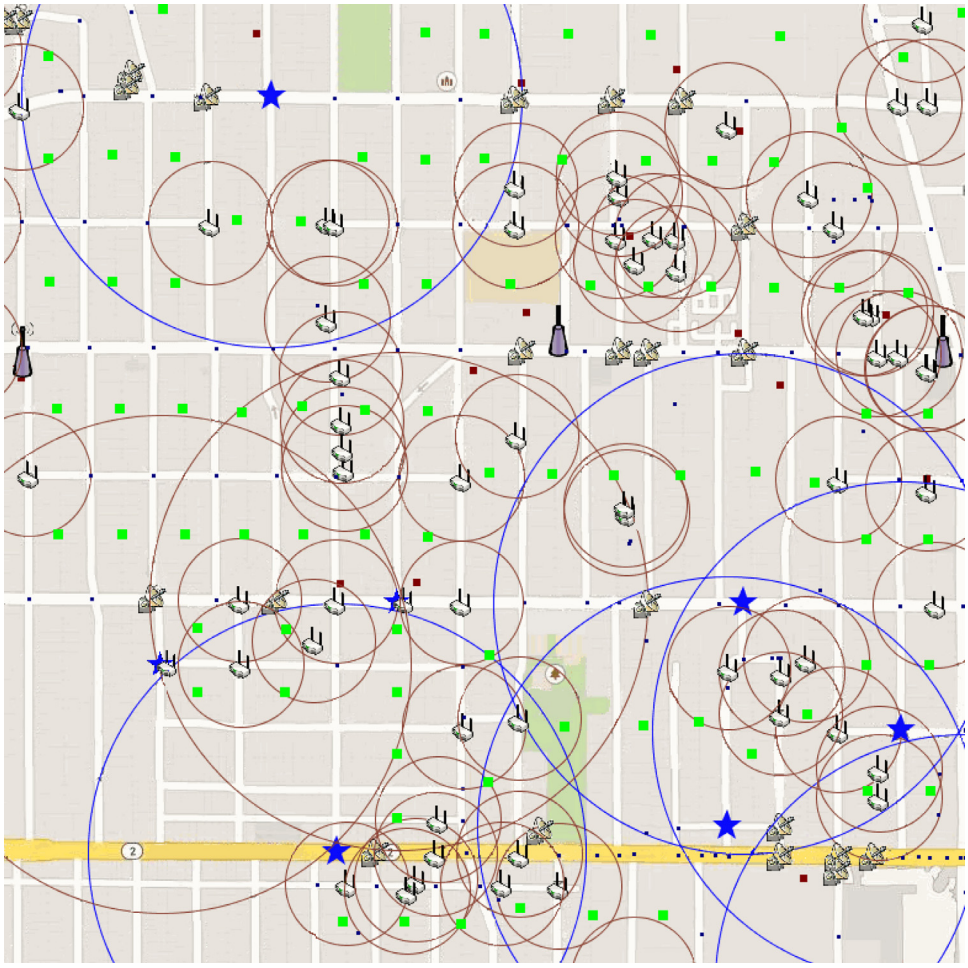


Fig. 6. Example of coverage. (For interpretation of the references to color in the text, the reader is referred to the web version of this article.)

large instances. Summarizing, BRKGA was better than MIP on large instances and the opposite was true for the small instances. For medium-size instances, both apparently performed in a similar way although we cannot affirm this since $p > 0.01$. For the unrestricted scenario, the statistical test only makes sense for the small instances since for the medium and large instances, MIP did not produce any feasible solution. In this case, BRKGA presented the median of 0.89, MS presented 0.01, and MIP presented 0.40. The tests indicate that all differences are significant, confirming the box plot.

Table 4 reports the performance of the algorithms. The first column indicates the instance class and the second column is the name of the algorithm. The two large blocks consider experiments limited to 1 h and 5 h, respectively. Each block has four columns. Column "% Best" represents the percentage of the number of instances for which the algorithm found a best solution; column "% Run" shows a percentage of the number of runs on which the algorithm found a best solution. The two columns under label "Prod. diff." show, respectively, the average of the proportional difference between the value of the best solution found and the achieved value (%), and its corresponding standard deviation (σ). First, note that MIP presents the same values for % Best and % Run since only one experiment is done per instance. Considering the restricted scenario, MIP

Table 4
Algorithm performance considering the best results found in restricted scenario.

