A MIP-based approach to solve the Prize-Collecting Local Access Network Design Problem

Ivana Ljubić[*] Peter Putz[*] Juan-José Salazar-González[†]

November 23, 2013

Abstract

This paper presents a new combinatorial optimization problem that can be used to model the deployment of broadband telecommunications systems in which optical fiber cables are installed between a *central office* and a number of *end-customers*. In this capacitated network design problem the installation of optical fiber cables with sufficient capacity is required to carry the traffic from the central office to the end-customers at minimum cost. In the situation motivating this research the network does not necessarily need to connect all customers (or at least not with the best available technology). Instead, some nodes are *potential* customers. The aim is to select the customers to be connected to the central server and to choose the cable capacities to establish these connections. The telecom company takes the strategic decision of fixing a percentage of customers that should be served, and aims for minimizing the total cost of the network providing this minimum service. For that reason the underlying problem is called the *Prize-Collecting Local Access Network Design* problem (PC-LAN).

We propose a branch-and-cut approach for solving small instances. For large instances of practical importance, our approach turns into a mixed integer programming (MIP) based heuristic procedure which combines the cutting-plane algorithm with a multi-start heuristic algorithm. The multi-start heuristic algorithm starts with fractional values of the LP-solutions and creates feasible solutions that are later improved using a local improvement strategy.

Computational experiments are conducted on small instances from the literature. In addition, we introduce a new benchmark set of real-world instances with up to 86 000 nodes, 116 000 edges and 1 500 potential customers. Using our MIP-based approach we are able to solve most of the small instances to proven optimality. For more difficult instances, we are not only able to provide high-quality feasible solutions, but also to provide certificate on their quality by calculating lower bounds to the optimal solution values.

[*]Department of Statistics and Operations Research, University of Vienna, Austria, ivana.ljubic@univie.ac.at, p.putz@univie.ac.at

[†]DEIOC, Universidad de La Laguna, Tenerife, Spain, jjsalaza@ull.es

This work was partially supported by the bilateral program "Acciones Integradas" between Austria (E15/2010) and Spain (AT2009-0021). I. Ljubić was partially supported by the APART fellowship of the Austrian academy of sciences and J.J. Salazar-González by the Spanish Government (MTM2009-14039-C06-01).

Keywords: Network Loading, Local Access, Prize-Collecting, Matheuristic Algorithm.

1 Introduction

Providing future-proof broadband Internet connections is currently a major infrastructural issue worldwide. More and more information is shared across the Internet and demand for higher data rates increases with new services. The *Digital Agenda for Europe*[1] of the European Commission stresses the importance of information and communications technologies and states that "Half of European productivity growth over the past 15 years was already driven by information and communications technologies [...] and this trend is likely to accelerate." It issues the goal of achieving "internet speeds of 30 Mbps or above for all European citizens, with half European households subscribing to connections of 100 Mbps or higher" by the year 2020. The German government decided to place strong emphasis on the expansion of broadband communications in one of its latest economic stimulus packages[2]. The rather challenging aim, formulated in 2009, is to provide 75% of all households nationwide with 50 Mbps connections by the end of 2014. Reaching this goal is only possible by rolling-out the fiber-optic access networks on a broad scale.

In telecommunication network planning, *customer nodes* are associated to physical locations representing buildings, business locations or single households. The following strategies (known under a common name *FTTx*) are used for the development of access networks:

- *Fiber-To-The-Curb* (FTTC) (or *Fiber-To-The-Node*, FTTN): Part of the connection from central offices to customers consists of optical fibers, but end-transmission lines are still made of copper. Besides the fiber-optic connections that need to be established, also *multiplexing* devices have to be installed. These devices receive signals from multiple customers via copper connections and aggregate them on a high-speed fiber-optic line.
- *Fiber-To-The-Building* (FTTB): Optical fiber runs all the way to a building. Multiplexing devices (usually installed in the basement) aggregate signals from short-distance copper lines to the subscribers within the building onto a fiber-optic line.
- *Fiber-To-The-Home* (FTTH): Connection between subscribers and central offices runs completely over an optical fiber.

Which strategy is employed in a particular case depends on various prerequisites. For instance, it depends on how densely the planning areas are populated (e.g., urban vs. rural areas).

Many local telecommunication carriers are realizing FTTH or FTTB projects. Deutsche Telekom AG announced plans for the connection of thousands of households in ten German cities with FTTH. Simultaneously, FTTC solutions are realized by extending VDSL connections[3]. The largest Austrian telecommunication provider, Telekom Austria Group, is going to invest one billion Euro in the

[1] *Digital Agenda* (May 2010), `http://europa.eu/rapid/pressReleasesAction.do?reference=IP/10/581`
[2] *Breitbandstrategie der Bundesregierung* (February 2009),
`http://www.zukunft-breitband.de/BBA/Navigation/Service/publikationen,did=290026.html`
[3] *Press release* (February 28, 2011), `http://www.telekom.com/dtag/cms/content/dt/de/996928`

modernization of the fixed net infrastructure[4].

The planning of such access networks is a highly complex task. Manual planning does not allow for finding provably close-to-optimal solutions. In the last years various uncapacitated optimization problems have been proposed in the context of FTTx planning (see, e.g. Arulselvan et al. [1], Leitner and Raidl [9], Gollowitzer and Ljubić [5], Gollowitzer et al. [6]). These optimization problems are mainly concerned with the design of the underlying network topology, ignoring many hardware parameters. On a more detailed level, the following aspects have to be considered in addition: There are cost/capacity relations for various active and passive components, such as transponders, splitters, fibers and cables. There are overhead cost for trenching. Also existing infrastructure has to be taken into account. There are two possibilities to deploy fiber-optic access networks: customers might be connected via *passive optical networks* (PON) or via *Point-to-point.* In the first case, signals for up to 64 customers are transmitted on a single fiber and are split on the optical level somewhere between the central office and the customer. In the second case a unique fiber starting at the central office is dedicated to each customer.

This paper deals with the detailed planning of point-to-point FTTH/FTTB telecommunication networks with a given *coverage rate* α, $(0 < \alpha \leq 1)$. This coverage rate is usually determined by a network carrier and represents the minimal fraction of potential customers that should be offered the service. Figure 1 shows two deployment scenarios for a real world instance with coverage rates of 0.6 and 0.9, respectively. Square nodes denote customers. The circle node denotes the *central office* r. The served customers are depicted with dark squares. Lines denote the denote installed connections in the solution.

In the context of FTTH/FTTB the potential customers are particular physical locations. Three important features are associated to each potential customer. Firstly, the number of subscribers (e.g., apartments and/or offices) in the building. This is denoted as the customers *prize.* Secondly, the number of optical fibers required to connect this potential customer is called its *demand.* Thirdly, there is a setup *cost* of installing a suitable device at the customer location. The available hardware (e.g., splitter devices in case of FTTB) determines the demand and cost for each customer.

For each customer selected for being served, there must be a fiber-optic line running to the central office. The company provides different types of cables. Each type of cable is characterized by two features. One is its capacity and represents the number of optical fibers. The other is its cost. For connecting two sites one may need several cables. Each combination of cables leads to a *module* with a given *capacity* and *cost.* The capacity of a module is simply the sum of the fibers included in the cables. The cost of a module is the sum of the cable costs plus the installation on the roads taking into account the length. The goal is to decide

- which subset of customers to connect so that at least a fraction of α of the overall customers' prizes is covered, and
- which modules should be installed along the edges so that the total demand for selected customers can be routed through the network at minimum cost.

[4] *Press release* (July 3, 2009),
`http://www.telekomaustria.com/presse/news/2009/0703-telecommunication-infrastructure--en1.php`

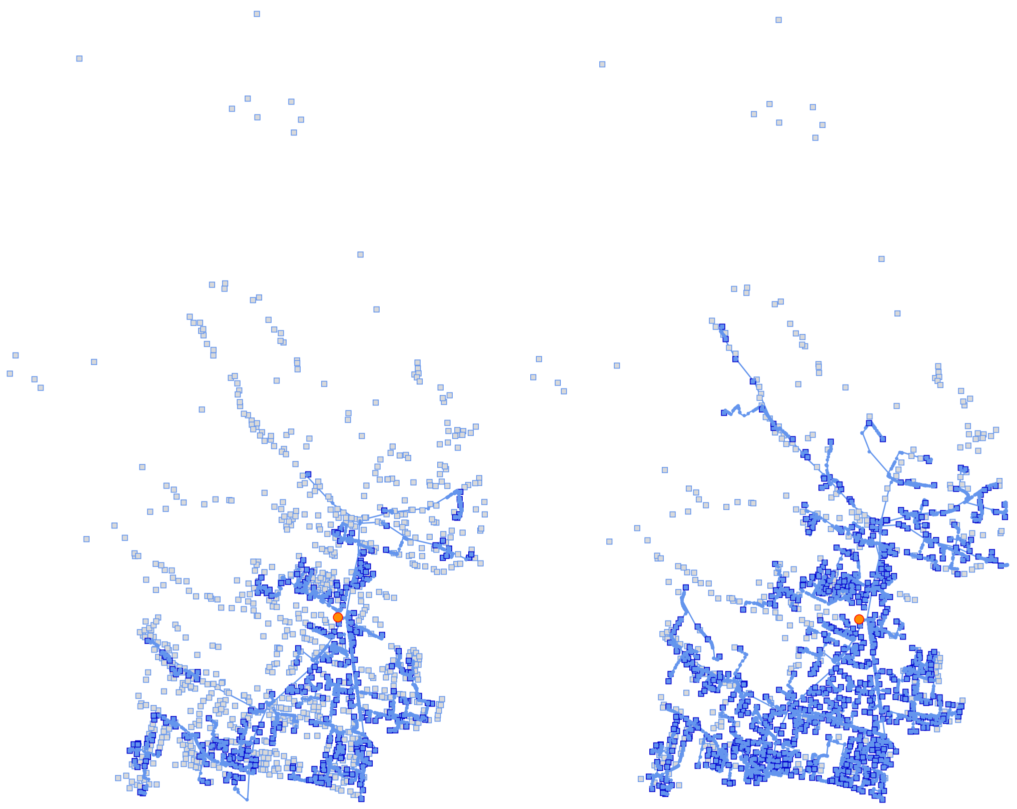
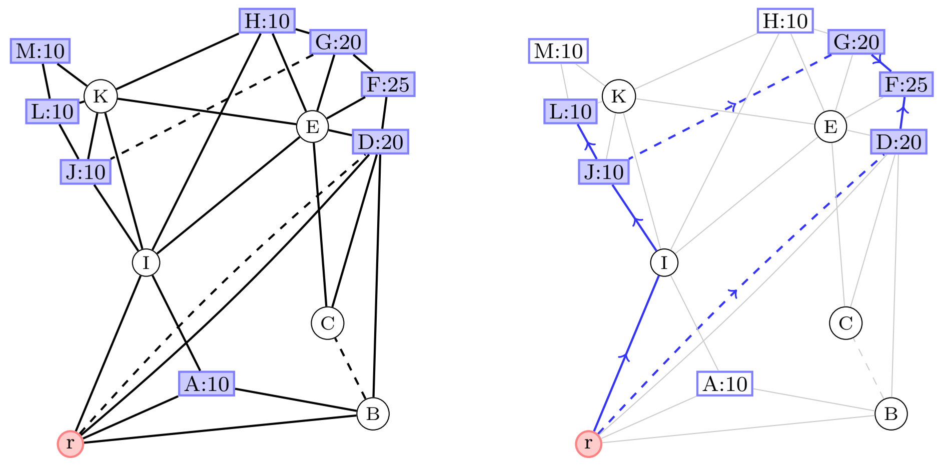


Figure 1: Realistic planning scenario with coverage rates of 0.6 up to 0.9.

