Solving Minimum-Cost Shared Arborescence Problems

Eduardo Álvarez-Miranda[a], Ivana Ljubić[b], Martin Luipersbeck[c], Markus Sinnl[c]

[a] *Department of Industrial Engineering, University of Talca, Curicó, Chile*
[b] *ESSEC Business School of Paris, Cergy-Pontoise, France*
[c] *Department of Statistics and Operations Research, Faculty of Business, Economics and Statistics, University of Vienna, Austria*

Abstract

In this work the minimum-cost shared network problem (MCSN) is introduced, where the objective is to find a minimum-cost subgraph, which is shared among multiple entities such that each entity is able to fulfil its own set of topological constraints. The topological constraints may induce structures like Steiner trees, minimum spanning trees, shortest paths, etc. The cost function to be minimized is a combination of the costs for the shared network and the costs incurred by each entity. The minimum cost shared Steiner arborescence problem (SStA) is a special case of the MCSN, in which the underlying structures take the form of Steiner trees. The SStA has been used in the literature to establish shared functional modules in protein interaction networks. A cut formulation for the SStA and Benders decomposition thereof are proposed in this article and computationally evaluated and compared with a previously proposed flow-based formulation. The effectiveness of the algorithms is illustrated on two types of instances derived from protein-interaction networks (available from the previous literature) and from telecommunication access networks.

Keywords: Combinatorial Optimization, Benders Decomposition, Integer programming, Telecommunications, Bioinformatics

1. Introduction and Motivation

In a typical Network Design (ND), or Network Optimization, setting, the decision maker's goal is to define, under one or more *criteria of optimality*, a network infrastructure and a corresponding operating regime that meets a certain set of topological and/or operative *requirements*.

Such a setting appears in different application such as supply chain management, telecommunications or bioinformatics, among many others [see 28, for a recent reference

Email addresses: ealvarez@utalca.cl (Eduardo Álvarez-Miranda), ivana.ljubic@essec.edu (Ivana Ljubić), martin.luipersbeck@univie.ac.at (Martin Luipersbeck), markus.sinnl@univie.ac.at (Markus Sinnl)

on ND]. Prominent examples of ND problems are optimal path problems, optimal tree problems, routing problems, or fixed-charge multi-commodity flow problems, to mention only a few.

In most of the decision making contexts of ND, it is assumed that a single entity (a city council, a company, a biological process, etc.) "operates" on the designed network. Although this assumption is reasonable in most of the cases, there are some more complex situations in which there is a managing operator that designs a network that is afterwards *shared* by different entities. Intuitively, the network to be designed in such case should allow a feasible operation for each of the entities, while ensuring cost efficiency individually but also globally. Take for instance the competitive environment in the telecommunication industry, where municipalities build and operate the shared network infrastructure, which is then used by various retailer companies for providing their local services (such as telephone, digital TV or high-speed internet). In a more general context, an *open-access network* is a business model in which the physical network is owned and managed by a single entity (e.g., a municipality) which creates an open access to internet service providers (ISPs) for delivering their services through the network [4]. The network owner creates a long-term revenue, whereas local service providers are given the opportunity for short-term gains. Such a business model has been shown to be useful in rural areas, where for any single ISP, building the network infrastructure is infeasible from the investment perspective [18]. In Australia, for example, a government owned corporation, NBN Co. Limited is creating the so-called National Broadband Network for providing the high-speed internet for more than 90% of the population. In this article, we approach the open-access network design problem from the perspective of the network operator, and search for the optimal shared infrastructure that minimizes the network installation cost plus the sum of operation costs incurred by the retailers.

In another context, suppose that a particular biological process, say a type of cancer, wants to be understood from a gene interaction perspective. Nowadays, different experimental procedures and conditions might help in establishing correlations between a type of cancer and different sets of genes (typically know as *hits*, i.e., genes that are statistically meaningful for explaining this particular biological process). Common bioinformatic approaches either aim at analyzing *signaling pathways* of each set of hits individually [see, e.g., 12, 31, 3], or merging these hits into a larger set and then analyzing the resulting data [see, e.g., 32]. Nonetheless, and as already motivated by [26], a more accurate analysis should be focused on answering the following question: is there any compact (in terms of size) *shared* functional module on which each of these hit sets establishes signaling-pathways simultaneously? The functional module that answers this question can help in understanding much better the underlying biological process, since it allows to capture its dynamics.

Contribution and Paper Outline The contribution of this paper is twofold. First, a general framework for min-cost shared network optimization is proposed; as it is shown,

such framework allows to model problems stemming from different application contexts. Second, for the Steiner arborescence variant of the proposed framework, which suits in both Telecommunication and Bioinformatics settings, two exact algorithmic strategies are designed, implemented and computationally compared using real and synthetic datasets coming from these two application contexts. The designed algorithms correspond to (i) a branch-and-cut based on the separation of so-called connectivity inequalities, and (ii) an enhanced Benders decomposition based on a specially tailored decomposable formulation. Extensive computational experiments on real instances show that the proposed Benders decomposition is, overall, more effective than the other branch-and-cut approach; moreover, the proposed enhancements for the decomposition (cut pool management, stabilization, parallelization, etc.) prove to be effective in improving its performance. The proposed models and algorithms are of interest for researchers and practitioners in both of the aforementioned areas, since instances of realistic size can be tackled efficiently.

The paper is organized as follows. A formal definition of the minimum cost shared network problem (MCSN) along with a literature review on related problems is presented in Section 2. In Section 3 a directed-cut based formulation is presented, and the details of the Benders decomposition are provided. The two types of separated cuts (Benders optimality cuts and integer L-shaped cuts) are described and the corresponding separation procedures are outlined. Complementary, the stabilization techniques are presented as well in the same section. More details regarding the implementation of the proposed algorithms and specially tailored enhancements are given in Section 4. In Section 5 the description of the test instances is given. An extensive characterization and analysis of the computational experiments is presented in Section 6. Finally, concluding remarks and paths for future work are given in Section 7.

2. Problem Definition and Previous Work

2.1 The Minimum-Cost Shared Network Problem

Let $G = (V, A)$ be a network where V is the set of nodes and A is the set of arcs. On this network a set L of entities or users require to function according to different and independent topological and/or operative requirements, say $\mathcal{Y}^l(G)$ for each $l \in L$. Moreover, such operation shall be carried out on a shared network which should also meet particular requirements embodied by $\mathcal{X}(G)$.

Consider that $\mathbf{c} : A \to \mathbb{R}_{\geq 0}$ is the sharing cost function, such that c_{ij} is the cost of taking $(i, j) \in A$ as part of the shared network (e.g., the installation cost). Likewise, let $\mathbf{w}^l : A \to \mathbb{R}_{\geq 0}$ be the cost function for user $l \in L$, such that w^l_{ij} is the cost of taking $(i, j) \in A$ as part of network that will used by l (e.g., the operating cost for providing the local service).

Assume that $\mathbf{X}$ is a subset of G that represents the shared network. Likewise, consider that for each $l \in L$ a subset of G, say $\mathbf{Y}^l$, is chosen for the operation of l ($\mathbf{Y} =$

$\left\{\mathbf{Y}^1, \ldots, \mathbf{Y}^{|L|}\right\}$). A pair $(\mathbf{X}, \mathbf{Y})$ will be considered as feasible if the following conditions are verified

$$\mathbf{X} \in \mathcal{X}(G), \tag{1}$$

$$\mathbf{Y}^l \in \mathcal{Y}^l(G), \qquad \forall l \in L \tag{2}$$

$$\mathbf{Y}^l \subseteq \mathbf{X}. \qquad \forall l \in L. \tag{3}$$

Expression (1) imposes that the shared network $\mathbf{X}$ must fulfill the requirements embodied by $\mathcal{X}(G)$; for example it could be empty or impose $\mathbf{X}$ to be a connected subgraph. Expression (2) forces that, for every l in L, the corresponding network $\mathbf{Y}^l$ must meet the topological requirements expressed by $\mathcal{Y}^l(G)$; finally, (3) points out that the (local) networks of each retailer must be contained in the shared network.

