Submitted to *INFORMS Journal on Computing*
manuscript (Please, provide the mansucript number!)

Exact Approaches for Designing Multi-Facility Buy-at-Bulk Networks

Ashwin Arulselvan
Department of Management Science, University of Strathclyde, ashwin.arulselvan@strath.ac.uk

Mohsen Rezapour
Department of Computing Science, University of Alberta, Canada, rezapour@ualberta.ca

Wolfgang A. Welz
Institute for Mathematics, TU Berlin, Germany, welz@math.tu-berlin.de

We study a problem that integrates buy-at-bulk network design into the classical facility location problem. We consider a generalization of the facility location problem where multiple clients may share a capacitated network to connect to open facilities instead of requiring direct links. In this problem, we wish to open facilities, build a routing network by installing access cables of different costs and capacities, and route every client demand to an open facility.

We provide a path based formulation and we compare it with the natural compact formulation for this problem. We then design an exact branch-price-and-cut algorithm for solving the path based formulation. We study the effect of two families of valid inequalities. In addition to this, we present three different types of primal heuristics and employ a hybrid approach to effectively combine these heuristics in order to improve the primal bounds. We finally report the results of our approach that were tested on a set of real world instances as well as two sets of benchmark instances and evaluate the effects of our valid inequalities and primal heuristics.

Key words: Facility Location, Buy-at-Bulk, Branch-Price-and-Cut Algorithm, Optical Access Networks.

1. Introduction

The problem under consideration integrates *buy-at-bulk network design* into the classical *uncapacitated facility location* problem. In the facility location problem, we want to decide which facilities to

open and how to assign clients to these open facilities so that the sum of the facility opening costs and client connection costs is minimized. This problem does not involve decisions concerning the routing of the clients' demands to the open facilities; once we decided on the set of open facilities, each client is served by the closest open facility. In the (single-sink) buy-at-bulk network design problem, on the other hand, we want to design and dimension a minimum cost routing network providing sufficient capacities to route all clients' demands to their sink. This problem involves deciding on the routing of each client's demand. But, in contrast to the facility location problem, the (single) destination of demands is predetermined. In many modern day applications, however, all these decisions are interdependent and affect each other. Hence, they should be considered simultaneously. In the planning of telecommunication networks, for example, this corresponds to locating routing and switching devices (facilities) and dimensioning access cables (e.g. fiber-optic cables) that are used to route the (un-splittable) traffic from clients to facilities. Such a multi-sink network design with facility location problem can be modeled as follows: The set of clients and potential facility locations together with the possible connections is represented by a network. On the edges of this network we may install multiple copies of an access cable of any capacity. The task now is to (1) open facilities at a subset of the potential locations and (2) determine the combination of cable types to be installed on the edges of the network, so that we have sufficient capacity to simultaneously route the entire demand of each client to an open facility via a single path through the network of cables. We will refer to the problem of finding such a solution with minimum total cost as the *Multi-Facility Buy-at-Bulk* network design problem, denoted by MFBB. Such a problem also has applications in transportation logistics (Ravi and Sinha 2006) where one has to locate manufacturing facilities, and select trucks of different capacities shipping goods to the clients so that the entire demand of each client is shipped by the same truck (un-splittable).

1.1. Problem definition

The MFBB problem can be formalized as follows: we are given an undirected graph $G = (V, E)$ with non-negative edge lengths $c_e \in \mathbb{R}_{\geq 0}$, $e \in E$; a set $F \subseteq V$ of facilities with cost $\mu_i \in \mathbb{R}_{\geq 0}$, $i \in F$; and a

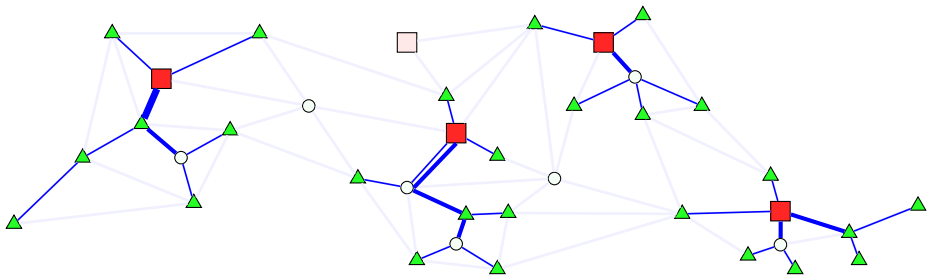

Figure 1 **A feasible solution to MFBB, where square nodes (in red) represent (open) facilities, triangle nodes represent clients, circle nodes represent intermediate nodes, and bold edges (of different thickness) represent access cables (of different capacities) connecting clients to facilities.**

set of clients $D \subseteq V$ with demands $d_j \in \mathbb{Z}_{>0}$, $j \in D$. We are also given K types of *access cables* that may be used to connect clients to open facilities. A cable of type k has capacity $u_k \in \mathbb{Z}_{>0}$ and cost (per unit length) $\sigma_k \in \mathbb{Z}_{\geq 0}$. We assume that the cable types obey economies of scale, i.e., we have $\sigma_1 < ... < \sigma_K$ and $\frac{\sigma_1}{u_1} > ... > \frac{\sigma_K}{u_K}$. This implies that the cost per unit capacity decreases from small to large cables. The task is to open a subset $\bar{F} \subseteq F$ of facilities and construct an *access network* whose edges use a combination of multiple cables of varying capacities, so that we can route the demands from the clients to the facilities. The entire demand of each client j must be routed via the access network to an open facility using exactly one path, denoted by P_j, and the sum of the capacities of cables installed on each edge must be capable of supporting all the demand flowing through that edge. Note that we allow the demand routed along a single edge to use different access cables, but the collection of edges traversed must be a path in G; see Figure 1. The objective is to minimize the total cost of opening facilities and installing access cables, where the cost of installing a single access cable of type i on edge e is $\sigma_i c_e$. This problem includes the classical facility location problem as a special case and is therefore NP-hard. The problem definition can be summarized as follows.

PROBLEM 1. **Multi-Facility Buy-at-Bulk (MFBB)**

Input:

- undirected graph $G = (V, E)$; edge lengths $c_e \in \mathbb{Z}_{\geq 0}$, $e \in E$
- potential facilities $F \subseteq V$ with opening costs $\mu_i \in \mathbb{Z}_{\geq 0}$, $i \in F$
- clients $D \subseteq V$ with demands $d_j \in \mathbb{Z}_{>0}$, $j \in D$

- access cable types K with

  —capacity $u_k \in \mathbb{Z}_{>0}$, $k \in K$

  —setup cost (per unit length) $\sigma_k \in \mathbb{Z}_{\geq 0}$, $k \in K$

  $\sigma_1 < ... < \sigma_K$ and $\frac{\sigma_1}{u_1} > ... > \frac{\sigma_K}{u_K}$

Solution:

- set of open facilities $\bar{F} \subseteq F$
- access network $\bar{A} \subseteq E$ through which each client $j \in D$ is connected to a facility $i_j \in \bar{F}$ and that the connection from client j to its facility i_j induces a path P_j.
- access cable installation $\omega : \bar{A} \times K \to \mathbb{Z}_{\geq 0}$ of sufficient capacity, i.e., $\sum_{j:\ e \in P_j} d_j \leq \sum_k u_k \omega_{e,k}$

Goal:

$$\min \sum_{i \in \bar{F}} \mu_i + \sum_{e \in A^*} \sum_{k \in K} \sigma_k c_e \omega_{e,k}$$

1.2. Related work

The combination of the facility location and the buy-at-bulk network problem has been considered for the first time in Meyerson et al. (2000). They show that this combination can be seen as a special case of the *single-sink non-uniform buy-at-bulk network design* (SSNBB) problem (a.k.a. the *Cost-Distance* problem): Each instance of MFBB can be transformed to an instance of SSNBB by adding an artificial sink (say r) and connecting it to every facility $i \in F$ where the cost of each (dummy) edge ir is set to be μ_i. They hence provide the first O(log(|D|)) approximating algorithm for MFBB using this transformation. Later, Ravi and Sinha (2006) developed an $O(K)$ approximation for this problem and called it *integrated logistics*. It is worth noting that there is a closely related variant of MFBB in the (computer science) literature, namely the facility location with deep-discount edge costs problem, where each cable type, instead of having a fixed cost and a fixed capacity, has unlimited capacity but a flow-dependent variable cost in addition to its fixed cost. This, for example, occurs in the planning of water and energy supply networks where the consideration of different connection types on the edges of the access network is not motivated by the different capacities but by the different shipping cost (per unit) of alternative technologies

or operational modes. One can transform between buy-at-bulk and deep-discount variants of the problem with factor 2 loss (see Friggstad et al. (2015)). Friggstad et al. (2015) proved an upper bound of O(K) on the integrality gap of a natural flow-based linear programming formulation of the problem. To the best of our knowledge there is still no $O(1)$ approximation or an exact algorithm for the MFBB problem in the literature.

The *Single-Sink (uniform) Buy-at-Bulk problem* (SSBB) can be seen as another simplification of MFBB, in which we are given exactly one single facility, referred to as a sink, and a solution must connect all clients to this sink. This problem has been widely studied in *operations research* as well as *computer science* communities. It should be noted that there are two variants of the SSBB problem in the literature, namely splittable SSBB (s-SSBB) and unsplittable SSBB (u-SSBB), depending on whether the demand of each client is allowed to be routed along several paths or not. We remark that the u-SSBB problem is a special case of our problem.

