Exact Approaches to the Single-Source Network Loading Problem

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

June 14, 2010

Abstract

This article considers the network design problem that searches for a minimum-cost way of installing capacities on the edges of a network in order to simultaneously route a flow from a given access point to a subset of nodes representing customers with positive demands. We first consider compact and exponential-sized MIP formulations of the problem and provide their theoretical and computational comparison. **We also consider a stronger disaggregated flow formulation.** To solve the problem in practice, we project out the flow variables and generate Benders cuts within a branch-and-cut framework. To the best of our knowledge the combination of Benders approach and this specific disaggregation has not been considered so far. R2.1

In an extensive computational study we compare the performance of compact MIP models against a textbook implementation and several normalization variants of Benders decomposition. We introduce a set of 32 real-world instances and use these, together with 64 other instances from the literature, to test our approaches. The results show that our branch-and-cut approach **outperformes** the best-performing compact formulation leading to the best exact algorithm today for solving the considered data set. R2.m1

Keywords: Local Access Network Design, Network Loading, Capacitated Network Design, Benders Decomposition.

1 Introduction

We consider the problem of deploying a broadband telecommunication system that lays optical fiber cable from a *central office* to a number of *end-customers*. In case of the *fiber to the home* technology (FTTH) the end-customers represent houses, whereas when deploying *fiber to the curb* technology (FTTC) the end-customers are usually multiplexor devices. In both cases, we are dealing with a capacitated network design problem that requires an installation of optical fiber cables with sufficient capacity to carry the traffic from the central office to the end-customers. We start with a network without capacities, or

*Department of Statistics and Decision Support Systems, University of Vienna, Austria
†DEIOC, Universidad de La Laguna, Tenerife, Spain

with some pre-installed capacities, and search for the installation of cable types on links that enables simultaneous routing of traffic so that the whole demand in the network is satisfied at minimum cost. This problem is known in the literature as the *Single-Source Network Loading Problem* (SSNLP) or *local access network design problem*.

Our Contribution This article provides a branch-and-cut approach based on Benders decomposition for solving SSNLP. **By adapting known disaggregation techniques for piecewise linear multi-commodity flow problems [16], we use a MIP formulation that has not been considered so far for solving the SSNLP.** From **a** theoretical point of view, the new formulation provides stronger lower bounds when compared to existing MIP models. To make the new model computationally tractable, we project out **the** flow variables by using strengthening, rounded Benders inequalities incorporated into a branch-and-cut framework. **The network loading problem has been intensively studied in the literature. However, to the best of our knowledge the combination of Benders approach and this specific disaggregation has not been investigated so far.** When comparing this approach to the best-performing compact formulation, we show that the average gap on a set of benchmark instances from the literature can be improved from 5.5% to 2.5%. Furthermore, when testing a new set of instances derived from a real-world telecommunication example, we report 8 optimal solutions, whereas the compact formulation does not solve a single one to optimality.

R2.1 R2.m2 R2.m3 R1.6

Problem Definition Let us consider an undirected and connected graph $G^u(V, E)$ with a designated root node $r \in V$ (representing the central office or access point to the backbone network) and a set of customers $D \subseteq V \setminus \{r\}$. Each customer $k \in D$ is associated with a positive demand $d_k \in \mathbb{R}_{>0}$. Each edge $e \in E$ is associated with a length $l_e \in \mathbb{R}_{\geq 0}$. It is allowed to install combinations of different cable types with positive costs and capacities on every edge. Salman et al. [38], **among others, have observed** that, by using dynamic programming, one can precompute the optimal combination of cable types for each level of flow and for every edge. This provides an increasing non-linear step cost function of flow for every edge of the network. We consider the optimization problem after this transformation, i.e., instead of searching for optimal *combination of cable types*, we are looking for the optimal *module* of the step cost function to be installed on every edge. Thereby, we assume that modules $\mathcal{N}_e = \{n_1, n_2, \ldots, n_{|\mathcal{N}_e|}\}$ are given for each edge $e \in E$, with capacities $u_{e,n} \in \mathbb{R}_{>0}$ and costs $c_{e,n} \in \mathbb{R}_{\geq 0}$ for each $1 \leq n \leq |\mathcal{N}_e|$. **Without loss of generality we assume that the module cost per edge are increasing with indices n.** We denote $|\mathcal{N}| := \max_{e \in E} |\mathcal{N}_e|$. Then SSNLP looks for a single-source multiple-sink routing and a link capacity assignment, with an installation of *at most one module* on every edge, to satisfy all customer demands.

R2.m4 R2.m5

Since we allow the flow between the access point and a customer to be split apart, we are speaking about a *bifurcated routing* formulation. The optimal solution of SSNLP is not necessarily a tree. Obviously, if there is only one module per edge providing sufficient capacity to route the total flow through it, then the optimal solution will be a tree, and the problem is equivalent to the Steiner tree problem on

the graph by considering all customers with positive demand as *terminals* while minimizing the sum of edge lengths ($l_e \cdot c_{e,1}$) taken into the solution.

Our definition of SSNLP works for the general setting: Economies of scale may not be given over all modules. We allow the number of modules and their costs and capacities to differ from edge to edge, in contrast to the previous approaches where the modules were considered to be uniform. In addition, capacities on an edge may be limited.

Previous Work Due to its importance in telecommunications, transportation, computer and energy supply networks, network loading problems have been widely studied in the literature. Many authors consider a more general variant in which a routing from multiple sources to multiple sinks is required. Polyhedral structures of the general network design problem with multiple sources and multiple sinks are studied in [4, 7, 12, 17, 24, 29, 40]. Benders decomposition approaches have been studied as well: for the multiple-source multiple-sink case, an exact algorithm based on the *expansion step cost model* from [17] was given in [22]; the latter approach has been improved recently in [19]. In [14] the authors study the relationship between metric and Benders inequalities for the general capacitated network design problem. In [35] the authors look into speeding up Benders decomposition by combining it with local branching. Metric inequalities for the network loading problem have been studied in [3]. The authors work on the multi-commodity flow problem and propose several variants for separating metric inequalities. We build some of our normalization models by extending their ideas, see Section 3. **We refer to the extensive survey given by Costa [13] for further references on network design models (including those with module-induced variables) and Benders approaches.** R1.6

SSNLP has been studied only under the assumption that costs and capacities satisfy the concept of *economies of scale*, i.e., that the cost per unit capacity of a thick (high capacity) cable is cheaper than that of a thin (low capacity) cable, thus buying capacities in bulks becomes more economical when the traffic increases. For that reason, in the computer science literature, SSNLP is also known as the *single-sink buy-at-bulk* network design problem (see, e.g., [37]). In terms of approximation algorithms, currently the best provable worst-case approximation ratio of 24.92 has been obtained by Grandoni and Italiano [25]. Chopra et al. [10] have shown that the network loading problem with only two cable types and with a single-source and a single-sink node remains NP-hard. Berger et al. [6] **handle the non-bifurcated case and** have proposed a tabu-search heuristic that relies on the computation of k shortest R2.m6 paths, in order to find alternative paths from the root to each customer node.

Salman et al. [38] proposed the *search by objective relaxation* (SOR) approach for solving SSNLP. The authors solve SSNLP by considering the flow problem with a non-linear step cost function. The step cost function is first approximated by its lower convex envelope. The obtained relaxed problem is solved by a combinatorial algorithm in polynomial time. The process is repeated in every node of the branch-and-bound tree in which branching is done by dividing the interval of possible flow values on an edge into subintervals. In Section 5 we refer to their results where we also use their benchmark instances

to test our approach.

Raghavan and Stanojević [34] pointed out that the linear programming (LP) relaxation of a single-commodity flow model for SSNLP (see Section 2.2) also approximates the step cost function by its lower convex envelope. Therefore the SOR approach can also be seen as a stylized branch-and-bound on a single-commodity flow model. The authors compared their stylized branch-and-bound approach against two MIP models based on the aggregated single-flow formulation. While the authors were able to **outperform** the explicit cost model, their results were always worse than those obtained by the incremental cost MIP model. R2.m1

The rest of the paper is organized as follows. In Section 2 we recall existing MIP formulations and propose a new disaggregated compact model. A hierarchy of MIP formulations is also given. Section 2.5 explains how to project out flow variables of the new compact model and how to generate stronger cutting planes. In Section 3 we propose several different ways for normalizing the Benders subproblem. Algorithmic aspects of our approach are discussed in Section 4. We implemented several different approaches for solving the Benders subproblem. An extensive computational comparison of compact vs. Benders approaches is provided in Section 5.

2 MIP Formulations

Network loading problems are often modeled using compact flow-based MIP formulations (see, e.g., [14, 24, 34, 38]) involving integer design variables and continuous flow variables. When disaggregating flow variables, we face the trade-off problem between the increasing quality of lower bounds and the growing size of the underlying linear program. In this section, we first recall two "natural" compact formulations for SSNLP. We then propose a new disaggregated flow-based formulation and a row-generation approach based on Benders decomposition for this model. We also recall a weak cut-set based model for the SSNLP. We finally provide a hierarchy of different MIP formulations with respect to the quality of lower bounds of their LP relaxations.

2.1 Transformation into Directed Problem

It is well known that, in general, the MIP formulations of uncapacitated network design problems on directed graphs provide better lower bounds than their undirected counterparts (see e.g., [11]). However, the MIP approaches to SSNLP up to now, considered in [34, 38], involve undirected graphs. We propose to work with directed graphs and for that purpose we transform our input graph $G^u = (V, E)$ into a directed graph $G = (V, A)$ where $A := \{(i,j),(j,i) \mid \{i,j\} \in E; i,j \neq r\} \cup \{(r,j) \mid \{r,j\} \in E\}$. The available modules on the arcs remain symmetric, i.e., $c_{ij,n} = c_{ji,n} = c_{\{i,j\},n}$, $u_{ij,n} = u_{ji,n} = u_{\{i,j\},n}$ for all $n \in \mathcal{N}_{\{i,j\}}$. To solve SSNLP, we now search for the *directed solution*, i.e., for the installation of at most one module on every *arc* so that there is enough capacity to route the flow from r to every $k \in D$.

Note that this transformation from an undirected into a directed graph is not valid for general multi-commodity network design problems. Instead one can easily prove that it is valid under our single-source assumption. More formally: R1.7

Lemma 2.1. *Let G be the graph as described above. Then, there always exists an optimal solution of the* directed SSNLP*, which is a directed acyclic subgraph of G.*

R1.5
R2.m7

Corollary 2.2. *Given the graph G^u, demands d_k, for all $k \in D$ and the cost and capacity functions c and u as described above, any optimal solution of SSNLP on G^u can be transformed into an equivalent* directed solution *on G with the same objective value, and vice versa.*

2.2 Single-Commodity Flow Formulation (SCF)

The single-commodity flow formulation, SCF, models the flow on every arc as the total amount of flow routed from the root toward the customers. To model a non-decreasing step cost function on every arc, binary variables need to be used. There is a possibility to model the problem using the *incremental cost model* or the *explicit cost* (also called the *multiple choice*) model [15].

The *incremental cost* model for the general multi-source multi-sink network design problem was introduced by Dahl and Stoer [17], while for SSNLP it was tested in [34, 38]. Raghavan and Stanojević [34] proved that for SSNLP both models are equivalent in terms of quality of lower bounds, and their LP relaxations both approximate the monotonically increasing step cost function by its lower convex envelope. A more general equivalence result for generic minimization problems with separable non-convex piecewise linear costs is given by Croxton et al. [15], see also Keha et al. [27].

