A Branch-Cut-And-Price Algorithm for a Combined Buy-at-Bulk Network Design Facility Location Problem*

Ashwin Arulselvan, Mohsen Rezapour, and Wolfgang A. Welz

Insitut für Mathematik, Technische Universtät Berlin
arulsel,rezapour,welz@math.tu-berlin.de

Abstract In the combined buy-at-bulk network design facility location problem we are given an undirected network with a set of potential facilities and a set of clients with demands. In addition, we are also provided a set of cable types with each cable type having a capacity and cost per unit length. The cost to capacity ratio decreases from large to small cables following economies of scale. A network planner is expected to determine a set of facilities to open and install cables along the edges of the network in order to route the demands of the clients to some open facility. The capacities of the installed cable must be able to support the demands of the clients routed along that edge.The objective is to minimize the cost that is paid for opening facilities and installing cables.
We model the problem as an integer program and propose a branch-cut-and-price algorithm for solving it. We study the effect of two family of valid inequalities that naturally emerge from the model. We present the results of our implementation that were tested on a set of large real world instances.

1 Introduction

We consider a problem that integrates *buy-at-bulk network design* into the classical *uncapacitated facility location* problem. We model applications in which a set of facilities must be located to serve a set of clients, while a capacitated network must be designed connecting every client to its facility. In telecommunication networks, for example, this corresponds to installing routing devices on the network nodes, and laying cables, e.g. fiber-optic cables, along the edges of the network so that the required demands can be routed through the routing devices in the resulting network. The facilities here could be central offices or distribution points where devices could be installed to serve the clients.

The *Buy-at-Bulk network design with uncapacitated Facility Location* problem, denoted by BBFL, can be stated as follows. We are given an undirected graph $G = (V, E)$ with nonnegative edge lengths $g_e \in \mathbb{R}_{\geq 0}$, $e \in E$; a set $F \subseteq V$ of facilities with cost $f_i \in \mathbb{R}_{\geq 0}$; and a set of clients $D \subseteq V$ with demands $d_j \in \mathbb{Z}_{>0}$,

* Supported by DFG Research Center Matheon "Mathematics for key technologies" in Berlin.

$j \in D$. We are also given K *types* of *access cables* that may be used to connect clients to open facilities. A cable of type i has capacity $u_i \in \mathbb{Z}_{>0}$ and cost (per unit length) $\sigma_i \in \mathbb{Z}_{\geq 0}$. The task is to open a subset $I \subseteq F$ of facilities and construct a forest in order to route the demands from the clients to the facilities. The entire demand of each client must be routed via the forest to an open facility by installing access cables (probably multiple copies) along the edges of the forest with a capacity capable of supporting the routed demand.[1]. The objective is to minimize the total cost of opening facilities and installing access cables, where the cost for installing a single access cable of type i on edge e is $\sigma_i g_e$. This problem includes the classical facility location problem as a special case and is therefore NP-hard.

If we already decide on the forest (edges supporting the solution), then we need just to choose the optimal combination of the cables for each edge separately with total capacity that covers the demand that flows through each edge. To model and solve the single-sink buy-at-bulk network design problem, Salman et.al [9] point out that one can precompute (via dynamic programming) the optimal combination of cable types for all flow levels. This gives a monotonically increasing step cost function for the flow on any edge in the network. This allows modeling the problem as a single commodity network flow problem (since there is a single sink) with additional variables to model the resulting nonlinear objective function. After this transformation, one can look for the optimal module of the step cost function to be installed on every edge. We will make use of this in our formulations. We assume that for each edge e a set $\mathcal{N}_e = \{n_1, n_2, \ldots, n_{\mathcal{N}_e}\}$ of modules 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 $g_{e,n}$ and a capacity of $u_{e,u}$.

1.1 Previous and related work

To the best of our knowledge, the combination of the facility location and the buy-at-bulk network problem has been considered for the first time in [14]. They show that this combination can be seen as a special case of the *Cost-Distance* problem, and thereby provide the first O(log($|D|$)) approximating algorithm for this problem. Ravi et al. [6] later developed an $O(K)$ approximation, where K is the number of cable types, for this problem and called it *Integrated logistics*. As far as we are aware, this is the first exact algorithm for this combination. However, the combination of the facility location and network design (with $K = 1$) problem has been studied in the literature; see [15,16] and references within them for more details.