Class	Alg.	1 h experiment				5 h experiment			
		Best solutions		Prop. diff.		Best solutions		Prop. diff.	
		% Best	% Run	%	σ	% Best	% Run	%	σ
Small	BRKGA	20.00	10.42	4.55	2.17	20.00	10.41	5.06	2.34
	MIP	90.00	90.00	3.30	–	90.00	90.00	2.36	–
	MS	0.00	0.00	94.14	124.78	0.00	0.00	94.37	124.71
Med.	BRKGA	60.00	2.50	3.27	2.60	0.00	0.00	6.06	2.91
	MIP	40.00	40.00	21.33	15.17	100.00	100.00	–	–
	MS	0.00	0.00	64.33	5.03	0.00	0.00	66.16	4.71
Large	BRKGA	90.00	3.75	3.63	2.94	90.00	3.75	3.29	4.68
	MIP	10.00	10.00	57.49	7.38	10.00	10.00	50.54	18.44
	MS	0.00	0.00	58.11	8.77	0.00	0.00	65.37	8.01
All	BRKGA	56.67	5.56	3.80	2.65	36.67	4.72	4.81	3.66
	MIP	46.67	46.67	40.54	22.76	66.67	66.67	45.72	23.12
	MS	0.00	0.00	72.20	73.88	0.00	0.00	75.57	74.37

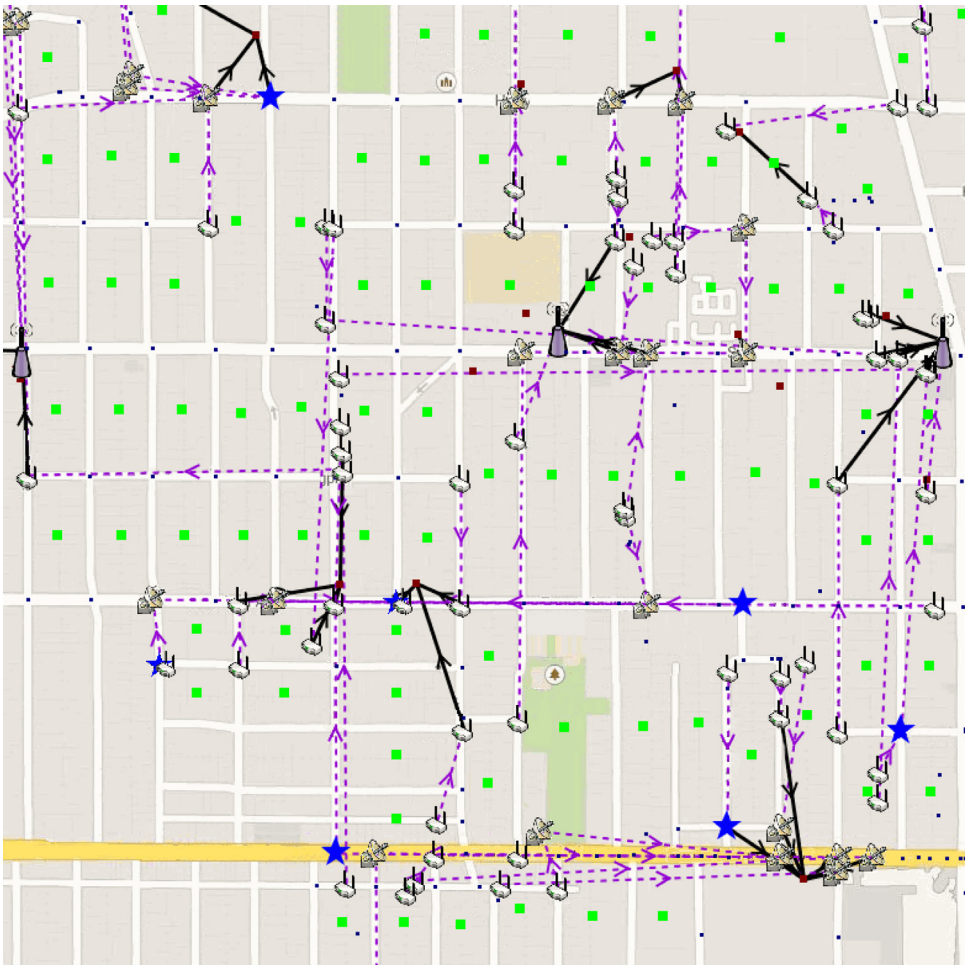


Fig. 7. Example of backhaul network. (For interpretation of the references to color in the text, the reader is referred to the web version of this article.)