Several approximation algorithms for this problem have been proposed in the computer science literature. The problem has first been studied from the perspective of approximations in Salman et al. (2001). For the unsplittable case, Garg et al. (2001) developed an $O(K)$ approximation, using LP (Linear Programming) rounding techniques. Hassin et al. (2004) provide the first constant factor approximation for the single cable version of the problem. The first constant factor approximation for the problem with multiple cable types is due to Guha et al. (2009). Talwar (2002) showed that an LP formulation of this problem has a constant integrality gap and provided a 216 approximation algorithm. Using sampling techniques, this factor was reduced to 145.6 by Jothi and Raghavachari (2004), and later to 40.82 by Grandoni and Rothvoß (2010). For the splittable case, Gupta et al. (2003) presented a simple 76.8-approximation algorithm using random-sampling techniques. Unlike the algorithms mentioned above, their algorithm does not guarantee that the solution is a tree. Gupta's algorithm was modified to improve the approximation guarantee for u-SSBB to 65.49 (see Jothi and Raghavachari (2004)) and later to 24.92 (see Grandoni and Rothvoß (2010)).

The SSBB problem is also well studied in the operation research literature. However, in the operation research literature, it is mostly known as *single-source network loading problem* (e.g.

(Ljubić et al. 2012)) or (in the case of telecommunication network planning) as *Local Access Network Design Problem (LAN)* (e.g. (Salman et al. 2008)).

Randazzo et al. (2001) considered the LAN problem with only two cable types under the assumption that the solution must be a tree (and therefore the flows are unsplittable). They provided a multicommodity flow formulation for the problem and solved it by applying Benders' decomposition. Salman et al. (2008) considered the LAN problem with multiple cable types where the cable types obey economies of scale. They applied a flow-based MIP (Mixed Integer Programming) formulation and worked with the relaxation obtained by approximating the step cost function on the capacities by a lower convex envelope to provide a special branch-and-bound algorithm for LAN design. Raghavan and Stanojević (2006) later reformulated this as a stylized branch-and-bound algorithm. Working with the approximate step cost function, as defined by Salman et al. (2008), Ljubić et al. (2012) considered a stronger multicommodity flow formulation for the problem by disaggregating the commodities, and applied a branch-and-cut algorithm based on Benders decomposition for solving the problem.

1.3. Contribution and organization

In this work, we undertook the first computational study for the multi-facility buy-at-bulk network design problem, which so far has been only addressed from the perspective of designing approximation algorithms. We justified this exact approach, since our literature review identified a lack of progress made by approximation algorithms in terms of theoretical guarantees. We modeled this problem as a path based formulation and compared it with a natural compact formulation of the problem. We then provided a branch-price-and-cut algorithm to solve the path based formulation.

We also studied two classes of valid inequalities to improve the lower bounds. In addition to this, we studied three different types of primal heuristics and employed a hybrid approach to effectively make use of these heuristics in order to improve the primal bounds.

The paper is organized as follows: Compact and exponential-sized integer programming formulations are presented in Section 2. Two families of valid inequalities are proposed in Section 3. The details of proposed branch-price-and-cut algorithm is described in Section 4. Finally, computational results are reported in Section 5 and conclusions are made in Section 6.

2. Integer programming formulations

In this section we first propose a natural compact IP (Integer Programming) formulation for the MFBB problem. Then, relying on (approximate) step cost functions that can be precomputed for each edge using dynamic programming, we propose some new IP formulations for the problem.

2.1. A natural IP formulation

We write a natural flow-based integer linear program for the MFBB problem. For each edge we create a pair of anti-parallel directed arcs, with same length as the original one. Let $\vec{E}$ be the set of these arcs. We denote the undirected version of an arc $\bar{e} \in \vec{E}$ by e. For an edge $e = lm$ between nodes l and m, we will use a notation (l, m) or (m, l) to explicitly specify the orientation of the corresponding arc $\bar{e} \in \vec{E}$. We will use the notation e and $\bar{e}$, when it is clear from the context, for the sake of compactness. In our model we will use the following variables: For every $\bar{e} \in \vec{E}$ and client $j \in D$, the binary variable $f_{\bar{e}}^j$ indicates if flow from client j uses arc $\bar{e}$; for $e \in E$ and access cable type k, x_e^k indicates the number of access cables of type k installed on edge e; and z_i indicates if facility i is open or not. We use the notation $\delta^+(u) = \{(u, v) \in \vec{E}\}$ and $\delta^-(v) = \{(u, v) \in \vec{E}\}$.

$$\text{(IP-1)} \quad \min \sum_{i \in F} \mu_i z_i + \sum_{e \in E} c_e \sum_{k=1}^{K} \sigma_k x_e^k$$

$$\sum_{\bar{e} \in \delta^+(j)} f_{\bar{e}}^j = 1 \qquad \forall j \in D \tag{1}$$

$$\sum_{\bar{e} \in \delta^+(v)} f_{\bar{e}}^j = \sum_{\bar{e} \in \delta^-(v)} f_{\bar{e}}^j \qquad \forall j \in D, v \in V \backslash F, v \neq j \tag{2}$$

$$\sum_{\bar{e} \in \delta^-(i)} f_{\bar{e}}^j - \sum_{\bar{e} \in \delta^+(i)} f_{\bar{e}}^j \leq z_i \qquad \forall j \in D, i \in F \tag{3}$$

$$\sum_{j \in D} d_j (f_{(l,m)}^j + f_{(m,l)}^j) \leq \sum_{k=1}^{K} u_k x_{lm}^k \qquad \forall\ lm \in E \tag{4}$$

$$x_e^k \ \text{ non-negative integers} \qquad \forall e \in E, k \in K$$

$$f_{\bar{e}}^j \in \{0, 1\} \qquad \forall \bar{e} \in \vec{E}, j \in D$$

$$z_i \in \{0, 1\} \qquad \forall i \in F$$

Constraints (1) impose that at least one unit of flow leaves each client. Constraints (2) are flow conservation constraints at non-facility nodes. Constraints (3) state that the flow only terminates at open facilities. Constraints (4) ensure that we install sufficient capacity to support the flow.

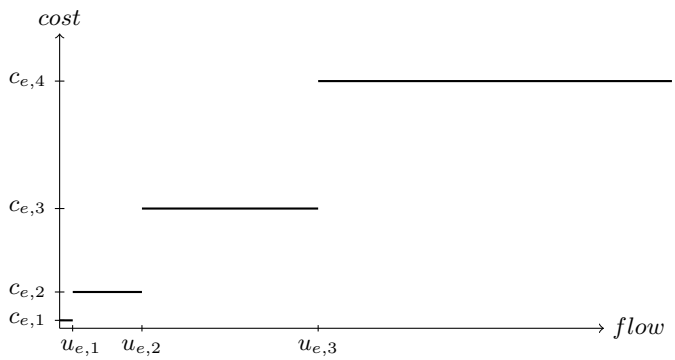

Figure 2 **An illustration for the cost function g_e**

It is not hard to show that an optimal fractional solution for the LP relaxation of IP-1 only uses the last cable type with the lowest cost per capacity ratio. Consider a simple complete graph containing only a single client with unit demand and a single facility with zero cost. Consider two types of access cables: $\sigma_1 = 1,\ u_1 = 1;\ \sigma_2 = 2,\ u_2 = L$. It is obvious that the cost of the optimal integral solution for such an instance is 1, while the cost of the optimal fractional solution is $\frac{2}{L}$. Hence, we have the following remark.

REMARK 1. The integrality gap of (IP-1) can be arbitrarily large.

2.2. An alternative IP modeling of MFBB

In this section, following the previous work for the single-sink buy-at-bulk network design problem (e.g., see Salman et al. (2008)), we first provide a slightly better IP formulation, namely (IP-2), for the problem. We then propose our main formulation which is a path based formulation of the problem with an exponential number of variables.

Notice that if the locations of the open facilities as well as the subgraph (edges supporting the access network) connecting clients to open facilities are given, then the problem reduces to the *integer minimum knapsack* problem for each edge, wherein one needs to choose the optimal combination of the cables for that edge in order to support the demand flowing through it. We compute the optimal combination of cable types for all possible flow levels on every edge (using dynamic programming). This provides a monotonically increasing step cost function for each edge e in the network, which we denote by g_e.

Next, we consider each piece of the resulting step cost function, for each edge, as a module with a specified cost and a specified capacity available for that edge; see Figure 2. More precisely, we assume that for each edge e a set $\mathcal{N}_e = \{n_1, n_2, \ldots, n_{\mathcal{N}_e}\}$ of modules (obtained by finding the optimal combination of cable types for all flow levels on e) is given, and at most one of these modules can be installed to support the corresponding flow along that edge. Each module n has a cost of $c_{e,n}$ and a capacity of $u_{e,n}$. Finally, to model this, we introduce, for each edge e and each module $n \in \mathcal{N}_e$, a variable $x_{e,n}$ which indicates whether module n has been installed on edge e or not. Intuitively speaking, this indicates whether piece n of the step cost function determines the optimal cable cost for edge e. Now, we reformulate the problem by replacing constraints (4) by constraints (5) and (6) as follows.

$$\text{(IP-2)} \quad \min \sum_{i \in F} \mu_i z_i + \sum_{e \in E} \sum_{n \in \mathcal{N}_e} c_{e,n} x_{e,n}$$

$$(1) - (3)$$

$$\sum_{j \in D} d_j (f^j_{(l,m)} + f^j_{(m,l)}) \leq \sum_{n \in \mathcal{N}_{lm}} u_{lm,n} x_{lm,n} \qquad \forall lm \in E \tag{5}$$

$$\sum_{n \in \mathcal{N}_{lm}} x_{lm,n} \leq 1 \qquad \forall lm \in E \tag{6}$$

$$x_{e,n} \quad \text{non-negative integers} \qquad \forall e \in E, n \in \mathcal{N}_e$$

$$f^j_{\bar{e}} \in \{0,1\} \qquad \forall \bar{e} \in \vec{E}, j \in D$$

$$z_i \in \{0,1\} \qquad \forall i \in F$$

We denote by $\text{proj}_{f,z}(P) = \{f, z \in [0,1]^{|\vec{E}| \times |D| \times |F|} | \exists (x, f, z) \in P\}$ as the projection of some polyhedron P on the space of f and z variables. Let P_1 and P_2 denote the feasible space of the LP relaxation corresponding to formulations IP-1 and IP-2, respectively. It is not hard to show that $\text{proj}_{f,z}(P_2) \subseteq \text{proj}_{f,z}(P_1)$. This in fact implies the following result.