Single-Sink Buy-at-Bulk problem (SSBB) can be seen as a further simplification of the problem above in which the set of open uncapacitated facilities is given in advance, and considered as a single-sink. The SSBB problem has been

[1] We allow the demand crossing a single edge to use different access cables, but the collection of edges traversed must be a path in G.

widely studied in the both *operations research* and *computer science* communities. It should be noticed 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. For the unsplittable case, Garg et al. [2] developed an $O(K)$ approximation, using LP rounding techniques, where K is the number of cable types. The first constant factor approximation for this problem is due to Guha et al. [4]. Talwar [7] 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 et al. [5], and later to 40.82 by Grandoni et al. [3]. For the splittable case, Gupta et al. [13] 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. Modifying Gupta's algorithm, the approximation for u-SSBB was later reduced to 65.49 by Jothi et al. [5], and then to 24.92 by Grandoni et al. [3].

In the operation research literature, this problem is also known as *single-source network loading problem* [8] or (in the case of telecommunication network planning) *Local Access Network Design Problem (LAN)* [9].

In [11] Randazzo et al. study the LAN problem with two cable types under the assumption that the solution must be a tree (with unsplittable flow). They provide a multicommodity flow based formulation for the problem and solve it by applying Benders' decomposition. Salman et al. [9] consider the LAN problem with multiple cable types where the cable types obey economies of scale. They apply flow-based MIP formulations and work with relaxations obtained by approximating the capacity step cost function by its lower convex envelope to provide a special branch-and-bound algorithm for LAN design. Raghavan and Stanojevic [12] later reformulate this approximation technique as a stylized branch-and-bound algorithm. Working with the approximate step cost function, as defined in [9], Ljubic et al. [8] study exact approaches, based on disaggregating the commodities and using Benders decomposition on a multicommodity flow based formulation.

Our contribution We model the problem as an integer program that uses an exponential number of variables. We propose a branch-cut-and-price approach to solve the problem. We discuss and study some valid inequalities in our experiments that strengthen the formulation. We present the results from our computational study on several real world instances.

Organization We begin with the integer programming formulation for combined buy-at-bulk network design facility location problem in section 2. We then discuss the valid inequalities we considered for the integer program in Section 3. In

Section 4, we explain the branch-cut-and-price procedure that discusses the pricing subproblem, branching strategies and primal heuristics. Finally in Section 5, we give the details of the implementation. We also explain the test instances and present our results before we conclude.

2 IP Formulation

We propose a path-based formulation for the problem with an exponential number of variables and solve it using column generation. We are given an undirected graph G=(V,E). For the sake of modelling paths, we first create a dummy root node r and connect all facilities with the root node. Let $E' = E \cup \left\{\bigcup_{i \in F}(i,r)\right\}$. Let $P(j)$ denote the set of all possible paths in $G' = (V \cup \{r\}, E')$ starting from client j and terminating at the root node 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. The binary variables z_i indicate whether facility i is open or not. For an edge $e = (k,l)$ and a module $n \in \mathcal{N}_{(k,l)}$ the variable $x_{kl,n}$ indicate whether module n has been installed on edge (k,l) or not. Then the problem can be formulated as follows:

$$(IP1) \quad \min \sum_{i \in F} f_i z_i + \sum_{(k,l) \in E} \sum_{n \in \mathcal{N}_(k,l)} g_{(k,l),n} \cdot x_{kl,n} \tag{1}$$

$$\sum_{p \in P(j)} y_p = 1, \qquad \forall j \in D \tag{2}$$

$$\sum_{j \in D} \sum_{\substack{p \in P(j) \\ \{(k,l),(l,k)\} \cap p \neq \emptyset}} d_j y_p \leq \sum_{n \in \mathcal{N}_{kl}} u_{kl,n} x_{kl,n}, \qquad \forall (k,l) \in E \tag{3}$$

$$\sum_{n \in \mathcal{N}_{kl}} x_{kl,n} \leq 1, \qquad \forall (k,l) \in E \tag{4}$$

$$\sum_{p \in P(j):(i,r) \in p} y_p \leq z_i, \qquad \forall i \in F, \forall j \in D \tag{5}$$

$$y_p, x_{kl,n}, z_i \in \{0,1\} \tag{6}$$

Constraints (2) force each client to be connected to a routing path. Constraints (3) ensure that we install sufficient capacity to support the flow along routing paths, Constraints (4) guarantee that at most one module is installed along each edge and Constraints (5) ensure that a serving facility is open.