found 9 of 10 best solutions, although only one was proved be optimal in the 5-h run. BRKGA found 20% of best solutions in about 10% of the runs. However, the BRKGA results are stable and its solutions are within about 5% of the best. MS did not find any good solution on any instance. For the medium and large instances, the roles of BRKGA and MIP were exchanged on the 1-h runs. However, MIP obtained all best solutions on the medium-size instances when it was given 5 h to run. In general, BRKGA did not find best solutions in several runs but presented very good alternative solutions. MIP found many best solutions but its results varied considerably. In the unrestricted scenario, BRKGA dominates MS and MIP since the latter could not solve any instance. BRKGA found the best solutions in 4.16% of the runs. The average proportional difference between the BRKGA results and the best solutions found was 13.94 ± 9.67. For detailed results, please refer to Tables B.3 and B.4 in the supplementary material.

8.4. Analyzing a solution

One can note that the WBNDP is a problem rich in structure from the point of view of network engineering. Similar to the number of input parameters, a typical output has more than 50 parameters, such as number of pieces of equipment of each type, flows, coverage, costs, revenue, and other metrics besides the network structure itself. In this section, we briefly analyze some of these output parameters considering solutions for the restricted scenario since it is based on real constraints.

Fig. 4 shows the evolution of revenue (dark blue line with dots) cost (red line with squares), profit (light blue with rhombus), and coverage (magenta dashed line with triangles) for instance re17 for a given run. Revenue, cost, and profit are scaled to the range [0 100]. Coverage is already represented in this range. One can note as the coverage increases, so does the revenue. This is expected since revenue is a function of traffic volume. Note that while the profit is monotone increasing, the revenue and cost display some bumps In the close-up figure showed on the top right, one can see that both revenue and cost vary up and down while profit always non-decreasing. Such cases show a phase transition. When both revenue and cost curves are sloped downwards, the algorithm has found a solution that is less expensive using less equipment. In bottom right close-up, note that the profit is constant between iterations 77 and 87, but both revenue and cost have slightly negative slopes.

Each tree in the backhaul forest has an average depth of 2.49 ± 0.50. Each node has an average of 1.381 ± 0.76 incoming neighbors which shows that the fan-in limit is rarely reached Indeed, the average of the maximum fan-in is 3.73 ± 0.64, and for some instances, no pole has more than two incoming neighbors. On average, 48.88 ± 14.32 % of the used poles have only Wi-Fi 19.51 ± 15.42 % have only LTE, and 9.22 ± 5.98 % have both technologies. In 22.38 ± 6.93 % of the used poles, only retransmitters are installed. In only 0.34 ± 0.51 % of used poles, can one find a small cell without a retransmitter. Such poles have no neighbors and are linked directly to a FAP or macrocell using fiber.

With respect to traffic, Wi-Fi was responsible for an average of 77.28 ± 16.50 % of the total covered access traffic while LTE

served only 17.14± 12.59% of the traffic. The macrocells served only 4.02± 6.70% of the traffic. Wi-Fi has shown itself as an important resource to serve the demand due to it large capacity and low cost when compared to LTE and HSPA equipment. The backhaul networks were able to serve, on average, 55.78± 16.82% of total demand.

In Fig. 5, we show an example which is a small portion of a given region where a backhaul network is to be built. This region has three macrocells represented by small antenna figures. Each light green square represents a set of demand points in a specific block. The average number of demand points is 30 per block, but can reach hundreds of residential and commercial buildings. The dark red squares represent the VRAD/FAPs. The very small dark blue squares represent the utility poles. Fig. 6 shows the equipment deployment and the coverage radii. The symbols with small equipment and two antennas represent Wi-Fi equipment while the blue stars represent LTE equipment. The poles with only retransmitters (backhaul equipment) are represented by a parabolic antenna. The brown and blue circles are, respectively, the Wi-Fi and LTE access radii. One can note that some blocks are not served. In particular, some blocks in the upper right-hand side of the figure are not covered. Fig. 7 shows the backhaul network. The dashed purple arrows are wireless links, the black solid arrows are fibered links, and the arrows indicate the direction of hte root nodes.