The cost of constructing the shared network is given by $c(\mathbf{X}) = \sum_{(i,j)\in\mathbf{X}} c_{ij}$, while the cost for operating the individual subnetworks on $\mathbf{X}$ is given by $w(\mathbf{Y}) = \sum_{l\in L} \sum_{(i,j)\in\mathbf{Y}^l} w^l_{ij}$. Therefore, the goal is the find a feasible pair $(\mathbf{X}, \mathbf{Y})$ such that, according to some criterion, the overall cost of the selected networks is minimized.

Considering all this elements, one can define the minimum-cost shared network problem (MCSN) as

$$\text{(MCSN)} \qquad OPT = \min\left\{f\left(c(\mathbf{X}), w(\mathbf{Y})\right) \mid (\mathbf{X}, \mathbf{Y}) \text{ satisfies (1)-(3)}\right\}. \tag{4}$$

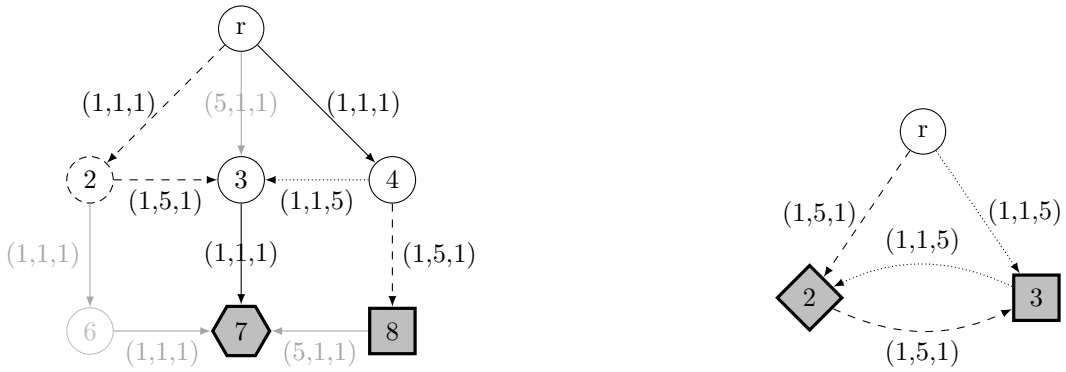
In this model $f\left(c(\mathbf{X}), w(\mathbf{Y})\right)$ is a positive function that depends on the particular application.

Note that this definition of the MCSN is quite generic. The conditions imposed by $\mathcal{X}(G)$ and $\mathcal{Y}^l(G)$ for the shared and the user networks, respectively, might constrain them to be spanning trees, Steiner trees or arborescences, Hamiltonian tours or similar (in their directed or undirected counterparts). Furthermore, function $f\left(c(\mathbf{X}), w(\mathbf{Y})\right)$ might define different types of balances between the cost of implementing the shared infrastructure and the cost of operating it.

The problem addressed in this paper is a special case of the MCSN and can be defined as follows. For each $l \in L$ there is a set $T_l \subset V$ of terminal nodes for l, so that a feasible network $\mathbf{Y}^l$ should be a Steiner arborescence spanning T_l and rooted at a dedicated root node $r \in V$ (common for all $l \in L$). The union of all these Steiner arborescences, i.e., $\mathbf{Y}$, must be embedded into the shared network $\mathbf{X}$, whose only requirement is to be connected. The cost for $(\mathbf{X}, \mathbf{Y})$ is calculated by the following function

$$f(\mathbf{X}, \mathbf{Y}) = (1-\alpha) \sum_{(i,j)\in\mathbf{X}} c_{ij} + \alpha \sum_{l\in L} \sum_{(i,j)\in\mathbf{Y}^l} w^l_{ij},$$

where $\alpha \in [0, 1]$ is a parameter (defined by the decision maker) that enables to establish the relative importance of one type of cost with respect to the other. So, if $\alpha = 0$, then the



(a) Example #1 (T_1={7, 8}, T_2={7})

(b) Example #2: (T_1={2}, T_2={3})

Figure 1: Examples for $|L|= 2$. Terminal nodes are drawn filled and such that their shapes mark their label-membership: diamond for T_1, square for T_2, hexagon for $T_1 \cap T_2$. For the given optimal solution, arcs and non-terminals are drawn based on the subnetwork they are part of: dotted for $\mathbf{Y}^1$, dashed for $\mathbf{Y}^2$, and continous for $\mathbf{Y}^1 \cap \mathbf{Y}^2$. Remaining arcs and nodes are drawn grayed out. For each arc (i, j) the triple $(c_{ij}, w^1_{ij}, w^2_{ij})$ denotes its costs.

cost of implementing the shared network is the only cost that must be considered; while if $\alpha = 1$, this means that the infrastructure is already available, and only the operating cost of individual users need to be considered. In this paper, this problem will be referred to as minimum cost shared Steiner arborescence problem (SStA).

Figure 1 shows two examples where $|L|= 2$. In Example #1, node 3 has two ingoing arcs but only one outgoing arc in $\mathbf{X}$ for the given optimal solution. In Example #2, the optimal shared network contains a 2-cycle. These examples illustrate that $\mathbf{X}$ is not necessarily an arborescence, and may even contain cycles. Hence, typical arborescence properties, namely, (i) in-degree of every node is at most one, and (ii) in-degree is less or equal than out-degree for every non-terminal node, do not hold. On the other hand, for a fixed label l the associated subgraph is an arborescence rooted at r with the corresponding terminals being the leaf nodes, and the properties (i) and (ii) being satisfied.

2.2 Previous Work and Related Problems

Results by Mazza et al. [26]. The SStA was proposed in [26], as a modelling tool for functional module reconstruction when different gene sets are available for a particular biological process. For this application, the elements in L was referred to as *labels*, $c_{ij} = w^l_{ij} = 1$ for all $(i, j) \in A$, $l \in L$, and $\alpha = 0.5$. Hence, the goal was to minimize, simultaneously, the cardinality of the shared network and the sum of the cardinalities of the networks associated to each label $l \in L$. The authors named the problem the Minimum $|L|$-labeling Problem. An Mixed Integer Programming (MIP) formulation was proposed,

and a heuristic preprocessing procedure was designed. Experiments were carried out on two datasets, one related to the response to influenza infection and the other to endoplasmic reticulum export regulation in humans. After heuristic preprocessing (which might have discarded the optimal solution), these datasets led to instances of 1598 proteins (nodes) and 8708 interactions (arcs), and 2 to 4 labels ($|L|$). Computational results using CPLEX showed that within two hours of computing time the proposed flow-based formulation (see below) was able to reach gaps no greater than 5%.

The following MIP formulation was used in [26] to solve the problem instances to optimality. We will refer to this formulation as a *single-commodity flow formulation* (SCF). For a given arbitrary subset of nodes $S \subseteq V$, let $\delta^-(S) = \{(i,j) \in A \mid i \in V \setminus S,\ j \in S\}$ and $\delta^+(S) = \{(i,j) \in A \mid i \in S,\ j \in V \setminus S\}$. Let $\mathbf{x} \in \{0,1\}^{|A|}$ be a set of binary variables such that $x_{ij} = 1$ iff $(i,j) \in A$ is selected as part of the shared network. Likewise, let $\mathbf{y} \in \{0,1\}^{|A|\times|L|}$ be a set of binary variables such that $y^l_{ij} = 1$ iff $(i,j) \in A$ is selected as part of the network associated with $l \in L$. Complementary, let $\mathbf{f} \in \mathbb{Z}^{|A|\times|L|}_{\geq 0}$ be a set of flow variables such that f^l_{ij} corresponds to the flow, from r, that passes through $(i,j) \in A$ and is associated with $l \in L$. Additionally, let $\mathbf{b} \in \{0,1\}^{|V|\times|L|}$ be an auxiliary coefficient such that $b^l_i = 1$ if $i \in T_l$ for $l \in L$, and $b^l_i = 0$ otherwise.