LEMMA 1. *IP-2 is at least as strong as IP-1 in terms of the lower bounds provided by their relaxations.*

The IP-2 formulation has $O(|D| \cdot |E|)$ variables and $O(|E|)$ constraints which may lead to quite large IP formulations with respect to the size of real-world applications; see Section 5. To get around

solving such a model with huge number of variables, we will propose a path based formulation for the problem and solve it using column generation.

2.2.1. Path based formulation We present a path based formulation for the problem with an exponential number of variables. For the sake of modeling paths, we first create a dummy root node r and connect all facilities with the root node. Let $\vec{E}' = \vec{E} \cup \left\{\bigcup_{i \in F}(i,r)\right\}$. Let $P(j)$ denote the set of all possible paths in $G' = (V \cup \{r\}, \vec{E}')$ starting from j and terminating at r.

Remember that the demand of each client must be routed to an open facility, and so to the root node, via a single path. For each $j \in D$ and for each $p \in P(j)$, we introduce a binary variable y_p which indicates if flow from j is routed along p. Then the problem can be formulated as follows:

$$\text{(IP-3)} \quad \min \sum_{i \in F} \mu_i z_i + \sum_{e \in E} \sum_{n \in \mathcal{N}_e} c_{e,n} \cdot x_{e,n}$$

$$\sum_{p \in P(j)} y_p = 1, \qquad \forall j \in D \tag{7}$$

$$\sum_{j \in D} \sum_{\substack{p \in P(j): \\ \{(l,m),(m,l)\} \cap p \neq \emptyset}} d_j y_p \leq \sum_{n \in \mathcal{N}_{lm}} u_{lm,n} x_{lm,n}, \qquad \forall lm \in E \tag{8}$$

$$\sum_{n \in \mathcal{N}_{lm}} x_{lm,n} \leq 1, \qquad \forall lm \in E \tag{9}$$

$$\sum_{p \in P(j):(i,r) \in p} y_p \leq z_i, \qquad \forall j \in D, i \in F \tag{10}$$

$$x_{e,n} \quad \text{non-negative integers} \qquad \forall e \in E, n \in \mathcal{N}_e$$

$$y_p \in \{0,1\} \qquad \forall j \in D, p \in P(j)$$

$$z_i \in \{0,1\} \qquad \forall i \in F$$

Constraints (7) force each client to be connected to a routing path. Constraints (8) ensure that we install sufficient capacity to support the flow along routing paths, constraints (9) guarantee that at most one module is installed along each edge and constraints (10) ensure that a facility is open.

It is worth noting that one could improve the lower bound provided by the linear relaxation of IP-3 by adding the following set of strengthening inequalities to the model:

$$\sum_{\substack{p \in P(j): \\ \{(l,m),(m,l)\} \cap p \neq \emptyset}} y_p \leq \sum_{n \in \mathcal{N}_{lm}} x_{lm,n}, \qquad \forall lm \in E, j \in D \tag{11}$$

This guarantees that there always is some module installed along any edge used by routing paths. However, adding the above set of valid inequalities leads to a model with $O(|D| \cdot |E|)$ constraints, increasing the number of constraints by a factor of $|D|$. Therefore, instead of adding them directly to the model, we will propose a family of strong valid inequalities (see Section 3.1) for IP-3 which contains Inequalities (11) as special cases.

3. Valid inequalities

We propose two families of valid inequalities that naturally emerge from the IP-3 model.

3.1. Cover Inequalities

In order to derive the cover inequalities corresponding to each constraint in set (8), we obtain a knapsack structure by complementing the x variables (replacing x by $1-x$) in the constraint. Consider the constraint in set (8) corresponding to edge $lm \in E$. It should be noted that these are sometimes referred to as strong linking constraints that link the path variables with the module variables (see Frangioni and Gendron (2012)). Let $U_{lm} = \sum_{n \in \mathcal{N}_{lm}} u_{lm,n}$. We define $\theta_{lm} = (D_\theta, M_\theta)$ to be a cover with respect to edge lm, where $D_\theta \subseteq D$ and $M_\theta \subseteq \mathcal{N}_{lm}$, if

$$\sum_{j \in D_\theta} d_j + \sum_{n \in M_\theta} u_{lm,n} > U_{lm}\,.$$

We say that a cover is minimal when removing any item either from D_θ or M_θ results in a set for which the above inequality does not hold. It is not hard to show that if θ_{lm} is a minimal cover, then the following inequalities are valid:

$$\sum_{j \in D_\theta} \sum_{\substack{p \in P(j): \\ \{(l,m),(m,l)\} \cap p \neq \emptyset}} y_p + \sum_{n \in M_\theta} (1 - x_{lm,n}) \leq |M_\theta| + |D_\theta| - 1 \Longleftrightarrow$$

$$\sum_{j \in D_\theta} \sum_{\substack{p \in P(j): \\ \{(l,m),(m,l)\} \cap p \neq \emptyset}} y_p \leq \sum_{n \in M_\theta} x_{lm,n} + |D_\theta| - 1 \tag{12}$$

Note that Ineq. (11) are dominated by Ineq. (12) with D_θ containing only a single client j.

Let (x^*, y^*, z^*) be the optimal fractional solution of the LP relaxation of model IP-3. Now, we present how to find a cover inequality corresponding to edge lm violated by (x^*, y^*, z^*). For each $j \in D$, we let

$$w_j^* = \sum_{\substack{p \in P(j): \\ \{(l,m),(m,l)\} \cap p \neq \emptyset}} y_p^* \tag{13}$$

And let $F_{lm} \subseteq \mathcal{N}_{lm}$ be the set of modules for the edge lm such that $x^*_{lm,n} > 0$. A most violated cover inequality is obtained by solving the following knapsack problem:

$$\min\ \gamma = \sum_{n \in F_{lm}} x^*_{lm,n} x_{lm,n} + \sum_{j \in D} (1 - w^*_j) w_j \tag{14}$$

$$\sum_{j \in D} d_j w_j + \sum_{n \in F_{lm}} u_{lm,n} x_{lm,n} \geq \sum_{n \in F_{lm}} u_{lm,n} + 1$$

$$x_{lm,n} \in \{0,1\}, \quad \forall n \in F_{lm}$$

$$w_j \in \{0,1\}, \quad \forall j \in D$$

which is equivalent to the following standard knapsack problem in maximization form by replacing x by $1 - \bar{x}$, and w by $1 - \bar{w}$.

$$\sum_{n \in F_{lm}} x^*_{lm,n} + \sum_{j \in D} (1 - w^*_j) - \max \sum_{n \in F_{lm}} x^*_{lm,n} \bar{x}_{lm,n} + \sum_{j \in D} (1 - w^*_j) \bar{w}_j$$

$$\sum_{j \in D} d_j \bar{w}_j + \sum_{n \in F_{lm}} u_{lm,n} \bar{x}_{lm,n} \leq \sum_{j \in D} d_j - 1$$

$$\bar{x}_{lm,n} \in \{0,1\}, \quad \forall n \in F_{lm}$$

$$\bar{w}_j \in \{0,1\}, \quad \forall j \in D$$

Let $D' \subseteq D$ and $F'_{lm} \subseteq F_{lm}$ be the optimal subsets that we obtained by solving the minimizing knapsack problem and let γ be the corresponding objective value. Observe that $(D', F'_{lm} \cup \{N_{lm} \backslash F_{lm}\})$ is a minimal cover with respect to edge lm. Note that $\gamma < 1$ implies the following

$$\sum_{j \in D'} \sum_{p \in P(j):(l,m) \in p} y^*_p > \sum_{n \in F'_{lm} \cup \{N_{lm} \backslash F_{lm}\}} x^*_{lm,n} + |D'| - 1$$

using (13), (14), and the definition of F_{lm}. Hence, we conclude that (x^*, y^*, z^*) violates (valid) Inequality (15) if $\gamma < 1$.

$$\sum_{j \in D'} \sum_{p \in P(j):(l,m) \in p} y_p \leq \sum_{n \in F'_{lm} \cup \{N_{lm} \backslash F_{lm}\}} x_{lm,n} + |D'| - 1 \tag{15}$$

3.2. Cut inequalities

The modules generated follow economies of scale and hence the optimal solution of the LP relaxation ends up fractionally picking the last module that has the lowest cost per capacity ratio. In this section, similar to previous work for capacitated network design problem (e.g., see Atamtürk and

Rajan (2002), Raack et al. (2011)), we will introduce a set of valid inequalities, called *cut inequalities*, to somewhat remedy this problem. Given a fractional optimal solution (x^*, y^*, z^*) of IP-3. For the graph $\bar{G} = (V \cup \{r\}, E \cup \{\bigcup_{i \in F} ir\})$, we take the edge capacities to be $u_{lm} = \sum_{n \in \mathcal{N}_{lm}} x^*_{lm,n}$, for all $lm \in E$, and z^*_i for all ir edges. For every $j \in D$, we solve the maximum flow problem with source j and sink r. If the flow value is less than one, then we obtain the following violated cut:

$$\sum_{lm:k \in \bar{S}} \sum_{n \in \mathcal{N}^j_{lm}} x_{lm,n} + \sum_{i \in \bar{S}} z_i \geq 1 \tag{16}$$

where $\bar{S}$ (containing j; not r) indicates the corresponding minimum cut set, and $N^j_{lm} = \{n \in \mathcal{N}_{lm} : u_{lm,n} \geq d_j\}$ indicates the modules available for edge lm with capacities greater than the demand of the client j. The validity of the cut follows from the fact that every client j needs to be connected to some facility along a path with every edge in the path having at least one module, with capacity greater than the demand d_j, installed.