The multiple choice modeling approach leads to the following *single-commodity flow* formulation: Binary variables $x_{ij,n} \in \{0,1\}$ decide whether the module n shall be installed on the arc (i,j), whereas flow variables $f_{ij} \geq 0$ describe the amount of flow on arc $(i,j) \in A$. Then the SCF model is: R2.2

$$\text{SCF} \quad : \quad \min \sum_{(i,j)\in A} l_{ij} \sum_{n\in\mathcal{N}_{ij}} c_{ij,n} x_{ij,n}$$

$$\text{s.t.} \quad \sum_{(i,j)\in A} f_{ij} - \sum_{(j,i)\in A} f_{ji} = \begin{cases} -d_i, & i \in D \\ \sum_{k\in D} d_k, & i = r \\ 0, & \text{otherwise} \end{cases} \qquad \forall i \in V \tag{1}$$

$$\sum_{n\in\mathcal{N}_{ij}} x_{ij,n} \leq 1 \qquad \forall (i,j) \in A \tag{2}$$

$$0 \leq f_{ij} \leq \sum_{n\in N} u_{ij,n} x_{ij,n} \qquad \forall (i,j) \in A \tag{3}$$

$$x_{ij,n} \in \{0,1\} \qquad \forall (i,j) \in A,\ \forall n \in \mathcal{N}_{ij}. \tag{4}$$

The *flow conservation constraints* (1) ensure that every customer receives desired amount of flow, while the *capacity constraints* (3) ensure that enough capacity is installed on every arc. The *disjunction constraints* (2) are typical for the *explicit cost model*, i.e., on every arc at most one module may be installed.

Observation 2.3. *Given an optimal solution* (x', f') *of an LP relaxation of* SCF*, the subgraph* G' *of* G *obtained by taking all arcs* $(i,j) \in A$ *such that* $\sum_{n \in \mathcal{N}_{ij}} x'_{ij,n} > 0$ *contains no directed cycle.*

Therefore, the subtour elimination constraints $x_{ij,n} + x_{ji,n} \leq 1$, for all $(i,j) \in A$, and all $n \in \mathcal{N}_{ij}$, are redundant for both the SCF model and its LP relaxation. **However, the LP relaxation is stronger when the disjunction constraints (2) are replaced by:** R2.m8

$$\sum_{n \in \mathcal{N}_{ij}} (x_{ij,n} + x_{ji,n}) \leq 1 \qquad \forall (i,j) \in A.$$

The SCF model contains $O(|A| \cdot |\mathcal{N}|)$ variables and constraints, but due to "big-M" constraints (3), it provides arbitrarily bad lower bounds. In case of economies of scale, the LP relaxation of the SCF model has an optimal solution in which at most one of $x_{ij,n}$ variables (the one with the lowest $c_{ij,n}/u_{ij,n}$ ratio) on every arc is non-zero (see also [38]).

2.3 Multi-Commodity Flow Formulation (MCF)

This formulation is commonly used for the multiple-source multiple-sink network design problems (see, e.g., [29]). In this model, f^k_{ij} describes the amount of flow of commodity $k \in D$ routed through the arc (i,j). Commodities in our case are source-sink pairs $(r,k), k \in D$, i.e., they are directly associated to customers $k \in D$. The MCF model reads as follows:

$$\text{MCF} : \qquad \min \sum_{(i,j) \in A} l_{ij} \sum_{n \in \mathcal{N}_{ij}} c_{ij,n} x_{ij,n}$$

$$\text{s.t.} \qquad \sum_{(i,j) \in A} f^k_{ij} - \sum_{(j,i) \in A} f^k_{ji} = \begin{cases} -d_k, & i = k \\ d_k, & i = r \\ 0, & \text{otherwise} \end{cases} \qquad \forall i \in V,\ \forall k \in D \tag{5}$$

$$\sum_{n \in \mathcal{N}_{ij}} x_{ij,n} \leq 1 \qquad \forall (i,j) \in A \tag{6}$$

$$\sum_{k \in D} f^k_{ij} \leq \sum_{n \in \mathcal{N}_{ij}} u_{ij,n} x_{ij,n} \qquad \forall (i,j) \in A \tag{7}$$

$$0 \leq f^k_{ij} \leq d_k \sum_{n \in \mathcal{N}_{ij}} x_{ij,n} \qquad \forall (i,j) \in A,\ \forall k \in D \tag{8}$$

$$x_{ij,n} \in \{0,1\} \qquad \forall (i,j) \in A,\ \forall n \in \mathcal{N}_{ij}. \tag{9}$$

The *flow conservation constraints* (5) and the *capacity constraints* (7) have the same meaning as for the SCF model. The *coupling constraints* (8) ensure that if there is a flow in any module n on the arc (i,j),

then the corresponding design variable need to be set up. Obviously, these constraints are redundant for the MIP formulation, but they improve the lower bound of the LP relaxation.

The MCF model contains $O(|A|\cdot|\mathcal{N}|+|A|\cdot|D|)$ variables and $O(|V|\cdot|D|+|A|\cdot|\mathcal{N}|+|A|\cdot|D|)$ constraints. The LP relaxations of the SCF model and MCF model without coupling constraints (8) produce the same lower bound.

2.4 Disaggregated Multi-Commodity Flow Formulation (DMCF)

For the multi-commodity capacitated network design problem, Croxton et al. [16] and Frangioni and Gendron [20] proposed a disaggregation by integer values in a MIP based on the multi-commodity flow formulation. **By adapting this disaggregation technique to SSNLP,** we disaggregate flow variables with respect to modules. Beside the binary design variables, $x_{ij,n} \in \{0,1\}$, we use the disaggregated flow variables $f^k_{ij,n}$ that define the amount of flow of commodity $k \in D$, routed through the arc (i,j) using the module $n \in \mathcal{N}_{ij}$. The DMCF model reads then as follows: R2.1

$$\text{DMCF}: \quad \min \sum_{(i,j)\in A} l_{ij} \sum_{n\in\mathcal{N}_{ij}} c_{ij,n} x_{ij,n} \tag{10}$$

$$\text{s.t.} \sum_{(i,j)\in A} \sum_{n\in\mathcal{N}_{ij}} f^k_{ij,n} - \sum_{(j,i)\in A} \sum_{n\in\mathcal{N}_{ji}} f^k_{ji,n} = \begin{cases} -d_k, & i = k \\ d_k, & i = r \\ 0, & \text{otherwise} \end{cases} \quad \forall i \in V,\ \forall k \in D \tag{11}$$

$$\sum_{n\in\mathcal{N}_{ij}} x_{ij,n} \le 1 \qquad \forall (i,j) \in A \tag{12}$$

$$\sum_{k\in D} f^k_{ij,n} \le u_{ij,n} x_{ij,n} \qquad \forall (i,j) \in A, \forall n \in \mathcal{N}_{ij} \tag{13}$$

$$0 \le f^k_{ij,n} \le d_k x_{ij,n} \qquad \forall (i,j) \in A, \forall k \in D, \forall n \in \mathcal{N}_{ij} \tag{14}$$

$$x_{ij,n} \in \{0,1\} \qquad \forall (i,j) \in A, \forall n \in \mathcal{N}_{ij}. \tag{15}$$

The *capacity constraints* (13) ensure that the total flow in module n on arc (i,j) must not exceed the capacity of the given module n. Constraints (14) are redundant for the MIP formulation, but they improve the optimal value of the LP relaxation.

The DMCF model contains $O(|A|\cdot|\mathcal{N}|\cdot|D|)$ constraints and $O(|A|\cdot|\mathcal{N}|\cdot|D|)$ variables, and it is very unlikely that even the most sophisticated MIP solvers may solve instances of moderate size using the DMCF formulation **as a single MIP including all variables and constraints.** Our computational experiments with the DMCF model confirmed this claim (see Section 5). To use the advantage of this strong model, we propose to project out the flow variables and to introduce Benders inequalities instead, keeping the quality of lower bounds, and even improving them by rounding techniques. **Frangioni and Gendron [20, 21] have proposed a column-and-row-generation algorithm for solving this disaggregated model.** R2.3 R2.3

Denote by

$$\begin{aligned}
\mathcal{P}_{\text{SCF}} &:= \{(x,f) \in [0,1]^{|A|} \times \mathbb{R}^{|A|}_{\geq 0} \mid (x,f) \text{ satisfy } (1)-(3)\} \\
\mathcal{P}_{\text{MCF}^-} &:= \{(x,f) \in [0,1]^{|A|} \times \mathbb{R}^{|A||D|}_{\geq 0} \mid (x,f) \text{ satisfy } (5)-(7)\} \\
\mathcal{P}_{\text{MCF}} &:= \{(x,f) \in [0,1]^{|A|} \times \mathbb{R}^{|A||D|}_{\geq 0} \mid (x,f) \text{ satisfy } (5)-(8)\} \\
\mathcal{P}_{\text{DMCF}^-} &:= \{(x,f) \in [0,1]^{|A|} \times \mathbb{R}^{|A||D||\mathcal{N}|}_{\geq 0} \mid (x,f) \text{ satisfy } (11)-(13)\} \\
\mathcal{P}_{\text{DMCF}} &:= \{(x,f) \in [0,1]^{|A|} \times \mathbb{R}^{|A||D||\mathcal{N}|}_{\geq 0} \mid (x,f) \text{ satisfy } (11)-(14)\}
\end{aligned}$$

the polytopes of the LP relaxations of the above MIP models for SSNLP. Denote by $\text{proj}_x(\mathcal{P}) = \{x \in [0,1]^{|A|} \mid \exists (x,f) \in \mathcal{P}\}$ the natural projection on the space of x variables, for any of the polyhedra $\mathcal{P}$ defined above. It is not difficult to see that the following result holds:

Lemma 2.4.

$$\text{proj}_x(\mathcal{P}_{\text{DMCF}}) \subseteq \text{proj}_x(\mathcal{P}_{\text{MCF}}) \subseteq \text{proj}_x(\mathcal{P}_{\text{SCF}}).$$

In addition, there exist instances of the SSNLP for which $\text{proj}_x(\mathcal{P}_{\text{DMCF}}) \subset \text{proj}_x(\mathcal{P}_{\text{MCF}}) \subset \text{proj}_x(\mathcal{P}_{\text{SCF}})$.

Section 5 provides computational evidence of this theoretical result. Furthermore, one can also show the following equivalence.

Lemma 2.5.

$$\text{proj}_x(\mathcal{P}_{\text{DMCF}^-}) = \text{proj}_x(\mathcal{P}_{\text{MCF}^-}) = \text{proj}_x(\mathcal{P}_{\text{SCF}}).$$

R1.5

2.5 Benders Decomposition for DMCF (Benders)

R2.4

Magnanti and Wong [31] emphasize that for any MIP, the tighter the LP relaxation of a model, the better Benders cuts can be produced. Therefore, we propose to solve the disaggregated model DMCF by projecting out flow variables by dynamically generating the corresponding violated Benders inequalities. Similar ideas **have** been applied to the weaker MCF model for several related problems, see e.g. [13, 14, 22].

R2.m11

R2.m12

Let the master problem be the one given by the objective function (10) subject to constraints (12) and (15). A solution $x' \in [0,1]^{|A||N|}$ of the master problem defines a feasible solution for the LP relaxation of the DMCF model if and only if there exist flow variables $f^k_{ij,n}$, $(i,j) \in A, k \in D, n \in \mathcal{N}$ satisfying the linear system of inequalities given by (11), (13) and (14), where $x = x'$.

Farkas' lemma states that a linear system of equations $\{Ax \leq b, x \geq 0\}$ has a solution if and only if $u^T b \geq 0$ for all $u \geq 0$ such that $u^T A \geq 0$. To apply Farkas' lemma to the system (11), (13), (14), we define a dual variable α^k_i associated to each equation (11), a dual variable $\beta^k_{ij,n}$ associated to each inequality (14), and a dual variable $\gamma_{ij,n}$ associated to each equation in (13). Then, the polyhedron defined by this system is non-empty if and only if the following Benders decomposition subproblem SUB is bounded (i.e., its optimal value is equal to zero):

$$\text{SUB} : \min \quad z(\alpha,\beta,\gamma,x') = \sum_{k \in D} d_k(\alpha^k_r - \alpha^k_k) + \sum_{(i,j)\in A} \sum_{n \in N_{ij}} \left(\sum_{k \in D} d_k \beta^k_{ij,n} + u_{ij,n}\gamma_{ij,n} \right) x'_{ij,n} \qquad (16)$$

$$\text{s.t. } \alpha_i^k - \alpha_j^k + \beta_{ij,n}^k + \gamma_{ij,n} \geq 0 \qquad \forall (i,j) \in A,\ \forall k \in D,\ \forall n \in \mathcal{N}_{ij} \tag{17}$$

$$(\alpha, \beta, \gamma) \geq 0. \tag{18}$$

Violated Benders inequalities can be found and used within a branch-and-cut framework as follows. We first solve the linear relaxation of the master problem, obtaining a fractional solution $x' = [x'_{ij,n} : (i,j) \in A, n \in \mathcal{N}_{ij}]$. With these values x' we define the corresponding subproblem SUB. Note that this subproblem can be either bounded or unbounded. In the latter case, there is an unboundedness direction $(\alpha', \beta', \gamma')$ for which $z(\alpha', \beta', \gamma', x') < 0$. To avoid this situation the *Benders cut*

$$\sum_{(i,j)\in A} \sum_{n \in N_{ij}} \left(\sum_{k \in D} d_k \beta'^k_{ij,n} + u_{ij,n} \gamma'_{ij,n} \right) x_{ij,n} \geq \sum_{k \in D} (\alpha'^k_k - \alpha'^k_r) d_k \tag{19}$$

must be added to the master problem. **Let $M = \sum_{k \in D} (\alpha'^k_k - \alpha'^k_r) d_k$. Observe that we can round down the coefficients multiplying $x_{ij,n}$ by setting them to $\mu_{ij,n} = \min(M, \sum_{k \in D} d_k \beta'^k_{ij,n} + u_{ij,n} \gamma'_{ij,n})$. The resulting inequality:**

$$\sum_{(i,j)\in A} \sum_{n \in N_{ij}} \mu_{ij,n} x_{ij,n} \geq M \tag{20}$$

is valid and it is called *rounded Benders inequality.* R2.5

The process is iterated until the linear relaxation of the master problem has been solved to optimality, that is, until the subproblem SUB is unable to find more violated Benders cuts. If $x_{ij,n}$ variables are all integer, the SSNLP is solved. Otherwise, we resort to branching.