3 Valid Inequalities

3.1 Cover Inequalities

In order to derive the cover inequalities corresponding to each constraint in set (3), we obtain a knapsack structure by complementing the x variables (re-

placing x by $1-x$) in the constraint. Consider the constraint in set (3) corresponding to edge $(k,l) \in E$. Let $U_{kl} = \sum_{n \in \mathcal{N}_{kl}} u_{kl,n}$. We define $c_{(k,l)} = (D_c, M_c)$ to be a cover with respect to edge (k,l), where $D_c \subseteq D$ and $M_c \subseteq \mathcal{N}_{kl}$, if

$$\sum_{j \in D_c} d_j + \sum_{n \in M_c} u_{kl,n} > U_{kl}$$

We say that a cover is minimal when just removing any item either from D_c or M_c results a cover for which the above inequality does not hold. It is not hard to show that if $c_{(k,l)}$ is a minimal cover, then the following inequalities are valid [17]:

$$\sum_{j \in D_c} \sum_{p \in P(j):(k,l) \in p} y_p + \sum_{n \in M_c} (1 - x_{kl,n}) \leq |M_c| + |D_c| - 1 \Longleftrightarrow$$

$$\sum_{j \in D_c} \sum_{p \in P(j):(k,l) \in p} y_p \leq \sum_{n \in M_c} x_{kl,n} + |D_c| - 1 \tag{7}$$

Let (x^*, y^*, z^*) be the optimal fractional solution of the LP relaxation of model IP1. Now, we present how to find a cover inequality corresponding to edge (k,l) violated by (x^*, y^*, z^*). For each $j \in D$, we let

$$w_j^* = \sum_{p \in P(j):(k,l) \in p} y_p^*$$

And let $F_{kl} \subseteq \mathcal{N}_{kl}$ be the set of modules for the edge (k,l) such that $x_{kl,n}^* > 0$. The most violated cover inequality can be detected by solving the knapsack problem shown below:

$$\min \ \gamma = \sum_{n \in F_{kl}} x_{kl,n}^* x_{kl,n} + \sum_{j \in D} (1 - w_j^*) w_j \tag{8}$$

$$\sum_{j \in D} d_j w_j + \sum_{n \in F_{kl}} u_{kl,n} x_{kl,n} \geq \sum_{n \in F_{kl}} u_{kl,n} + 1 \tag{9}$$

$$x_{kl,n} \in \{0,1\}, \forall n \in F_{kl} \tag{10}$$

$$w_j \in \{0,1\}, \forall j \in D \tag{11}$$

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

$$\sum_{m \in F_{kl}} x_{kl,n}^* + \sum_{j \in D} (1 - w_j^*) - \max \sum_{n \in F_{kl}} x_{kl,n}^* \bar{x}_{kl,n} + \sum_{j \in D} (1 - w_j^*) \bar{w}_j \tag{12}$$

$$\sum_{j \in D} d_j \bar{w}_j + \sum_{n \in F_{kl}} u_{kl,n} \bar{x}_{kl,n} \leq \sum_{j \in D} d_j - 1 \tag{13}$$

$$\bar{x}_{kl,n} \in \{0,1\}, \forall m \in F_{kl} \tag{14}$$

$$\bar{w}_j \in \{0,1\}, \forall j \in D \tag{15}$$

Let $D' \subseteq D$ and $F'_{kl} \subseteq F_{kl}$ be the optimal subsets of the minimizing knapsack problem. Then, it is easy to show that (x^*, y^*, z^*) violates the following cover inequality if $\gamma < 1$.