9. Final considerations

In this paper, we proposed a new problem called the wireless backhaul network design problem (WBNDP) which resembles variants of the Steiner tree and the facility location problems. The objective is to build a forest to collect and route wireless traffic. Differing from other problems in the literature, WBNDP uses routed traffic to compute the profit. This traffic is constrained to the network infrastructure with several real-world constraints. We proposed a biased random-key genetic algorithm (BRKGA) to solve the WBNDP. Its decoder relies on building the forest in a bottom-up fashion. We also proposed a mixed integer linear programming model to solve the WBNDP.

BRKGA presented solid results with little variation. It was able to overcome the IBM ILOG CPLEX 12.6 using MIP (3), in several medium and large instances of 1-h runs. For longer experiments, BRKGA excelled on large instances. Such results enable BRKGA to be used as an important tool in the planning phase of a wireless backhaul network where, usually fast iterations are required. However, BRKGA also showed itself valuable for longer optimizations, mainly on large instances. The stable results produced by the BRKGA give network engineers a better understanding of the characteristics of an optimal network and makes it easier to modify some assumptions if needed.

Acknowledgments

The authors wish to thank Byoung-Jo Kim for his help with the radio modeling portions of this paper. Carlos E. Andrade is supported by São Paulo Research Foundation (FAPESP, Brazil) grants 2010/05233-5 and 2012/08222-0. Flávio K. Miyazawa is supported by National Council for Scientific and Technological Development (CNPq, Brazil) grants 306860/2010-4 and 477692/2012-5. The work of the second author was done while he was employed at AT&T Labs Research in Middletown, NJ.

Appendix A. Supplementary data

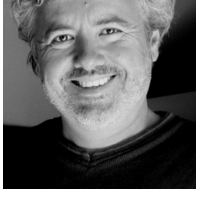
Supplementary data associated with this article can be found, in the online version, at http://dx.doi.org/10.1016/j.asoc.2015.04.016