The fiber-optic lines that are necessary to connect a certain customer need not to run along the same single path through the network.

Our Contribution: In this work we study exact and heuristic approaches to this problem of practical relevance. We first propose a branch-and-cut approach that is capable of solving small instances. However, due to the complexity of the problem and size of the instances in real applications, it is difficult to establish algorithmic approaches that ensure global cost-minimal solutions. In this work we also propose a new MIP-based approach that is based on an interplay between the cutting plane approach and a multi-start heuristic. The multi-start heuristic starts with fractional values of the LP-solutions and creates feasible solutions that are improved using a local improvement strategy. We introduce a new benchmark set of real-world instances with up to 86 000 nodes, 116 000 edges and 1 500 potential customers. For these instances our computational experiments show that the MIP-based approach outperforms the alternative approach of only using the multi-start heuristic without MIP information.

A preliminary version of this paper was presented in the *International Network Optimization Conference 2011* (Ljubić et al. [11]).



(a) Instance of the PC-LAN design problem. (b) Optimal solution of the PC-LAN instance.

Figure 2: Input graph and optimal solution of a PC-LAN instance.

2 Problem definition

We are given an undirected and connected graph $G = (V, E)$ with a node $r \in V$ representing the central office (or central server or access to the backbone network) and a set of potential customers $K \subseteq V \setminus \{r\}$. To each potential customer $k \in K$, a positive demand d_k, a positive prize p_k and a positive setup cost c_k are assigned. We denote by $p_0 = \alpha \sum_{k \in K} p_k$ the minimum customer prize to collect. Let M_e be the set of modules that can be installed on edge $e \in E$, each one $m \in M_e$ associated with a positive capacity $u_{e,m}$ and a cost $c_{e,m}$. We assume that modules are sorted such that $u_{e,m} < u_{e,m+1}$. The optimization problem of our interest is the selection of the customers to be served, the single-source multiple-sink routing, and installation of *at most one module* on every edge. The connection from the central office to a customer can be seen as a *flow* that is allowed to split apart. Thus we are speaking of a *bifurcated* flow. As a result, an optimal solution of the problem is not necessarily a tree in the graph. If $\alpha = 1$, the problem is known as the *Local Access Network Design Problem* (LAN), or the *Single Source Network Loading Problem* (SSNLP). For $0 < \alpha < 1$, we refer to this problem as the *Prize-Collecting Local Access Network Design Problem* (PC-LAN). Figure 2 depicts an example with an input graph and a solution of the PC-LAN instance with $\alpha = 0.7$. The graph $G = (V, E)$ has 14 nodes. For each edge e the length l_e of e is the Euclidean distance. Solid lines represent modules with $u_{e,m} = 100$ and $c_{e,m} = 120l_e$. Dashed lines represent the module with $u_{e,m} = 40$ and $c_{e,m} = 10l_e$. Rectangle nodes are customers with their demands d_k written at the corresponding labels. Customer prizes and cost are defined as $p_k = d_k, c_k = d_k/2$, respectively. A coverage rate of 70% is requested, hence $p_0 = 0.7 \sum_{k \in K} d_k = 80.5$. The served demand by the optimal solution in the figure is 85.

2.1 Related work

The literature contains some approaches for solving LAN, which is a special case of PC-LAN where $\alpha = 1$. LAN has been studied in Raghavan and Stanojević [15] and Salman et al. [17] where the authors assume that the stepwise link capacities satisfy economies of scales. Both papers consider flow-based MIP formulations and work with relaxations obtained by approximating the noncontinuous stepwise function by its lower convex envelope. In Ljubić et al. [10], exact approaches for LAN with general stepwise functions, based on Benders' decomposition on various MIP models are proposed. More general multiple-source multiple-sink variants of the LAN problem have also been approached. See e.g. recent works in Frangioni and Gendron [4], among others. However all these articles deal with problems where all customers must be served, while in PC-LAN we also need to deal with the problem of selecting a subset of the customers to be served.
More practice-oriented approaches have been studied in Martens et al. [12]. By using a two-step approach with suitable MIP formulations, it is possible to optimize fiber-optic networks in realistic scenarios (see Martens et al. [13]). Also Orlowski et al. [14] conducted various practice-oriented case studies that originated from planning scenarios by a German carrier. Finally we point out that there are many other works in the literature focusing on other designing aspects of fiber-to-the-home networks. For example, in recent works of Gualandi et al. [7] and Kim et al. [8], the authors concentrate on facility location aspects rather than on network design ones.

2.2 A compact MIP formulation

We now show a Single Commodity Flow (SCF) formulation of the PC-LAN problem. This and other formulations can be derived from the models given in Ljubić et al. [10] for the problem where all customers must be in the network. We make use of a binary variable y_k to model whether a potential customer k is served or not. Observe that there always exists an optimal solution of the problem which is a directed acyclic graph connecting the central node with the subset of selected customers. The SCF formulation uses this result and works on a bi-directed graph G. The set of directed arcs is denoted by A. The modules available for the arcs (i, j) and (j, i) are the same as for the corresponding edge $\{i, j\}$, hence $M_{(i,j)} = M_{(j,i)} = M_{\{i,j\}}$. To model the non-decreasing step cost function on every arc, binary variables are used. A binary variable $x_{a,m}$ decides whether the module m shall be installed on the arc a. For each subset $S \subset V$, we will denote the ingoing cut by $\delta^-(S) := \{(i, j) \in A \mid i \in V \setminus S, j \in S\}$.

To make sure that sufficient capacity is available to install cables between the central node and the selected customers, we additionally introduce continuous *flow variables* $f_{ij} \geq 0$ that represent the total amount of flow routed from the center towards the customers.

SCF Model

The MIP model reads now as follows:

$$\min \sum_{a \in A} \sum_{m \in M_a} c_{a,m} x_{a,m} + \sum_{k \in K} c_k y_k \tag{1}$$

subject to

$$\sum_{a\in\delta^+(i)} f_a - \sum_{a\in\delta^-(i)} f_a = \begin{cases} -d_i y_i, & i \in K \\ \sum_{k\in K} d_k y_k, & i = r \\ 0, & \text{otherwise} \end{cases} \qquad \forall i \in V \tag{2}$$

$$0 \le f_a \le \sum_{m\in M_a} u_{a,m} x_{a,m} \qquad \forall a \in A \tag{3}$$

$$\sum_{k\in K} p_k y_k \ge p_0 \tag{4}$$

$$\sum_{m\in M_a} x_{a,m} \le 1 \qquad \forall a \in A \tag{5}$$

$$x_{a,m} \in \{0,1\} \qquad \forall a \in A,\ \forall m \in M_a \tag{6}$$

$$y_k \in \{0,1\} \qquad \forall k \in K. \tag{7}$$

Objective function (1) consists of two terms: the cost for the installation of modules and the cost issued at customers' sites. Constraints (2) are the *flow preservation constraints*: they make sure that the amount of flow sent to the customers is equal to their demands, in case that they are served. *Capacity constraints* (3) ensure that on each link, a module is installed that allows the necessary amount of flow to be routed through it. The *disjunction constraints* (5) ensure that at most one module is installed on every arc. Finally, the *coverage constraint* (4) states that a subset of customers achieving at least the requested target prize p_0 has to be connected. The presented SCF model is compact, but plugging it in a black-box MIP solver may not produce satisfactory results, mainly due to the big-M constraints (3). These constraints require hundreds of thousands of branch-and-bound nodes to be enumerated before proving the optimality of a solution (see, e.g. Ljubić et al. [10] where similar results are obtained for the related LAN problem).

2.3 Branch-and-cut approach

Rather than working with the compact model presented above, we remove the flow variables and use an exponential-size set of constraints to model the problem. These constraints are dynamically separated using a cutting plane approach. Furthermore, additional strengthening inequalities are added to improve the lower bounds and consequently reduce the search space and the number of enumerated nodes within a branch-and-bound tree. In this section we describe the basic ingredients of our branch-and-cut approach.

CUT Model

The directed cut-set formulation is obtained by projecting out flow variables from the SCF model and replacing constraints (2) and (3) with the following ones:

$$\sum_{a\in\delta^-(S)} \sum_{m\in M_a} u_{a,m} x_{a,m} \ge \sum_{k\in S} y_k d_k \quad \forall S \subset V \text{ s.t. } S \cap K \ne \emptyset \text{ and } r \notin S \tag{8}$$

The *cut-set inequalities* (8) state that every subset of nodes S, containing at least one customer and not containing r, must have enough *incoming capacity* to route the total demand requested inside the set. It is not difficult to see (see, e.g., Ljubić et al. [10]) that the new model, that we will refer to as the cut-set model (CUT), exhibits the same lower bounds as SCF. These bounds can be further strengthened as follows. Since $(\boldsymbol{x}, \boldsymbol{y})$ variables are binary and the coefficients are non-negative, the cut-set inequalities can be easily strengthened by rounding down some left-hand side coefficients:

$$\sum_{a\in\delta^-(S)} \sum_{m\in M_a} \min\left(u_{a,m}, \sum_{k\in S} d_k\right) x_{a,m} \geq \sum_{k\in S} d_k y_k \qquad \forall S \subseteq V \text{ s.t. } S\cap K \neq \emptyset \text{ and } r \notin S. \tag{9}$$

Further Valid Inequalities

The CUT model can be further strengthened with the following *connectivity cuts*. Every set of nodes containing at least one customer must have at least one incoming arc if the customer is included in the solution:

$$\sum_{a\in\delta^-(S)} \sum_{m\in M_a} x_{a,m} \geq y_k \qquad \forall S \subseteq V \setminus \{r\}, \forall k \in S \cap K. \tag{10}$$

¿From the knapsack constraint (4), we can derive a family of traditional *cover inequalities* as follows. Let J be a minimal subset of customers such that its complement $K \setminus J$ cannot satisfy the required coverage demand, i.e., $\sum_{k\in K\setminus J} p_k < p_0$ and $p_l + \sum_{k\in K\setminus J} p_k \geq p_0$, for any $l \in J$. The set J is called a *minimal cover* with respect to coverage requirement p_0. Let COV be the collection of all minimal covers with respect to p_0. Then, the following *minimal cover inequalities* are valid for our problem:

$$\sum_{k\in J} y_i \geq 1 \qquad \forall J \in COV. \tag{11}$$

These inequalities can be further strengthened by lifting. In general the separation of cover inequalities is NP-hard. See, e.g., the recent work of **?**] where heuristic separation approaches are proposed.