We refer to this implementation of the separation of violated Benders inequalities as the *textbook implementation*. In Section 5, we compare this basic implementation with more elaborated variants proposed in Section 3.

2.6 Directed Cut-Set Formulation (DCut)

We now recall the cut-set formulation for SSNLP on directed graphs. For each subset $S \subset V$, we will denote by $\delta^+(S) := \{(i,j) \in A : i \in S, j \in V \setminus S\}$ and $\delta^-(S) := \{(i,j) \in A : i \in V \setminus S, j \in S\}$, outgoing and ingoing cuts, respectively.

Projecting out aggregated flow variables $f_{ij} = \sum_{k \in D} \sum_{n \in \mathcal{N}_{ij}} f^k_{ij,n}$ for all $(i,j) \in A$ leads to the following cut-set inequalities:

$$\sum_{(i,j)\in\delta^+(S)} \sum_{n \in \mathcal{N}_{ij}} u_{ij,n} x_{ij,n} \geq \sum_{k \in D \setminus S} d_k \qquad \forall S \subset V : r \in S, S \cap D \neq D. \tag{21}$$

The separation problem of cut-set inequalities in general multiple-source multiple-sink case is NP-hard and can be reduced to the max-cut problem [4]. However, inequalities (21) for SSNLP can be separated in polynomial time as follows. For a given fractional solution x', we define the directed *support graph* $G' = (V', A')$ where $V' := V \cup \{t\}$ with an additional sink t, and $A' := A_1 \cup A_2$ being $A_1 := \{(i,j) \in A : \sum_{n \in N_{ij}} u_{ij,n} x'_{ij,n} > 0\}$ and $A_2 := \{(k,t) : k \in D\}$. The capacity associated to each arc $a = (i,j) \in A_1$

is set to $\sum_{n\in N} u_{ij,n}x'_{ij,n}$, and the capacity of each arc $a=(k,t)\in A_2$ is set to d_k. If the minimum cut between r and t in G' is less than $\sum_{k\in D} d_k$, there is a violated inequality (21).

Since $x_{ij,n}$ variables are binary, the cut-set inequalities can also be strengthened as follows:

$$\sum_{(i,j)\in\delta^+(S)}\sum_{n\in\mathcal{N}_{ij}} \min\left(u_{ij,n}, \sum_{k\in D\setminus S} d_k\right) x_{ij,n} \geq \sum_{k\in D\setminus S} d_k.$$

Observe that, due to "big-M" constants referring to capacities, the LP relaxation obtained by solving the DCut formulation can be arbitrarily bad. In fact, constraints (21) correspond to Benders inequalities obtained by projecting out flow variables from the SCF model and therefore the DCut model is equivalent to the SCF model with respect to the quality of lower bounds of their LP relaxations.

The DCut model can be improved with additional *connectivity constraints*, i.e.:

$$\sum_{(i,j)\in\delta^+(S)}\sum_{n\in\mathcal{N}} x_{ij,n} \geq 1 \quad \forall S\subset V : r\in S, S\cap D\neq D, \tag{22}$$

leading to a new model DCut$^+$. Constraints (22) can be derived from the MCF model by dualizing flow-constraints (5) and (8).

Denote by

$$\begin{aligned}\mathcal{P}_{\text{DCut}} &:= \{x\in[0,1]^{|A|} \mid x \text{ satisfy (21), (2)}\},\\ \mathcal{P}_{\text{DCut}^+} &:= \{x\in\mathcal{P}_{\text{DCut}} \mid x \text{ satisfy (22) }\}.\end{aligned}$$

Model DCut$^+$ is weaker than the MCF model because the flow variables in the MCF model simultaneously need to satisfy the capacity constraints (7) and (8), while the cut-set inequalities (21) and (22) ensure the existence of two independent flows only. Thus, we have:

Lemma 2.6.

$$\text{proj}_x(\mathcal{P}_{\text{MCF}}) \subseteq \mathcal{P}_{\text{DCut}^+} \subseteq \mathcal{P}_{\text{DCut}} = \text{proj}_x(\mathcal{P}_{\text{SCF}}).$$

In addition, there exist instances of the SSNLP for which $\text{proj}_x(\mathcal{P}_{\text{MCF}}) \subset \mathcal{P}_{\text{DCut}^+} \subset \mathcal{P}_{\text{DCut}}$ *holds.*

2.7 Further Strengthening Inequalities

We now address two families of valid inequalities that may improve the lower bound obtained by solving the LP relaxations of the above MIP models for SSNLP.

2.7.1 Degree-Balance Constraints

Non-customer nodes $V\setminus(D\cup\{r\})$ cannot have incoming (or outgoing) arcs only. Therefore, we can add the following *degree-balance constraints* that only work for **the** single source case: R2.m9

$$\sum_{(l,i)\in A, l\neq j}\sum_{n\in\mathcal{N}_{li}} x_{li,n} \geq \sum_{n\in\mathcal{N}_{ij}} x_{ij,n} \qquad \forall (i,j)\in A, i\notin D, i\neq r \tag{23}$$

$$\sum_{(j,l)\in A, l\neq i}\sum_{n\in\mathcal{N}_{jl}} x_{jl,n} \geq \sum_{n\in\mathcal{N}_{ij}} x_{ij,n} \qquad \forall (i,j)\in A, j\notin D, j\neq r. \tag{24}$$

Inequality (23) states that if an arc (i,j) emanating from a non-customer node i is being used in the solution, there must be at least one arc entering i. Thanks to Lemma 2.1 the opposite arc (j,i) can be excluded from the summation on the left hand side. Inequality (24) states the opposite case for an arc (i,j) entering a non-customer node j.

Observe that capacitated versions of these cuts, i.e.:

$$\sum_{(l,i)\in A, l\neq j} \sum_{n\in\mathcal{N}_{li}} u_{li,n}x_{li,n} \geq \sum_{n\in\mathcal{N}_{ij}} u_{ij,n}x_{ij,n} \qquad \forall (i,j)\in A, i\notin D, i\neq r$$

$$\sum_{(j,l)\in A, l\neq i} \sum_{n\in\mathcal{N}_{jl}} u_{jl,n}x_{jl,n} \geq \sum_{n\in\mathcal{N}_{ij}} u_{ij,n}x_{ij,n} \qquad \forall (i,j)\in A, j\notin D, j\neq r.$$

are *not valid* in general, but only if there is a uniform capacity/cost structure on edges.

2.7.2 Cover Inequalities

The following inequalities are of particular interest for the DCut model. Given a cut-set inequality (21) defined by $S\subset V, r\in S$, define the index set $I(S) := \{(i,j,n) \mid (i,j)\in\delta^+(S), n\in\mathcal{N}_{ij}\}$ and $B := \sum_{k\in D\setminus S} d_k$. Set $M\subset I(S)$ is called a *cover* with respect to $I(S)$ if $\sum_{(i,j,n)\in M} u_{ij,n} < B$ and a *maximal cover* if, in addition, for all M' such that $I(S)\supseteq M'\supset M$: $\sum_{(i,j,n)\in M'} u_{ij,n}\geq B$. If M is a maximal cover with respect to $I(S)$, then the following *cover inequalities* are valid:

R1.4

R2.m10

$$\sum_{(i,j,n)\in I(S)\setminus M} x_{ij,n} \geq 1. \tag{25}$$

In general, the separation problem of cover inequalities is NP-hard. We show that the problem of finding the most violated cover inequality (25) is equivalent to solving the *precedence constrained knapsack problem*. **Recall** that indices $n\in\mathcal{N}_{ij}$ are sorted according to increasing arc capacities. To model any cover M with respect to $I(S)$, we introduce the binary variables $z_{ij,n}$ that are equal to one if and only if $(i,j,n)\in M$. For every arc $(i,j)\in\delta^+(S)$, we define $u_{ij,0}=0$.

R2.m5

For a given fractional solution x' and an index set $I(S)$ induced by a cut-set inequality, the *most violated cover inequality* can be found by solving the following model:

$$\begin{aligned}
\text{KNAP}: \quad & \max \sum_{(i,j,n)\in I(S)} x'_{ij,n} z_{ij,n} \\
& \sum_{(i,j,n)\in I(S)} (u_{ij,n} - u_{ij,n-1}) z_{ij,n} < B \qquad (26)\\
& z_{ij,n} \geq z_{ij,n+1}, && \forall (i,j,n)\in I(S),\ n < |\mathcal{N}_{ij}| \\
& z_{ij,n}\in\{0,1\}, && \forall (i,j,n)\in I(S).
\end{aligned}$$

Let z' be an optimal solution of model KNAP. The corresponding cover inequality reads then as follows:

$$\sum_{(i,j,n)\in I(S)} (1 - z'_{ij,n}) x_{ij,n} \geq 1.$$

If all capacities and demands are integers, (26) can be replaced by $\sum_{(i,j,n)\in I(S)}(u_{ij,n} - u_{ij,n-1})z_{ij,n} \leq B - 1$. The cover inequalities are similar to the *band inequalities* for the incremental cost model given in [17].

2.8 Hierarchy of Formulations

The hierarchical scheme given in Figure 1 summarizes the relationships between the LP relaxations of the basic MIP models considered throughout this paper for SSNLP. A filled arrow specifies that the target formulation is strictly stronger than the tail formulation. An empty arrow specifies that the target formulation is at least as strong as the tail formulation. DCut$^+$ denotes the cut-set formulation extended by connectivity inequalities (22). Benders$^+$ denotes the model with rounded Benders cuts (20).

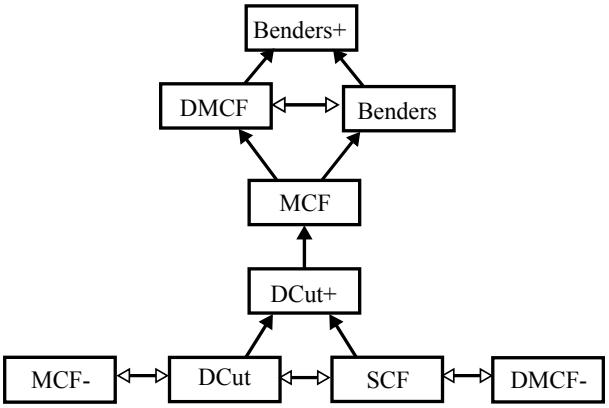


Figure 1: Hierarchy of LP relaxations.