4. Solution procedure

For the basics of column generation, we refer to Dantzig and Wolfe (1960) and (Gamrath 2010, Chapter 4). Since the path based formulation presented above contains an exponential number of variables, our solution procedure is based on the column generation technique. We consider as the restricted master problem the continuous relaxation of the IP-3 model including all the constraints and the x and z variables, but only the y variables corresponding to a subset $P'(j) \subseteq P(j)$ of paths for each $j \in D$.

4.1. Initialization

We enrich the restricted master problem with solutions obtained from a few runs of a randomized greedy algorithm. The description is provided in Algorithm 1 and it works as the following. First, we pick a random permutation $\Pi = (j_1, j_2, ..., j_{|D|})$ of clients. We then construct a solution in a greedy fashion: we start with an empty network, i.e. no modules installed and no facilities opened. In each step of the algorithm, we pick a client j_t (according to the picked permutation) and route its entire demand to some facility via a routing path which requires the minimum total cost of capacity installations plus the facility opening cost over the network constructed so far. We continue

Algorithm 1 `GreedyAlgorithm`

1: $\bar{I} \leftarrow \emptyset$; $\bar{P} \leftarrow \emptyset$.

2: Pick a random permutation of clients in D.

Let $\Pi = (j_1, j_2, ..., j_{|D|})$ be the picked permutation.

3: **for** $t = 1, 2, \cdots, |D|$ **do**

4: Find the cheapest cost routing path p_t as described above.

5: Let i_t be the facility with $(i_t, r) \in p_t$. Open facility i_t, if it has not been opened yet.

6: Route d_{j_t} units of demand from j_t to facility i_t via path p_t; update $\bar{f}_{\bar{e}}$ accordingly for $\bar{e} \in \vec{E}$.

7: $\bar{I} \leftarrow \bar{I} \cup \{i_t\}$; $\bar{P} \leftarrow \bar{P} \cup \{p_t\}$.

8: **end for**

this process for all clients. Let $\bar{P}$ be the set of routing paths returned by the algorithm. In what follows we explain how such a routing path can be obtained at each stage t. Recall the increasing step cost function g_e (see Section 2.2). Let $\bar{f}_{\bar{e}}$ be the amount of flow which has been routed along each arc $\bar{e}$ and let $\bar{I}$ be the set of facilities opened so far. We construct a weighted graph $G_t = \left(V \cup \{r\}, \vec{E} \cup \left\{\bigcup_{i \in F}(i, r)\right\}\right)$ as follows: set the weight of each arc $\bar{e} \in \vec{E}$ to be $g_e(\bar{f}_{\bar{e}} + d_{j_t}) - g_e(\bar{f}_{\bar{e}})$ (this is, the marginal increase in the cost due to transporting additional d_{j_t} units of demand along that edge); and set the weight of each arc (i, r) to be μ_i if $i \notin \bar{I}$ and zero otherwise. Now the routing path from j_t to r, namely p_t, can be obtained by computing a (j_t, r) shortest path in graph G_t.

It should be noticed that such shortest-path based approaches (with some modifications) can be employed in order to design algorithms with a (logarithmic) approximation guarantee for the problem; we refer the readers to Charikar and Karagiozova (2005) for more details.

4.2. Column generation

We iteratively solve the restricted master problem and search for new columns having negative reduced cost that is computed using the optimal dual solution. Let the dual variables corresponding to constraints (7) be ρ_j, for all $j \in D$. We will refer to the dual variables corresponding to constraints (8) with the notation as π_{lm}, for all $lm \in E$ and the dual variables corresponding to

constraints (10) by γ_i^j, for all $i \in F, j \in D$. For each j, we determine if a path p in $P(j) \setminus P'(j)$ could improve the current (fractional) solution. The pricing problem associated with client j is:

$$\min_{p \in P(j)} \quad -\left(\rho_j + \sum_{\substack{p \in P(j): \\ \{(l,m),(m,l)\} \cap p \neq \emptyset, \\ l \neq r}} d_j \pi_{lm} + \sum_{i \in F} \mathbb{I}_i^p \gamma_i^j \right)$$

where $\mathbb{I}_i^p$ is an indicator variable denoting whether edge ir is in the path p or not (r being the root node). Given a graph $\bar{G} = (V \cup \{r\}, E \cup \{\bigcup_{i \in F} ir\})$, we take the weight of an edge lm to be $-d_j \pi_{lm}$, for all $lm \in E$ and weights $-\gamma_i^j$, for all ir edges, $i \in F$. We now find the shortest path in $\bar{G}$ from j to the root node r. Note that the dual vectors $\boldsymbol{\pi}, \boldsymbol{\gamma} \leq 0$ and so Dijkstra's algorithm can be used to find the shortest path. If the solution to this shortest path problem has length less than ρ_j, then the solution is not optimal for the master problem and the variable corresponding to this path should be added into our restricted master problem. The new restricted master problem is re-solved and the process is iterated as long as the pricing problems corresponding to the clients generate new columns.

We notice that the feasible solutions space of the restricted master problem may be empty during the loop mentioned above, due to branching constraints (see Section 4.4) or in the beginning when no columns have been generated yet. In this case, we use Farkas' Lemma to add columns that gradually move the solutions space closer to the feasible region. Note that this is the same problem as the pricing problem considering the so called dual Farkas values. This method has been called *Farkas pricing*, and provided in Achterberg (2009) within the SCIP framework (see Section 5).

4.3. Cut generation

Once the column generation is over, we start searching for the cover inequalities violated by the current fractional solution. We search for such cuts as described in Sections 3.1 and 3.2. Cuts are only added in the root node, while the computationally less expensive cut inequalities are added with higher priority. For both types of cuts a limit on the number of cuts has been introduced after which the relaxation is solved again before adding further cuts. To avoid that using this approach cuts are only generated for the first variables, the generation is then performed in a round-robin

fashion, starting with those variables that were not considered in the last iteration. The generated cover cuts (15) will not change the structure of the pricing problem, however the weights associated with edges of the network may be changed. Hence the column generation process should be repeated considering the new pricing problems, once new cuts are added.

Each e has multiple cover inequalities associated with it. Let Θ_e be the set of covers associated with edge e. Let $\Theta = \bigcup_{e \in E} \Theta_{(e)}$. For a cover $\theta \in \Theta$, let D_θ be the set of clients involved in the cover and α_θ be the corresponding dual variable.

The new pricing problem associated with client j is:

$$\min_{p \in P(j)} \quad - \left(\rho_j + \sum_{\substack{p \in P(j): \\ \{(l,m),(m,l)\} \cap p \neq \emptyset, \\ l \neq r}} d_j \pi_{lm} + \sum_{i: i \in F} \mathbb{I}_i^p \gamma_i^j + \sum_{(l,m) \in p} \sum_{\substack{\theta \in \Theta_{lm}: \\ j \in D_\theta}} \alpha_\theta \right) \tag{17}$$

Note that cut inequalities (16) involving x and z variables improve the quality of the bound without affecting the pricing problem.

Such a loop is repeated until neither new columns nor cuts are added.

4.4. Branching Strategies

So far, we have described how we employ the column generation and cut separation methods for solving the master problem. However, the optimal solution to the master problem might not be integral at the end of the price-and-cut loop. Integer linear programs are typically solved by using *Branch-and-Bound*, a widely known technique, which uses branching to handle integrality. This technique, when used together with column generation and cut separation is called Branch-Price-and-Cut.

The facility and module variables, however, are not generated but only specify certain bounds for the paths variable. Thus, we first branch on the former variables. But even if all facilities and modules are integral this does not guarantee integral path variables. As branching on the generated variables is not efficient in the branch-and-price context, an alternative can be to implement standard branching in the space of the compact formulation by adding a constraint that give us lower and upper bounds on the capacity of an edge that currently has a fractional flow value. However, this branching decision destroys the pricing subproblem structure. To get around these issues,

similar to the one used by Barnhart et al. (2000) for multi-commodity flow, we impose implicit branching constraints as the following. Consider any client $j \in D$ whose demand is routed along more than one path, say two distinct paths p_1 and p_2, in the current (fractional) solution to the master problem. Note that paths have at least node j in common. Consider the first node at which these two paths split. We partition the set of edges emanating from this node into two subsets E_1 and E_2 such that E_1 (E_2, respectively) intersects p_1 (p_2, respectively). Then, we create two branches with one imposing $\sum_{p \in P(j): p \cap E_2 \neq \emptyset} y_p = 0$ and the other imposing $\sum_{p \in P(j): p \cap E_1 \neq \emptyset} y_p = 0$.

More precisely, when we are given a fractional solution, we create the next branch as follows:

1. Out of those clients whose demand is split, choose the $j \in D$ with the highest demand.

2. Identify the two paths p_1 and p_2 with the most fractional y_{p_1} and y_{p_2} of client j. (If the fractionalities are identical, we use the objective value of the corresponding variables as a tiebreaker.)

3. By traversing the path starting from the client node, identify the last common node d in both paths and then create two subsets from the outgoing edges. In order to generate balanced branches, the sets E_1 and E_2 are selected in such a way that $|E_1|$ and $|E_2|$ differ by at most one. For an efficient implementation it is further important to only add arcs to the set, that are not already forbidden in the current node of the branch-and-bound tree.

We remark that these branching decisions requires no changes in the basic structure of the pricing problem, which remains a simple shortest path problem through all the enumeration process. Our subproblems will work with the subgraph with the corresponding forbidden edges deleted.