Another family of cover inequalities can be derived from the cut-set inequalities (8). Given a cut-set inequality (8) defined by $S \subset V, r \notin S$, define the index set $I(S) := \{(a,m) \mid a \in \delta^-(S), m \in M_a\}$ and the demand inside of S as $B := \sum_{k\in K\cap S} d_k$. Set $J \subset I(S)$ is called a *cover* with respect to $I(S)$ if $\sum_{(a,m)\in J} u_{a,m} < B$ and a *maximal cover* if, in addition, for all J' such that $I(S) \supseteq J' \supset J$: $\sum_{(a,m)\in J'} u_{a,m} \geq B$. If J is a maximal cover with respect to $I(S)$, then the following *maximal cover inequalities* are valid:

$$\sum_{(a,m)\in I(S)\setminus J} x_{a,m} \geq 1 + \sum_{k\in K\cap S} (y_k - 1). \tag{12}$$

These inequalities are also NP-hard to separate. However, modern MIP solvers have build-in mechanisms for detecting knapsack-type inequalities (like (4) or (8) in our case) and using heuristic techniques for separating cover inequalities (like (11) or (12)).

Non-customer nodes $V \setminus (K \cup \{r\})$ cannot have incoming (or outgoing) arcs only. Therefore, we can add the following *degree-balance constraints* that are valid due to the fact that the final solution is an acyclic graph:

$$\sum_{(l,i)\in A, l\neq j} \sum_{m\in M_{li}} x_{li,m} \geq \sum_{m\in M_{ij}} x_{ij,m} \qquad \forall (i,j)\in A, i\notin K, i\neq r \tag{13}$$

$$\sum_{(j,l)\in A, l\neq i} \sum_{m\in M_{jl}} x_{jl,m} \geq \sum_{m\in M_{ij}} x_{ij,m} \qquad \forall (i,j)\in A, j\notin K, j\neq r. \tag{14}$$

The following *in-degree inequalities* are very useful for the initialization of the LP model:

$$\sum_{a\in\delta^-(i)} \sum_{m\in M_a} x_{a,m} \geq y_i \qquad \forall i\in K \tag{15}$$

Notice that these are the cut-set inequalities (10) associated to all the singletons S.
Finally, the following constraints are valid:

$$\sum_{m\in M_{ij}} (x_{ij,m} + x_{ji,m}) \leq y_i \quad \forall (i,j)\in A. \tag{16}$$

They are implied by the family of connectivity constraints (10), together with the in-degree constraints (15). However we have empirically observed that they speed-up the branch-and-cut performance if added at the initialization phase.

Separation

The cutting plane approach is performed at each node of the branch-and-bound tree. It starts with the linear programming relaxation of the CUT model without (8). This relaxation is strengthened with the rounded cut-set inequalities (9) associated to all the singletons S and with the inequalities (13), (14), (15) and (16) mentioned above. Other inequalities are generated in an iterative way as it is described below.

At each iteration a fractional solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ is given. The separation problem of cut-set inequalities in general multiple-source multiple-sink case is NP-hard and can be reduced to the max-cut problem Barahona [2]. However, inequalities (8) for PC-LAN can be separated in polynomial time as follows. For a given fractional solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$, we define the directed *support graph* $G' = (V', A')$ where $V' := V \cup \{t\}$ with an additional sink t, and $A' := A_1 \cup A_2$ being $A_1 := \{a \in A \mid \sum_{m\in M_a} u_{a,m} x^*_{a,m} > 0\}$ and $A_2 := \{(k,t) \mid k \in K\}$. The capacity associated to each arc $a \in A_1$ is set to $\sum_{m\in M_a} u_{a,m} x^*_{a,m}$, and the capacity of each arc $a = (k,t) \in A_2$ is set to $d_k y^*_k$. If the minimum cut between r and t in G' is less than $\sum_{k\in K} d_k y^*_k$, it defines a violated inequality (8). Finally, the inequality is rounded and then inserted into the LP model.

In addition, also violated connectivity constraints (10) are dynamically separated. To this end, given a fractional solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$, we define a network from G where the capacity associated to each arc $a \in A$ is $\sum_{m\in M_a} x^*_{a,m}$. Then, if the minimum cut between r and any customer k in G is less than y^*_k this cut defines a violated inequality (10).

In general, a minimum cut problem has several optimal solutions. Especially when the dimension of the network is very large as is the case in our PC-LAN instances. Therefore, it is possible to find several violated inequalities from a given fractional solution. Multiple, nested cuts are produced for a fractional solution, as follows: The minimum cut problem is solved. Then the capacity of the arcs in the cut is increased. This defines a new minimum cut problem. This new problem is solved and if the associated inequality is violated it is added to the model. This procedure is repeated to produce multiple, nested cuts. Among all cuts with the same max-flow value, we prefer sparser ones. To detect them, we use the technique of adding an epsilon-capacity to every arc before calculating the flows. That way, cuts with less arcs are preferred to the more dense ones.

3 MIP-based heuristic approach

This section describes a MIP-based approach to find high quality feasible solutions to large-sized instances of the PC-LAN. It consists of three main ingredients:

1. Cutting Plane phase: The cutting plane approach works with relaxations of the CUT model described above. In the *separation phase* (cf. Section 2.3), a new set of violated inequalities is inserted into the LP. The LP is resolved and the optimal LP-solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ is taken as input for the following Network Construction phase.
2. Network Construction phase: First, a set of customers is selected according to the fractional values $\boldsymbol{y}^*$. Next, a network is constructed iteratively by using shortest path calculations on the graph with adapted edge weights. The fractional values $\boldsymbol{x}^*$ are taken into account for this construction.
3. Local Improvement phase: The solution found in the construction phase is subjected to a local improvement procedure. Flow routed along an *expensive* edge together with affected customers are removed, leaving a partial solution. Then the partial solution is repaired by adding new customers and extending the network design. Two different definitions of *expensive* are alternated.

The first phase is described in Section 2.3, and we now describe the remaining two phases. Before that, we first introduce the notation that is used in this section.

3.1 Notation

For the sake of a simpler description of the heuristic algorithm we use the following notation. We represent the modules M_e by numbers in $\{1, 2, \ldots\}$. A network design can be represented by a vector $\boldsymbol{z} \in \mathbb{N}^{|E|}$ consisting of module indices $z_e \in \{0\} \cup M_e$. For example, $z_e = 3$ means that the third available module for e is installed; $z_e = 0$ means that there is no installation on e. Capacities per edge are denoted by $\boldsymbol{g} \in \mathbb{R}_{\geq 0}$. A flow through the network is represented by a vector $\boldsymbol{f} \in \mathbb{R}^{|A|}_{\geq 0}$. The function $\mu_e : \mathbb{R}_{\geq 0} \mapsto M_e$ maps some required capacity to the index of the *most appropriate module* on edge e, i.e., the cheapest module with sufficient capacity, or the largest module if there is no module with sufficient capacity. More formally, for some required capacity $b \geq 0$:

$$\mu_e(b) = \begin{cases} 0 & \text{if } b = 0 \\ \arg\min_{\{m \in M_e | u_{e,m} \geq b\}} c_{e,m} & \text{if } b > 0 \text{ and } \exists m \in M_e | \, u_{e,m} \geq b \\ |M_e| & \text{otherwise} \end{cases} \qquad \forall e \in E.$$

An edge e is said to be *saturated* by a required capacity g_e if the largest module is already used on this edge and no free capacity is left, i.e., $z_e = |M_e|$ and $g_e = u_{e,z_e}$. Given a current capacity vector

$\boldsymbol{g}$, a suitable design vector $\boldsymbol{z}$ and some additionally required capacity $b \geq 0$, we define the following *edge weight approximations*:

$$w_e\left(g_e, z_e, b\right) = \begin{cases} c_{e,\mu_e(g_e+b)} - c_{e,z_e}, & \text{if } g_e < u_{e,|M_e|} \\ \infty, & \text{otherwise} \end{cases} \qquad \forall e \in E. \tag{17}$$

Hence w_e represents the cost for expanding the installation on e from the currently selected module z_e to the module $\mu_e(g_e + b)$. If the required capacity g_e saturates the edge e, such an expansion is impossible and the edge weight is infinite.

3.2 Network construction

Starting from a fractional solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ obtained after solving the LP-relaxation, we build a feasible solution by applying the following three procedures.

3.2.1 Rounding

Let $\boldsymbol{y}^*$ be a solution of the LP-relaxation of the CUT model. We sort the customer indices in order of decreasing fractional values y_k^*. We then define an integer feasible selection $\boldsymbol{y}$ by greedily setting indices of customers with large fractional values to one until the coverage constraint (4) is satisfied. The fractional vector $\boldsymbol{x}^*$ is used to compute a vector of minimum required capacities $\boldsymbol{g}^*$:

$$g_e^* = \sum_{m \in M_e} u_{e,m}(x_{ij,m}^* + x_{ji,m}^*) \qquad \forall e = \{i, j\}.$$

Note that g^* may not define the undirected capacity vector of a feasible network design. The fractional solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ may not satisfy all cut-set inequalities (8) associated to every set S, but only to those inequalities that have been separated so far.