3 Separation of Benders Cuts

R2.5

In this section we propose several ways of generating Benders cuts associated to extreme rays. We show different normalization techniques to make the primal subproblem feasible. An extreme point that solves the normalized subproblem induces a Benders cut associated to an extreme ray. Costa et al. [14] have proposed an approach for strengthening Benders cuts associated to non-extreme rays for a general multi-commodity network design problem. It should be noted that their ideas could similarly be extended to our Benders cuts in case non-extreme rays were generated. R2.5

3.1 Separation Models

R2.5

Any implementation of Benders cut's separation heavily affects the overall performance of a MIP approach. In our work, we considered the straight-forward implementation of solving the dual subproblem SUB as defined in Section 2.5, and three *normalization approaches* obtained by closing the dual un-

bounded cone (see Table 1 for an illustration). For each of these models, we explicitly solved the dual or the primal version of the subproblem.

Dual	Primal	Explanation
SUB	-	see Section 2.5
SUBc	PSUBc	SUB extended by $(\alpha, \beta, \gamma)^t \cdot \mathbf{1} = 1$
SUBn	PSUBn	SUB extended by $\sum_{k \in D} d_k(\alpha_k^k - \alpha_r^k) = 1$
SUBf	PSUBf	SUB extended by $\alpha^t \cdot \mathbf{1} = 1$

Table 1: Different normalization approaches for separating Benders cuts.

SUB **Model:** In order to get a violated Benders inequality, we search for an extreme ray of the unbounded subproblem SUB. As already observed in [5, 18], this approach has a significant drawback: it returns a randomly chosen extreme ray without having any positive influence on the quality of the violated cut found. An advantage of this method is that it returns a violated constraint much faster than the corresponding more sophisticated methods described below.

SUBc **and** PSUBc **Models:** Instead of solving the subproblem on the pointed unbounded cone, one can close it by adding the following hyperplane:

$$\sum_{(i,j) \in A} \sum_{n \in N_{ij}} \sum_{k \in D} \beta_{ij,n}^k + \sum_{(i,j) \in A} \sum_{n \in N_{ij}} \gamma_{ij,n} + \sum_{i \in V} \sum_{k \in D} \alpha_i^k = 1. \tag{27}$$

Obviously, the Benders cut generated by solving SUB extended by (27) is violated if and only if the objective value is strictly less than zero. Furthermore, each vertex of such obtained polyhedron (except the origin) corresponds to an extreme ray of the unbounded subproblem.

One easily observes that the model SUBc is equivalent to the similar problem of maximizing the value of $\Theta \leq 0$ subject to constraints (11), (13) and (14) in which Θ is added to the left-hand side of each of them. This primal model is denoted by PSUBc.

SUBn **and** PSUBn **Models:** Recall that before inserting a cut into the master problem we check its violation according to (28). Since we are interested in looking for a maximally violated Benders cut, one can normalize the subproblem by fixing the right-hand side to one. The SUB model is therefore extended by the following constraint:

$$\sum_{k \in D} d_k(\alpha_k^k - \alpha_r^k) = 1.$$

The resulting subproblem SUBn is bounded and its solution (if negative) always corresponds to the most violated Benders cut according to (28). Again, the master solution x' is infeasible if and only if the solution of SUBn is strictly less than zero. The primal of SUBn, denoted by PSUBn, is known as the *maximum concurrent flow model* (see, e.g., [8]), and it has been used by Avella et al. [3] for separation of metric inequalities.

SUBf **and** PSUBf **Models:** One can also consider the following flow feasibility subproblem:

$$\text{PSUBf}: \quad \max \Theta$$

$$\sum_{(i,j)\in A} \sum_{n\in\mathcal{N}_{ij}} f^k_{ij,n} - \sum_{(j,i)\in A} \sum_{n\in\mathcal{N}_{ji}} f^k_{ji,n} + \Theta = \begin{cases} d_k, & i = r \\ -d_k, & i = k \\ 0, & \text{otherwise} \end{cases} \qquad \forall i \in V,\ \forall k \in D$$

$$\sum_{k\in D} f^k_{ij,n} \le u_{ij,n} x'_{ij,n} \qquad \forall (ij) \in A,\ \forall n \in \mathcal{N}_{ij}$$

$$0 \le f^k_{ij,n} \le d_k x'_{ij,n} \qquad \forall (i,j) \in A,\ \forall k \in D,\ \forall n \in \mathcal{N}_{ij}$$

$$\Theta \le 0.$$

This problem has a nice flow structure that can easily be recognized by an LP solver (like Cplex), therefore we consider it as another alternative normalization approach for finding a violated Benders inequality. In the corresponding dual variant of the model, denoted by SUBf, we extend SUB with $\sum_{k\in D} \sum_{i\in V} \alpha^k_i = 1$.

SUBcap **and** PSUBcap **Models:** In our preliminary computational experiments we also tried the variant in which Θ is added only to capacity and coupling constraints (13) and (14), respectively. The latter model is a generalization of the *capacity reduction problem*, used by Avella et al. [3] to generate the so-called *strong metric inequalities* for the MCF model of NLP. However, in our preliminary results, both SUBcap and PSUBcap were significantly outperformed by other models mentioned above. In particular, these submodels could not be solved to optimality within our default time limit for solving Benders subproblems. Therefore, we do not report results for these models in Section 5.

3.2 Further Potential Enhancements

We implemented the following additional techniques that are known to significantly improve the performance of (Benders) separation algorithms, in general.

Connectivity Cuts: We separate *nested*, *backward* and *minimum cardinality* connectivity cuts (22), that are used as a standard procedure for accelerating cutting plane methods (see, e.g. [28]). The idea of separating nested cuts is to look for violated inequalities whose coefficient vector is orthogonal to an already detected cut. Using backward cuts one can generate two violated inequalities per a single max-flow computation. Minimum cardinality cuts search for a cut set with the smallest number of arcs having the same max-flow value. We separate up to 100 connectivity cuts per iteration. For finding the maximum flow in a directed graph, we used an adaptation of Cherkassky and Goldberg's maximum flow algorithm [9]. **This procedure was turned on for producing the results reported in Section 5.**

R1.5

Disjoint Benders Cuts: We also developed a separation procedure that produces several disjoint Benders cuts starting from the same solution of the master problem. It works as follows: assume that $(\alpha', \beta', \gamma')$ corresponds to an unbounded direction in the current Benders subproblem. We fix to zero the variables β and γ with positive value in this solution. Now, we again solve the Benders subproblem with the fixed variables. This may give us another unbounded direction. By repeating this procedure, one may generate a set of disjoint Benders cuts.

This separation procedure is very similar to the one used to separate disjoint (*nested*) connectivity inequalities mentioned above. However, whereas nested connectivity cuts drastically improve the performance, there is a drawback of separating disjoint Benders cuts: solving a single Benders subproblem is computationally more expensive than resolving the primal master LP. Therefore, this procedure was turned off for producing the results reported in Section 5. R2.7

Magnanti-Wong Implementation: The ideas of **Magnanti and Wong [31] have** been widely used for accelerating separation of Benders cuts (see, e.g., [30, 35]). The authors proposed to accelerate the convergence of the basic Benders algorithm by adding Pareto-optimal Benders cuts. A Pareto-optimal Benders cut is given by the following definition: a cut $z(\alpha'', \beta'', \gamma'', x) \geq 0$ *dominates* another cut $z(\alpha', \beta', \gamma', x) \geq 0$ if and only if $z(\alpha', \beta', \gamma', x) \geq z(\alpha'', \beta'', \gamma'', x)$ for all $x \in \{0,1\}^{|A||\mathcal{N}|}$ satisfying (2), and the strict inequality holds for at least one x. A Benders cut is said to be *Pareto-optimal* if no other cut dominates it. In case that there are multiple optimal solutions to the Benders subproblem, Magnanti and Wong have proposed an approach to search for a Pareto-optimal cut by solving an additional subproblem in the separation phase: R2.m13 R2.m14

1. Given a fractional solution x', solve the Benders subproblem SUBx to get a violated cut defined by $(\alpha', \beta', \gamma')$. If $z(\alpha', \beta', \gamma', x') = 0$, no violated cut exists. Stop.
2. Set $z' := z(\alpha', \beta', \gamma', x')$.
3. Solve the new subproblem defined as:

$$\min\{z(\alpha, \beta, \gamma, x_0) \mid (\alpha, \beta, \gamma) \text{ satisfy the constrains in SUBx and } z(\alpha, \beta, \gamma, x') = z'\}.$$

4. Denote by $(\alpha'', \beta'', \gamma'')$ the solution to this subproblem. Then, the cut $z(\alpha'', \beta'', \gamma'', x) \geq 0$ is inserted into the master problem.

In this procedure, SUBx denotes any of the normalized variants (bounded subproblems) described above and x_0 is a given fractional solution called *core point*, i.e., a point that belongs to the relative interior of the convex hull of all binary vectors x satisfying (2). As already observed by Papadakos [33], for the above procedure to work **efficiently,** one needs to start it with a different core point every time the R2.m15

procedure is applied. For that purpose, we start with a randomly chosen point from the interior, and later we generate a random convex combination of two incumbent solutions.

The obvious drawback of this procedure **is that** we have to solve two time-consuming subproblems within each separation. Furthermore, solving the Magnanti-Wong subproblem is computationally more expensive **than** solving the master problem. **In our default implementation, the separation of Pareto-optimal cuts is turned off. In Section 5, we report on the effects obtained by turning on this procedure.** R2.m16 R2.m17 R1.5

4 Branch-and-Cut Algorithm

The overall approach is a branch-and-cut algorithm in which the LP relaxation of the compact model SCF is solved first, followed by the separation of further strengthening inequalities. More precisely the approach works as follows:

1. Initialization:
   (a) Initialize the master problem with the variables and constraints of the SCF model extended by additional inequalities (see Section 4.1).
   (b) Solve the LP relaxation of the master.
2. In every k-th node of the branch-and-bound tree:
   (a) As long as there are violated connectivity inequalities (22), add them to the master LP and resolve it. R1.3
   (b) Based on the current fractional solution x', create the Benders subproblem.
   (c) Solve the subproblem. If this results in a violated Benders cut, add it to the master LP and resolve it.

Since in Step 1(a) the master is initialized with the SCF model, Benders cuts generated in Step 2(b) are only used to strengthen the LP bounds. In other words, Benders cuts are not needed to guarantee the feasibility of a solution. This inspires another strategy in which Benders cuts are added only at the root node of the branch-and-bound tree.

In the context of a multiple-source multiple-sink variant of the problem, the above strategies are not valid. Either the master must be initialized with a much larger (multi-commodity flow) model or k must be equal to 1. R1.7

As an alternative approach to the initialization of lower bounds by the SCF model, we also **implemented a branch-and-cut algorithm for solving the DCut formulation. The cut-set inequalities (21)** can be separated very fast. However, our preliminary results have shown that using SCF directly results in a clearly preferable approach in practice. **For that reason, we do not report DCut results in Section 5. Furthermore, since cover inequalities (25) are derived from cut-set** R1.3 R1.3

inequalities (21), their separation is not implemented in our computational framework. R1.4

4.1 Initialization of Lower Bounds

In order to obtain good, yet easily computable, lower bounds that will avoid expensive computation of "trivial" Benders cuts, we extend the SCF model by constraints (23), (24) and the following in-degree inequalities and root-out-degree inequalities:

$$\sum_{(i,j)\in A}\sum_{n\in\mathcal{N}_{ij}} x_{ij,n} \geq 1 \qquad \forall j \in D$$
$$\sum_{(r,j)\in A}\sum_{n\in\mathcal{N}_{rj}} x_{rj,n} \geq 1.$$

Obviously, these inequalities are special cases of connectivity cut-set inequalities (22) for $S = V \setminus \{j\}$ and $S = \{r\}$, respectively.

4.2 Branch-and-Cut Parameters

To improve the overall performance and to avoid numerical difficulties we consider the following two standard ingredients:

- Tailing Off: If the relative improvement of the lower bound is less than `Eps`% in the last `It` iterations of the separation procedure, we stop the separation and resort to branching. The general setting of (`It`,`Eps`) is $(20, 10^{-3})$. However, if only the computationally more expensive Benders cuts were separated in recent iterations, a stricter setting of (`It'`,`Eps'`) $= (10, 10^{-3})$ is applied.