References

[1] J. O'Toole, Mobile apps overtake PC Internet usage in U.S., CNN Money (2014, February 28) http://money.cnn.com/2014/02/28/technology/mobile/mobile-apps-internet
[2] Cisco, VNI Mobile Forecast Highlights 2014–2019, Cisco Website, 2015 http://www.cisco.com/assets/sol/sp/vni/forecast_highlights_mobile
[3] J.F. Gonçalves, M.G.C. Resende, Biased random-key genetic algorithms for combinatorial optimization, J. Heuristics 17 (2011) 487–525, http://dx.doi.org/10.1007/s10732-010-9143-1, ISSN 1381-1231.
[4] R. Mahindra, H. Viswanathan, K. Sundaresan, M.Y. Arslan, S. Rangarajan, A practical traffic management system for integrated LTE-WiFi networks, in: Proceedings of the 20th Annual International Conference on Mobile Computing and Networking, MobiCom'14, ACM, New York, NY, USA, 2014, pp. 189–200, http://dx.doi.org/10.1145/2639108.2639120
[5] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of $\mathcal{NP}$-Completeness, Freeman, San Francisco, 1979.
[6] S. Voß, Steiner tree problems in telecommunications, in: M.G.C. Resende, P.M. Pardalos (Eds.), Handbook of Optimization in Telecommunications, Springer, US, 2006, pp. 459–492, http://dx.doi.org/10.1007/978-0-387-30165-5_18, ISBN 978-0-387-30662-9.
[7] M.X. Goemans, D.P. Williamson, A general approximation technique for constrained forest problems, SIAM J. Comput. 24 (2) (1995) 296–317, http://dx.doi.org/10.1137/S0097539793242618
[8] M.X. Goemans, D.P. Williamson, The primal dual method for approximation algorithms and its application to network design problems, in: D. Hochbaum (Ed.), Approximation Algorithms for NP-Hard Problems, P.W.S. Publishing Co., Boston, MA, USA, 1996, pp. 144–191.
[9] D.S. Johnson, M. Minkoff, S. Philips, The prize collecting tree problem: theory and practice, in: Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms, Baltimore, Maryland, USA, 1999, pp. 769–790.
[10] S.A. Canuto, M.G.C. Resende, C.C. Ribeiro, Local search with perturbations for the prize-collecting Steiner tree problem in graphs, Networks 38 (1) (2001) 50–58, http://dx.doi.org/10.1002/net.1023, ISSN 1097-0037.
[11] A. Lucena, M.G.C. Resende, Strong lower bounds for the prize collecting Steiner problem in graphs, Discret. Appl. Math. 141 (1–3) (2004) 277–294, http://dx.doi.org/10.1016/S0166-218X(03)00380-9, ISSN 0166-218X.
[12] I. Ljubić, R. Weiskircher, U. Pferschy, G.W. Klau, P. Mutzel, M. Fischetti, An algorithmic framework for the exact solution of the prize-collecting Steiner tree problem, Math. Program. 105 (2–3) (2006) 427–449, http://dx.doi.org/10.1007/s10107-005-0660-x, ISSN 0025-5610.
[13] G.W. Klau, I. Ljubić, A. Moser, P. Mutzel, P. Neuner, U. Pferschy, G. Raidl, R. Weiskircher, Combining a memetic algorithm with integer programming to solve the prize-collecting Steiner tree problem, in: K. Deb (Ed.), Genetic and Evolutionary Computation – GECCO 2004, Vol. 3102 of Lecture Notes in Computer Science, Springer, Berlin/Heidelberg, 2004, pp. 1304–1315, http://dx.doi.org/10.1007/978-3-540-24854-5_125, ISBN 978-3-540-22344-3.
[14] A.S. da Cunha, A. Lucena, N. Maculan, M.G.C. Resende, A relax-and-cut algorithm for the prize-collecting Steiner problem in graphs, Discret. Appl. Math. 157 (6) (2009) 1198–1217, http://dx.doi.org/10.1016/j.dam.2008.02.014
[15] L. Gouveia, Using the Miller–Tucker–Zemlin constraints to formulate a minimal spanning tree problem with hop constraints, Comput. Oper. Res. 22 (9) (1995) 959–970, http://dx.doi.org/10.1016/0305-0548(94)00074-I, ISSN 0305-0548.
[16] G. Dahl, L. Gouveia, C. Requejo, On formulations and methods for the hop-constrained minimum spanning tree problem, in: M.G.C. Resende, P.M Pardalos (Eds.), Handbook of Optimization in Telecommunications, Springer, US, 2006, pp. 493–515, http://dx.doi.org/10.1007/978-0-387-30165-5_19, ISBN 978-0-387-30662-9.
[17] L. Gouveia, A. Paias, D. Sharma, Restricted dynamic programming based neighborhoods for the hop-constrained minimum spanning tree problem, J. Heuristics 17 (1) (2011) 23–37, http://dx.doi.org/10.1007/s10732-009-9123-5, ISSN 1381-1231.
[18] L. Gouveia, L. Simonetti, E. Uchoa, Modeling hop-constrained and diameter-constrained minimum spanning tree problems as Steiner tree problems over layered graphs, Math. Program. 128 (1–2) (2011) 123–148, http://dx.doi.org/10.1007/s10107-009-0297-2, ISSN 0025-5610.
[19] A.M. Costa, J.-F. Cordeau, G. Laporte, Fast heuristics for the Steiner tree problem with revenues, budget and hop constraints, Eur. J. Oper. Res. 190 (1) (2008) 68–78, http://dx.doi.org/10.1016/j.ejor.2007.06.012, ISSN 0377-2217.
[20] A.M. Costa, J.-F. Cordeau, G. Laporte, Models and branch-and-cut algorithms for the Steiner tree problem with revenues, budget and hop constraints, Networks 53 (2) (2009) 141–159, http://dx.doi.org/10.1002/net.20274, ISSN 1097-0037.
[21] S.B. Layeb, I. Hajri, M. Haouari, Solving the Steiner tree problem with revenues, budget and hop constraints to optimality, in: 5th International Conference on Modeling, Simulation and Applied Optimization (ICMSAO), Hammamet, Algeria, 2013, pp. 1–4, http://dx.doi.org/10.1109/ICMSAO.2013.6552674
[22] Z.-H. Fu, J.-K. Hao, Breakout local search for the Steiner tree problem with revenue, budget and hop constraints, Eur. J. Oper. Res. 232 (1) (2014) 209–220, http://dx.doi.org/10.1016/j.ejor.2013.06.048, ISSN 0377-2217.
[23] D.R. Karger, M. Minkoff, Building Steiner trees with incomplete global knowledge, in: Proceedings of 41st Annual Symposium on Foundations of Computer Science, FOCS'00, IEEE Computer Society, 2000, pp. 613–623, http://dx.doi.org/10.1109/SFCS.2000.892329, ISBN 0-7695-0850-2.