3.2.2 Construction

Using the previously generated vector $\boldsymbol{y}$ and $\boldsymbol{g}^*$, this procedure constructs a feasible network design of the PC-LAN. Algorithm 3.1 describes the main steps. The initialization phase defines a *demand per node* $\boldsymbol{b} \in \mathbb{R}_{\geq 0}^{|V|}$ as:

$$b_k = \begin{cases} d_k y_k, & \text{if } k \in K \\ 0, & \text{otherwise.} \end{cases}$$

An initial network design $\boldsymbol{z}$ is defined via the most appropriate module per edge with respect to $\boldsymbol{g}^*$, i.e., $z_e := \mu_e(g_e^*)$ for all $e \in E$. The algorithm subsequently modifies $\boldsymbol{b}$, creates an undirected flow $\boldsymbol{g}$ and updates the design $\boldsymbol{z}$. In each iteration a node v with positive demand $b_v > 0$ is chosen. Denote the demand to be transported as $b := b_v$ and cancel the node demand of v: $b_v := 0$. The values of $\boldsymbol{g}$, $\boldsymbol{z}$ and b uniquely determine the edge weight approximation $\boldsymbol{w}$ via (17). This vector $\boldsymbol{w}$ defines the edge weights for the shortest paths calculation on G. A shortest path from v to r is computed: $SP_{\boldsymbol{w}}(v) = \langle v, v_1, v_2, \dots, r\rangle$. Along this path, the current demand b is *transported.* Denote the remaining capacity on e by $\bar{u}$ and the maximum that can be transported by $\bar{b}$. The flow $\boldsymbol{g}$ is increased: $g_e := g_e + b$ and the necessary installations $z_e := \mu_e(g_e)$ are made for all $e \in SP_{\boldsymbol{w}}(v)$.

Once Line 23 is reached, the node demand has been transported from v to r and the next iteration starts.

A special case occurs when an edge e on $SP_{\boldsymbol{w}}(v)$ does not offer sufficient remaining capacity $\bar{u}$, i.e., $b > \bar{u}$ in Line 18. Then, only this maximum available capacity $\bar{u}$ is transported on e. Now the node demands are changed appropriately at each endpoint of e. The node closer to v, denoted i, receives an additional demand of $b - \bar{u}$. The node closer to r, denoted j, receives an additional demand of $\bar{u}$. The edge e becomes saturated and the heuristic continues by picking the next randomly chosen node with positive node demand in Line 5. The heuristic terminates when $\boldsymbol{b} = \boldsymbol{0}$ and $\boldsymbol{z}$ is feasible for the chosen subset of customers represented by $\boldsymbol{y}$.

Of course, no shortest path may exist. This can be due to an infeasible input or due to the greedy decisions taken in the course of the algorithm. In this case the heuristic terminates in Line 11 without finding a feasible solution.

3.2.3 Flow Calculation

After the construction has produced a feasible solution $\boldsymbol{z}$, redundant capacities may have been installed along the edges. To reduce the installation cost, a minimum-cost flow problem is defined on the subgraph of G induced by $z_e > 0$. The flow cost are defined as $\frac{c_{e,z_e}}{u_{e,z_e}}$, and the capacity is set to u_{e,z_e} for all edges. The min-cost flow problem is solved and yields a directed flow vector $\boldsymbol{f}$. A new design vector $\boldsymbol{z}'$ can be derived from $\boldsymbol{f}$ by setting $z'_e := \mu_e(f_{ij} + f_{ji})$ for all $e \in E$. Clearly, $z'_e \leq z_e$ for all $e \in E$. The directed flow vector $\boldsymbol{f}$ also allows to express the design in terms of directed $x_{a,m}$ variables:

$$x_{ij,m} := \begin{cases} 1, & \text{if } f_{ij} > 0 \text{ and } m = \mu_{\{ij\}}(f_{ij}) \\ 0, & \text{otherwise.} \end{cases}.$$

3.3 Local improvement

Given an integer feasible solution represented by the vector $(\boldsymbol{z}, \boldsymbol{y}, \boldsymbol{f})$, we attempt the following Local Improvement strategy. The main steps are given in Algorithm 3.2. Initialize the new solution $(\boldsymbol{z}', \boldsymbol{y}', \boldsymbol{f}')$ as $\boldsymbol{z}' := \boldsymbol{0}$, $\boldsymbol{y}' := \boldsymbol{y}$, $\boldsymbol{f}' := \boldsymbol{f}$. Decompose the flow on each arc a into commodity flows, i.e., compute a flow per customer per arc. Pick an edge $\tilde{e}$ maximizing c_{e,z_e}. Those customers k that have a positive flow on this edge $\tilde{e}$ are removed from the selection, i.e., set $y'_k := 0$. In addition, the flow for these customers is removed from $\boldsymbol{f}'$. Compute the required capacity per edge $\boldsymbol{g}'$ on behalf of the reduced flow $\boldsymbol{f}'$. Define the edge weight approximation $\boldsymbol{w}$ as described by equation (17). Let l_k be the length of the shortest path $SP_{\boldsymbol{w}}(k)$ from k to r for all currently unselected customers, i.e., $y'_k = 0$. Select customers with small l_k values and add them to a new set $\boldsymbol{y}''$ until the combined selection $\boldsymbol{y}' + \boldsymbol{y}''$ satisfies the coverage constraint $\sum_{k \in K} p_k(y'_k + y''_k) \geq p_0$. Now start the Network Construction (Algorithm 3.1) with the minimum required capacity $\boldsymbol{g}'$ and the set of newly selected customers $\boldsymbol{y}''$. The result is a new network design $\boldsymbol{z}'$. Set $\boldsymbol{y}' := \boldsymbol{y}' + \boldsymbol{y}''$. If the new solution $(\boldsymbol{z}', \boldsymbol{y}')$ has a smaller objective value than the currently best found solution $(\boldsymbol{z}, \boldsymbol{y})$, the new solution $(\boldsymbol{z}', \boldsymbol{y}')$ becomes the new best solution. Otherwise the edge that had been selected in Line 6 is added to a taboo list T.

In order to achieve more diverse results we alternate the two criteria in Line 6 and Line 21 as

Algorithm 3.1 Network Construction.

```
Input: customer selection $\boldsymbol{y} \in \{0,1\}^{|K|}$, minimum required capacity $\boldsymbol{g}^* \in \mathbb{R}^{|E|}_{\geq 0}$
1: init node demand $\boldsymbol{b} \in \mathbb{R}^{|V|}_{\geq 0} : b_k := y_k d_k$ for all $k \in K$ and $b_v = 0$ for all $v \notin K$
2: init design $z_e := \mu_e(g^*_e)$ for all $e \in E$
3: init undirected flow $g_e := 0$ for all $e \in E$
4: while $\exists v \in V : b_v > 0$ do
5:   pick a random node $v \in V : b_v > 0$
6:   $b := b_v$ // the demand to be transported
7:   $b_v := 0$
8:   define edge weight $\boldsymbol{w} : w_e(g_e, z_e, b)\ \forall\ e \in E$ according to (17)
9:   compute a shortest path $SP_{\boldsymbol{w}}(v)$ from $v$ to $r$ in $\langle G, \boldsymbol{w} \rangle$
10:  if there is no shortest path then
11:    return failed
12:  end if
13:  for $e = (i,j) \in SP_{\boldsymbol{w}}(v) = \langle v, v_1, v_2, \ldots, r \rangle$ do
14:    $\bar{u} := u_{e,\mu_e(g_e+b)} - g_e$ // remaining capacity
15:    $\bar{b} := \min(b, \bar{u})$ // maximum that can be transported
16:    $g_e := g_e + \bar{b}$
17:    $z_e := \mu_e(g_e)$
18:    if $b > \bar{u}$ then // insufficient remaining capacity
19:      $b_i := b_i + b - \bar{u}$
20:      $b_j := b_j + \bar{u}$
21:      goto Line 5
22:    end if
23:  end for
24: end while
```

follows. The criterion for picking an edge with highest absolute cost c_{e,z_e} in Line 6 is modified to pick the edge with highest relative cost, i.e.,

$$\tilde{e} := \operatorname*{arg\,max}_{e=\{i,j\}\in E,(f_{ij}+f_{ji})>0, e\notin T} \left(c_{e,z_e}/f_e\right).$$

The criterion for choosing new customers with smallest shortest path lengths l_k in Line 21 is modified to choose customers mimizing the ratio of prizes over costs, i.e.,

$$\tilde{k} := \arg \max_{k\in K, y'_k+y''_k=0} \left(\frac{p_k}{c_k + l_k}\right).$$

The two options for these two criteria give four variations of Algorithm 3.2, that are cyclicly repeated. until 20 edges have been considered for deletion without improving the objective value.

3.4 Multi start modifications

As stated in Section 3, the Network Construction phase followed by the Local Improvement Phase is repeated in a multi-start fashion. To get a wide variation in the solutions during multi-starting we implemented the following modifications to the base algorithms provided in Sections 3.2 and 3.3.

For some PC-LAN instances it is advantageous to send the flow to groups of customers along the same path. If economies of scale are given, as is frequently the case with this type of network design problems, larger modules have a smaller relative cost $u_{e,m}/c_{e,m}$ than smaller modules. In situations like this the Network Construction heuristic described in Section 3.2 should be modified to first *cluster* some neighboring demands and second search for a routing to the access point for the combined demands. On the other hand, sometimes the opposite is true: Economies of scale are not given. For example, when smaller modules represent existing infrastructure and larger modules represent new connections that involve a high setup cost. For these problems it can be crucial to facilitate the existing infrastructure as much as possible and avoid the larger modules. Under these circumstances the heuristic should do the opposite of routing clustered demands together. Instead, it would be better to split the given demands apart and route the partial demands individually in order to facilitate the existing infrastructure in an optimal way.