4.5. Primal Heuristic

For the overall performance of a Branch-and-Price approach it is crucial that good primal solutions of the problem are found, however it is usually unlikely for the branch-and-bound process alone to find good upper bounds fast. We therefore present two simple heuristic algorithms in the following.

4.5.1. LP based greedy heuristic The following heuristic is invoked at each node of the (B&B) tree. Given the fractional optimal solution of the current node, resulting in a fractional $x_{e,n}$ and z_i variables, run algorithm `GreedyAlgorithm`, described in Section 4.1, while the weight

assignment to the arcs of the graph G' is based on the LP solution: For i^{th} client according to the picked permutation, assign a weight $\left(g_e(\bar{f}_{\bar{e}} + d_{j_i}) - g_e(\bar{f}_{\bar{e}})\right) \cdot \left(1 - \sum_{n \in \mathcal{N}_e} x_{e,n}\right)$ to each arc $\bar{e} \in \vec{E}$; set weights for (i, r) arcs to be $\mu_i \cdot (1 - z_i)$ if $i \notin \bar{I}$, otherwise 0; then route its demand to r via a routing path with the minimum total cost in G'.

4.5.2. IP based primal heuristic We treat the problem including all variables generated so far as a complete integer program and then solve that program using IBM ILOG CPLEX. The result gives us the best possible solution that can be achieved by only using the current paths from $P'(j)$ without adding any new variables. Since the number of variables is fixed, this problem is easier to solve and even if the solution process of the resulting IP is canceled after a certain amount of time, the best solution found is still a feasible solution to our problem. But as the IPs considered are still rather big, the solution process is very time consuming. In order to implement this idea as efficient as possible, the CPLEX-problem is solved in the background so that the main CPU-thread can still continue to perform the regular branch-and-bound process using SCIP. This way only some minor coordination and communication overhead needs to be performed within the actual SCIP-plugin, while the expensive solution process can be performed in other threads.

If CPLEX finds a new improving solution the main thread is signaled and the solution is then copied into SCIP so that it can be immediately used as an upper bound in the branch-and-bound process. After either CPLEX reaches a certain node limit or after too many new variables have been generated since the last start of the heuristic, the new variables are added and the solver is restarted. By adding variables to the existing CPLEX-problem we assure that solutions found in previous runs are still available and automatically used as a primal bound in the new computation. To improve the performance even further, we also add the valid inequalities (see Section 3) as CPLEX user-cuts. Those inequalities are problem dependent and thus improve the automatically generated cuts by CPLEX.

As the branch-and-price algorithm is single threaded, this approach allows us to perform this IP based heuristic on modern multi-core processors with basically no additional computation time.

5. Computational results

The Branch-Price-and-Cut approach has been implemented using the framework provided by SCIP (Achterberg 2009). In this context SCIP handles all the underlying Integer Programming specific aspects and has been extended with problem dependent plugins for the pricing and branching as well as the cut generation and primal heuristic. As explained in Section 4.2 the pricing problem corresponds to solving a shortest path problem for every client, which is solved via an implementation of Dijkstra's Algorithm.

To avoid that this computation is performed for all the clients in every iteration, we implemented a two step approach: In the first step one single-source shortest paths problem is solved to find lower bounds for the shortest paths to every clients. If we take a look at the arc costs corresponding to the problem for client j, we notice that only the dual variables corresponding to the constraints (10) and to the cover cuts depend on j. The ρ_j and the capacities d_j can be applied after the shortest paths have been calculated. However, different dual variables resulting from the constraints (10) are selected for different clients. Here the smallest of the corresponding values is used as an arc weight. This leads to the following optimization problem, where the shortest r-j-paths in the resulting graph represent a lower bound to the shortest paths in the actual pricing problem:

$$-\rho_j + d_j \left(\min_{p \in P(j)} \left(\sum_{\substack{(l,m) \in p \\ l \neq r}} -\pi_{lm} + \sum_{i : i \in F} \mathbb{I}_i^p \cdot \min_{j \in D} \frac{-\gamma_i^j}{d_j} \right) \right)$$

As the dual variables α_c are all not positive, this gives us a lower bound of the pricing problem (17). In the second step the graph with client dependent weights is then only solved for the clients j that had a non-negative path in the first step.

For the cover inequalities we use the solver provided as part of SCIP that uses dynamic programming to find an optimal Knapsack solution. The min-cut computations for the cut inequalities are performed using a push-relabel maximum flow algorithm.

5.1. Preprocessing

Some basic preprocessing techniques have been used to reduce the size of the network. As a first step, there are some very basic rules that can be used to identify edges that will never carry any

flow. This is for example the case for edges that cannot be reached by any facility or for edges adjacent to a node with a degree of one that is not a facility or a client.

This idea can also be extended for clients with only one adjacent edge. In this case we know that in any optimal solution this one edge will carry a flow of exactly the demand d of that client. It is therefore possible to relocate the client to the other end of the edge by adding a certain offset to the objective function which corresponds to the cost required to route d along the edge. This approach is especially useful if the clients are arranged in a star like fashion around one node which is a common scenario for certain network types. Since our problem is uncapacitated and all the modules follow economies of scale, it is now possible to aggregate the clients located in the same node into one single client corresponding to their total demand.

As our application represents real-world maps and networks, the graph G is usually sparse. And since there might also be certain bottlenecks in the network, the graph G often contains bridges. These bridges can then be used to identify upper and lower bounds of the demand routed across certain edges: As a first step we detect bridges using Tarjan's Bridge-finding algorithm (Tarjan 1974). Deleting a bridge separates the graph into two components; we now let d_{bridge} denote the total demand of all clients in the component corresponding to one side of the bridge. If this side does not contain any facilities, we know that a value of exactly d_{bridge} has to be routed across the bridge and that further a value of at most d_{bridge} will flow through any edge on that side. This information can then be used to eliminate unnecessary modules on these edges. If an upper bound of zero on an edge is detected the edge can even be entirely eliminated.

All the preprocessing rules mentioned above are used in our implementation. For sparse graphs they help to considerably reduce the number of edges and hence the number of x variables. This helps in significantly speeding up the pricing process and strengthening the LP bound as well.

5.2. Instances details

We used three different tests sets, namely RW, PA, and JMP instances.

- The *RW* (real-world) instances tested correspond to real world network planning problems (see `http://www.zib.de/projects/tools-planning-fttx-networks`). The networks were generated

from the publicly available information obtained through geographic information systems, arising from the German research project FTTx-Plan (see FTT (2014)).

Each instance correspond to a region in Germany and was constructed bearing in mind the potential client and facility locations. The street segments form the edges, while the street intersections and traffic circles provide the intermediate nodes; see Figure 3. The information about the different cable types along with their costs and capacities were provided by our industry partners.

- The *JMP* instances were created as in Johnson et al. (2000). They randomly distribute n nodes in a unit square. Two nodes within a distance of $\frac{2}{\sqrt{n}}$ are then connected by an edge. The edge costs were taken proportional to the distance between them. We modified the adaptation of this model carried out in Álvarez-Miranda et al. (2014) for a single commodity robust network design problem. 20% of the terminals were taken as facilities and the remaining were taken as clients. The largest demand is taken as an argument and demands were randomized between 0 and this maximum demand value. The cable types were taken to be the same as in RW instances.

- The *PA* instances were created based on the model designed by Barabási and Albert (1999) in order to create realistic networks. In this generator, nodes are iteratively added and an added node i gets connected to β, a parameter chosen as input, existing nodes. An existing node j gets connected to i with a probability inversely proportional to the degree of j. In Cacchiani et al. (2014), the authors adapted this model to solve a single commodity network design problem. We used this model to generate our instances. We give the maximum demand as an input and randomize our demands for all clients. We also take the cost of the facilities based on our real world instances (it is taken to be a constant for all facilities). The authors also reported the difficulty in solving JMP instances over the PA instances for their network design problem. In our experiments, we observed a similar behavior.

5.3. Computational experiments

All computations were performed on Intel Xeon E5-2630, 2.3 GHz CPUs using one thread for SCIP and three threads for the CPLEX- Heuristic. We used CPLEX 12.6.0 and SCIP version 3.1.0.

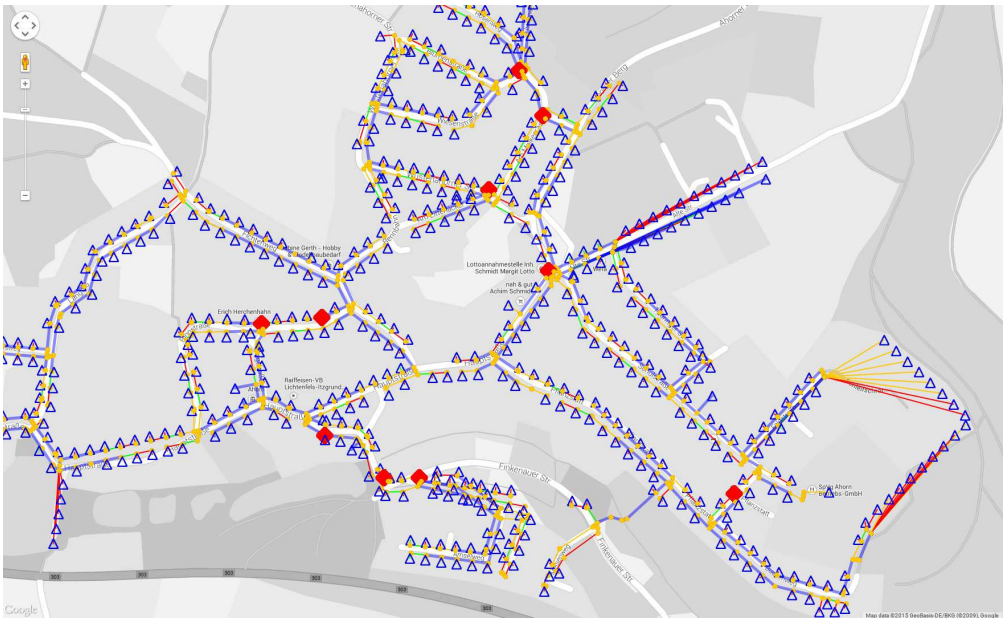

Figure 3 **Some part of a solution to instance "a" where squares, triangles and circles represent potential facilities, clients and nodes, respectively. This map has been created using the Google Maps API.**

Description of the tables: In Table 1, we report the performance of our branch-price-and-cut approach (which is based on the path based formulation) compared to that of a standard branch-and-bound (B&B) algorithm based on the compact formulation (IP-2) solved using CPLEX directly. Both approaches have been applied for the real-world instances and a run time of 36 000 s (ten hours). We then report the performance of our approach applied for the much smaller generated instances after a run time of 7200 s (two hours) in Tables 2 and 3. A gap limit of 2.0 % has been used for all the instances.