Using these elements, the following MIP formulation is valid for the SStA:

$$\text{(SCF)} \qquad \min\ (1-\alpha) \sum_{(i,j)\in A} c_{ij} x_{ij} + \alpha \sum_{l\in L} \sum_{(i,j)\in A} w^l_{ij} y^l_{ij} \tag{5}$$

$$\text{s.t.} \qquad y^l_{ij} \leq f^l_{ij} \leq |T_l|\ y^l_{ij} \qquad \forall (i,j) \in A,\ l \in L \tag{6}$$

$$\sum_{(j,i)\in\delta^-(i)} y^l_{ji} \leq 1 \qquad \forall i \in V,\ l \in L \tag{7}$$

$$\sum_{(r,i)\in\delta^+(r)} f^l_{ri} = |T_l| \qquad \forall l \in L \tag{8}$$

$$\sum_{(j,i)\in\delta^-(i)} f^l_{ji} = \sum_{(i,j)\in\delta^+(i)} f^l_{ij} + b^l_i \qquad \forall i \in V \setminus \{r\},\ l \in L \tag{9}$$

$$y^l_{ij} \leq x_{ij} \qquad \forall (i,j) \in A, l \in L \tag{10}$$

$$\mathbf{x} \in \{0,1\}^{|A|}\ ,\ \mathbf{y} \in \{0,1\}^{|A|\times|L|}\ ,\ \mathbf{f} \in \mathbb{Z}^{|A|\times|L|}_{\geq 0} \tag{11}$$

The objective function (5) accounts for both the shared network cost and the total cost incurred by the labels; the relative importance of each of these costs is measured by α (which has been defined before). Constraints (6)-(9) ensure that, for every $l \in L$, any feasible vector $\mathbf{y}^l$ is associated with a Steiner arborescence connecting terminals T_l. Finally, constraints (10) link the shared network, embodied by $\mathbf{x}$, with each of networks induced by $\mathbf{y}^l$, for all $l \in L$. Clearly, due to the common root node, connectivity of the shared network is ensured by these constraints. Finally, the nature of the variables is given by (11).

Formulation (SCF) allows to compute solutions for the SStA without the need of any

sophisticated implementation. However, this is possible only for instances of a limited size. The weak bounds provided by the Linear Programming (LP) relaxation of (6)-(9) render this formulation impractical already for medium size instances.

Connection to the Fixed Charge Network Design problem (FCND) The MCSN shares some similarities with the uncapacitated FCND [see, e.g., 25]. In the latter problem we are given an undirected graph and a set of node pairs, called commodities. The objective is to find an minimum-cost subgraph $\mathbf{X}$ that contains a path for each given commodity. The cost function is given as the sum of the cost for the shared network plus the costs of each path. The SStA becomes the FCND, if the set T_l is singleton for every $l \in L$. For the FCND, Benders decomposition has been suggested as a possible solution approach [see 9, 25]. In contrast to the FCND, the SStA (and the MCSN, in general) requires the subnetworks supported by $\mathbf{X}$ to be of a more complicated nature, which makes the application of Benders decomposition more involved, due to the non-convexity of the underlying Benders subproblems.

Finally, note that, due to the given applications, we focus on the case where both $\mathbf{X}$ and $\mathbf{Y}^l$ are directed, but the proposed methodology can easily be adapted to deal with undirected networks.

3. Benders Decomposition

In this section, a MIP formulation based on connectivity inequalities is first presented. Afterwards, a scheme for projecting out a large amount of decision variables in a Benders decomposition fashion is outlined. We start by presenting a standard way of imposing connectivity for each l using connectivity cuts. We then discuss how to project out variables representing arborescences associated to each l, and propose some stabilization techniques for the separation of the underlying Benders cuts.

Recall that $G = (V, A)$ is a graph, with a dedicated root node r. Associated with each element $l \in L$ (referred to as label as said before), there is a set $T_l \subset V$.

Directed-cut Formulation As mentioned above, formulation (SCF) becomes impractical when large instances need to be solved. A standard alternative way to deal with connectivity is to use the so-called connectivity cuts that, although exponential in number, enable an effective implementation within a branch-and-cut procedure.

Let $\mathbf{x}$ and $\mathbf{y}$ be defined as before. Let $\mathcal{W}_l$ be the set of all subsets of V inducing connectivity cuts on G with respect to T_l, i.e., $\mathcal{W}_l = \{W \subseteq V \mid W \cap T_l \neq \emptyset, r \in W \setminus V\}$.

The following formulation models the SStA in terms of connectivity cuts:

$$\text{(CUT)} \qquad \min\ (1-\alpha)\sum_{(i,j)\in A} c_{ij}x_{ij} + \alpha\sum_{l\in L}\sum_{(i,j)\in A} w_{ij}^l y_{ij}^l \tag{CUT.1}$$

$$\sum_{(i,j)\in\delta^-(W)} y_{ij}^l \geq 1 \qquad \forall\, W\in\mathcal{W}_l,\ l\in L \tag{CUT.2}$$

$$y_{ij}^l \leq x_{ij} \qquad \forall (i,j)\in A,\ l\in L \tag{CUT.3}$$

$$\mathbf{x}\in\{0,1\}^{|A|}\ ,\ \mathbf{y}\in\{0,1\}^{|A|\times|L|} \tag{CUT.4}$$

The connectivity cuts are given by constraints (CUT.2); intuitively speaking, they ensure that, for each $l\in L$, variables $\mathbf{y}^l$ induce a directed path from r to every element in T_l. The shared network is obtained by the coupling constraints (CUT.3).

3.1 Benders Reformulation

Although the cut-based formulation (CUT) presented above allows to devise more effective algorithmic strategies (c.f. §6), it might still fail to tackle instances that are either too large or whose characteristics, e.g., cardinality of L, might be burdensome. As an alternative, and due to the structure of the problem, a Benders-like decomposition seems a natural algorithmic alternative for the problem; variables $\mathbf{x}$ will belong to the (outer level) master problem, while the underlying variables $\mathbf{y}^l$, for all $l\in L$ will be projected out. For each fixed value of the master problem, a (possibly empty) set of violated inequalities is derived by finding the optimal solutions of the corresponding (inner level) subproblems.

Before presenting the *Benders reformulation* (B), based on the previously outlined idea, the following notation is needed. Let $\mathcal{W}=\bigcup_{l\in L}\mathcal{W}_l$ and let $\boldsymbol{\theta}\in\mathbb{R}^{|L|}_{\geq 0}$ be a vector of continuous variables such that θ_l models the operating cost associated to label l, $l\in L$. We reformulate the original problem as follows:

$$\text{(B)} \qquad \min \qquad (1-\alpha)\sum_{(i,j)\in A} c_{ij}x_{ij} + \alpha\cdot\sum_{l\in L}\theta_l \tag{B.1}$$

$$\text{s.t.} \qquad \sum_{(i,j)\in\delta^-(W)} x_{ij} \geq 1 \qquad \forall W\in\mathcal{W} \tag{B.2}$$

$$\theta_l \geq \Phi_l(\mathbf{x}) \qquad \forall l\in L \tag{B.3}$$

$$\mathbf{x}\in\{0,1\}^{|A|}\ ,\ \boldsymbol{\theta}\in\mathbb{R}^{|L|}_{\geq 0}, \tag{B.4}$$

where constraints (B.2) ensure the existence of a directed path between the root and any possible terminal $t\in\bigcup_{l\in L}T_l$ – thus the shared network is feasible, and we refer to (B.2) as *Benders feasibility cuts*. For each $l\in L$, the function $\Phi_l(\mathbf{x})$ is a real-valued function in

$\mathbf{x}$, given as

$$\text{(S)} \qquad \Phi_l(\mathbf{x}^*) = \min \sum_{(i,j)\in A} w_{ij}^l y_{ij}^l \tag{S.1}$$

$$\text{s.t.} \qquad y_{ij}^l \le x_{ij}^*, \qquad \forall (i,j) \in A \tag{S.2}$$

$$\sum_{(i,j)\in\delta^-(W)} y_{ij}^l \ge 1, \qquad \forall W \in \mathcal{W}_l \tag{S.3}$$

$$\mathbf{y}^l \in \{0,1\}^n \qquad . \tag{S.4}$$

Given the minimization nature of the problem, constraints (B.3) in fact guarantee $\theta_l = \Phi_l(\mathbf{x}^*)$ for the optimal value of $\mathbf{x}^*$, for each $l \in L$. For a given $\mathbf{x}$ (representing a feasible shared network due to (B.2)), function $\Phi_l(\mathbf{x})$ provides the optimal solution value for building subnetwork required by each label $l \in L$. Constraints (S.3) guarantee that each terminal $t \in T_l$ is connected to the root using the arcs from y^l, whereas the coupling constraints (S.2) guarantee that arcs can be shared among the users, only when they are installed. Finally, since the shared network $\mathbf{x}$ does not necessarily represent a tree, we have to explicitly impose integrality of $\mathbf{y}$ variables to ensure that each subgraph for the given label $l \in L$ is a tree.