$$\sum_{j \in D'} \sum_{p \in P(j):(k,l) \in p} y_p \leq \sum_{n \in F'_{kl,m} \cup \{N_{kl} \setminus F_{kl}\}} x_{kl,n} + |D'| - 1 \tag{16}$$

3.2 Cut inequalities

The modules generated follow economies of scale and hence the LP relaxation ends up fractionally picking the last module -with the lowest cost/capacity rate- at optimality for most edges. Indeed, this is the main reason for the integrality gap of IP1 to be arbitrarily large. In this section we try to remedy this difficulty by introducing a set of valid inequalities, called *cut inequalities*, as follows: Given a fractional optimal solution (x^*, y^*, z^*), for the graph $G' = (V \cup \{r\}, E')$, we take the edge capacities to be, $u_{kl} = \sum_{n \in \mathcal{N}_{kl}} x^*_{kl,n}$, for all $(k, l) \in E$ and 1 for all other edges. For every customer $s \in D$ we solve the maximum flow problem with source as s and sink as r. If the flow value is less than 1, we obtain the following violated cut

$$\sum_{(k,l) \in C} \sum_{n \in \mathcal{N}_{kl}} x_{kl,n} \geq 1 \tag{17}$$

Where C is the corresponding the minimum cut. The validity of the cut follows from the fact that every customer needs to be connected to some facility along a path with every edge in the path having at least one module installed. Note that not all modules need to be picked in the generated inequality. Since we are allowed to install at most one module on edge, it is enough if we consider those modules with capacities greater than the demand of the customer.

4 Solution procedure

Since the path based formulation presented above contains an exponential number of variables, our solution procedure is based on the column generation techniques. We consider as the restricted master problem the continuous relaxation of the IP1 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 Column generation

We iteratively solve the restricted master problem and search for new columns having negative reduced cost that is computed using the the optimal dual solution. Let the dual variables corresponding to Constraints (2) be μ_j. We will refer to the dual variables corresponding to Constraints (3) with the notation as

π_{kl}, forall $(k,l) \in E$ and the dual variables corresponding to Constraints (5) by γ_i^j, forall $i \in F, j \in D$. For each $j \in D$, 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:

$$\min_{p \in P(j)} \quad -\left(\mu_j + \sum_{\substack{(k,l) \in p \\ l \neq r}} d_j \pi_{kl} + \sum_{i \in F} \mathbb{I}_i^p \gamma_i^j\right)$$

where $\mathbb{I}_i^p$ is an indicator variable denoting whether edge (i,r) is in the path p or not (r being the root node). Given the graph $G' = (V \cup \{r\}, E')$, we take the weight of an edge (k,l) to be $-d_j \pi_{kl}$, for all $(k,l) \in E$ and weights $-\gamma_i^j$, for all $i \in F, (i,r) \in E'$. We now find the shortest path in E' 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 μ_j, then the solution is not optimal for the master problem and 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 4.3) 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 above considering the so called dual Farkas values. This method has been called *Farkas pricing*, and provided in [10] within the SCIP framework (see Section 5).

4.2 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. The generated cover cuts (16) 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 (k,l) has multiple cover inequalities associated with it. Let $C_{(k,l)}$ be the set of covers associated with edge (k,l). Let $C = \bigcup_{(k,l) \in E} C_{(k,l)}$. For a cover $c \in C$, let D_c be the set of clients involved in the cover and α_c be the corresponding dual variable.

The new pricing problem associated with client j is:

$$\min_{p \in P(j)} \quad -\left(\mu_j + \sum_{\substack{(k,l) \in p \\ l \neq r}} d_j \pi_{kl} + \sum_{i : i \in F} \mathbb{I}_i^p \gamma_i^j + \sum_{(k,l) \in p} \sum_{\substack{c \in C_{(k,l)} \\ j \in D_c}} \alpha_c\right) \tag{18}$$

Note that cut inequalities (17) involving x 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.