- Degree of Violation: Assume that after solving the Benders subproblem for a given fractional value x', we obtain a violated cut defined by a vector $(\alpha', \beta', \gamma')$. Before inserting the corresponding cut into the master LP, we normalize it by dividing it with its right-hand side (which is always positive) and calculate its violation by the current fractional solution x' as follows:

$$\text{violation}(\alpha', \beta', \gamma', x') = 1 - \sum_{(i,j)\in A}\sum_{n\in N_{ij}} \frac{\sum_{k\in D} d_k \beta'^k_{ij,n} + u_{ij,n}\gamma'_{ij,n}}{\sum_{k\in D} d_k(\alpha'^k_k - \alpha'^k_r)} x'_{ij,n} \tag{28}$$

  If $\text{violation}(\alpha', \beta', \gamma', x') < 10^{-4}$, the cut will not be considered as violated and will not be inserted into the system.

The flowchart in Figure 2 explains how we implemented our cut separation procedure.

4.3 Primal Heuristic

We employ the following rounding heuristic based on min-cost-flow. Denote the total installed capacity on an arc by $X_{ij}(x) = \sum_{n\in\mathcal{N}_{ij}} u_{ij,n}x_{ij,n}$. The cheapest fitting module to support a certain capacity U is denoted by $n(U) = \arg\min_{\{n\in\mathcal{N}_{ij}|u_{ij,n}\geq U\}} c_{ij,n}$. Starting from a fractional solution x, we create a binary solution x' and subsequently a cheaper binary solution x''. Initialize $x' := 0$. Now for every arc

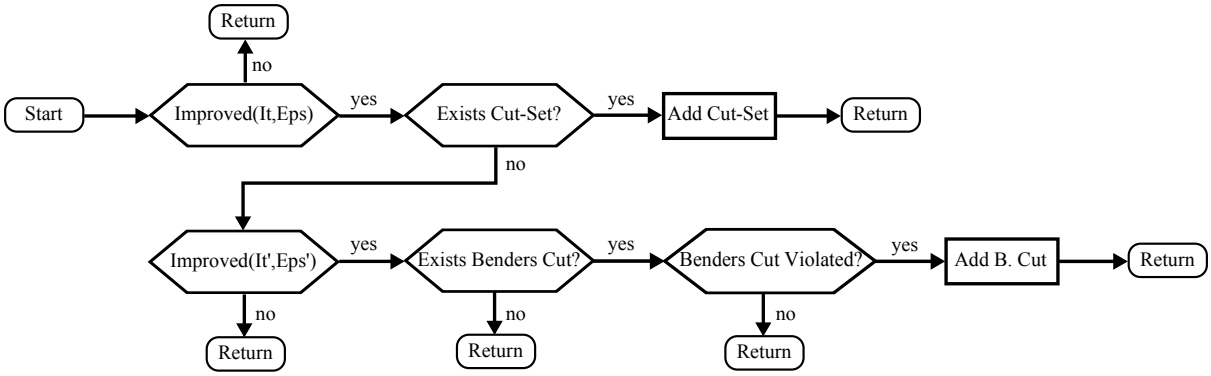


Figure 2: Separation of cuts in the branch-and-cut framework.

(i,j) install the cheapest fitting module, i.e. $x'_{ij,n(U)} := 1$, **with $U := X_{ij}(x)$.** The resulting x' is integer feasible, but also typically overly generous and can be improved. To this end we use an augmented graph with an additional sink t, similar to the one from Section 2.6: Let $G' = (V', A')$ where $V' = V \cup \{t\}$ and $A' = A \cup \{(k,t) : k \in D\}$. The arc capacities are set to $X_{ij}(x')$ for all $(i,j) \in A$ and d_k for all (k,t), $k \in D$. Arc costs are defined as $\sum_{n \in \mathcal{N}_{ij}} c_{ij,n} x'_{ij,n}$. Initialize $x'' := 0$. We now compute the min-cost-flow $f \in \mathbb{R}^{|A|}$ in G'. This induces our new incumbent candidate $x'' : x''_{ij,n(f_{ij})} := 1$. R2.m18

We use the min-cost-flow implementation based on capacity scaling and successive shortest path computation found in the commercial library LEDA, 5.2 (see [1, 2]). This algorithm only works for integer capacity and cost values. Therefore we round these values to the nearest integer prior to the min-cost-flow computation. A result of this rounding is that x'' will, on rare occasions, be infeasible. This is easily detected by a subsequent computation of a max-flow and an infeasible x'' is discarded.

5 Computational Results

This section reports on our computational experience with the proposed branch-and-cut framework. We implemented our algorithms using C++ and Cplex 11.1 [26]. An Intel Core 2 personal computer with 1.8 GHz and 3.25 GB of RAM was used for testing purposes. If not mentioned otherwise, the default Cplex settings are used.

We set a time limit of 1000 seconds for solving benchmark instances. For all instances that cannot be solved to optimality because that limit was reached, we report the gap between the best known upper bound (UB) and the lower bound (LB) obtained, calculated as $\frac{UB-LB}{UB} \cdot 100\%$. The time limit for each single separation of Benders cuts was set to 45 seconds.

5.1 Preprocessing

All reported instances are preprocessed according to the following rules:

(i) Each customer node $k \in D$ with degree one is joined with his neighbor using the cheapest module

that allows the flow of d_k to be routed.

(ii) Each non-customer node with degree one is simply deleted.

(iii) Each non-customer-node with degree two and with the same modules on both sides is replaced by a single edge, with the same modules.

(iv) Each non-customer-node with degree two and different modules on both **sides** is replaced by an edge with all possible module combinations. R2.m19

(v) Rules (iii) and (iv) may result in parallel edges. Parallel edges are replaced by a single one with all possible module combinations.

(vi) Rules (iv) and (v) may lead to dispensable modules. A module $n \in \mathcal{N}_e$ is dispensable if there exists another module $n' \in \mathcal{N}_e$ with $u_{e,n'} \geq u_{e,n}$ and $c_{e,n'} \leq c_{e,n}$. Dispensable modules are deleted.

(vii) Sets of excess modules $N'_e = \{n \in \mathcal{N}_e \mid u_{e,n} \geq \sum_{k \in D} d_k\}$ are replaced by a single module n' with $c_{e,n'} = min_{n \in N'_e} c_{e,n}$ and $u_{e,n'} = \sum_{k \in D} d_k$.

Observe that rules (iv) and (v) may generate instances with non-uniform modules, even if the modules of the original instance were uniform.

5.2 Benchmark Instances

Instances from Salman [36]: Salman instances include four problems originally defined in [23] (problems ARPA, OCT, USA, and RING) and 60 problems randomly generated by Salman [36], also used in [38]. For the latter ones, there are 12 groups with 20, 30 and 40 nodes. There are 9 cable types obeying economies of scale. The cheapest cable type has capacity of 6 – see [6, 38] for a detailed description. The convex combinations of these generate up to $\lceil \frac{\sum_k d_k}{6} \rceil$ modules. The notation `e(n)(s)(d)` provides summary information on the instances: `n` denotes the number of nodes, `s` explains the location of the root node (`c` stands for *central*, `r` stands for *random position*), `d` explains the level of demand (`l` stands for *low demand*, which is randomly generated between 0 and 30; `h` stands for *high demand*, randomly generated between 0 and 60). In [34, 38] two kinds of experiments were performed: using all 9 cable types and using only 4 of them. Since our method does not depend on the number of cable types, but on the number of modules, we only performed the more challenging variant involving all 9 cable types. Table 3 provides input information on Salman instances: each of twelve `e(n)(s)(d)` groups contains 5 instances, and the average values per group are reported. The number of nodes ($|V|$), the number of edges ($|E|$), the number of customers ($|D|$), and the number of modules ($|\mathcal{N}|$) represent the averaged values obtained after preprocessing.

Real-World Instances (Bregenz): We used the street map of the Austrian city Bregenz with 1014 nodes and 1191 edges as underlying network. As customers we considered 4 different sets of nodes

with cardinalities $|D| \in \{29, 36, 45, 67\}$. We classified the instances into two groups: Group `H` contains graphs with *higher demands*, i.e., each customer has a demand randomly chosen from $\{4, 8, 12, 16, 20\}$; Group `L`, in contrast, contains graphs with *lower* demand, i.e., each customer is assigned a demand of 4 units.

We employed up to four different modules as displayed in Table 2. These modules imitate real-world data we obtained from an Austrian telecommunication company. In particular, not all of these modules obey economies of scale: it is possible that there are empty conduits with limited modular capacity available at low costs, but if higher capacities need to be installed, new trenches need to be prepared, which involves very high investment costs.

Type	$\|\mathcal{N}\|$	(capacity $u_{ij,n}$, cost $c_{ij,n}$) ...
A	2	(120, 7.0), (1020, 146.0)
B	2	(30, 2.2), (1020, 146.0)
C	3	(30, 2.2), (60, 4.0), (1020, 146.0)
D	4	(30, 2.2), (60, 4.0), (120, 7.0), (1020, 146.0)

Table 2: The four different sets of modules used for Bregenz instances.

The preprocessing greatly reduces the size of the graph and the number of customers goes down to 28, 33, 41 and 61. Furthermore, although we start with uniform modules, we end up with non-uniform ones. Table 5 illustrates that: the number of minimal, maximal and average number of modules per arc is given in columns $|\mathcal{N}_{\min}|$, $|\mathcal{N}_{\max}|$ and $|\mathcal{N}_{\text{avg}}|$, respectively.

5.3 Solving Salman Instances

We first report on the results with the three compact MIP models presented in Section 2: DMCF, MCF and SCF. We also compared the branch-and-cut approaches based on different separation models as explained in Section 3.1. The main goal of this study was: a) to compare the qualities of lower bounds obtained by solving compact models versus branch-and-cut approaches, and b) to determine whether there is a difference in the performance of the branch-and-cut approach when the textbook implementation is compared against normalized separation approaches. For that purpose, we wanted to ensure that the obtained results are not biased by the quality of incumbent solutions found by the MIP solver. **In previous computations we determined best known upper bounds (BKUB) for all instances using the heuristic described in Section 4.3. For the reported results we initialized all the models with BKUB and turned off the heuristic calls.** For this particular test, we also turned Cplex cuts and the presolver off. Benders cuts are separated at the root and in each 10th node of the branch-and-bound tree. R1.5

Table 3 provides values averaged over 5 instances per group, for `e(n)(s)(d)` instances, and the values for additional four instances from [23].

Gap at the root node: In Table 3 we first report on the quality of LP relaxations of three compact models and the corresponding value of the LP relaxation at the root node of the branch-and-bound tree for the SUBc approach. The gaps between obtained lower bounds and the best known upper bound (provided in column *UB*) are shown. These results are coherent with our theoretical discussion provided in Section 2.8. The SUBc approach was the one among all branch-and-cut approaches to provide the tightest lower bounds at the root node. The average (median) gap over all 64 instances of the SUBc approach is 6.0% (5.9%). The worst LP relaxation gap among Benders approaches is obtained by solving the SUB model: the average (median) gap is 7.5% (7.5%).

Comparing compact formulations, we observe that the average (median) gap of the SCF model of 19.6% (18.6%) can be improved to 9.1% (9.1%) by solving the MCF model, which can further be improved to 5.9% (6.0%) by solving the DMCF formulation. **However, the LP relaxation of the DMCF model was not solved for 4 out of 20 instances of the group e40 within the time limit.** R1.8

Looking at gaps of the SUBc approach and the DMCF model, we can observe two different effects. In some cases SUBc produces better gaps. This results from tightening Benders cuts by rounding down the coefficients (see groups `e20_c_l`, `e40_c_h` in Table 3). In other cases the gap of SUBc is slightly worse than the one of DMCF. This is explained by tailing-off and violation checks. Particularly, if at some point the current Benders cut does not satisfy the violation test (28) and we decide not to add this particular cut and instead resort to branching, the lower bound at the root node will be slightly worse than the value of the LP relaxation of DMCF.