In order to accommodate for these two contradicting ideas, we employ several variants of the basic algorithm from Section 3.2.2. To enable a clustering of demands, the loop in Lines 13-23 of Algorithm 3.1 is changed so that the installation is not necessarily done along the whole path right from v to the central office r, but instead stops at some earlier node j. Two criteria are used to select j: (i) j is the first node with a positive demand $b_j > 0$ encountered along the path, or (ii) j is at most q edges away from v. Observe that the demands are clustered if criterion (i) is applied and the parameter q is set to a small number. To implement this variant, these next instructions are inserted between Lines 22 and 23:

```
if criterion (i) or (ii) then
    $b_j := b_j + b$
    goto Line 5
end if.
```

Algorithm 3.2 Local Improvement.

```
Input: design $\boldsymbol{z} \in \mathbb{N}^{|E|}$, customer selection $\boldsymbol{y} \in \{0,1\}^{|K|}$, flow $\boldsymbol{f} \in \mathbb{R}_{\geq 0}^{|A|}$
1: $s := 0$ // improvement counter
2: $T = \varnothing$ // taboo list
3: while $s < 20$ do
4:    $\boldsymbol{z}' := \boldsymbol{0}, \boldsymbol{y}' := \boldsymbol{y}, \boldsymbol{f}' := \boldsymbol{f}$
5:    compute $f_a^k \in \mathbb{R}_{\geq 0}^{|A| \times |K|}$ such that $\sum_{k \in K} f_a^k = f_a \, \forall a \in A$ // flow decomposition
6:    $\tilde{e} := \arg\max_{e=\{i,j\} \in E, (f_{ij}+f_{ji})>0, e \notin T} (c_{e,z_e})$ // pick edge $\tilde{e}$
7:    for all $k \in K$ with $f_{(ij)}^k > 0$ or $f_{(ji)}^k > 0$ on $\tilde{e} = \{i,j\}$ do // reduction
8:       $y_k' := 0$
9:       $f_a' := f_a' - f_a^k \, \forall a \in A$
10:   end for
11:   $\boldsymbol{g}' \in \mathbb{R}_{\geq 0}^{|E|} := \boldsymbol{0}$ // required capacity
12:   for all $e = \{i,j\} \in E$ do
13:      $g_e' := f_{ij}' + f_{ji}'$
14:   end for
15:   $\boldsymbol{y}'' := 0$ // additional customers
16:   for all $k \in K$ with $y_k' + y_k'' = 0$ do // compute shortest path lengths
17:      define edge weight $\boldsymbol{w} : w_e(g_e', z_e', d_k) \forall \, e \in E$ according to (17)
18:      compute the shortest path $SP_{\boldsymbol{w}}(k)$ and denote the length by $l_k$
19:   end for
20:   while $\sum_{k \in K} p_k(y_k' + y_k'') < p_0$ do
21:      $\tilde{k} := \arg\min_{k \in K, y_k' + y_k''=0} l_k$
22:      $y_{\tilde{k}}'' := 1$
23:   end while
24:   $(\boldsymbol{z}', \boldsymbol{f}') :=$ Network Construction $(\boldsymbol{g}', \boldsymbol{y}'')$
25:   $\boldsymbol{y}' := \boldsymbol{y}' + \boldsymbol{y}''$
26:   if $\sum_{e \in E} c_{e,z_e'} + \sum_{k \in K} c_k(y_k') < \sum_{e \in E} c_{e,z_e} + \sum_{k \in K} c_k(y_k)$ then // improvement
27:      $\boldsymbol{z} := \boldsymbol{z}', \boldsymbol{y} := \boldsymbol{y}', \boldsymbol{f} := \boldsymbol{f}'$ // keep new best solution
28:      $i = 0$
29:   else // no improvement
30:      $i := i + 1$
31:      $T := T \cup \{\tilde{e}\}$ // the edge from Line 6 becomes taboo
32:   end if
33: end while
34: return $(\boldsymbol{z}, \boldsymbol{y}, \boldsymbol{f})$
```

The idea of this clustering is to merge customers that are *close* to each other with respect to the stepwise edge cost function. To provide an *anti-clustering* variant of the algorithm, two additional modifications of the algorithm are introduced. The first is a redefinition of the node demands. Instead of one number b_v per node v, we use a list of subdemands $B_v = \{b_{v,1}, b_{v,2}, \dots\}$ for every node that can be treated independently. The initialization in Line 1 changes to

$$B_k := \{y_k d_k\} \text{ for all } k \in K$$
$$B_v := \varnothing \text{ for all } v \in V \setminus K.$$

In Line 5, *one* of the subdemands $b_{v,t}$ of a node v with at least one positive subdemand is chosen and Lines 6-7 become

$$b := b_{v,t}$$
$$B_v := B_v \setminus \{b_{v,t}\}.$$

The update of the node demands in case of insufficient remaining capacity in Lines 19-20 becomes:

$$B_i := B_i \cup \{b - \bar{u}\},$$
$$B_j := B_j \cup \{\bar{u}\}.$$

Whereas the update of node demands in case of criterion (i) or (ii), introduced above, becomes:

```
if criterion (i) or (ii) then
    $B_j := B_j \cup \{b\}$;
    goto Line 5
end if.
```

The second modification to help with anti-clustering is to initially split the demands in two $B_k = \{y_k d_k/2, y_k d_k/2\}$ or three $B_k = \{y_k d_k/3, y_k d_k/3, y_k d_k/3\}$ partial demands in Line 1.
A specific variant of the Network Construction algorithm can be chosen with four parameters:

- Activate criterion (i), or do not activate it.
- Select a value for $q \in \{1, \dots, |V|\}$ for criterion (ii).
- Join node demands by using *one* value b_v per node, or do not join but use a list of values B_v.
- Select a splitting ratio $\in \{1, 2, 3\}$ for the initial definition of node demands.

Each time the Network Construction algorithm is executed, a specific variant is chosen by the means of a learning adaptation mechanism known as *Reactive Search Optimization.* Initially pre-specified settings for the four parameters are used. Then the settings for the parameters are varied from a diversification of the settings towards an intensification. That is, from randomly perturbed settings towards settings that have produced the best objective values so far.

4 Computational results

This section discusses the performance of the approaches described in Section 3 on two sets of benchmark instances: (i) **small instances** that are graphs generated by Salman et al. [17] and used for testing the LAN problem, and (ii) new set of **large-scale instances** arising from the real-world motivation of our research. The computations are performed on a computer with Intel Xeon 2.6 GHz and 3 GB RAM. We have used CPLEX 12.2 to solve the linear programs and also as a MIP framework.

4.1 Benchmark instances

Small Instances

This is a group of 60 randomly generated problems originally published in Salman [16]. They were also used in [17, 3, 10]. The instances contain 20, 30 and 40 nodes and there are 12 groups in total with 5 instances per group. There are 9 cable types obeying economies of scale in each of the instances. The cheapest cable type has a capacity of 6. The convex combinations of these cable types generate up to $\lceil \sum_{k \in K} d_k/6 \rceil$ modules. See [3, 17] for a detailed description. The groups are created according to the number of nodes, location of the root node ('c' being *central*, or 'r' being *random position*), and the level of demand ('l' stands for *low demand*, which is randomly generated between 0 and 30; 'h' stands for *high demand*, randomly generated between 0 and 60).

Large instances

Starting from real-world inputs, we have generated three new large-sized benchmark instances using real-world locations. The instances are named A, B and C. The number of nodes in our instances is 86 745, 48 247 and 77 329, respectively. The number of edges is 116 750, 65 304 and 107 696 and the number of potential customers is 1 157, 720 and 1 498, respectively. The other features of the PC-LAN instances are generated following the procedure described here. We are given three types of nodes: physical locations of customers, location of the central office and locations of intermediate nodes. For each customer location, we are given the number of subscribers associated to this location. Usually, several splitter devices with various splitting ratios (e.g., 1:4, 1:16, 1:32) are available. Their costs obey economies of scales. For example, to connect 16 subscribers, a device must be installed that costs €2000 and one optical fiber should come in that building. To connect a building with 17 subscribers, a device that costs €3000 and 2 fibers are needed and this larger device is sufficient to support up to 32 subscribers. It is not feasible to connect only a fraction of subscribers at the customer node. Instead, decisions have to be made whether all subscribers or none of them are going to be served. This allows for preprocessing of input data and exact calculation of customers' demands and the corresponding set-up costs.

These instances consider three types of links: existing fibers, existing ducts and public streets. Existing and currently unused optical fiber cables can be used with a very small cost. In existing ducts, a limited number of additional cables can be installed for relatively little cost. Along street segments, new trenches can be built and new ducts and cables can be laid. In addition to the cost for the ducts and cables there is a significant overhead cost for new trenches. Different cable technologies are available. They differ in terms of the number of fibers per cable and cost per meter. Existing fiber and existing ducts can be used simultaneously. If new trenches are dug, any existing infrastructure is removed and replaced by the new installations.

In addition the practical situation will be further complicated by the availability of different cable technologies. They can provide different numbers of fibers per cable. Other more expensive technologies can provide the same number of fibers in a cable of smaller diameter. Using these would allow to put more cables, hence more fibers in an existing duct. Furthermore between two locations, there will sometimes be different existing ducts. Independent decisions about how many cables of which technology to put into each of them will lead to many different possible combinations. Taking these aspects into account, we pre-computed the available modules for each edge. The minimum, average and maximum number of modules per edge in instance A are 3, 9 and 131, respectively. For instance B they are 3, 9 and 84 and for instance C, 3, 10 and 161. This shows the high diversity in the input data defining our instances.

We considered values of $\alpha \in \{0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$. Hence the set of benchmark instances contains 21 examples of PC-LAN.

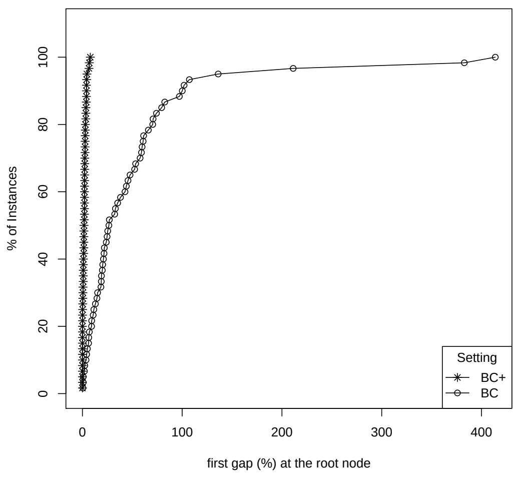
4.2 Performance of the branch-and-cut algorithm

We first present the results of our branch-and-cut algorithm on the set of small instances. For the set of small instances, the main results are summarized in Table 1 and Figure 3. We consider the following two settings: (1) BC+, which is the branch-and-cut approach with the MIP heuristic as described in Section 3, and (2) BC, which is the branch-and-cut approach with the default CPLEX heuristics. For each of the 12 groups in Table 1 we report the following values: "c/r" and "h/l" columns describe the type of instances (as explained above). In addition, for the two branch-and-cut settings, with and without our proposed MIP-heuristic, we report the following values: the first obtained percentage gap at the root node ($gap_f[\%]$) and the time in CPU seconds when this gap is achieved ($t_f[s]$), gap at the root node ($gap_r[\%]$) and the time in CPU seconds needed to solve the LP-relaxation at the root node ($t_r[s]$), and the final gap obtained after the time limit of 1 000 seconds is reached ($gap[\%]$). The value $t[s]$) provides the total running time. All values are averaged over 5 instances per each group. Let LB denote the best lower bound (or the optimal solution, if known) of an instance, and let UB be the value of a feasible solution. The reported gaps are calculated as $100(UB - LB)/UB$.