The main difficulty in solving the Benders-like reformulation (B) lies in the description of the functions $\Phi_l(\mathbf{x})$ which are neither convex, nor continuous. As this is usually done in two-stage stochastic/robust programming with binary recourse variables [see, e.g., 2, 6, 24], constraints (B.3) will be linearized as follows. We will slighly abuse the notation, and refer to (B) as a Benders reformulation in which inequalities (B.3) are replaced by the following two types of linear inequalities:

1. *Benders optimality cuts*, which are underestimators of $\Phi_l(\mathbf{x})$, $l \in L$, obtained using subgradients of the convex functions

$$\underline{\Phi_l}(\mathbf{x}) = \min\{ \sum_{(i,j)\in A} w_{ij}^l y_{ij}^l \mid \mathbf{y}^l \text{ satisfies (S.2), (S.3)}, \mathbf{y}^l \in [0,1]^{|A|}\}. \tag{$\underline{\text{S}}$}$$

2. *Integer L-shaped cuts* [see 21], that will make sure that this underestimation is tight whenever we encounter a feasible solution $(\mathbf{x}^*, \boldsymbol{\theta}^*)$ of (B) with binary vector $\mathbf{x}^*$.

As both families of cuts may be exponential in size, they are typically embedded within a branch-and-cut framework. In the following, we provide further details on how to derive these two types of inequalities and how to exploit the solution properties specific to the shared subtrees.

Benders Optimality Cuts Each $\underline{\Phi_l}(\mathbf{x}^*)$ can be obtained from the dual subproblem (notice that the dual subproblem is well defined since we assume that the problem is

bounded, and since each vector $\mathbf{x}^*$ is assumed to satisfy all Benders feasibility cuts (B.2)). So we have

$$(D^l) \qquad \underline{\Phi}_l(\mathbf{x}^*) = \max \quad \sum_{W \in \mathcal{W}} \beta_W - \sum_{(i,j)\in A} x^*_{ij}\alpha_{ij} \tag{D.1}$$

$$\text{s.t.} \quad \sum_{\substack{W:\\ (i,j)\in\delta^-(W)}} \beta_W - \alpha_{ij} \leq w^l_{ij} \qquad \forall (i,j) \in A \tag{D.2}$$

$$\boldsymbol{\alpha} \geq \mathbf{0}\ ,\ \boldsymbol{\beta} \geq \mathbf{0}, \tag{D.3}$$

where $(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)$, so-called dual multipliers, are the vectors associated to constraints (S.2) and (S.3), respectively (superscript l is dropped, for the ease of notation). By convexity of $\underline{\Phi}_l(\mathbf{x})$, the following underestimation is valid:

$$\theta_l \geq \Phi_l(\mathbf{x}) \geq \underline{\Phi}_l(\mathbf{x}) \geq \underline{\Phi}_l(\mathbf{x}^*) + \boldsymbol{\sigma}^*(\mathbf{x} - \mathbf{x}^*), \tag{12}$$

where $\boldsymbol{\sigma}^* \in \partial\underline{\Phi}_l(\mathbf{x}^*)$ is a subgradient of $\underline{\Phi}_l$ at $\mathbf{x}^*$. An optimal solution $(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)$ to (D^l), corresponds to a subgradient of the function $\underline{\Phi}_l(\mathbf{x})$ at point $\mathbf{x}^*$ (more precisely, $\underline{\Phi}_l(\mathbf{x}^*) = \sum_{W\in\mathcal{W}_l} \beta^*_W - \sum_{(i,j)\in A} x^*_{ij}\alpha^*_{ij}$ and $-\boldsymbol{\alpha}^* \in \partial\underline{\Phi}_l(\mathbf{x}^*)$), so that the following *Benders optimality cut* is obtained:

$$\theta_l \geq \sum_{W\in\mathcal{W}_l} \beta^*_W - \sum_{(i,j)\in A} x_{ij}\alpha^*_{ij}. \tag{BC}$$

Notice that any dual multiplier $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ being a feasible solution to (D_l) defines a valid Benders cut (BC), and hence a valid lower bound on θ_l. Consequently, (D_l) does not necessarily have to be solved to optimality, as long as a violated inequality can be found (cf. §4).

Integer L-shaped Cuts In the case of $\mathbf{x}^*$ being integral, one first separates Benders optimality cuts (BC) as long as they are violated. When none of these cuts is violated, one requires additional cuts to close the duality gap between the LP-relaxation of the dual subproblem $\underline{\Phi}_l(\mathbf{x}^*)$ and its MIP optimal solution $\Phi_l(\mathbf{x}^*)$. Let $G_S = (V_S, A_S)$ be the support graph induced by $\mathbf{x}^*$, let $\phi^*_l := \Phi_l(\mathbf{x}^*)$ be the optimal cost for connecting r to terminals T_l on G_S, and let $\underline{\theta}_l$ be a globally valid lower bound on the cost for connecting r to terminals T_l on G. For each label $l \in L$ the integer L-shaped cut is given by

$$\theta_l \geq \phi^*_l \cdot (1 - IND) + \underline{\theta}_l \cdot IND \tag{iLS}$$

where IND is an integer indicator variable such that when it is equal to zero, this constraint is activated (i.e., we request $\theta_l \geq \phi^*_l$) and when it is ≥ 1, the constraint is not binding. Cut (iLS) corresponds to an integer L-shaped cut [see 21, for further details], in which one

chooses the Hamming distance to $\mathbf{x}^*$ to define the indicator variable, so we have:

$$IND = \sum_{(i,j) \notin A_S} x_{ij} + \sum_{(i,j) \in A_S} (1 - x_{ij}).$$

The global lower bound $\underline{\theta}_l$ for SStA is obtained by computing $\Phi_l(\mathbf{1})$ (i.e., by solving the Steiner arborescence problem with terminals T_l, assuming that all arcs from G are available) for each label. Alternatively, slightly weaker bounds $\underline{\theta}_l$ can be obtained by solving the Linear Programming (LP)-relaxation of the Steiner tree problem using a dual ascent procedure [11]. The latter variant is chosen in the proposed implementation, since solving the dual ascent is much faster than computing the optimal Steiner tree on G (which is in general much larger than G_S), and the provided bounds are typically close to the optimum.

For the SStA, one can strenghten the cuts (iLS) by redefining the indicator variable, using problem specific knowledge. Observe that no subgraph of G_S may contain a Steiner tree of cost that is $< \phi_l^*$. Thus, (iLS) is strengthened by setting $IND = \sum_{(i,j) \notin A_S} x_{ij}$. Following a similar argument, consider the augmented graph G'_S constructed from G_S by adding to A_S an arbitrary set of arcs A'. G'_S may only contain a Steiner tree of lower cost than ϕ_l^* if at least one arc of A' is incident to some node in V_S. Otherwise, A' would induce one or more components disconnected from all terminal nodes, and would thus be redundant. A consequence is that (iLS) may be further strengthened by setting

$$IND = \sum_{(i,j) \in \delta(V_S)} x_{ij}.$$

Note also that the following type of "no-good" cut

$$IND \geq 1 \tag{NG}$$

is sufficient to cut off sub-optimal solutions. Constraints (NG) are more numerically stable than (iLS), but unfortunately do not affect the value of θ_l. Preliminary experiments have been performed in which both families of inequalities are added whenever $\mathbf{x}^*$ is integral. However, no significant speedup could be achieved by adding (NG), and thus only (iLS) are separated during the final experiments.

3.2 Separation Procedure

Solving the Benders reformulation (B) requires the separation of three types of inequalities: Benders feasibility cuts (B.2), Benders optimality cuts (BC) and integer L-shaped cuts (iLS). The separation of (B.2) is handled by employing the maximum-flow computations, in a similar manner as it is done for the (CUT) formulation (see § 4.1 for further details). The general separation procedure is outlined in Algorithm 1.