In (all) Tables 1, 2 and 3, we report the instance size as well as the number of path based variables and cuts that were added during the branch-price-and-cut process. In addition, we report the time (in seconds) that was spent solving the root node. We also report the gaps (in percent) at various stages of the process. The LP gap is the average percentage gap of the dual LP bound (before any cuts were added), while the root gap corresponds to the gap of the dual bound after all cuts were added. We report the final gap we obtained within the time limit. All of them have been calculated with respect to the best primal solution found during the entire process. The column labeled $|E'|$ in Table 1 denotes the number of edges after preprocessing. The *flow-vars* and *flow-gap*

columns of Table 1 show the number of flow based variables needed in the compact formulation of each (real-world) instance and report the final gap we obtained by running CPLEX using the default settings on one thread for ten hours. In Table 2 and 3 each row of the table represents the average results for 10 different random instances and we also give the number of instances that could be solved to 2.0 % within two hours. The average gaps are only provided over the instance which did not reach this limit. Note, that for the sake of clarity, we denote by "$*$" the final gap of those cases (in Tables 2 and 3) for which all 10 different instances were solved to 2.0 % before the time limit.

Experiments: The last columns of Table 1 show that our branch-price-and-cut algorithm together with implemented primal heuristics and problem specific valid inequalities can be used to solve very large real-world instances to roughly 20.0 %. Only instance "e" with over 12 000 edges and "c" with more than 7000 edges lead to worse gaps. The flow-gap column of Table 1, however, shows that the CPLEX solver by itself could not even solve the root LP for most of these instances (those with ∞ as the flow gap) within the time limit of ten hours. In particular, comparing the 'flow-vars' column with the 'path-vars' column of Table 1 shows the success of our approach in reducing the number of used variables. In fact we believe this is the main reason why our approach based on the path based formulation is doing much better than the one based on the compact formulation.

Tables 2 and 3 show the performance of our approach for slightly smaller data. Instances JMP (Table 3) turn out to be much more challenging (with respect to their sizes). However, we were able to find solutions which are guaranteed to be far less than 5 % away from the optimal solution in most of the cases.

In order to make the computational study comprehensive, we generated additional JMP and PA instances with different graph sizes, demand and fixed cost structures. The demands were drawn uniformly at random between (1 and 50), (50 and 100), (100 and 200) and (200 and 500) constituting the four demand structures. The three fixed cost structures were drawn uniformly at random between (500 and 1000), (1000 and 5000) and (5000 and 15,000). Demands less than

50 and facility costs around 5000 are the cost and demand structures that are closest to our real world instances. For each size and demand or cost structure, we generated 10 instances and the performance of the algorithm is shown through Figures 5 to 8 that gives the boxplots of their root and final gap. As we can see, the root gap of the PA instances are around 10% and the final gap is around 2% for node sizes up to 100. As the node size increases the final gap seems to be centered around 3 to 7%. We also can observe quite a bit of variance when we plot with respect to facility cost structure. The JMP instances have the root gap around 10% and final gap at around 2% for graphs with nodes up to 50 and it increases slightly with the graph size. The observations for JMP instances is fairly uniform across all instances. The average gap gradually increases with the instance size with low variance in the demands and the facility cost categories, as one would expect. The PA instances, however, have quite a bit of variance especially for the facility cost category. We believe the demand 500 group is an outlier for larger instances as the routing cost would dominate the low facility cost causing it to open more facilities. There is another intuitive explanation for this difference in behavior: Topologically these two graphs are quite different. For example, JMP instances are created by uniformly distributing the nodes in space, whereas PA instances are created with preferential attachment in mind. In networks of the latter type, a few nodes usually have high connectivity to the majority of the nodes. It is intuitive to open them all as facilities in a large network, especially when facility costs are cheaper compared to routing high demands.

Table 1: Results for the RW instances

ins.	$\|V\|$	$\|E\|$	$\|E'\|$	$\|F\|$	$\|D\|$	#flow-vars	flow-gap	#path-vars	#cuts	root-time	lp-gap	root-gap	final-gap
a	1,675	1,722	883	104	604	$2.08 \cdot 10^6$	27.2	12,079	5,089	864	57.6	18.9	18.2
b	4,110	4,350	2,300	230	1,670	$1.45 \cdot 10^7$	∞	23,418	13,692	7,472	74.8	23.7	23.3
c	6,750	7,262	3,992	531	2,440	$3.54 \cdot 10^7$	∞	33,211	7,165	36,000	75.0	32.7	32.7
d	4,227	4,482	2,384	319	1,490	$1.34 \cdot 10^7$	∞	31,261	10,865	36,060	64.0	20.5	20.5
e	11,544	12,350	6,699	890	4,275	$1.06 \cdot 10^8$	∞	43,478	3,759	36,000	80.5	53.0	53.0
f	637	758	459	101	39	59,124	12.7	50,739	1,749	266	53.1	19.5	16.1
g	3,055	3,177	671	49	591	$3.76 \cdot 10^6$	∞	2,976	2,134	61.9	34.3	12.1	10.7
h	2,271	1,419	887	498	349	$9.9 \cdot 10^5$	∞	32,081	2,325	32,700	56.2	21.5	21.3
i	1,315	1,422	759	148	238	$6.77 \cdot 10^5$	15.8	50,167	5,685	12,360	80.5	16.7	15.9

Table 2: Results of the branch-price-and-cut algorithm on PA instances

$\|V\|$	$\|E\|$	$\|F\|$	$\|D\|$	solved	root-time	#path-vars	#cuts	lp-gap	root-gap	final-gap
50	97	2.5	9.5	10	0.1	511.2	79.4	27.1	5.6	*
50	143.7	2.8	9.2	9	0.6	4,098.1	130.8	19.3	4.6	2.4
75	147	5.2	12.8	10	0.4	1,470.0	156.2	32.5	4.8	*
75	218.7	4.1	13.9	7	2.5	7,066.8	261.9	26.6	4.8	3.4
100	197	6.1	18.9	7	1.4	5,850.1	271.7	44.5	7.0	2.9
100	293.8	5.1	19.9	2	3.6	15,982.4	311.5	37.7	7.2	3.9
125	247	6.3	24.7	7	1.6	4,244.8	281.9	52.6	6.2	3.7
125	368.8	6.8	24.2	5	12.7	11,709.9	403.0	41.9	5.6	3.8
150	297	8.2	28.8	5	5.6	6,308.5	380.3	59.4	6.5	3.7
150	443.4	7.3	29.7	1	9.9	16,461.1	471.8	47.7	6.9	4.2
175	347	9.2	33.8	2	3.6	8,858.3	427.6	62.4	7.9	4.7
175	518.8	8.1	34.9	2	18.8	12,142.2	607.7	56.2	7.1	4.8
200	397	9.9	40.1	0	6.9	8,750.7	515.3	72.0	10.2	4.9
200	593.6	10.5	39.5	0	27.2	14,564.7	739.0	59.8	7.9	5.2
250	497	13.2	48.8	0	11.3	12,154.8	687.2	74.6	8.3	5.3
250	743.7	13.2	48.8	0	48.5	19,729.3	867.9	61.8	8.0	5.6
300	597	16.1	58.9	0	29.5	13,482.4	827.4	79.0	9.3	6.7
300	893.5	17.1	57.9	0	53.8	18,309.4	993.6	70.3	9.3	6.3

Table 3: Results of the branch-price-and-cut algorithm on JMP instances

$\|V\|$	$\|E\|$	$\|F\|$	$\|D\|$	solved	root time	#path-vars	#cuts	lp-gap	root-gap	final-gap
25	48.6	2.4	5.7	10	0.1	660.3	68.6	16.1	11.6	*
30	60.3	2.4	6.4	10	0.3	414.2	88.9	9.6	4.4	*
35	70.1	2.4	6.8	9	0.3	888.2	94.5	16.8	13.9	3.7
40	88.4	2.4	9.5	7	0.5	4,847.1	170.0	25.1	17.6	2.2
45	99.3	2.7	8.7	10	0.8	1,112.0	135.8	10.1	3.1	*
50	114.7	3.8	9.3	8	0.5	3,796.2	163.7	17.4	12.4	3.1
55	117.9	3.4	11.6	6	1.8	7,568.2	225.0	17.0	9.4	4.0
60	133.7	3.1	13.9	2	2.6	9,401.6	287.4	13.4	6.1	4.8

To take a closer look at the performance of the proposed heuristics we refer to Figure 4, which shows the progress of the upper bound during the solution process of instance “a”. We observe that all approaches already reach good upper bounds within a few minutes. The LP based greedy heuristic is fast and can also return primal solutions that consist of variables currently not in the restricted master problem. The IP based heuristic basically uses the power of all the problem independent primal heuristics bundled into CPLEX with respect to the currently available variables. This is computationally expensive but it also helps to find very good primal solutions. We observe that the hybrid strategy, which uses both types of heuristics, leads to the best results. Here, the greedy heuristic helps to construct new promising paths that can then also be taken into account for the CPLEX heuristic.