Gap after the time limit: For the SCF model and for Benders separation approaches Table 3 also reports the lower-bound gap after the time limit was reached. Every single variant of our branch-and-cut approach **outperformes** the compact SCF model. The best results are obtained by solving the SUBf approach: the average (median) gap after 1000 seconds is 2.5% (2.5%), while SCF terminates at 5.5% (5.6%). R2.m1

SUBf solves 14 out of 20 instances of group `e20` to optimality, while SCF finds optima only in 7 out of 20 cases. Despite the bad quality of gaps of the LP relaxation, the model SCF succeeds to improve the final gap by drawing the advantage of branching. The average number of branch-and-bound nodes when solving SCF is close to 750 000, while the number of nodes processed by our Benders implementations varies between 212 (SUBn) and 6043 (SUBf). Due to the huge number of branch-and-bound nodes, in 21 out of 64 cases SCF terminates **abnormally due to** memory **overload** [1]. R2.m20 R1.11

The rightmost column in Table 3 shows the average gaps reported by Salman et al. [38] obtained by solving SORb2 approach. The average gaps obtained by Raghavan and Stanojević [34] were always worse than those obtained in [38], therefore we report only on the latter ones. **In [38], the authors set the**

[1] **According to Cplex documentation, increasing the available memory (by tuning the corresponding parameters) slows down the performance of the algorithm. In those cases where "out of memory" happens, Cplex generates hundreds of thousands of branch-and-bound nodes to solve the SCF model. Hence, tuning the corresponding parameters will only replace the "out of memory" by the "time limit" message.**

time-limit to 5400 seconds and used Cplex 9.1 with default settings. We used the time-limit of 1000 seconds and Cplex 11.2 with Cplex cuts and presolver turned off. According to the performance evaluation tests provided in [39], our computer is approximately 1.2 times faster. Comparing the values in the column SUBf and the last column in Table 3, we may conclude that in most cases our approach outperforms the approach of Salman et al. [38]. R1.12

Table 4 reports on the correlations between the average time needed to solve the subproblem, the number of branch-and-bound nodes and the tightness of the bounds at the root node of the branch-and-bound tree. The average values over all 64 Salman instances for the following parameters are provided: Time_0 and Gap_0 denote the running time and the gap at the root node of the branch-and-bound tree, respectively; Benders_0 denotes the number of Benders cuts separated at the root node; $\text{Time}_0/\text{Benders}_0$ provides the ratio between the total time spent and the number of Benders cuts. The values $\text{Gap}_{\text{total}}$ and $\text{Time}_{\text{total}}/\text{Benders}_{\text{total}}$ are the corresponding values provided for the total running time $\text{Time}_{\text{total}}$ of 1000 seconds. The last row shows how many branch-and-bound nodes have been processed within the time limit. For the results after 1000 seconds, the two best performing approaches are shown in bold face.

The normalized Benders subproblems have a complicated flow structure with two kinds of capacity constraints. Therefore, the problem of solving a normalized subproblem by closing the unbounded cone with an additional constraint may become a difficult task. Row $\text{Time}_0/\text{Benders}_0$ of Table 4 provides an estimate of an average time (in seconds) needed to solve each Benders subproblem. The fastest subproblems are SUBf and PSUBn (followed by the separation of extreme rays with the SUB approach). Correspondingly, these two variants are first to be finished at the root node of the branch-and-bound tree. Therefore, they are also separating the most Benders cuts and traversing the most nodes of the branch-and-bound tree. However, the SUBf bounds obtained at the root node are tighter than the corresponding bounds of the PSUBn model, which makes the SUBf approach the winner, when solving this data set.

This study shows that:

- Using rounded Benders cuts derived from the DMCF formulation **outperforms** the compact SCF model. R2.m1
- Two important aspects decide on the quality of our Benders approach: a) the running time needed to solve the Benders subproblem, and b) the quality of the derived Benders cuts. The model that succeeds to balance the trade-off between these two aspects is the most desirable one.

5.4 Solving Real-World Instances

We now show the comparison of results obtained for the set of real-world instances derived from Bregenz, a city in Austria.

Problem								Gap at the Root Node Cplex presolver off				Gap after 1000s Cplex cuts and presolver off								Gap
	s	d	$\|V\|$	$\|E\|$	$\|D\|$	$\|N\|$	UB	DMCF	MCF	SCF	SUBc	SCF	SUB	SUBc	SUBn	SUBf	PSUBc	PSUBn	PSUBf	[Salman et al.]
e20	c	l	18.2	37.8	8.6	12.6	111.9	5.5	11.2	32.3	**5.4**	0.3	**0.0**	**0.0**	0.4	**0.0**	**0.0**	**0.0**	**0.0**	0.0
e20	r	l	17.4	35.4	10.0	13.0	143.3	**6.6**	10.9	33.2	6.8	4.7	1.6	2.0	2.5	**1.2**	1.7	1.5	1.5	1.9
e20	c	h	18.2	38.0	8.6	27.6	194.2	**6.0**	9.8	22.1	**6.0**	1.8	2.4	2.6	4.3 [1]	1.3	2.0	2.0	**1.2**	0.7
e20	r	h	17.6	36.2	9.2	23.2	239.8	**5.8**	9.4	19.7	**5.8**	2.8	1.6	1.7	3.8 [2]	**0.4**	0.9	1.1	0.9	1.0
e30	c	l	26.6	54.4	14.6	24.2	323.5	**6.6**	10.8	23.9	7.1 [1]	9.2 (2)	4.0	4.3 [1]	6.1 [2]	**2.7**	3.5	3.2	3.4	7.1
e30	r	l	26.8	55.2	13.8	22.2	290.1	**5.9**	9.1	22.9	**5.9**	7.8	4.0	4.4	6.0 [2]	**2.8**	4.0	3.3	3.5	7.6
e30	c	h	26.2	53.6	14.2	47.4	529.1	**5.1**	8.8	15.2	**5.1** [1]	6.2	4.4	4.4 [1]	7.5 [5]	**3.0**	3.7	3.5	3.5	6.2
e30	r	h	26.6	54.8	12.2	33.4	469.0	**4.9**	6.9	13.7	**4.9**	5.5 (1)	3.7	4.1	5.5 [3]	**2.8**	3.6	3.1	3.3	4.5
e40	c	l	36.6	75.2	19.0	27.4	409.1	**6.8**	11.0	28.1	7.1 [2]	15.5 (5)	6.5	6.5 [2]	8.6 [1]	**4.5**	6.4 [2]	5.2	5.6	14.7
e40	r	l	35.6	74.6	19.4	31.4	572.4	**5.8**	9.0	22.2	6.1 [2]	12.4 (5)	5.2	5.7 [2]	6.8 [2]	**3.9**	5.2	4.4	5.0	10.4
e40	c	h	35.8	73.8	19.0	49.4	674.8	6.1 [2]	8.2	16.0	**5.8** [4]	9.5 (3)	5.1	5.8 [4]	6.6 [2]	**3.7**	5.3 [1]	4.2	4.7 [1]	7.9
e40	r	h	33.8	71.8	16.8	46.2	745.6	**5.1** [2]	7.2	14.1	5.2 [4]	8.0 (3)	4.8 [1]	5.0 [4]	6.3 [5]	**3.4**	4.5	4.0	4.4	6.3
oct			16	20	14	39	2432.3	**6.1**	8.8	17.5	6.3	3.8	2.8	4.0	5.2	1.4	3.0	**1.2**	2.5	0.0
ring			26	54	17	47	1391.3	**7.2**	11.5	25.6	7.3	11.1	5.6	5.9	7.9	**3.8**	5.4	4.4	4.6	6.5
usa			26	39	16	44	2233.2	**7.4**	10.4	19.5	**7.4**	8.4 (1)	6.2	6.2	8.2 [1]	**4.6**	5.6	5.1	5.4	4.8
arpa			16	21	12	35	2571.4	**5.0**	9.4	15.0	5.1	1.5	1.5	2.2	4.5	**0.5**	1.7	1.7	1.3	0.0

Table 3: Results for Salman's instances. The upper part of the table shows average values over 5 instances in each class `e(n)(s)(d)`. The lower part shows results for the four instances from Gavish and Altinkemer [23]. We report the gaps to the best known upper bound obtained at the root node and after the time limit. The numbers in squared brackets denote in how many (out of 5) cases the LP solution at the root node was not found within the time limit. For the SCF model the numbers in round brackets show how many experiments terminated because Cplex ran out of memory.

Average	DMCF	MCF	SCF	SUB	SUBc	SUBn	SUBf	PSUBc	PSUBn	PSUBf
Time_0	503.2	3.2	0.1	68.3	422.5	603.5	22.0	247.0	35.3	170.0
Benders_0	-	-	-	40.9	38.3	67.8	57.2	36.0	115.5	52.9
$\text{Time}_0/\text{Benders}_0$	-	-	-	1.7	11.0	8.9	0.4	6.9	0.3	3.2
Gap_0	5.9	9.1	19.6	7.5	6.0	6.9	6.4	6.1	6.9	6.2
$\text{Benders}_{\text{total}}$	-	-	-	740.8	164.7	159.2	**1281.5**	303.4	**1498.6**	556.0
$\text{Time}_{\text{total}}/\text{Benders}_{\text{total}}$	-	-	-	1.3	6.1	6.3	**0.8**	3.3	**0.7**	1.8
$\text{Gap}_{\text{total}}$	4.5	4.5	8.0	3.6	3.9	5.4	**2.5**	3.5	**3.0**	3.1
Nodes	1031	100264	687068	2152	492	212	6043	1326	3655	2501

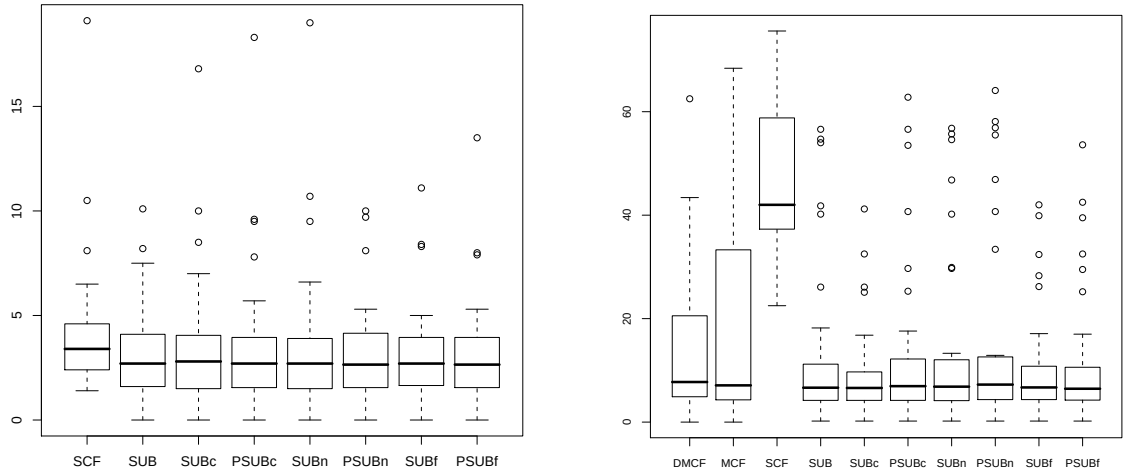
Table 4: Average values over all 64 Salman's instances.

LP relaxations: We first compare the gap of LP relaxations for three compact models, SCF, MCF and DMCF, whose values are given in Table 5. We turned Cplex cuts and the presolver off, to be able to compare the gaps achieved with the proposed LP models. For all 32 instances, the LP relaxation of the SCF model was solved within 1 or 2 seconds, but the average (median) gap over all Bregenz instances is 46.6% (42.0%). As expected, lower bounds obtained by solving the MCF model are significantly better: 19.1% (7.1%), but the LP relaxations of only 13 out of 32 instances **were** solved to optimality in less than 1000 seconds (within 383.8 seconds, on average). Finally, the average (median) gap obtained by solving the DMCF model is 13.9% (7.8%), but only in 3 out of 32 cases the LP **relaxations were** solved to optimality within the given time limit (in 55 seconds, on average). This also explains why some of the presented gaps of the MCF model are better than the corresponding DMCF ones (LP relaxations are solved by dual simplex method). R2.m21 R2.m22

According to these experiments we concluded that the only compact model that can be directly solved without a row and/or column generation technique is the SCF model. In order to use the strength of the DMCF model, we apply a row-generation technique to it. A column-generation technique for a similar problem has been presented in [20, 21]. R1.9 R2.m32

Solving MIPs: For the SCF model and for the seven branch-and-cut variants described above, we ran the code for 1000 seconds, with default Cplex settings **and the primal heuristic described in Section 4.3.** Only when solving the SUB model, the Cplex presolver needs to be turned off. Since the separation of Benders cuts may become a time-consuming task for instances of that size, we separate them only at the root node of the branch-and-bound tree. R1.5

Box-plots in Figure 3 provide an overview of the obtained gaps at the root node of the branch-and-bound tree. We observe that the huge gaps of the SCF model (46.6% average and 42.0% median value) can be reduced down to an average (median) value of 4.2% (3.4%), by turning on Cplex cuts and the presolver. By applying Benders cuts, all proposed methods return better results: the average gap varies between 3.1% (PSUBf) and 3.6% (PSUBc), and the corresponding median values vary between 2.6%



(a) With Cplex default settings.