Data: Solution $(\mathbf{x}^*, \boldsymbol{\theta}^*)$ of (B)
Result: Set of violated cuts C to (B)

1 Let $T = \bigcup_{l \in L} T_l$
2 $C \leftarrow$ `separateFeasibilityCuts`$(\mathbf{x}^*, T)$
3 **if** $C = \emptyset$ **then**
4 | `OptCut`$= 0$
5 | **for** $l \in L$ **do**
6 | | Solve (D^l) and let $\underline{\Phi}_l(\mathbf{x}^*)$ be the corresponding objective function value and $(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)$ the dual multipliers.
7 | | **if** $\theta_l^* < \underline{\Phi}_l(\mathbf{x}^*)$ **then**
8 | | | Add the optimality Benders cut (BC) to C and `OptCut`++
9 | **if** *`OptCut`*$= 0$ *and* $\mathbf{x}^*$ *is integer* **then**
10 | | **for** $l \in L$ **do**
11 | | | Solve (S.1)-(S.4) and let $\Phi_l(\mathbf{x}^*)$ be the corresponding objective function value.
12 | | | **if** $\theta_l^* < \Phi_l(\mathbf{x}^*)$ **then**
13 | | | | Add the integer L-shaped cut (iLS) to C
14 **return** C

Algorithm 1: Separation procedure for Benders reformulation (B)

In Step 2 the separation of connectivity cuts is performed over set T, which contains terminals of all labels (see Step 1). If none of such cuts is violated by the current $\mathbf{x}^*$ solution, Benders optimality cuts are then separated. For each label $l \in L$ the LP-relaxation of the Benders subproblem is solved in Step 6; if the obtained solution induces a violated optimality cut (i.e., $\theta_l^* < \underline{\Phi}_l(\mathbf{x}^*)$), then the corresponding cut of type (BC) is added in Step 8. If no optimality cut is violated (i.e., `OptCut`$= 0$) and the current master solution $\mathbf{x}^*$ is integer, then integer L-shaped cuts are separated. For each label $l \in L$ the Benders subproblem (S.1)-(S.4) is solved to optimality and the corresponding integer L-shaped cuts (iLS) are added in Step 13 if violated.

To solve the Benders subproblem (S.1)-(S.4) (or its LP-relaxation), which is the Steiner arborescence problem on the support graph G_S, another branch-and-cut algorithm based on the separation of connectivity cuts (S.3) is employed. Thus, the proposed algorithm resembles the so-called *two-stage branch-and-cut* approach from [6] in which the Benders reformulation is implemented as a branch-and-cut algorithm, where in each node of the branch-and-bound tree, a cutting plane procedure is invoked for solving the LP-relaxation of the Benders subproblem $\underline{\Phi}_l(\mathbf{x}^*)$. In some cases, when $\mathbf{x}^*$ is integer and no further violated cuts associated with $\underline{\Phi}_l(\mathbf{x}^*)$ can be found, this cutting plane procedure turns into a branch-and-cut (i.e., the problem $\Phi_l(\mathbf{x}^*)$ is solved and possible violated integer L-shaped cuts are separated). This justifies the name of the approach.

It should be noted that, due to the strength of the LP-relaxation of the MIP model (S.1)-(S.4) and sparsity of the support graph G_S, in most of the cases, Benders subproblems $\Phi_l(\mathbf{x}^*)$ are solved without branching (i.e., we have $\underline{\Phi}_l(\mathbf{x}^*) = \Phi_l(\mathbf{x}^*)$). Furthermore, when $\mathbf{x}^*$

induces a support graph which is a tree, the optimal value of $\Phi_l(\mathbf{x}^*)$ is found by performing a simple breath-first-search procedure. Further implementation details are given in the next section.

3.3 Stabilization Techniques

Another important performance factor for branch-and-cut implementations of Benders reformulations is the number of cutting plane iterations performed at the master level [17]. Since cutting plane procedures based on Benders decomposition are known to exhibit a strong tailing-off effect, solving the problem at hand to optimality may prove to be difficult. Techniques to reduce this tailing-off effect are known as stabilization methods. In the proposed framework, a variant of the *in-out* separation studied in [5, 15, 17] has been used. Intuitively, the idea is to choose a separation point $\mathbf{x_{sep}}$ intelligently such that the separated Benders inequalities cut off larger parts of the infeasible region with respect to (B). In the in-out method, the separation point is chosen as a convex combination between an interior (i.e., feasible) point $\mathbf{x_{in}}$ and an outer (i.e., infeasible) point $\mathbf{x_{out}}$. In the framework, $\mathbf{x_{out}}$ corresponds to the current solution $\mathbf{x}^*$ of (B), and $\mathbf{x_{in}} := (1, \ldots, 1)^{|A|}$. Clearly, this point is a feasible solution to (B). A point to be separated is computed based on the parameter $\Lambda \in (0, 1]$:

$$\mathbf{x_{sep}} := (1 - \Lambda)\, \mathbf{x_{in}} + \Lambda\, \mathbf{x_{out}}$$

Choosing $\mathbf{x_{sep}}$ close to $\mathbf{x_{in}}$ when $\mathbf{x}^*$ is infeasible is likely to generate a feasible point. Thus optimality cuts can be computed earlier on, avoiding the generation of unnecessary Benders feasibility cuts. In the scheme presented in [5], in each cutting plane iteration separation is performed for multiple points, until a violated inequality is found. In the proposed framework, a more conservative variant has been implemented, since the separation of optimality cuts is very time-consuming. For this, a set of parameters $\{\Lambda_0, \ldots, \Lambda_k\}$ is used. Given $\Lambda_0 := 0.1$ each other Λ_i is computed as recursive bisection of the interval $[0.1, 1]$, for $k = 5$. A priority is assigned to each parameter, which is defined as the average increase in the lower bound when this parameter is used. In each cutting plane iteration, the parameter with the highest priority is chosen.

Algorithm 2 presents the stabilized separation procedure for formulation (B). If the in-out separation point $\mathbf{x_{sep}}$ fails to produce any cutting plane, the separation is repeated using $\mathbf{x}^*$. This guarantees that the cutting plane procedure does not terminate prematurely. Procedures `separateOptimalityCuts` and `separateLshapedCuts` refer to the separation shown in Algorithm 1 (cf. lines 5-8 and lines 10-13). Separation of optimality cuts is continued within the branch-and-bound tree, but the number of cutting plane iterations is limited to five per node. Branching is performed on node variables $\mathbf{x}$.

Data: Solution $(\mathbf{x}^*, \boldsymbol{\theta}^*)$ of (B), in-out parameter $\Lambda \in (0, 1]$
Result: Set of violated cuts C to (B)

$\mathbf{x_{sep}} \leftarrow \Lambda\, \mathbf{x}^* + (1 - \Lambda)\mathbf{x_{in}}$
$C \leftarrow$ `separateFeasibilityCuts`$(\mathbf{x_{sep}}, T)$
if $C = \emptyset$ **then**
  $C \leftarrow C \cup$ `separateOptimalityCuts`$(\mathbf{x_{sep}}, \boldsymbol{\theta}^*)$
if $C = \emptyset$ **then**
  $C \leftarrow C \cup$ `separateOptimalityCuts`$(\mathbf{x}^*, \boldsymbol{\theta}^*)$
if $C = \emptyset$ *and* $\mathbf{x}^*$ *is integer* **then**
  $C \leftarrow C \cup$ `separateLshapedCuts`$(\mathbf{x}^*, \boldsymbol{\theta}^*)$

Algorithm 2: Stabilized separation procedure for Benders reformulation (B)

4. Implementation Details

In this section, the main components of the algorithmic scheme designed to tackle the SStA are presented. First, the cutting plane generation strategies for both the (CUT) formulation and the Benders reformulation (B) are outlined. Afterwards, a primal procedure for providing feasible solutions along the optimization task is presented. Finally, preprocessing techniques aiming at reducing the size of the input instances are described.

4.1 Formulation (CUT): separation details

The MIP model induced by the cut-based model (CUT.1)-(CUT.4) is algorithmically approached by an on-the-fly separation of connectivity cuts (CUT.2).