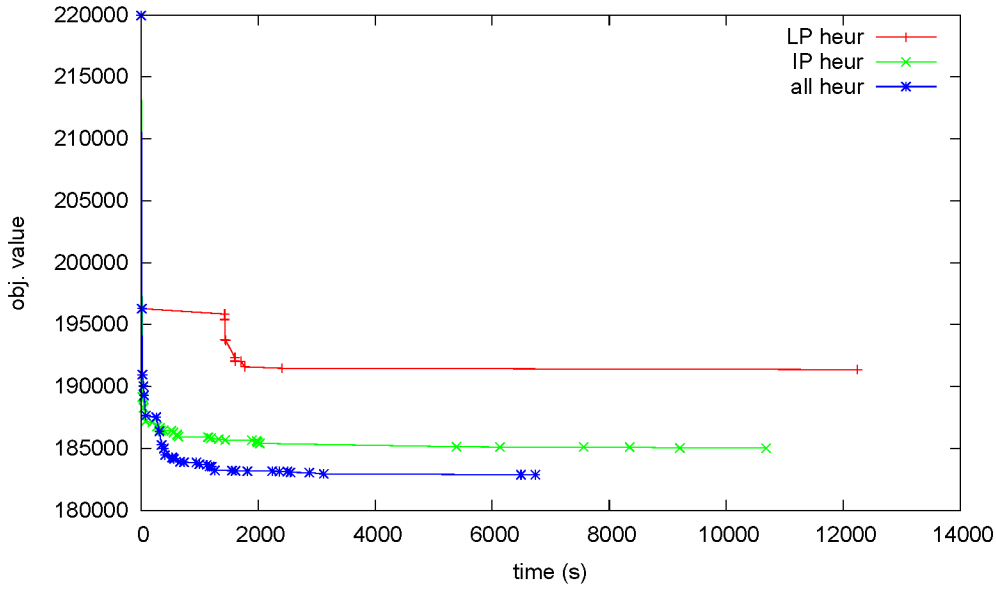

Figure 4 **Progress of the upper bound for the instance "a" using only the LP based greedy heuristic, only the IP based primal heuristic and both heuristics.**

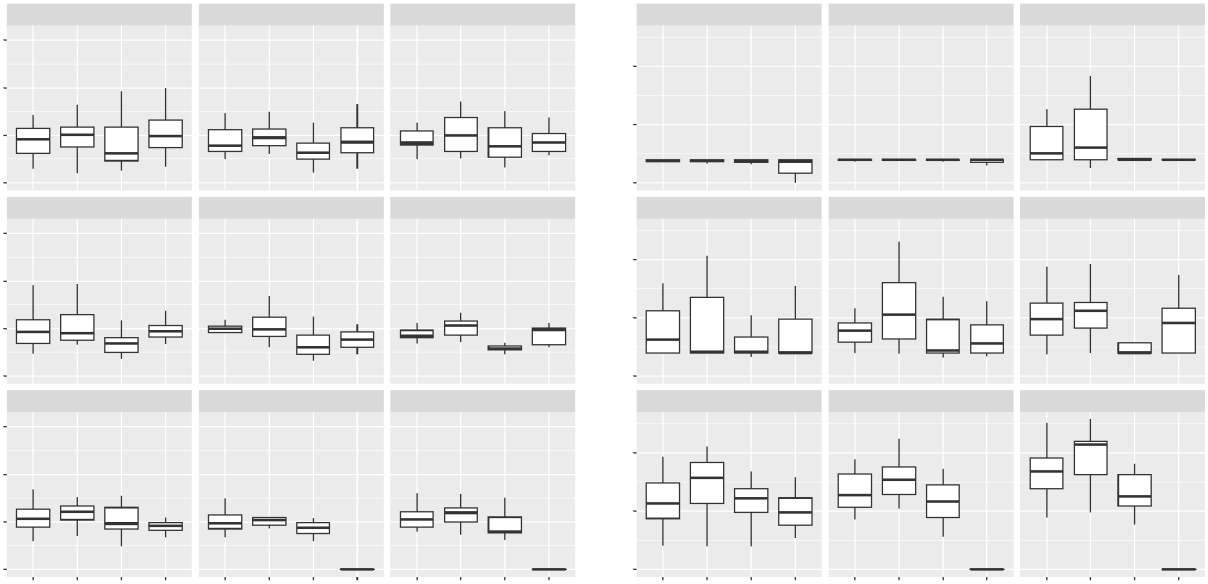

Figure 5 **The root LP gap and the final gap obtained for PA instances with node sizes varying between 50 and 300 for different demand structures.**

6. Conclusion

In this work we considered an integrated buy-at-bulk network design facility location problem. We proposed compact and exponential-sized Integer Programming (IP) formulations for the problem.

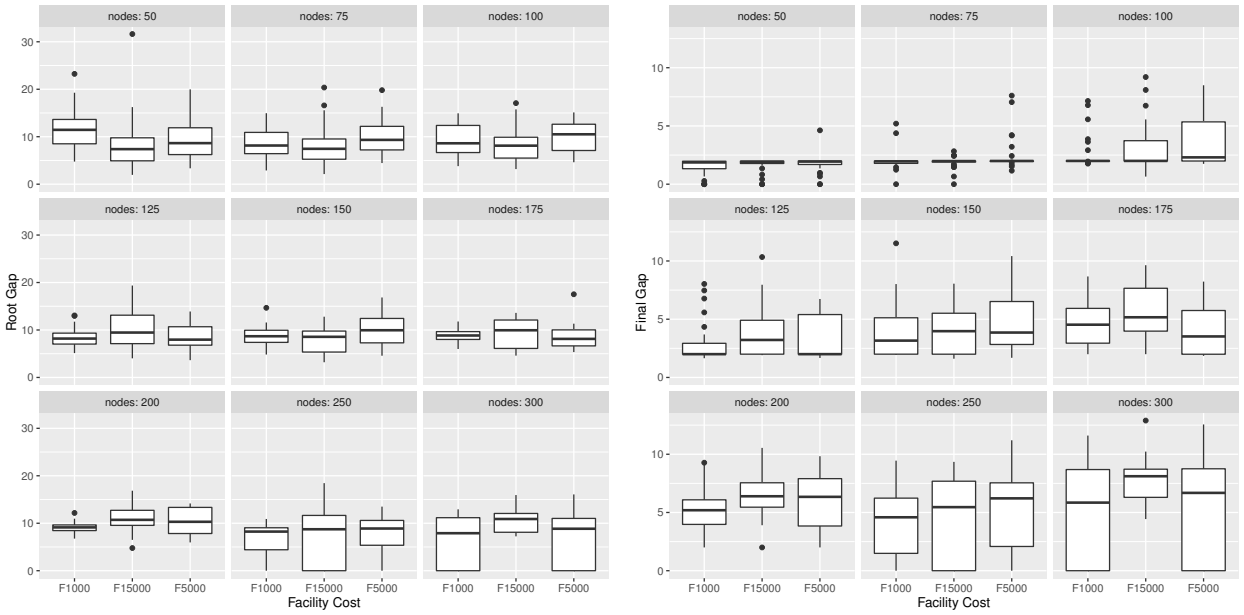

Figure 6 **The root LP gap and the final gap obtained for PA instances with node sizes varying between 50 and 300 for different fixed cost structures.**

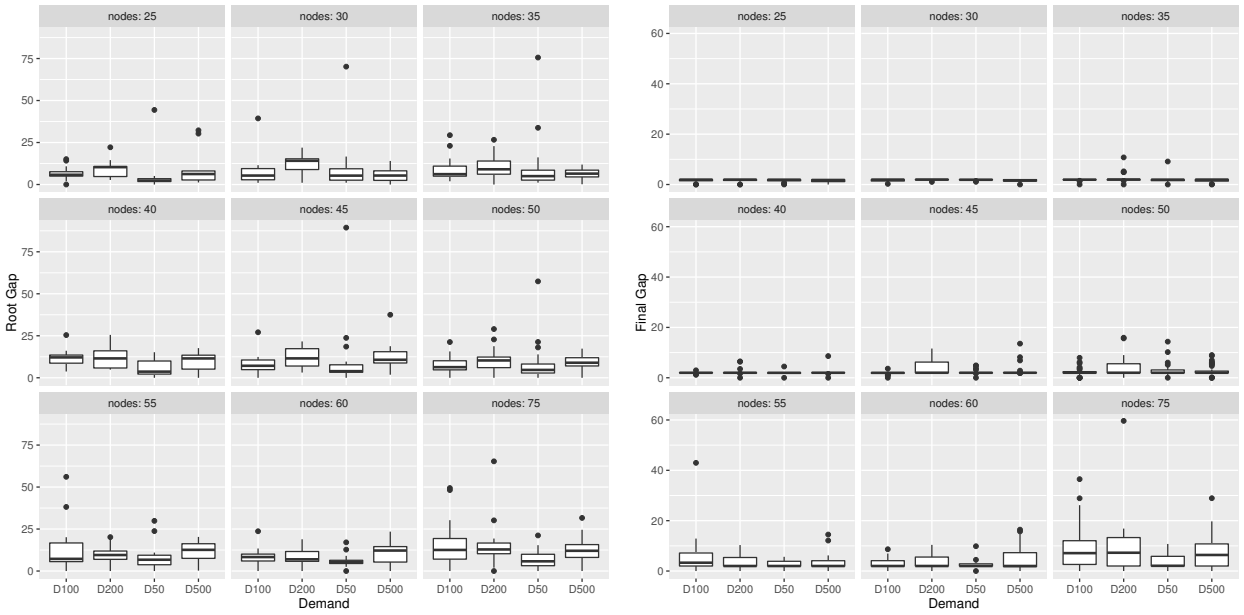

Figure 7 **The root LP gap and the final gap obtained for JMP instances with node sizes varying between 25 and 75 for different demand structures.**