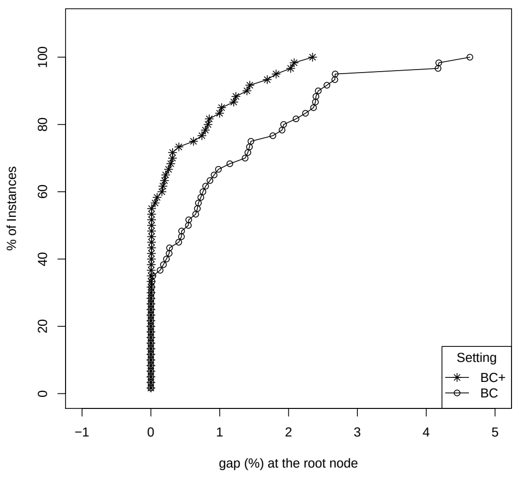
Additional information regarding the performance of the two branch-and-cut settings is available at Figure 3. Each point (x, y) on these charts is to be interpreted as follows: $y\%$ of all instances obtained the gap which is $\leq x$. Three charts correspond to the: first gaps at the root node, gaps at the root node and final gaps after 1 000 seconds, respectively.

Summarizing the information of Table 1 and Figure 3 we conclude that at the beginning of the optimization process, BC+ finds feasible solutions of relatively good quality much faster than the BC setting with the default CPLEX heuristic. More precisely, in less than a second, BC+ obtains solutions that are at most 4% above the lower bounds, on average. At the same time, BC needs a few seconds to detect solutions that are between 13% and 185% above the lower bounds, on average. Even after finishing the complete cutting plane phase at the root node, there are still significant differences in the quality of upper bounds obtained by BC+ and those obtained by BC (cf. Figure 3(b)). This can also be observed by comparing the columns denoted by $gap_r[\%]$ in Table 1 for the two settings.

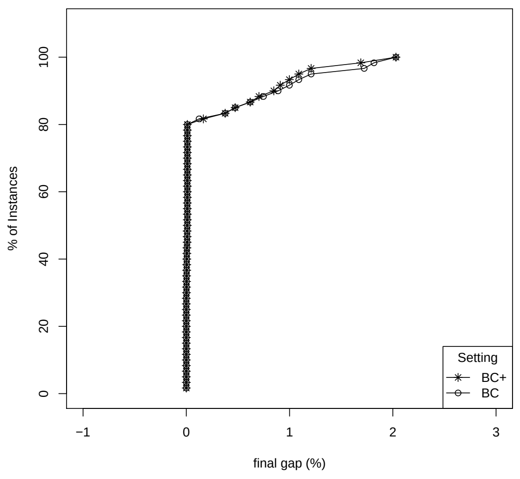
Finally, if sufficient running time is available, then both approaches succeed in closing the gap, and we observe that BC+ becomes more successful with the increasing instance size.



(a)



(b)



(c)

Figure 3: Results on small instances: (a) gaps of first feasible solutions detected at the root node, (b) gaps at the root node, (c) final gaps.

			BC+						BC					
Inst.	c/r	h/l	gap_f[%]	$t_f[s]$	gap_r[%]	$t_r[s]$	gap[%]	$t[s]$	gap_f[%]	$t_f[s]$	gap_r[%]	$t_r[s]$	gap[%]	$t[s]$
20	c	h	1.9	0.3	0.4	4.4	0.0	7.5	52.0	0.1	0.5	0.8	0.0	1.7
20	c	l	0.8	0.3	0.0	4.0	0.0	6.1	34.1	0.0	0.0	0.6	0.0	1.3
20	r	h	2.6	0.3	0.1	4.3	0.0	43.3	82.1	0.3	1.3	1.2	0.0	24.1
20	r	l	1.1	0.3	0.1	3.7	0.0	6.5	21.9	0.0	0.2	0.7	0.0	3.1
30	c	h	0.9	0.5	0.5	11.9	0.2	439.9	30.8	0.8	0.5	6.0	0.2	445.1
30	c	l	2.1	0.5	0.1	7.5	0.0	470.4	64.4	0.1	0.9	4.8	0.0	190.0
30	r	h	1.9	0.4	0.4	6.2	0.0	251.9	34.3	0.4	0.8	3.3	0.0	95.9
30	r	l	0.8	0.5	0.0	7.8	0.0	32.6	52.4	0.6	0.7	4.3	0.0	11.5
40	c	h	1.6	0.7	0.5	17.8	0.4	970.7	25.2	4.2	1.1	13.2	0.4	635.1
40	c	l	4.0	0.5	0.8	13.0	0.5	497.4	13.1	2.8	1.6	8.4	0.5	453.6
40	r	h	3.3	0.7	1.2	12.6	0.7	878.0	30.2	1.7	2.4	6.2	0.7	915.3
40	r	l	2.8	0.7	0.8	15.3	0.4	823.8	185.3	2.4	1.6	9.3	0.5	823.3

Table 1: Comparison of the branch-and-cut algorithm with and without the heuristic on small instances with $\alpha = 0.7\%$.

4.3 Solving the large instances

In this section we hybridize the proposed cutting plane approach with the multi-start heuristic, in order to provide high-quality solutions for large-scale instances and to give the certificate of their quality at the same time. For this purpose, we ran the code for 10 hours with particular settings described below.

Separation Settings: In preliminary experiments, we tested different configurations and selected the following scheme of alternating two configurations, that worked best for our instances. In configuration (I), we favor connectivity cuts, and to this end we produce at most two nested cut-set inequalities (9) and at most 2 000 nested connectivity cuts (10). In the second configuration (II), we separate at most 20 nested cut-set inequalities (9) and do not generate connectivity cuts. We apply configuration (I) iteratively until no more violated inequalities are found, or until the improvement of the objective value in the last ten iterations is too small. Then we apply configuration (II) iteratively until no more violated inequalities are found, or until the improvement is too small. Let o be the current objective value, let o_{I} be the objective value derived ten iterations ago with configuration (I), and let o_{II} be the objective value derived ten iterations ago with configuration (II). The relative improvement for configuration (I) is said to be *too small* once $(o - o_{\mathrm{I}})/o$ drops below $\epsilon = 10^{-4}$. Note that $(o - o_{\mathrm{I}})/o$ may again become greater than ϵ while configuration (II) is active. Thus the algorithm may switch back to configuration (I) and vice-versa. Once no connectivity cuts, nor cut-set inequalities exist, or both values $(o - o_{\mathrm{I}})/o$ and $(o - o_{\mathrm{II}})/o$ are below ϵ, we resort to branching.

Non-MIP variant of the heuristic To measure the impact of the MIP information in our heuristic approach we have also designed an alternative heuristic variant which does not make use of a MIP solver. This variant may be of interest for practical purposes since a company may desire not to purchase and install a black-box MIP solver in order to heuristically solve instances. Our non-MIP variant works as follows.

1. Compute a selection of customers $\boldsymbol{y}$ that satisfies the coverage constraint (4).
   This is done similarly to the customer selection in the Local Improvement Algorithm 3.2, Lines 15-23. Since there are no currently selected customers, this set is empty $\boldsymbol{y}' = \mathbf{0}$. Also, initially there is no current installation, i.e., $\boldsymbol{z}' = \mathbf{0}$ and no minimum required capacity, i.e., $\boldsymbol{g}' = \mathbf{0}$. Now $\boldsymbol{y} := \boldsymbol{y}''$ denotes a customer selection, feasible with respect to inequality (4).
2. Apply the construction phase 3.2.2 on $\boldsymbol{y}$ and no initial required capacity, i.e., $\boldsymbol{g}^* = \mathbf{0}$

These two steps compensate for the missing fractional solution in the algorithmic framework.

Computational results This section compares the MIP-based approach described in Section 3 and the non-MIP variant when solving our benchmark instances. We apply a time limit of 10 hours for both approaches. The MIP-based approach applies only the Cutting Plane phase in the first 2 hours. In the remaining 8 hours, every Cutting Plane phase is followed by multi-starting Network Construction, followed by Local Improvement as long as a better solution is produced. Inside the Local Improvement, the removal of different edges is repeatedly tried, until 20 recent attempts did

#	α	LB	UB	gap	Gap_{non}	Gap_{MIP}	$\lvert V^*\rvert$	$\lvert V_1^*\rvert$
A	0.4	894318	1103084	18.93	9.10	**0.00**	1712	14
A	0.5	1245096	1558613	20.12	6.27	**0.00**	2462	11
A	0.6	1617097	2064569	21.67	4.99	**0.00**	3098	47
A	0.7	2032880	2699669	24.70	2.30	**0.00**	4196	51
A	0.8	2599170	3433859	24.31	**0.00**	0.57	5454	108
A	0.9	3400201	4386960	22.49	**0.00**	1.60	6776	19
A	1.0	7188015	8584895	16.27	**0.00**	3.01	9970	18
B	0.4	522753	568518	8.05	13.43	**0.00**	1130	15
B	0.5	715968	778720	8.06	11.75	**0.00**	1288	9
B	0.6	938149	1003232	6.49	12.84	**0.00**	1654	15
B	0.7	1228323	1322599	7.13	7.76	**0.00**	1932	19
B	0.8	1601173	1771221	9.60	3.72	**0.00**	2521	26
B	0.9	2126598	2349981	9.51	2.63	**0.00**	3206	31
B	1.0	3463753	3916245	11.55	**0.00**	1.08	5286	26
C	0.4	1005973	1106272	9.07	16.02	**0.00**	2968	26
C	0.5	1383417	1565520	11.63	18.94	**0.00**	3446	46
C	0.6	1844854	2249367	17.98	6.53	**0.00**	4089	55
C	0.7	2349758	3018622	22.16	2.73	**0.00**	5691	90
C	0.8	3011135	3927335	23.33	**0.00**	1.14	7047	100
C	0.9	4016022	5180962	22.49	**0.00**	1.17	8552	97
C	1.0	6278802	7384655	14.98	**0.00**	2.40	11607	112

Table 2: Results of the MIP-based heuristic versus the non-MIP variant on large instances.

not improve the solution. The non-MIP approach multi-starts until the time limit of 10 hours is up. Inside the Local Improvement the iteration continues until the solution has not improved in the 200 recent attempts.