Formulation (CUT) is enhanced by initializing it with additional constraints that help improving the corresponding LP bounds or speed up the cutting plane procedure. In particular, the following *flow-balance* inequalities (CUT.5) and *in-degree* inequalities (CUT.6) are considered, see also [13, 20, 23]:

$$\sum_{(j,i)\in\delta^-(i)} y^l_{ji} \leq \sum_{(i,j)\in\delta^+(i)} y^l_{ij} \qquad \forall i \in V \setminus (T_l \cup \{r\}),\ l \in L \qquad \text{(CUT.5)}$$

$$\sum_{\substack{(k,i)\in\delta^-(i)\\ k\neq j}} y^l_{ki} \geq y^l_{ij} \qquad \forall i \in V \setminus (T_l \cup \{r\}),\ (i,j) \in \delta^+(i),\ l \in L \qquad \text{(CUT.6)}$$

Recall that flow-balance constraints might be infeasible for the shared network $\mathbf{x}$, since an optimal solution does not necessarily correspond to a tree (it may have nodes with multiple ingoing arcs).

Due to the formulation's potentially large number of variables and constraints, careful separation of violated inequalities becomes a crucial factor. Preliminary experiments have shown that the most robust strategy is to add (CUT.2), (CUT.5) and (CUT.6) dynamically. Although initializing the model with a subset of inequalities may help to speed up the

solution of smaller instances, for larger instances the performance is affected negatively, due to the larger LPs.

Data: LP solution $\mathbf{x}^*$ of (CUT)
Result: Set C of inequalities violated by $\mathbf{x}^*$

1 **for** $l \in L$ **do**
2 | Add (CUT.5) and (CUT.6) to C violated by $\mathbf{x}^*$
3 | Add (CUT.2) to C violated by $\mathbf{x}^*$ from cut pool (computed by dual ascent)
4 | Add (CUT.2) to C violated by $\mathbf{x}^*$ (separated by maximum-flow/BFS)

Algorithm 3: Separation procedure for formulation (CUT)

Algorithm 3 shows the separation procedure. For each label, the steps of the algorithm are equivalent to the separation procedure applied to solve the directed-cut formulation of the Steiner tree problem [16]. Thus, improvements proposed in literature also apply to the SStA with some minor adaptations [see, e.g., 16].

Cut Separation using a Dual Ascent Procedure A common technique for Steiner arborescence [see, e.g., 11, 27] for decreasing the number of cutting plane iterations is to initialize the model with a subset of (CUT.2) computed by a dual ascent algorithm [30]. For the SStA, the (CUT) model may be initialized by running dual ascent once for each label, using both, the arc weights and the terminals associated to the label. Depending on instance size and the number of labels, the amount of generated cuts may be too large to solve the initial LP-model within a reasonable time limit. Therefore, in the proposed framework, cuts are stored within a cut pool and only added to the LP model when violated (cf. line 3 of Algorithm 3).

Exact Separation of Cuts using Maximum Flows Exact separation of (CUT.2) is performed using the preflow-push maximum flow algorithm [7]. Instead of separating using the original LP solution values $\mathbf{x}^*$ to define the so-called support graph G_S, a small ε is added to them. This allows to separate arc *minimal* cuts, which are known to be much stronger than the regular cuts [see 20, 23, for further details]. To generate more cuts during one iteration, a limited nested cut separation is applied, i.e., the maximum flow is computed at most $\mathrm{N_{max}}$ times for each terminal (in the computations, $\mathrm{N_{max}}$ is set to 10).

Let G_S be the support graph induced by $\mathbf{x}^*$. To further speed up separation, the maximum flow computation is skipped for each terminal that is already reachable from the root on G_S. Reachability can be checked efficiently by executing a breadth-first-search (BFS) on G_S. Each time a cut is separated, its arcs are added to G_S and reachability is updated. Furthermore, if $\mathbf{x}^*$ is integral, separation by maximum flow is skipped completely and it is instead performed by BFS.

Note that the separation is independent for each label, and may be run in parallel. However, since separation only takes a negligible amount of time at least on the tested instances, this improvement has not been implemented.

4.2 Benders Reformulation (B): Separation Details

Following the general separation framework outlined in Algorithm 1, this section provides further details regarding the stabilization and handling of cutting planes, both at the master level, and for the subproblems.

As for (CUT), when solving $\underline{\Phi}_l(\mathbf{x}^*)$ for a given $\mathbf{x}^*$, these Benders subproblems can be initialized with inequalities (CUT.5) and (CUT.6). However, the addition of flow-balance inequalities (CUT.5) could result into an infeasible LP associated to $\underline{\Phi}_l(\mathbf{x}^*)$. That would require additional Benders feasibility cuts (different than (B.2)) to be added to the master problem. The strategy was evaluated in preliminary experiments, but was discarded for simplicity because it did not result in an additional speedup. On the other hand, in-degree inequalities (CUT.6) are valid for any LP-optimal solution $\mathbf{x}^*$ satisfying (B.2), and are thus separated.

Cut Pools for Benders Subproblems When solving $\underline{\Phi}_l(\mathbf{x}^*)$, some of the cuts of type (S.3) may remain violated, even when the values of $\mathbf{x}^*$ change. Thus, it pays off to store the already separated inequalities and use them to initialize the branch-and-cut pool for solving $\underline{\Phi}_l(\mathbf{x}^*)$ in all subsequent iterations. To prevent the underlying MIP model (S.1)-(S.4) from becoming too large, cuts with positive slack in the current LP solution are removed at regular intervals. This cleanup step is performed every twenty iterations. As in the separation of connectivity cuts, adding a small ε to $\mathbf{x}^*$ appears to also improve the strength of separated Benders optimality cuts.

Parallelization Solving the Benders subproblem $\Phi_l(\mathbf{x}^*)$, $l \in L$, is clearly a bottleneck for an efficient implementation. One can observe that for each $l \in L$, the Benders subproblem $\Phi_l(\mathbf{x}^*)$ (or its LP-relaxation) can be solved independently, and thus, the separation procedure may be parallelized. In contrast to formulation (CUT), this option has been implemented into the framework, since it has a measurable impact on performance.

Underestimating the Value of $\underline{\Phi}_l(\mathbf{x}^*)$ Given an optimal solution $(\mathbf{x}^*, \boldsymbol{\theta}^*)$ of the Benders master problem, recall that a violated Benders optimality cut (BC) is found whenever $\theta_l^* < \underline{\Phi}_l(\mathbf{x}^*)$ for some $l \in L$. However, finding the optimal solution of $\underline{\Phi}_l(\mathbf{x}^*)$ may not be necessary, and even a valid underestimation of this value, say $LB_l < \underline{\Phi}_l(\mathbf{x}^*)$, is often sufficient to cut off a solution $(\mathbf{x}^*, \boldsymbol{\theta}^*)$ of the master problem. In that case, the condition $\theta_l^* < \underline{\Phi}_l(\mathbf{x}^*)$ (see line 7 of Algorithm 1) is replaced by a valid condition $\theta_l^* < LB_l$. Especially during the first few cutting plane iterations, when (B) does not contain much information in the form of inequalities, solving $\underline{\Phi}_l(\mathbf{x}^*)$ to optimality usually does not pay off.
Two possible ways to calculate LB_l are considered.

1. Solving $\underline{\Phi}_l(\mathbf{x}^*)$ only for a limited number of cutting plane iterations, provides a valid lower bound LB_l. In the proposed framework, the number of cutting plane iterations for solving $\underline{\Phi}_l(\mathbf{x}^*)$ is by default limited to one. If such obtained LB_l yields no violated inequality (i.e., $\theta_l^* \geq LB_l$), $\underline{\Phi}_l(\mathbf{x}^*)$ is solved again with the iteration limit removed.

2. Alternatively, every feasible solution $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ to the dual of the Benders subproblem yields a valid lower bound LB_l. Consequently, if $\boldsymbol{\alpha}$ is fixed, (D^l) corresponds to the dual of the directed cut formulation for the Steiner arborescence problem, and the remaining variables $\boldsymbol{\beta}$ can be calculated heuristically by applying the dual ascent algorithm on G [see, e.g. 11], given terminals T^l and modified weights $c'_{ij} = w^l_{ij} + \alpha_{ij}, \forall (i,j) \in A$. Various heuristic schemes have been devised for choosing $\boldsymbol{\alpha}$ based on $\mathbf{x}^*$, e.g., $\alpha_{ij} = (1 - x^*_{ij})c_{ij}, \forall (i,j) \in A$. The intuition behind this approach is to penalize the use of arcs not part of the shared network induced by $\mathbf{x}^*$. Note that if $\mathbf{x}^*$ covers the whole graph G, for each $(i,j) \in A$, $\alpha_{ij} = 0$, and the computed lower bound is a global lower bound.