(b) By turning off Cplex cuts and the presolver.

Figure 3: Box-plots over 32 `Bregenz`-instances: the gaps (in %) of lower bounds at the root node of the branch-and-bound tree.

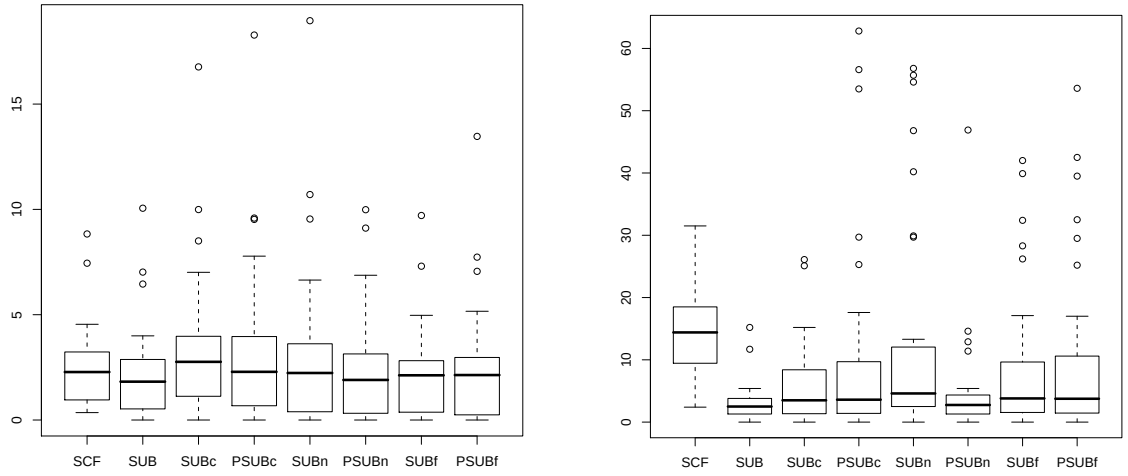
(PSUBf) and 2.8% (SUBc).

Figure 4 shows the gaps after the time limit was reached. Looking at the overall gaps after the given time limit, we observe that it is difficult to point out the differences between particular normalization approaches when default Cplex settings are used (see Figure 4(a)). Therefore, we cannot say that there is a clear winner among different Benders approaches. Although the Benders cuts obtained by solving the SUBc model are among the tightest ones (see, e.g., Figure 3), the separation was not finished at the root node of the branch-and-bound tree in 24 out of 32 cases. Figure 5 illustrates a typical situation in which SUBc gets stuck in the separation phase, while the SUB approach, for example, can draw an advantage out of branching.

Looking at Figures 3(a) and 4(a) one could get the impression that the simple SCF model (solved with the default Cplex settings) is competitive with the much more complex Benders approach. However, Table 5 presents detailed results obtained with default Cplex settings and shows that this is a wrong impression. Indeed, even on the set of real-world instances, the Benders decomposition approach is able to outperform the SCF model. For R2.m1 **8 out of 32 instances, the branch-and-cut approach (PSUBf) is able to find the optimal solution within the given time limit, while the SCF model did not solve a single instance to optimality. Furthermore, for 18 out of the remaining 24 instances, better gaps were produced using the branch-and-cut approach rather than solving the SCF model.** R1.10

	Problem						best	best	Gap at the root			Gap after 1000s							
									Cplex presolver off			with Cplex default settings							
Instance	$\|V\|$	$\|E\|$	$\|D\|$	$\|\mathcal{N}_{\min}\|$	$\|\mathcal{N}_{\text{avg}}\|$	$\|\mathcal{N}_{\max}\|$	**LB**	**UB**	DMCF	MCF	SCF	SCF	SUB	SUBc	SUBn	SUBf	PSUBc	PSUBn	PSUBf
29_A_H	322	488	28	2	2.0	3	**110998.28**	**110998.28**	3.8	3.9	58.5	1.1	**0.0**	1.1	0.3	0.2	1.2	**0.0**	**0.0**
29_B_H	322	488	28	2	2.0	4	227009.39	228146.84	30.6	68.4	71.1	**0.5**	0.6	7.0	6.6	2.1	7.8	0.7	2.1
29_C_H	322	488	28	3	3.0	7	55166.67	56779.45	24.6	10.7	25.7	3.3	3.1	3.3	2.9	2.9	4.0	**2.8**	2.9
29_D_H	322	488	28	4	4.1	10	50447.35	52546.87	21.7	11.6	33.1	4.5	**4.0**	5.5	5.0	4.1	5.0	4.6	4.3
36_A_H	325	490	33	2	2.0	4	**174491.64**	**174491.64**	3.5	3.7	42.0	1.0	0.5	0.4	0.4	0.2	0.5	0.2	**0.0**
36_B_H	325	490	33	2	2.0	4	807815.62	819877.83	12.8	37.1	37.4	**1.5**	2.1	4.5	5.8	2.8	5.7	1.9	2.1
36_C_H	325	490	33	3	3.0	7	192314.14	193821.25	14.5	56.1	60.8	**0.8**	1.0	1.8	1.8	**0.8**	2.4	0.9	**0.8**
36_D_H	325	490	33	4	4.1	11	82432.96	84676.01	16.0	10.6	26.0	2.8	2.6	3.2	3.0	2.7	**2.6**	3.1	2.7
45_A_H	333	498	41	2	2.0	4	**206863.16**	**206863.16**	3.1	2.7	43.7	1.1	0.2	1.0	**0.0**	**0.0**	0.1	**0.0**	**0.0**
45_B_H	333	498	41	2	2.0	4	851564.22	915891.12	33.6	40.0	39.9	7.4	**7.0**	10.0	10.7	7.3	9.6	9.1	7.7
45_C_H	333	498	41	3	3.0	7	224778.79	230380.49	22.3	58.9	59.9	**2.6**	6.5	8.5	9.5	5.0	9.5	6.9	7.1
45_D_H	333	498	41	4	4.1	11	100670.85	104114.25	19.4	19.6	27.3	3.3	3.3	3.7	3.4	**3.0**	3.7	3.5	3.6
67_A_H	351	516	61	2	2.0	4	236879.59	238483.81	6.4	5.9	37.2	1.2	0.8	1.8	0.7	0.8	1.2	**0.7**	**0.7**
67_B_H	351	516	61	2	2.0	4	1677023.89	1839517.90	43.4	67.7	64.8	**8.8**	10.1	16.8	19.0	9.7	18.3	10.0	13.5
67_C_H	351	516	61	3	3.0	7	611825.02	625948.67	14.2	46.8	42.0	**2.4**	3.7	5.0	5.1	3.8	5.1	4.9	5.2
67_D_H	351	516	61	4	4.1	11	138159.10	141382.31	26.4	29.5	22.5	**2.3**	2.6	3.4	3.1	2.4	3.4	3.2	3.1
29_A_L	322	488	28	1	1.0	1	**101951.52**	**101951.52**	**0.0**	**0.0**	75.6	0.4	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
29_B_L	322	488	28	2	2.0	4	38318.98	38472.82	7.7	7.4	55.7	1.6	0.7	1.2	0.7	2.5	0.8	**0.4**	0.5
29_C_L	322	488	28	3	3.0	4	34927.38	35758.16	5.1	4.6	59.1	3.2	2.3	3.3	2.3	**2.2**	2.4	2.3	2.3
29_D_L	322	488	28	3	3.0	4	34734.91	35533.78	4.7	4.0	60.5	3.8	**2.2**	3.1	2.4	2.4	2.3	2.3	2.3
36_A_L	325	490	33	2	2.0	2	**163386.07**	**163386.07**	0.6	0.6	53.4	0.4	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
36_B_L	325	490	33	2	2.0	4	**59201.52**	**59201.52**	6.4	6.0	36.0	0.7	**0.0**	1.5	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
36_C_L	325	490	33	3	3.0	5	57067.61	57797.48	7.2	6.3	39.7	2.3	1.6	2.2	1.2	1.4	**1.1**	1.3	1.5
36_D_L	325	490	33	4	4.0	5	54881.13	56167.45	5.8	4.6	41.9	3.1	2.3	2.7	2.3	**2.1**	2.2	2.4	2.6
45_A_L	333	498	41	2	2.0	2	**193052.96**	**193052.96**	0.1	0.1	56.7	0.6	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
45_B_L	333	498	41	2	2.0	4	70211.46	70546.13	7.8	6.8	37.7	1.3	0.7	0.6	**0.4**	0.5	0.6	0.5	0.7
45_C_L	333	498	41	3	3.0	6	67156.12	68224.05	7.1	5.7	40.7	2.3	1.6	2.5	**1.5**	1.9	1.9	1.9	1.7
45_D_L	333	498	41	4	4.0	6	64901.08	66471.38	6.4	4.8	43.8	3.3	2.4	2.8	**2.3**	**2.3**	**2.3**	2.5	2.6
67_A_L	351	516	61	2	2.0	3	**218267.74**	**218267.74**	1.1	1.1	54.6	0.6	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
67_B_L	351	516	61	2	2.0	4	200000.95	201352.36	62.5	61.1	67.9	0.9	1.0	2.8	**0.7**	0.9	2.0	**0.7**	1.5
67_C_L	351	516	61	3	3.0	7	80926.47	82804.18	12.2	12.0	35.5	2.5	2.3	2.8	2.2	**2.1**	2.7	2.3	2.5
67_D_L	351	516	61	4	4.1	8	79031.51	81906.07	12.8	13.7	41.0	4.0	3.5	4.2	3.8	**3.4**	3.9	4.2	3.8

Table 5: Lower bounds obtained by solving LP relaxations of three compact approaches (with Cplex presolver turned off) are compared. The overall gaps (obtained with default Cplex settings, after 1000 seconds) of the compact model SCF and seven branch-and-cut variants are provided.



(a) With Cplex default settings.

(b) By turning off Cplex cuts and the presolver.

Figure 4: Box-plots over 32 `Bregenz`-instances: the overall gaps (in %) obtained after 1000 seconds.

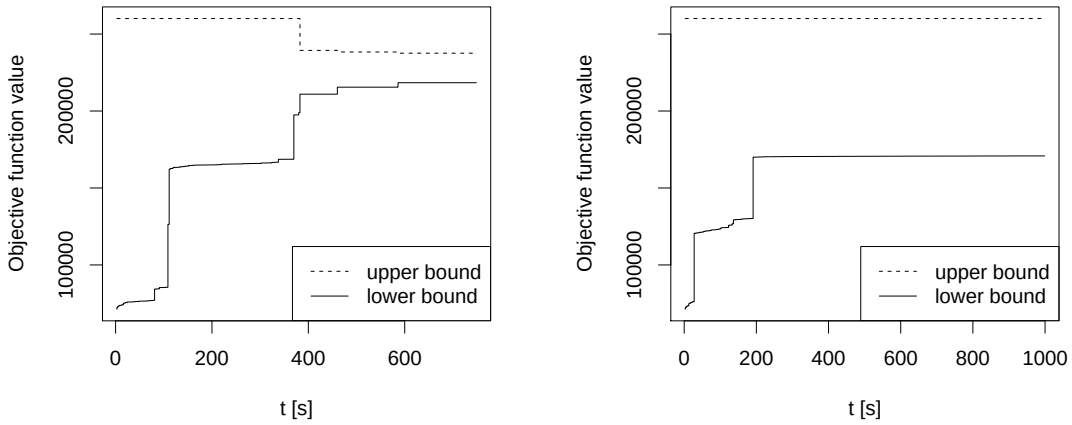
5.5 Testing Potential Enhancements

We report on negative results when trying to enhance the Benders decomposition with disjoint Benders cuts and by using Magnanti-Wong (MW) approach. In particular, solving the LP subproblem related to disjoint cuts is a time-consuming task which takes longer than resolving the primal problem. Since the only purpose of disjoint cuts is to speed up the separation without resolving the master problem, we did not obtain better results by using them.