[24] I. Ljubić, S. Gollowitzer, Layered graph approaches to the hop constrained connected facility location problem, INFORMS J. Comput. 25 (2) (2013) 256–270, http://dx.doi.org/10.1287/ijoc.1120.0500
[25] F. Rayal, LTE Peak Capacity Explained: How to Calculate It? Personal Blog, 2011, June 27 http://frankrayal.com/2011/06/27/lte-peak-capacity
[26] A.V. Goldberg, R.E. Tarjan, A new approach to the maximum-flow problem, J. ACM 35 (4) (1988) 921–940, http://dx.doi.org/10.1145/48014.61051, ISSN 0004-5411.
[27] J.F. Gonçalves, M.G.C. Resende, A parallel multi-population genetic algorithm for a constrained two-dimensional orthogonal packing problem, J. Comb. Optim. 22 (2011) 180–201.
[28] C.E. Andrade, F.K. Miyazawa, M.G.C. Resende, Evolutionary algorithm for the *k*-interconnected multi-depot multi-traveling salesmen problem, in: Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation, GECCO'13, ACM, New York, NY, USA, 2013, pp. 463–470, http://dx.doi.org/10.1145/2463372.2463434, ISBN 978-1-4503-1963-8.
[29] C.E. Andrade, M.G.C. Resende, H.J. Karloff, F.K. Miyazawa, Evolutionary algorithms for overlapping correlation clustering, in: Proceedings of the 16th Conference on Genetic and Evolutionary Computation, GECCO'14, ACM, New York, NY, USA, 2014, pp. 405–412, http://dx.doi.org/10.1145/2576768.2598284, ISBN 978-1-4503-2662-9.
[30] C.E. Andrade, R.F. Toso, M.G.C. Resende, F.K. Miyazawa, Biased random-key genetic algorithms for the winner determination problem in combinatorial auctions, Evol. Comput. 23 (2) (2015), http://dx.doi.org/10.1162/EVCO_a_00138 (in press).
[31] M. Matsumoto, T. Nishimura, Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator, ACM Trans. Model. Comput. Simul. 8 (1998) 3–30, http://dx.doi.org/10.1145/272991.272995, ISSN 1049-3301.
[32] B. Dezsõ, A. Jüttner, P. Kovács, LEMON – an open source C++ graph template library, Electron. Notes Theor. Comput. Sci. 264 (5) (2011) 23–45, http://dx.doi.org/10.1016/j.entcs.2011.06.003, ISSN 1571-0661.
[33] M. Birattari, Z. Yuan, P. Balaprakash, T. Stützle, F-Race and iterated F-Race: an overview, in: Experimental Methods for the Analysis of Optimization Algorithms, Springer, Berlin/Heidelberg, 2010, pp. 311–336.
[34] M. López-Ibáñez, J. Dubois-Lacoste, T. Stützle, M. Birattari, The `irace` Package, Iterated Race for Automatic Algorithm Configuration, Tech. Rep. TR/IRIDIA/2011-004, IRIDIA, Université Libre de Bruxelles, Belgium, 2011 http://iridia.ulb.ac.be/IridiaTrSeries/IridiaTr2011-004.pdf
[35] D. Whitley, S. Rana, R.B. Heckendorn, The island model genetic algorithm: on separability, population size and convergence, J. Comput. Inf. Technol. 7 (1998) 33–47.
[36] W.J. Conover, Practical Nonparametric Statistics, 2nd ed., John Wiley & Sons, 1980.
[37] M.P. Fay, M.A. Proschan, Wilcoxon–Mann–Whitney or *t*-test? On assumptions for hypothesis tests and multiple interpretations of decision rules, Stat. Surveys 4 (2010) 1–39.
[38] Y. Benjamini, Y. Hochberg, Controlling the false discovery rate: a practical and powerful approach to multiple testing, J. R. Stat. Soc. Ser. B: Methodol. 57 (1) (1995) 289–300, ISSN 00359246, http://www.jstor.org/stable/2346101