The second approach seems promising since computing lower bounds only involves the execution of a fast heuristic instead of solving a potentially time-consuming LP. Unfortunately, preliminary experiments showed that none of the considered schemes for choosing $\boldsymbol{\alpha}$ managed to produce lower bounds competitive with the ones computed by the first approach. Thus in all final experiments, only the first approach is applied.

4.3 Primal and Initialization Heuristic

Algorithm 4 describes a heuristic procedure that works for both (CUT) and (B). It computes a feasible solution from an LP solution $(\mathbf{x}^*, \mathbf{y}^{1*}, \ldots, \mathbf{y}^{|L|*})$. For the Benders reformulation, variables $\mathbf{y}$ are projected out, but they are easily retrieved from the LP-solutions of Benders subproblems, $\underline{\Phi}_l(\mathbf{x}^*)$, for the given $\mathbf{x}^*$, $l \in L$. Adapted costs $\bar{\mathbf{w}}^\mathbf{l}$ are then computed based on $(\mathbf{x}^*, \mathbf{y}^*)$ and the original arc costs for each $l \in L$. Given T_l and $\bar{\mathbf{w}}^\mathbf{l}$, a heuristic Steiner tree S_l is constructed for each $l \in L$. To encourage arc sharing, for each new Steiner tree S_l, the shared costs are temporarily set to zero for all $(i,j) \in S_l$. Finally, a feasible solution S to the SStA is computed as the union of all S_l. The heuristic Steiner tree construction is performed using a shortest path heuristic [see, e.g., 10, 29]. A heuristic starting solution is constructed by executing the algorithm with all LP values set to zero. For each formulation, the primal heuristic is called once after each cutting plane iteration and in each tenth branch-and-bound node.

Data: LP values $(\mathbf{x}^*, \mathbf{y}^{1*}, \ldots, \mathbf{y}^{|L|*})$
Result: feasible solution S

$\bar{c}_{ij} \leftarrow c_{ij} \qquad \forall (i,j) \in A$
for $l \in L$ **do**
  $\bar{w}^l_{ij} \leftarrow (1-\alpha)(1 - x^*_{ij})\bar{c}_{ij} + \alpha(1 - y^{l*}_{ij})w^l_{ij},\ \forall (i,j) \in A$
  $S_l \leftarrow \texttt{constructHeuristicST}(T_l, \bar{\mathbf{w}}^\mathbf{l})$
  $\bar{c}_{ij} \leftarrow 0,\ \forall (i,j) \in S_l$
$S \leftarrow \bigcup_{l \in L} S_l$

Algorithm 4: Primal heuristic

4.4 Preprocessing

The following preprocessing tests have been derived from tests originally proposed for the Steiner tree problem on undirected graphs (see, e.g., [14]). For this purpose let $V \setminus (T \cup \{r\})$ denote the set of non-terminal nodes, where T was previously defined as $T := \bigcup_{l \in L} T_l$.

Test 1. Non-Terminal of Degree 1 (NTD1): A non-terminal node with exactly one adjacent node can be removed because there is always an optimal solution without it.

Test 2. Terminal of Degree 1 (TD1): Given a terminal node with exactly one incoming arc, this arc must be part of all optimal solutions, and may be fixed.

Test 3. Non-Terminal of Degree 2 (NTD2): Given a non-terminal node i with exactly two adjacent nodes j and k, as well as incoming arc (j, i) and outgoing arc (i, k), these arcs are either both part of an optimal solution or neither of them are. Thus (j, i) and (i, k) may be replaced by a new arc (j, k) with costs $c_{jk} := c_{ji} + c_{ik}$ and $w^l_{ji} := w^l_{ji} + w^l_{ik}$ for all $l \in L$.

Test 4. Non-Terminal Reachability (NTR): A non-terminal not reachable from the root through a directed path may be removed.

The validity of Tests 1–4 follows directly from their definition. Note that for Test NTD2, if there already exists an arc (j, k) such that its costs (c_{jk} and w^l_{jk}, for all $l \in L$) are incomparable to the newly inserted arc's costs, neither can replace the other, making G a multigraph. For simplification, in the proposed implementation the test is always skipped in this specific case, which only occurs for an relatively few non-terminals. Otherwise, all tests are applied applied exhaustively.

5. Test Instances

In the following, the benchmark instances considered in the computational experiments are described. As already explained in the introduction, these instances come from two very different application contexts.

Biological Network Analysis Instances proposed in [26] are considered as part of this dataset. They are based on the same biological network or *interactome*, which is comprised of 10 169 nodes and 44 738 bidirectional interactions. The first instance, `ER-export`, consists of the original network with two labels (52 and 33 terminals). The second instance, `Influenza`, consists of four labels (with 7, 14, 17 and 34 terminals, respectively); for this instance, the network is augmented by adding an artificial root node, additional nodes and interactions, resulting in 25 additional nodes and 144 additional bidirectional interactions. Note that, as in the original paper [26], terminals not reachable from the root are excluded.

Each bidirectional interaction is represented by two anti-parallel arcs, whose arc weights are set to one; hence, the number of interactions is minimized. In the following these two instances will be referred to as `BIO` dataset.

Additional instances associated to Bioinformatics applications have been generated from the `ACTMOD` dataset, originally proposed for the prize-collecting Steiner tree problem [1] and related to the problem presented in [12]. This set is composed of 8 instances. These instances represent undirected biological networks, in which nodes and arcs are assigned non-negative weights. For the purpose of generating SStA instances, each edge is replaced by two anti-parallel arcs, and nodes with weight greater than the weight of each adjacent edge are treated as terminals. For each graph, instances with 2, 4 and 6 labels have been generated, where each label contains 5% to 55% of all terminals, selected at random. This leads to a total of 24 SStA instances. Note that all arc weights are set to one.

Telecommunication Network Design In [22], two groups of network design instances have been generated from real-world telecommunication networks, that are anonymized for the data protection purposes. Starting with these instances (originally proposed for the Steiner tree problem), the following subsets are considered: The first group, which is labeled as `VIENNA`, contains instances that have been preprocessed using the Steiner tree preprocessing procedures; 20 instances have been selected from the original dataset instances with less than 5500 nodes (after preprocessing). The second group contains instances to which no preprocessing has been applied (`GEO`). Due to the fact that many instances in this group are of similar structure, only the first 7 instances have been selected from the original dataset (G101-G107). These instances are, on average, as large as the others within this dataset. Complementary information on this dataset is available in [22].

For each original instance, new SStA instances with 3, 6 and 9 labels have been generated, leading to a total of 60 SStA instances in `VIENNA` and 21 in `GEO`. Labels are divided equally into three groups depending on the size of the corresponding set of terminals. Each label group is regarded as small, medium and large retailer, with 10-20%, 40-60% and 70-90% of randomly selected terminals, respectively. Each edge from the original instance is split into two anti-parallel arcs. Original arc weights are used for the shared network costs, while the costs for each label are assigned randomly in the range of $\pm 20\%$ of the shared network's arc costs. The resulting graphs contain between 160 and 5195 nodes, with the number of arcs ranging from 474 to 15722. The instances are available from authors upon request.

6. Computational Results

Experimental Setting The algorithmic scheme described in §4 has been implemented in C++ and compiled using GCC 4.9. The framework is based on OGDF [8] for graph data structures and CPLEX Concert framework 12.6 for the solution of MIP formulations. The experiments have been performed on an Intel Xeon CPU with 2.5 GHz and 20 cores. A

single core has been used for formulation (CUT) and (SCF). The master problem of Benders reformulation (B) is processed by a single core, while the subproblems are processed in parallel using up to four cores. A fixed memory limit of 16 GB RAM has been imposed, which is increased to 32 GB for the large-scale GEO instances. Unless stated otherwise, computational results report the relative optimality gap between the lower and upper bound computed per algorithm, which is abbreviated as "gap" within tables.