4.3 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, even 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 enforce integrality. This technique, when used together with column generation and cut separation is called Branch-Cut-and-Price. The most intuitive branching rule is, given variable y_p with a continuous value, to create two branches, the first one with $y_p = 0$ and other with $y_p = 1$. But this cannot be done due to the column generation algorithm. If a path p is fixed to zero, the pricing subproblem will find this route again on the next iteration and will return it to the restricted master. An alternative can be to implement standard branching in the space of the compact formulation. We branch by adding constraints that gives lower and upper bound 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, we follow closely the so called path-splitting branching rule used for multi-commodity flow in [1] where branching constraints bound the number of paths that use a common starting path from a client as follows. Consider any $j \in D$ 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 consider the edges e_1 and e_2 present in p_1 and p_2 respectively emanating from this node. 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 $\sum_{p \in P(j): p \cap \{e_1\} \neq \emptyset} y_p = 0$. In [1], they partition the set of edges emanating from this node into two sub-sets E_1 and E_2 such that E_1 (E_2, respectively) intersects p_1 (p_2, respectively). Then, they impose $\sum_{p \in P(j): p \cap E_2 \neq \emptyset} y_p = 0$ in one branch and $\sum_{p \in P(j): p \cap E_1 \neq \emptyset} y_p = 0$ in the other branch. This branching strategy is more efficient than the one we implemented in terms of convergence. We plan to follow this in our future work. 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. In our implementation, the branching priorities for $\boldsymbol{x}$ and $\boldsymbol{z}$ are higher than that of the $\boldsymbol{y}$ variables.

4.4 Primal Heuristic

For the overall performance of a Branch-and-Price approach it is crucial that good primal solutions to the problem are found fast using heuristics. One possible idea for such a heuristic is to treat the problem including all variables generated hitherto as a fixed Integer Problem and then solve it using a commercial solver

like CPLEX. The result gives us the best possible solution that can be achieved without adding new variables and only using the paths from $P'(j)$. Since the number of variables is fixed, this problem is much easier to solve and even if the solution process of the sub-IP is canceled after a certain amount of time, the best solution found is still a feasible solution to our problem.

5 Experiments

The Branch-Cut-and-Price approach has to been implemented using the framework provided by SCIP [10]. In this context SCIP handles all the underlying Integer Programming specific aspects and has be extended with problem dependent plugins for the pricing and branching as well as the cut generation and primal heuristic. As explained in Section 4.1 the pricing problem corresponds to solving a shortest path problem for every customer, which is solved via an implementation of Dijkstra's Algorithm.
To avoid that this computation is performed for all the customers 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 customers. 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 (5) and to the cover cuts depend on j. The μ_j and the capacities d_j can be applied after the shortest paths have been calculated. However, different dual variables resulting from the Constraints (5) are selected for different customers. 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:

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

As the dual variables α_c are all not positive, this gives us a lower bound of the pricing problem (18).
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.

As described in Section 4.4 CPLEX is used to find good solutions using only paths form the current $P'(j)$. To make this process as efficient as possible the CPLEX-problem is created by adding all variables corresponding to $P'(j)$, which is then solved in the background so that the main CPU-thread can still continue to perform the regular branch-and-bound process using SCIP. If CPLEX finds a new improving solution the main thread is being signaled and this solution is than added as a new solution for SCIP so that it can immediately be used as an upper bound in the branch-and-bound. 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 in the new computation. To improve the solution process we also add the found valid inequalities to the problem. Those inequalities are problem dependent and thus improve the automatically generated cuts by CPLEX.

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.

Test instances The instances tested correspond to real world network planning problem. The networks were generated from the publicly available information obtained through geographic information systems. Each instance correspond to a region in Germany and was constructed bearing in mind the potential customer and facility locations. The street segment form the edges, while the street intersections and traffic circles provide the intermediate nodes. The information about the different cable types along with their costs and capacities were provided by our industry partners.