Carlos E. de Andrade received his Computer Science degree from Federal University of Lavras, and his master degree (2006) and Ph.D. (2015) in computer science from University of Campinas, Brazil. He is interested in combinatorial optimization tools to solve hard problems that arise in both tactical and strategic decision-making. In particular, he is interested in exact methods using integer programming and heuristics. Before joining University of Campinas, he was assistant professor at Federal Institute for Education, Science and Technology Southern of Minas Gerais, Inconfidentes, Brazil, from 2006 to 2010.



Mauricio G.C. Resende Resende is Principal Research Scientist at Amazon.com in Seattle, Washington. Previously, he was Lead Inventive Scientist at the AT&T Labs Research in Middletown New Jersey. Mauricio's research interests include combinatorial optimization, metehauristics, engineering of algorithms, and networks and graphs. He has published over 180 peer-reviewed papers, edited five books, and holds 15 U.S. patents. He received an electrical engineer degree from Pontificial Catholic University of Rio de Janeiro, a master of operations research from the Georgia Institute of Technology, and a Ph.D. in operations research from the University of California at Berkeley. He was awarded the title of Doctor Honoris Causa from Universidad Nacional de San Agustín de Arequipa (UNSA), Arequipa, Peru.

Weiyi (Max) Zhang is currently a Senior Inventive Scientist of the Network Evolution Research Department at AT&T Labs Research, Middletown, NJ. Before joining AT&T Labs Research, he was an Assistant Professor at the Computer Science Department, North Dakota State University, Fargo, North Dakota. His research interests include routing, scheduling, and cross-layer design in computer networks, localization and coverage in wireless sensor networks, survivable design and quality-of-service provisioning. He has published more than 80 refereed papers in his research areas. He received AT&T Labs Research Excellence Award in 2013, Best Paper Award in 2007 from IEEE GLOBECOM'2007, and Best Paper Award in 2014 from IEEE ICC'2014.

Rakesh K. Sinha is a Lead Inventive Scientist in the Network Evolution Research Department of AT&T Labs Research. He has broad research interests in the areas of network architecture, design, and optimization. Prior to joining AT&T, he worked at the routing and signaling group of Ciena Core Director switches and before that at the Networking Research Department of Lucent Bell Laboratories. He received my B.Tech. (Computer Science) from Indian Institute of Technology (I.I.T.), Kanpur, India and his Ph.D. (Computer Science) from University of Washington Seattle.

Kenneth C. Reichmann holds a B.S. in physics and a M.S in electrical engineering. He is a Principal Inventive Scientist in the Network Evolution Research Department at AT&T Labs, concentrating on broadband wireless and optical access technologies. He is a Senior Member of the IEEE

Robert D. Doverspike received his undergraduate degree from the University of Colorado and Masters and Ph.D degrees from Rensselaer Polytechnic Institute (RPI). He is Executive Director of Network Evolution Research in AT&T Labs Research. Dr. Doverspike has made extensive contributions to the field of optimization of multi-layered transmission and switching networks and pioneered the concept of packet transport in metro and long distance networks. He also pioneered work in spearheading the deployment of new architectures for transport and IP networks, network restoration, and integrated network management of IP-over-optical-layer networks and Software Defined Networking. He has over 1500 citations to his books and articles over diverse areas/publications.

Flávio K. Miyazawa received his Computer Science (1990 degree from the Federal University of Mato Grosso do Sul the M.Sc. (1993) and D.Sc. (1997) in Applied Mathematics from the University of Sã Paulo, Brazil. Since 1998, he has been affiliated with the Institute of Computing of the University of Campinas, Brazil. From 2012 to the present he has been a full Professor at the same university. His main research activities are in the field of combinatorial optimization. He has published over 80 peer-reviewed papers in high quality journals and conferences, supervised 8 Ph.D. students and 19 master students.