Table 2 compares the performance of the MIP heuristic and the non-MIP variant. For the three instances (A, B and C) and for each coverage rates the following results are reported:

- # is the instance character.
- α is the coverage rate.
- LB gives the lower bound obtained while running the MIP-based heuristic.
- UB gives the best upper bound obtained by the MIP-based heuristic or the non-MIP variant.
- gap shows the optimality gap $(UB - LB)/UB$ in percent.
- Gap_{MIP} denotes the relative distance $(UB_{\text{MIP}} - UB)/UB$ in percent, where UB_{MIP} is the upper bound obtained by the MIP-based heuristic.
- Gap_{non} denotes the relative distance $(UB_{\text{non}} - UB)/UB$ in percent, where UB_{non} is the upper bound obtained by the non-MIP variant.

- $|V^*|$ gives the number of nodes in the best solution with value UB.
- $|V_1^*|$ gives the number of nodes in the best solution with in-degree greater than 1, where edges are understood to be oriented in the direction of a flow from r to the customers.

A value of 0.00 in Gap_{MIP} and Gap_{non} implies that the corresponding heuristic approach found the best upper bound. Bold values indicate which of the approaches provides the best upper bound. A value of 0 in column $|V_1^*|$ would imply that the solution is a tree, thus this column gives a measure of *deviation from tree*.

For none of the 21 instances the MIP-based approach finished the root node of the branch and bound tree in the time limit of 10 hours. Table 2 shows that the MIP-based heuristic found the best solution in 14 out of the 21 instances. The average value of Gap_{MIP} is 0.52 while the average value of Gap_{non} is 5.67. So on average the MIP-based heuristic is closer to the best found solution than the non-MIP variant. And also the largest advancement of MIP over non-MIP is more pronounced than the other way around. The largest advancement of the MIP aproach over the non-MIP approach, seen for C with $\alpha = 0.5$ is $|Gap_{\text{non}} - Gap_{\text{MIP}}| = 18.94$. While the largest improvement of the non-MIP approach over the MIP-approach, seen for B with $\alpha = 1.0$ is 3.01. The MIP-based heuristic is clearly better on instances with smaller values of α, which can be seen from the relatively larger distances of Gap_{non}. This can be explained by the fact that our MIP-based heuristic is better designed to exploit the combinatorial nature of the problem by selecting the appropriate subset of customers, guided by the LP-information. On pure LAN instances, all customers have to be served, i.e., $y_i = 1$, for all $i \in K$, and therefore the advantage of using the LP-information is not given anymore. Similar situation happens for large coverage values, where many of y-variables are close to one.

Comparing the growth of the cost of the optimal solution (by assuming that the provided lower bound are safe estimates of the optimal values), we observe that increasing the coverage rate by 10%, may lead to the total increase of the investment cost by more than 100% (cf. instance A, and the increase from $\alpha = 0.9$ to $\alpha = 1.0$). In the remaining two instances, the corresponding increase from the coverage of 90% to the full coverage, results in the increase of the investment cost by more than 50%. This clearly explains why the decision makers prefer the PC-LAN model over the LAN model when deploying the local access networks. Typical coverage values considered by decision makers clearly depend on the available budget, but they usually range between 60% and 90%.

To take a closer look at the performance of the two heuristic approaches during the 10 hours, we have selected two plots, one with the coverage rate of 70%, where the MIP-heuristic significantly outperforms the non-MIP variant, and one with the coverage rate of 100%, where the MIP-heuristic gives a solution which is 1% percent worse than the best solution found by the non-MIP variant. However, the advantage of the MIP-heuristic is that it also provides the lower bounds as certificates of the solution quality. We point out that the instance B, which is the smallest among the three considered graphs, contains more than 48 000 of nodes and more than 65 000 of edges, and that the final gaps we report lie between 6.5% and 11.5%. Figure 4 illustrates the performance of the two approaches on the instance B with $\alpha = 0.7$, where the MIP-based heuristic ends with a better solution than the non-MIP variant. MIP Heuristic - Upper Bound and Lower Bound show a value in every iteration of the MIP approach. Non-MIP - Upper Bound shows a value every time an

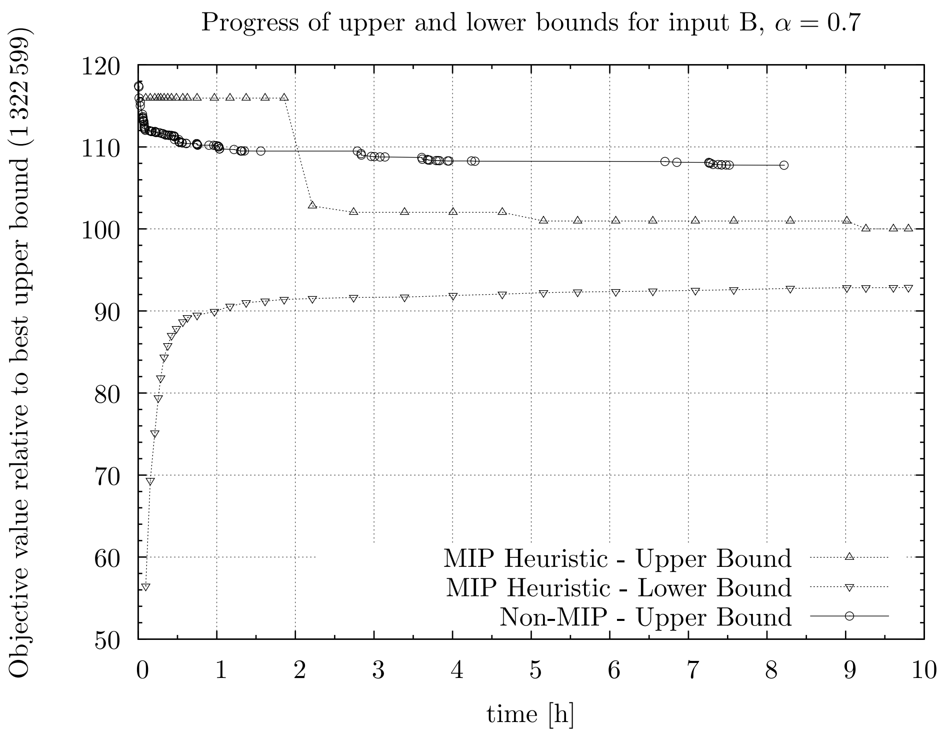


Figure 4: Instance B with $\alpha = 0.7$, where the MIP-based heuristic performs better than the non-MIP variant.

improved solution is found. A value of 100 for the relative objective value in the figure corresponds to $Gap_{\text{MIP}} = 0.0$ in Table 2. The time is represented in hours. Note that the MIP approach starts with an initial heuristic solution computed similarly to the non-MIP variant. This initial heuristic multi-starts as long as the solution gets better in every iteration. The next 20%, or 2 hours of the runtime are spent, iterating in the Cutting Plane phase to build up a set of cutting planes before Network Construction and Local Improvement algorithms are executed. We observe that the non-MIP variant slowly improves the quality of the solution and in the last almost two hours there is no more improvement. In contrast to this, the MIP approach finds a significantly better feasible solution on the first execution of Network Construction and Local Improvement after 2 hours. Furthermore, the MIP approach slowly improves the quality of upper and lower bounds providing the final gap of 7.13% between the lower and the upper bound.

Figure 5 illustrates the performance on the instance B with coverage rate 1.0, where the non-MIP variant ends with a better solution than the MIP-based heuristic. A value of 100 for the relative objective value in the figure corresponds to $Gap_{\text{non}} = 0.0$ in Table 2. We observe that the non-MIP variant, again, slowly improves the quality of the solution. On the other hand, the MIP approach does not seem to draw a noticeable advantage of the information from the fractional solutions. The MIP approach behaves similar to the non-MIP variant with the exception of investing much of the runtime in cutting planes and LP solutions. Thus it does not quite achieve the same solution quality as the non-MIP variant.

Table 3 presents further results of the MIP-based heuristic on our 21 instances. The meaning of the columns is the following:

- gap' is the percentage deviation between the initial solution value UB' (before solving the first linear relaxation) and the best lower bound obtained while running the MIP-based heuristic (LB). It is computed as $100(UB' - LB)/UB'$.
- Gap'_{MIP} denotes the gap of the initial solution before solving the first linear relaxation. As in the previous table, this gap has been calculated with respect to the best upper bound UB, and it is a percentage, i.e. it is computed as $100(UB' - UB)/UB$.
- t'_{MIP} is the time (in seconds) to produce the initial solution.
- t_{MIP} is the total time (in seconds) of the MIP-based heuristic minus the time consumed to compute the lower bound (i.e., separation phase and MIP solver).
- n_{MIP} gives the number of improved solutions found during the MIP-based heuristic.
- n_{LP} gives the number of iterations of the cutting-plane algorithm, i.e., the number of LP solutions.
- $n(9)$ shows the number of generated rounded cut-set inequalities (9).
- $n(10)$ shows the number of generated connectivity cuts (10).
- $\overline{n}(9)$ shows the average number of generated rounded cut-set inequalities (9) per fractional solution in the second half of the iterations of the cutting-plane algorithm.
- $\overline{n}(10)$ shows the average number of generated connectivity cuts (10) per fractional solution in the second half of the iterations of the cutting-plane algorithm.

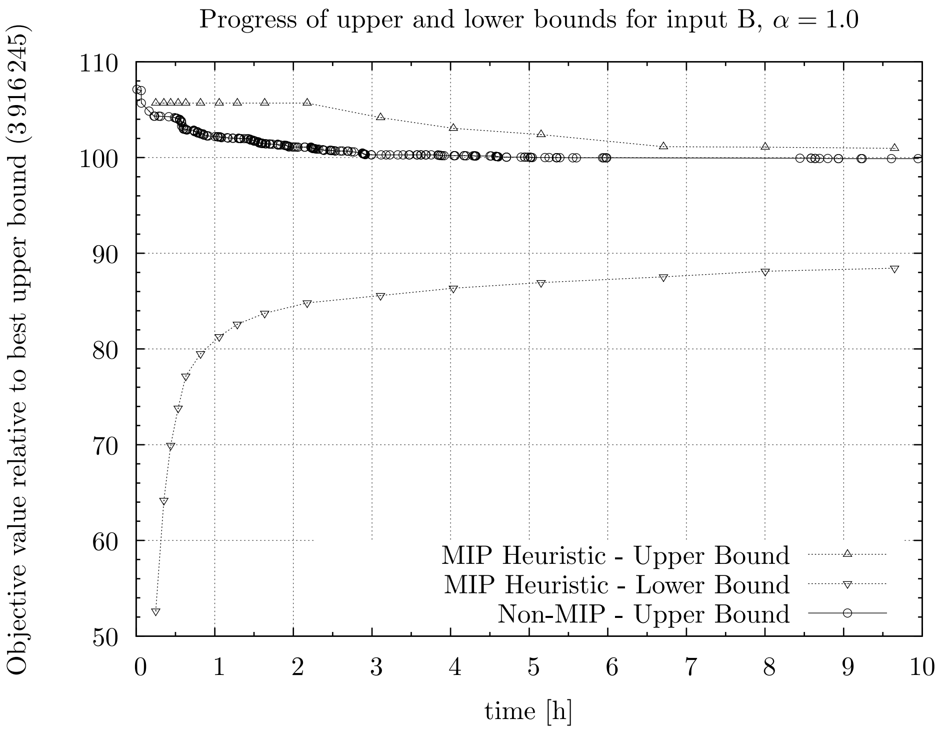


Figure 5: Instance B with $\alpha = 1.0$, where the non-MIP variant performs better than the MIP-based heuristic.