As already observed above, if Cplex general purpose cuts are turned on, it is difficult to point out the differences between different variants of Benders separation models. Therefore, to test the effects of applying the MW approach, we turned the Cplex cuts off. The MW approach generates most improving cuts when applied to the SUBn approach, for which we report the gaps obtained within the time limit of 1000 seconds (see Figure 6). We observe that the MW approach slows down the performance: the overall number of included Benders cuts is reduced while there is no significant improvement in the quality of lower bounds obtained per iteration.

6 Conclusions

We have presented a new disaggregated flow formulation DMCF **which is a byproduct of the model introduced in [16].** It induces tighter gaps than the MCF model which is typically used for network loading problems. Using Benders decomposition, we solve 8 of our 32 new single-source instances to optimality within a reasonable time limit. For 18 out of the remaining 24 instances, we report better gaps than the best performing compact formulation. R2.1



(a) (SUB)

(b) (SUBc)

Figure 5: Lower bound growth vs. time (CPU seconds) with models SUB and SUBc for instance `Bregenz29_B_H`. (a) The first huge increase of lower bound is due to two subsequently found Benders cuts, the second increase is due to branching. (b) The separation at the root node is not finished when solving SUBc.

Comparing normalization strategies for the Benders decomposition, we see that depending on the structure of the inputs, different normalizations are preferable. However, in contrast to a common belief, the separation of extreme rays, which is also called the *textbook implementation*, provides relatively good results across all instances.

There are several arguments explaining this observation:

1. We solve the problem starting from a compact formulation (the SCF model) and we use Benders cuts only in order to improve the quality of lower bounds, i.e., they are *not necessary* for SSNLP to have a complete MIP formulation. This is a first difference with respect to known approaches for solving the multiple-source multiple-sink network loading, where Benders inequalities are separated in a similar way.

2. In opposite to our objective function, many related problems consider settings with flow-dependent objective values. In such cases, Benders cuts are used to separate both, feasibility and optimality cuts. The quality of optimality cuts is essential for such problems and therefore enhancing approaches (like those given in e.g. [18, 31, 32, 35]) play a crucial role to make Benders decomposition work.

3. We confirm the claim of Magnanti and Wong [31], that the crucial role in the generation of efficient Benders separation approaches is played by the size (see our Lemma 2.4) of the convex hull of the relaxed master problem. We show that the textbook implementation of Benders separation

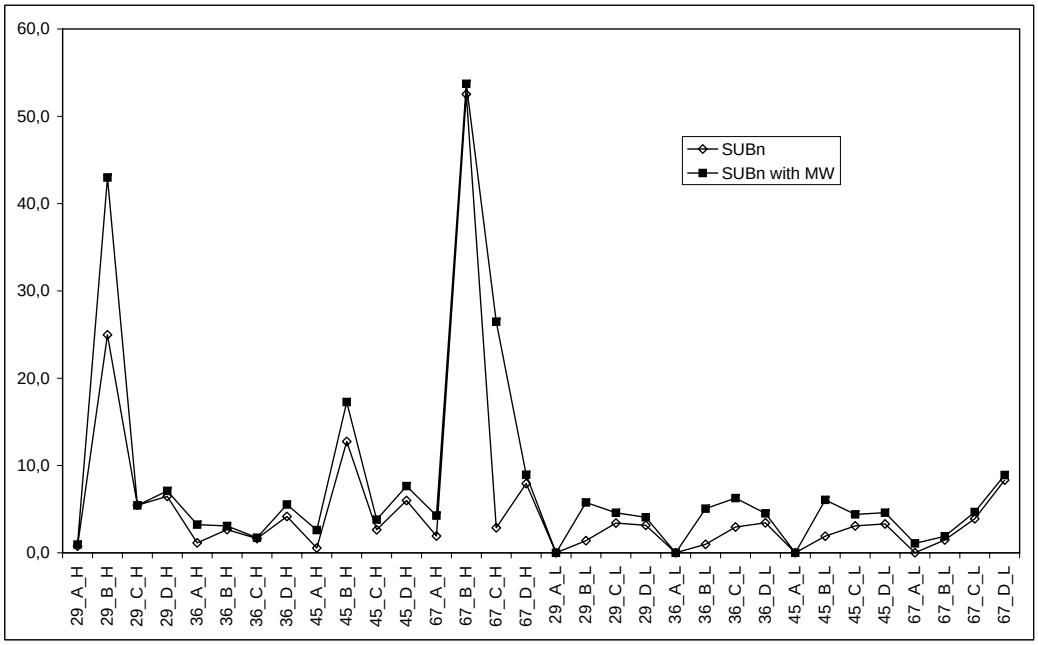


Figure 6: Comparing the gaps obtained within the time limit of 1000 seconds (Cplex cuts turned off): SUBn and SUBn extended by Magnanti-Wong cuts.

is not the worst possible choice, if a "good" LP-model is used to generate the corresponding cuts. Typically there is a trade-off problem in Benders decomposition approaches between the strength of the subproblem and the running time needed to solve it. To overcome this problem, the separation of extreme rays turns out to be a good compromise: an extreme ray is usually found much faster than an optimal extreme point of a bounded subproblem.

The algorithmic framework has been developed to solve large single-source instances arising in the design of telecommunication networks. Some ingredients in our approach exploit the single-source assumption. For example, degree-balance constraints or primal heuristic guarantee that the final solution is a directed acyclic graph. Also, based on the single-source assumption, Benders cuts are being added to strengthen the LP-relaxation, but they are not necessary for the feasibility of a solution of the SCF master model. However, the presented approaches for separating Benders cuts derived from the disaggregated formulation could also be applied to more general multi-commodity versions of the capacitated network design problem. R1.7

Acknowledgments

The first author is supported by the Hertha-Firnberg Fellowship of the Austrian Science Fund (FWF). The second author is supported by the Austrian Research Promotion Agency, FFG, within Bridge program (812973). The third author thanks the support of "Ministerio de Ciencia e Innovación" through the research project MTM2009-14039-C06-01.

References

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows*. Prentice Hall, 1993.

[2] Algorithmic Solutions Software Gmbh. LEDA Library for Efficient Data types and Algorithms. `http://www.algorithmic-solutions.com`; visited on May 1st 2010.

[3] P. Avella, S. Mattia, and A. Sassano. Metric inequalities and the network loading problem. *Discrete Optimization*, 4(1):103–114, 2007.

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

[5] J. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4:238–252, 1962.

[6] D. Berger, B. Gendron, J.-Y. Potvin, S. Raghavan, and P. Soriano. Tabu search for a network loading problem with multiple facilities. *Journal of Heuristics*, 6(2):253–267, 2000.

[7] D. Bienstock, S. Chopra, O. Günlük, and C.-Y. Tsai. Mininum cost capacity installation for multicommodity flows. *Mathematical Programming*, 81(2):177–199, 1998.

[8] D. Bienstock and O. Raskina. Asymptotic analysis of the flow deviation method for the maximum concurrent flow problem. *Mathematical Programming*, 91(3):479–492, 2002.

[9] B. V. Cherkassky and A. V. Goldberg. On implementing push-relabel method for the maximum flow problem. *Algorithmica*, 19:390–410, 1997.

[10] S. Chopra, I. Gilboa, and S. T. Sastry. Source sink flows with capacity installation in batches. *Discrete Applied Mathematics*, 85(3):165–192, 1998.

[11] S. Chopra and M. R. Rao. The Steiner tree problem I: Formulations, compositions and extension of facets. *Mathematical Programming*, 64(1–3):209–229, 1994.

[12] M. Chouman, T. G. Crainic, and B. Gendron. A cutting plane algorithm for multicommodity capacitated fixed-charge network design. Technical Report 2009-20, CIRRELT, 2009.

[13] A. M. Costa. A survey on Benders decomposition applied to fixed-charge network design problems. *Comput. Oper. Res.*, 32(6):1429–1450, 2005.

[14] A. M. Costa, J.-F. Cordeau, and B. Gendron. Benders, metric and cutset inequalities for multicommodity capacitated network design. *Computational Optimization and Applications*, 42:371–392, 2009.

[15] K. L. Croxton, B. Gendron, and T. L. Magnanti. A comparison of mixed-integer programming models for non-convex piecewise linear cost minimization problems. *Management Science*, 49:1268–1273, 2003.

[16] K. L. Croxton, B. Gendron, and T. L. Magnanti. Variable disaggregation in network flow problems with piecewise linear costs. *Operations Research*, 55(1):146–157, 2007.

[17] G. Dahl and M. Stoer. A cutting plane algorithm for multicommodity survivable network design problems. *INFORMS Journal on Computing*, 10(1):1–11, 1998.

[18] M. Fischetti, D. Salvagnin, and A. Zanette. A note on the selection of Benders' cuts. *Mathematical Programming*, 2010. To appear.

[19] B. Fortz and M. Poss. An improved Benders decomposition applied to a multi-layer network design problem. Technical report, GOM, Université Libre de Bruxelles, Belgium, 2008.

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

[21] A. Frangioni and B. Gendron. A Stabilized Structured Dantzig-Wolfe Decomposition Method. Technical Report 2010–02, CIRRELT, 2010.

[22] V. Gabrel, A. Knippel, and M. Minoux. Exact solution of multicommodity network optimization problems with general step cost functions. *Operations Research Letters*, 25(1):15–23, August 1999.

[23] B. Gavish and K. Altinkemer. Backbone network design tools with economic tradeoffs. *ORSA Journal on Computing*, 2(3):236–252, 1990.

[24] O. Günlük. A branch-and-cut algorithm for capacitated network design problems. *Mathematical Programming*, 86(1):17–39, 1999.

[25] F. Grandoni and G.F. Italiano. Improved approximation for single-sink buy-at-bulk. In *In International Symposium on Algorithms and Computation (ISAAC)*, pages 111–120, 2006.

[26] IBM ILOG. CPLEX. `http://www.ilog.com/products/cplex/`; visited on May 1st 2010.

[27] A. B. Keha, I. R. de Farias Jr., and G. L. Nemhauser. Models for representing piecewise linear cost functions. *Operations Research Letters*, 32(1):44–48, 2004.

[28] I. Ljubić, R. Weiskircher, U. Pferschy, G. Klau, P. Mutzel, and M. Fischetti. An algorithmic framework for the exact solution of the prize-collecting Steiner tree problem. *Mathematical Programming*, 105(427–449), 2006.

[29] T. L. Magnanti, P. Mirchandani, and R. Vachani. Modeling and solving the two-facility capacitated network loading problem. *Operations Research*, 43(1):142–157, 1995.

[30] T. L. Magnanti, P. Mireault, and R.T. Wong. Tailoring Benders decomposition for uncapacitated network design. *Mathematical Programming Study*, 18(1):112–154, 1984.

[31] T. L. Magnanti and R. T. Wong. Accelerating Benders decomposition: Algorithmic enhancement and model selection criteria. *Operations Research*, 29(3):464–484, 1981.

[32] T. L. Magnanti and R. T. Wong. Network design and transportation planning: Models and algorithms. *Transportation Science*, 18(1):1–55, 1984.

[33] N. Papadakos. Practical enhancements to the Magnanti-Wong method. *Operations Research Letters*, 36(4):444 – 449, 2008.

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

[35] W. Rei, J.-F. Cordeau, M. Gendreau, and P. Soriano. Accelerating Benders decomposition by local branching. *INFORMS Journal on Computing*, 21(2):333–345, 2009.

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

[37] F. S. Salman, J. Cheriyan, R. Ravi, and S. Subramanian. Approximating the single-sink link-installation problem in network design. *SIAM Journal on Optimization*, 11(3):595–610, 2000.

[38] F. S. Salman, R. Ravi, and J. N. Hooker. Solving the capacitated local access network design problem. *INFORMS Journal on Computing*, 20(2):243–254, 2008.

[39] SPEC. Standard performance evaluation corporation. `http://www.spec.org/`; visited on May 1st 2010.

[40] S. P. M. van Hoesel, A. M. C. A. Koster, R. L. M. J. van de Leensel, and M. W. P. Savelsbergh. Bidirected and unidirected capacity installation in telecommunication networks. *Discrete Applied Mathematics*, 133(1-3):103–121, 2003.