Computational results 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 with a time limit of 10 hours. We used CPLEX 12.4 and SCIP version 3.0.2.

inst	$\|V\|$	$\|E\|$	$\|F\|$	$\|D\|$	#vars generated	#cover inequalities	#cut inequalities	final gap	root gap
a	1675	1730	104	604	15757	3774	4493	21.23%	21.55%
b	4110	4350	230	1670	29098	8191	11487	25.78%	25.92%
c	6750	7352	531	2440	35381	0	9716	38.00%	38.00%
d	4227	4484	319	1490	40703	1853	10291	28.60%	28.60%
e	11544	12478	890	4275	43975	0	6044	70.44%	70.44%
f	637	826	101	39	66556	1222	577	19.08%	21.72%
g	6750	7352	531	2440	36315	0	10244	37.77%	37.77%
h	2271	1419	498	349	30874	642	1732	23.59%	23.73%
i	1315	1434	148	238	52563	4514	3150	21.45%	21.75%

From the results we could see that we have trouble solving the root LP in the larger instances (with more than 6000 nodes). This is an expected behavior with column generation as it has stabilization issues. For the smaller instances, we were able to close the gap to less than 25%. The issue with stability needs to be addressed for the larger instances by generating more useful columns. For the larger instances, the pass on the cut inequalities added a sufficient number of

cuts reaching the cut count limit for the iteration and hence the cover inequality call was bypassed in every iteration. The root gap presented was calculated using the best integer solution. The observation to be made is that the branching strategies need to be revised as the lower bounds are not improving much once the branching commences for most instances. We observed that the valid inequalities introduced helped in significantly improving the lower bound.

6 Conclusion

We presented a branch-cut-and-price algorithm for solving the multisink buy-at-bulk network design problem. We studied the effect of the two families of valid inequalities. The model shows promise in solving reasonably large real world instances. As future work, we intend to address the stabilization of column generation process and device sophisticated branching strategies. We also intend to design an efficient primal heuristic that would eventually replace the CPLEX based heuristic.

References

1. C. Barnhart, C. A. Hane, and P. H. Vance. "Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems." Operations Research 48.2 2000, pages 318-326.
2. N. Garg, R. Khandekar, G. Konjevod, R. Ravi, F. S. Salman and A. Sinha. On the integrality gap of a natural formulation of the single-sink buy-at-bulk network design formulation. In Proc. of IPCO 2001, pages 170-184.
3. F. Grandoni and T. Rothvoß. Network design via core detouring for problems without a core. In Proc. of ICALP 2010, pages 490-502.
4. S. Guha, A. Meyerson, K. Munagala. A constant factor approximation for the single sink edge installation problems. In Proc. of STOC 2001, pages 383-388.
5. R. Jothi and B. Raghavachari. Improved approximation algorithms for the single-sink buy-at-bulk network design problems. In Proc. of SWAT 2004, pages 336-348.
6. R. Ravi, A. Sinha. Integrated logistics: Approximation algorithms combining facility location and network design. In Proc. of IPCO 2002, pages 212-229.
7. K. Talwar. The single-sink buy-at-bulk LP has constant integrality gap. In Proc. of IPCO 2002, pages 475-486.
8. I. Ljubic, P. Putz, and J. J. SalazarGonzalez. Exact approaches to the singlesource network loading problem. Networks 59.1, 89-106, 2012.
9. F. S. Salman, R. Ravi and J. Hooker. Solving the Local Access Network Design Problem. INFORMS J. on Computing, 20:2, 243-254, 2008.
10. T. Achterberg. SCIP: Solving constraint integer programs. Mathematical Programming Computation 2009, pages 1-41.
11. C. D. Randazzo, H. P. L. Luna, and P. Mahey. Benders decomposition for local access network design with two technologies. Discrete Mathematics & Theoretical Computer Science 4.2 2001, pages 235-246.
12. S. Raghavan, and D. Stanojevic. A note on search by objective relaxation. Telecommunications planning: innovations in pricing, network design and management. Springer US, 2006, pages 181-201.

13. A. Gupta, A. Kumar, and T. Roughgarden. Simpler and better approximation algorithms for network design. In STOC 2003, pages 365-372.
14. A. Meyerson, K. Munagala, and S. Plotkin. Cost-distance: Two metric network design. In FOCS 2000, pages 624-630.
15. S. Melkote and M.S. Daskin. An integrated model of facility location and transportation network design. Transportation Research Part A, 35(6) 2001, pages 515-538.
16. S. Melkote and M.S. Daskin. Capacitated facility location-network design problems. European Journal of Operational Research, 129(3) 2001, pages 481-495.
17. G. L. Nemhauser and L. A. Wolsey, Integer and combinatorial optimization, 1988, Wiley-Interscience, NY, USA,