#	α	gap'	Gap'_{MIP}	Gap_{MIP}	t'_{MIP}	t_{MIP}	n_{MIP}	n_{LP}	$n(9)$	$n(10)$	$\overline{n}(9)$	$\overline{n}(10)$	$t(9)$	$t(10)$
A	0.4	27.44	11.73	0.00	68	9411	4	17	34	10074	2.0	492.3	99	5999
A	0.5	27.86	10.74	0.00	23	8938	4	17	34	11891	2.0	595.5	74	5102
A	0.6	28.19	9.07	0.00	142	12188	6	15	30	11957	2.0	658.4	79	6933
A	0.7	30.06	7.67	0.00	303	14365	6	13	26	12601	2.0	826.3	84	7316
A	0.8	27.57	4.50	0.57	39	12915	5	13	26	14116	2.0	943.3	81	7640
A	0.9	23.84	1.77	1.60	659	8370	2	12	24	15576	2.0	1159.6	52	5724
A	1.0	18.72	3.01	3.01	381	8421	1	8	16	13608	2.0	1514.0	15	6204
B	0.4	22.89	19.24	0.00	15	18477	6	125	790	4419	6.1	9.1	918	1603
B	0.5	22.10	18.03	0.00	86	18818	4	89	394	5601	3.6	13.0	421	2022
B	0.6	23.74	22.62	0.00	30	16329	10	61	302	6290	8.0	8.1	485	2318
B	0.7	19.90	15.95	0.00	91	12675	5	33	84	7692	3.1	87.6	110	2994
B	0.8	17.68	9.82	0.00	171	14659	5	25	50	9835	2.0	214.4	59	3461
B	0.9	17.93	10.27	0.00	186	19103	7	23	46	10623	2.0	256.2	51	3576
B	1.0	16.32	5.69	1.08	461	13909	7	16	32	12733	2.0	553.7	17	4120
C	0.4	26.85	24.31	0.00	459	23623	3	37	74	8738	2.0	43.3	447	4886
C	0.5	29.94	26.13	0.00	455	23863	8	24	48	9287	2.0	105.0	423	5953
C	0.6	27.44	13.04	0.00	426	21522	8	20	40	12010	2.0	271.7	291	7059
C	0.7	28.53	8.91	0.00	1523	22666	7	14	28	11288	2.0	476.3	270	8598
C	0.8	28.05	6.56	1.14	1324	20981	8	14	28	13683	2.0	666.5	253	10102
C	0.9	26.57	5.57	1.17	844	15655	6	15	30	16787	2.0	825.9	153	7992
C	1.0	18.17	3.90	2.40	3157	17509	5	12	24	18288	2.0	1173.4	60	8981

Table 3: Details of the MIP-based heuristic on the large instances.

- $t(9)$ shows the time consumed in separating rounded cut-set inequalities (9).
- $t(10)$ shows the time consumed in separating connectivity cuts (10).

¿From this table it can bee seen that around half of the total time of 10 hours (36 000 seconds) in the MIP-based heuristic is consumed by the cutting-plane procedure where inequalities are separated and linear programming relaxations are solved. However, as previously observed, this time consumption increases the solution quality when the coverage rate α is small. The objective value of the initial heuristic solution computed before the first iteration of the cutting-plane procedure is similar to the objective value of the best solution when α is large. In particular, we even observe that on instance A with 1.0 of coverage rate, the initial solution available in the first 5 minutes of the computation was not improved during the whole 10 hours of the MIP-based heuristic. The situation is different when the coverage rate is small. Note that the number of generated inequalities (9) and (10) is strongly affected by the separation settings described in Section 2.3. We also tested different settings to encourage finding more rounded cut-set inequalities, but the overall performance of the approach was not better. As shown in the table, on average we generate two rounded cut-set inequalities from each fractional solution, which means that configuration (I) was executed once on each cutting-plane iteration. Columns $\overline{n}(9)$ and $\overline{n}(10)$ indicate the average numbers of inequalities generated at the end of the cutting-plane process. For example, in the last 30 cutting-plane iterations only an average of 8.1 connectivity cuts are generated when solving instance B with 0.6 coverage rate. Finally, we also observe that the number of separated connectivity cuts (10) increases with the increasing coverage rates.

Conclusion

We have proposed a MIP-based approach to solve a new network design problem arising in a telecommunication context where not all customers need to be served. Instead, the company is interested in finding a good feasible solution to reach at least a given percentage. The problem is called PC-LAN and to our knowledge this is the first work to solve it. Based on a mathematical formulation, we propose a branch-and-cut approach. On small instances this approach finds optimal solutions. On large instances it has been adapted to find satisfactory solutions. Our procedure creates feasible solutions from the fractional ones obtained by separating two families of inequalities. A local search approach is used to improve each feasible solution, and all the approach is embedded in a multi-start framework. To measure the convenience of having a MIP solver inside our heuristic approach, we have also implemented a variant without using the cutting-plane ingredient. Our experiments have shown that the MIP-based approach significantly outperforms the non-MIP variant for coverage rates below $\alpha = 0.8$. These coverage rates are also typical for real-world applications motivating our research.

References

[1] A. Arulselvan, A. Bley, S. Gollowitzer, I. Ljubić, and O. Maurer. MIP modeling of incremental connected facility location. In J. Pahl, T. Reiners, and S. Voß, editors, *Proceedings of the International Network Optimization Conference (INOC)*, volume 6701 of *Lecture Notes in Computer Science*. Springer, 2011.

[2] F. Barahona. Network design using cut inequalities. *SIAM Journal on Optimization*, 6(3): 823–837, 1996.

[3] D. Berger, B. Gendron, J.-Y. Potvin, S. Raghavan, and P. Soriano. Tabu search for a network loading problem with multiple facilities. *Journal of Heuristics*, 6(2):253–267, 2000. URL `http://www.iro.umontreal.ca/~potvin/`.

[4] A. Frangioni and B. Gendron. 0-1 reformulations of the multicommodity capacitated network design problem. *Discrete Applied Mathematics*, 157(6):1229 – 1241, 2009.

[5] S. Gollowitzer and I. Ljubić. MIP models for connected facility location: A theoretical and computational study. *Computers & Operations Research*, 38(2):435–449, 2011.

[6] S. Gollowitzer, L. Gouveia, and I. Ljubić. A node splitting technique for two level network design problems with transition nodes. In J. Pahl, T. Reiners, and S. Voß, editors, *Proceedings of the International Network Optimization Conference (INOC)*, volume 6701 of *Lecture Notes in Computer Science*. Springer, 2011.

[7] S. Gualandi, F. Malucelli, and D. L. Sozzi. On the design of the next generation access networks. In A. Lodi, M. Milano, and P. Toth, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, 7th International Conference, CPAIOR 2010, Bologna, Italy, June 14-18, 2010. Proceedings*, volume 6140 of *Lecture Notes in Computer Science*, pages 162–175. Springer, 2010.

[8] Y. Kim, Y. Lee, and J. Han. A splitter location-allocation problem in designing fiber optic access networks. *European Journal of Operational Research*, 210(2):425 – 435, 2011.

[9] M. Leitner and G. R. Raidl. Branch-and-cut-and-price for capacitated connected facility location. *Journal of Mathematical Modelling and Algorithms*, 2011. To appear.

[10] I. Ljubić, P. Putz, and J. Salazar. Exact approaches to the single-source network loading problem. *Networks*, 2011. To appear.

[11] I. Ljubić, P. Putz, and J. J. Salazar. A heuristic algorithm for a prize-collecting local access network design problem. In J. Pahl, T. Reiners, and S. Voß, editors, *Proceedings of the International Network Optimization Conference (INOC)*, volume 6701 of *Lecture Notes in Computer Science*. Springer, 2011.

[12] M. Martens, E. Patzak, A. Richter, and R. Wessäly. Werkzeuge zur Planung und Optimierung von FTTx-Netzen. In *16. ITG-Fachtagung Kommunikationskabelnetze, Köln, Germany*, volume 218, pages 37–41. VDE-Verlag, 2009.

[13] M. Martens, S. Orlowski, A. Werner, R. Wessäly, and W. Bentz. FTTx-PLAN: Optimierter Aufbau von FTTx-Netzen. In *Breitbandversorgung in Deutschland*, volume 220 of *ITG-Fachbericht*. VDE-Verlag, March 2010.

[14] S. Orlowski, A. Werner, R. Wessäly, K. Eckel, J. Seibel, E. Patzak, H. Louchet, and W. Bentz. Schätze heben bei der Planung von FTTx-Netzen: optimierte Nutzung von existierenden Leerrohren – eine Praxisstudie. In *Breitbandversorgung in Deutschland*, volume 227 of *ITG-Fachbericht*. VDE-Verlag, March 2011.

[15] S. Raghavan and D. Stanojević. A note on search by objective relaxation. In *Telecommunications Planning: Innovations in Pricing, Network Design and Management*, volume 33 of *Operations Research/Computer Science Interfaces Series*, pages 181–201. Springer US, 2006.

[16] F. S. Salman. *Selected Problems in Network Design: Exact and Approximate Solution Methods*. PhD thesis, Carnegie Mellon University, Pittsburgh, 2000.

[17] F. S. Salman, R. Ravi, and J. N. Hooker. Solving the Capacitated Local Access Network Design Problem. *INFORMS Journal on Computing*, 20(2):243–254, 2008.