We presented a branch-price-and-cut algorithm for solving the path based IP formulation of the problem. We studied the effect of the two families of valid inequalities. Some pre-processing tech-

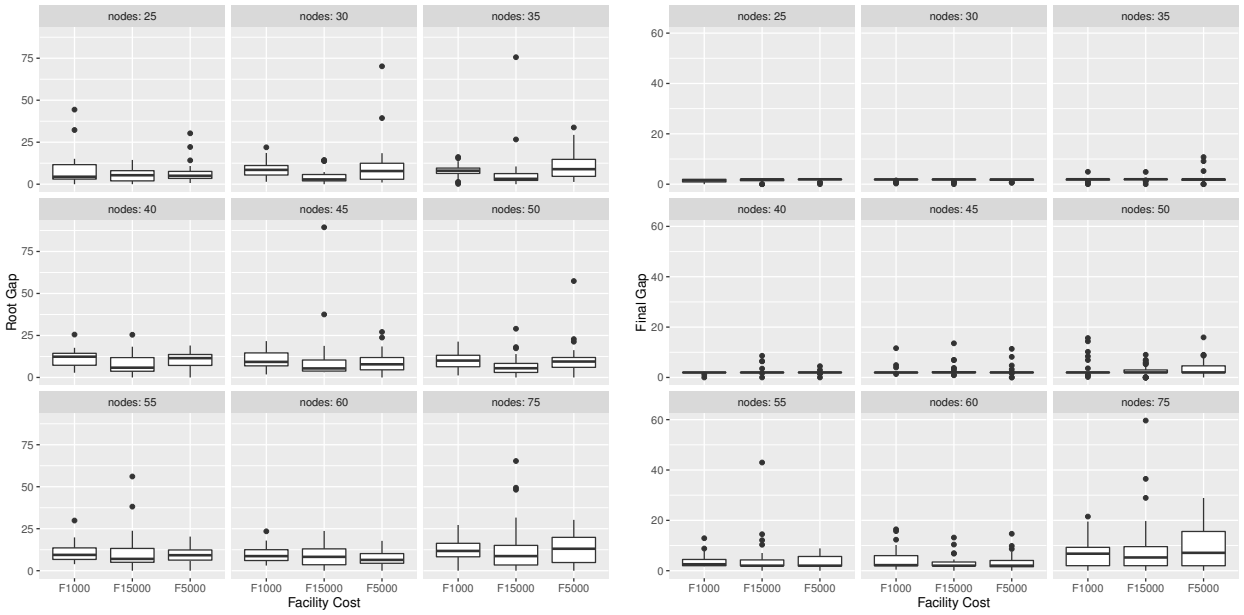

Figure 8 **The root LP gap and the final gap obtained for JMP instances with node sizes varying between 25 and 75 for different fixed cost structures.**

niques were proposed, which helped in effectively handling sparse instances. We also proposed two efficient IP- and LP-based primal heuristics to obtain good integer solutions. We gave a parallel implementation for the IP heuristic and demonstrated that a hybrid approach to combine the two heuristics provided better primal bounds. We test our approach on a set of real world instances and two different sets of computer generated instances. Using the proposed branch-price-and-cut approaches in combination with the primal heuristics allowed us to solve most of the tested real-world instances with a gap of less than 20 %. We empirically observed that the number of variables generated by the column generation approach is much lesser than the variables in the corresponding compact formulation.

Facilities in real world applications are usually capacitated and this is not considered in the current model. One can extend our model to this variant by introducing a capacitated module of cost μ_i corresponding to each (i, r), $i \in F$, and then seeking for a feasible routing network that allows simultaneous routing of clients' demands (not only to their facilities but also) to r without violating the capacities of (i, r) edges.

We are also currently considering models where clients need bi-connectivity with open facilities. This could also be solved using a branch-and-price framework with some alterations to the pricing problem.

Acknowledgments

The authors are much indebted to Daniel Schmidt for helpful discussions on the topic of this paper, especially on the generation of the test instances. The authors also would like to thank the anonymous reviewers for their valuable comments that helped us significantly improve the paper.

References

Achterberg, Tobias. 2009. SCIP: solving constraint integer programs. *Mathematical Programming Computation* **1**(1) 1–41.

Álvarez-Miranda, Eduardo, Valentina Cacchiani, Andrea Lodi, Tiziano Parriani, Daniel R Schmidt. 2014. Single-commodity robust network design problem: Complexity, instances and heuristic solutions. *European Journal of Operational Research* **238**(3) 711–723.

Atamtürk, Alper, Deepak Rajan. 2002. On splittable and unsplittable flow capacitated network design arc–set polyhedra. *Mathematical Programming* **92**(2) 315–333.

Barabási, Albert-László, Réka Albert. 1999. Emergence of scaling in random networks. *science* **286**(5439) 509–512.

Barnhart, Cynthia, Christopher A Hane, Pamela H Vance. 2000. Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems. *Operations Research* **48**(2) 318–326.

Cacchiani, Valentina, Michael Jünger, Frauke Liers, Andrea Lodi, Daniel R Schmidt. 2014. Single-commodity robust network design with finite and hose demand sets. *Technical Report OR-14-11, University of Bologna* .

Charikar, Moses, Adriana Karagiozova. 2005. On non-uniform multicommodity buy-at-bulk network design. *In Proceedings of the thirty-seventh annual ACM symposium on Theory of Computing (STOC)* 176–182.

Dantzig, George B, Philip Wolfe. 1960. Decomposition principle for linear programs. *Operations Research* **8**(1) 101–111.

Frangioni, Antonio, Bernard Gendron. 2012. A stabilized structured dantzig–wolfe decomposition method. *Mathematical Programming* **140**(1) 45–76.

Friggstad, Zachary, Mohsen Rezapour, Mohammad R Salavatipour, José A Soto. 2015. LP-based approximation algorithms for facility location in buy-at-bulk network design. *In Proceedings of the 14th International Symposium on Algorithms and Data Structures (WADS)* 373–385.

FTT. 2014. Fttx-plan. fttx-plan: Kostenoptimierte planung von fttx-netzen. http://www.fttx-plan.de/ .

Gamrath, Gerald. 2010. Generic branch-cut-and-price. Diploma thesis, Technische Universtät Berlin.

Garg, Naveen, Rohit Khandekar, Goran Konjevod, R Ravi, F Sibel Salman, Amitabh Sinha. 2001. On the integrality gap of a natural formulation of the single-sink buy-at-bulk network design problem. *In Proceedings of Integer Programming and Combinatorial Optimization (IPCO)* 170–184.

Grandoni, Fabrizio, Thomas Rothvoß. 2010. Network design via core detouring for problems without a core. *In Proceedings of Automata, Languages and Programming (ICALP)* 490–502.

Guha, Sudipto, Adam Meyerson, Kamesh Munagala. 2009. A constant factor approximation for the single sink edge installation problem. *SIAM Journal on Computing* **38**(6) 2426–2442.

Gupta, Anupam, Amit Kumar, Tim Roughgarden. 2003. Simpler and better approximation algorithms for network design. *In Proceedings of the thirty-fifth annual ACM Symposium on Theory of Computing (STOC)* 365–372.

Hassin, Refael, R Ravi, F Sibel Salman. 2004. Approximation algorithms for a capacitated network design problem. *Algorithmica* **38**(3) 417–431.

Johnson, David S, Maria Minkoff, Steven Phillips. 2000. The prize collecting steiner tree problem: theory and practice. *In Proceedings of ACM-SIAM Symposium on Discrete Algorithms (SODA)* .

Jothi, Raja, Balaji Raghavachari. 2004. Improved approximation algorithms for the single-sink buy-at-bulk network design problems. *In Proceedings of Algorithm Theory (SWAT)* 336–348.

Ljubić, Ivana, Peter Putz, Juan-José Salazar-González. 2012. Exact approaches to the single-source network loading problem. *Networks* **59**(1) 89–106.

Meyerson, Adam, Kamesh Munagala, Serge Plotkin. 2000. Cost-distance: Two metric network design. *In Proceedings of the 41st Annual Symposium onFoundations of Computer Science (FOCS)* 624–630.

Raack, Christian, Arie MCA Koster, Sebastian Orlowski, Roland Wessäly. 2011. On cut-based inequalities for capacitated network design polyhedra. *Networks* **57**(2) 141–156.

Raghavan, S, Daliborka Stanojević. 2006. A note on search by objective relaxation. *Telecommunications planning: innovations in pricing, network design and management* 181–201.

Randazzo, CD, Henrique Pacca Loureiro Luna, P Mahey, et al. 2001. Benders decomposition for local access network design with two technologies. *Discrete Mathematics & Theoretical Computer Science* **4**(2) 235–246.

Ravi, R, Amitabh Sinha. 2006. Approximation algorithms for problems combining facility location and network design. *Operations Research* **54**(1) 73–81.

Salman, F Sibel, Joseph Cheriyan, Ramamoorthi Ravi, Sairam Subramanian. 2001. Approximating the single-sink link-installation problem in network design. *SIAM Journal on Optimization* **11**(3) 595–610.

Salman, F Sibel, R Ravi, John N Hooker. 2008. Solving the capacitated local access network design problem. *INFORMS Journal on Computing* **20**(2) 243–254.

Talwar, Kunal. 2002. The single-sink buy-at-bulk lp has constant integrality gap. *In Proceedings of Integer Programming and Combinatorial Optimization (IPCO)* 475–486.

Tarjan, R Endre. 1974. A note on finding the bridges of a graph. *Information Processing Letters* **2**(6) 160–161.