Effect of Preprocessing The results obtained by applying the preprocessing tests described in §4.4 are reported in Table 1 for datasets `BIO`, `ACTMOD` and `GEO`. In the case of `BIO` instances, two labels are considered for the ER-export instance, and 4 labels for the Influenza instance. In the case of `ACTMOD` dataset, instances with six labels are taken into account. Finally, instances with 9 labels are considered from the `GEO` dataset. Results for other numbers of labels are omitted due to the fact that the preprocessing effectiveness is nearly independent from the number of labels. The results on biological networks show that the proposed preprocessing is highly effective for the metabol_expr_mice instances, where more than 60% of all arcs are eliminated ($|A_P|/|A| < 0.40$). The tests are much less effective on the other instances, in which networks are densely connected and most nodes have a high degree. More sophisticated tests could possibly reduce these graphs further. For the `GEO` dataset, even slightly more than half of all arcs can be eliminated, mainly due to the NTD2 test.

For dataset `VIENNA`, detailed preprocessing results have also been omitted, since the performed tests are unable to remove any nodes or arcs on this instance type. The reason is that `VIENNA` has been generated from preprocessed Steiner tree instances, where already several tests stronger than the ones proposed in this paper have been applied.

Benefits of Stabilizing and Parallelizing The effect of the stabilization and parallelization on the decomposition scheme associated with Benders reformulation (B) has been evaluated. Figure 2 shows the influence of the in-out separation on a example of instance I053a ($|L| = 9$) from the `VIENNA` dataset. As expected, by employing the stabilization, the tailing-off effect is strongly decreased [see also 17]. Curve B (Benders decomposition without stabilization) contains steps. This behavior can be explained by the fact that without stabilization, Benders feasibility cuts (i.e., cuts ensuring the connectivity of the shared network) have to be separated until the support graph G_S is connected. Using the in-out separation, moving into the direction of the chosen stabilization point increases the likelihood of connectivity, enabling earlier separation of optimality cuts (and consequently, an earlier improvement of lower bounds).

The benefits of the stabilization can be better appreciated when comparing, in Figure 2, the curves induced by the (CUT) formulation and the stabilized Benders; although at the very begging the (CUT) formulation provides tighter bounds, the stabilized Benders yields much better results already after a couple of seconds.

Table 2 shows a computational comparison of different parameter settings used to apply the stabilization procedure, described before, for datasets `VIENNA` and `ACTMOD`. For

	original			*preprocessed*			
instance	$\|V\|$	$\|A\|$	$\|T\|$	$\|V_P\|$	$\|A_P\|$	$\|T_P\|$	$\|A_P\|/\|A\|$
HCMV	3481	57420	66	2763	55622	66	0.97
drosophila001	5226	186788	125	3911	183638	125	0.98
drosophila005	5226	186788	242	3925	183678	242	0.98
drosophila0075	5226	186788	484	3947	183740	473	0.98
lymphoma	2034	15512	73	1260	13824	73	0.89
metabol_expr_mice_1	2786	7458	156	701	2852	142	0.38
metabol_expr_mice_2	2777	7432	69	643	2686	67	0.36
metabol_expr_mice_3	2192	5654	97	444	1882	88	0.33
ER	10169	89476	86	6151	80902	86	0.90
Influenza	10194	89764	58	6168	81178	57	0.90
(average)							0.73
G101	67966	164970	100	22874	71220	100	0.43
G102	111707	321008	2052	62288	220304	2052	0.69
G103	135543	403606	3033	82369	295890	3033	0.73
G104	158212	480044	3914	100866	364156	3914	0.76
G105	79244	202378	550	31824	104266	550	0.52
G106	204621	636272	5556	137526	501138	5556	0.79
G107	85568	228226	938	39120	132838	938	0.58
(average)							0.64

Table 1: Preprocessing results for datasets ACTMOD ($|L|= 6$), GEO ($|L|= 9$) and BIO . The columns show the number of nodes, arcs and terminals before and after preprocessing. The last column contains the ratio between the number of arcs before and after preprocessing.

each strategy the total number of instances solved to optimality ("#solved"), the average gap (in %) attained by those not solved to optimality ("gap"), and the average running time ("time") in seconds, are reported. Columns "W/O STAB." show results when no stabilization is used. Columns "IN-OUT 1" show the effect of incorporating the in-out separation by choosing the closest violated cut to the interior point; here separation is executed for multiple points, and separation is stopped as soon as a violated optimality cut has been found [see 5, for a further explanation]. In columns "IN-OUT 2" the effect of implementing the in-out separation (c.f. §3.3) using the on average best stabilization parameter is reported. In all cases, the best results are marked in bold. The results show that the first variant already has a strong effect on the average gap. One less instance is solved than without in-out stabilization. This can be attributed to the fact that on average the time spent solving problems to optimality is longer, and thus less cutting plane iterations are performed within the time limit. Variant 2 spends less time finding separation points, but manages to preserve the quality of lower bounds due to the improved way of choosing the parameter Λ. Due to the faster separation, six more instances from VIENNA

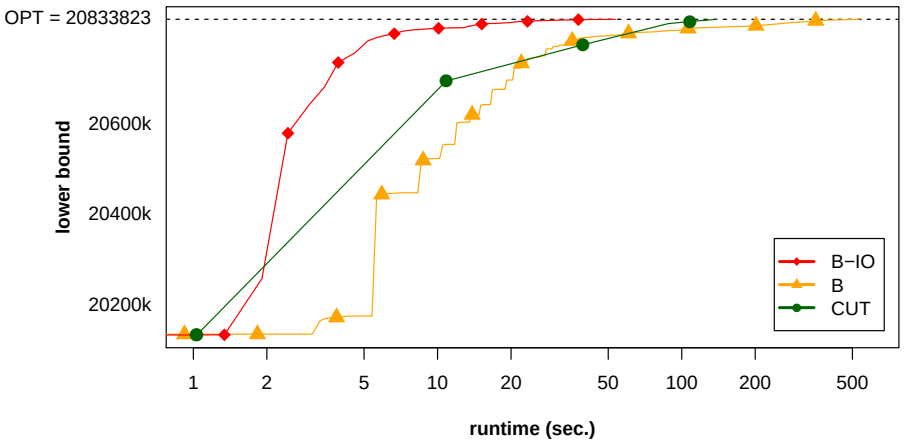


Figure 2: Effect of the stabilization procedures for instance VIENNA/I053a ($|L|= 9$). (B-IO) shows the performance with in-out separation and (B) shows performance without stabilization. CUT shows the lower bound evolution corresponding to the (CUT) formulation. These lower bound values are drawn in relation to the runtime in seconds, given in logarithmic scale. The optimal objective value is displayed by OPT and the dashed line.

		W/O STAB.			IN-OUT 1			IN-OUT 2		
dataset	#inst.	#solved	gap (%)	time (s)	#solved	gap (%)	time (s)	#solved	gap (%)	time (s)
VIENNA	60	15	0.92	2791	16	0.16	2740	**22**	**0.14**	**2490**
ACTMOD	24	10	4.25	2309	9	2.19	2403	**10**	**2.11**	**2136**

Table 2: Parameter comparison for stabilization procedures applied to all instances from groups VIENNA and ACTMOD.

are solved to optimality. Thus Variant 2 has been selected for all subsequent experiments.

Table 3 compares the influence of subproblem parallelization for formulation (B). The results show that for both the VIENNA and ACTMOD dataset, the improvements are only minor on average. For explaining this behavior, it must be noted that in the proposed implementation for each label optimality cuts are separated until the next master problem is solved. This may introduce a large amount of idle time if a single subproblem requires much more computation time than the rest, thus limiting the achievable speedup. However, parallelization is still a considerable factor when dealing with instances that have a very high number of labels.

Comparing Performances: (CUT) v/s (B) Tables 4 and 5 show detailed computational results on biological networks from ACTMOD and BIO, respectively. In both cases, performance measures are provided in columns "gap" (as %) and "time" (in seconds), for the different algorithmic strategies and for different number of labels. Additionally, for the BIO the gap calculated with respect to the best found primal solution is also provided in

		1 THREAD			2 THREADS			4 THREADS		
dataset	#inst.	#solved	gap	time	#solved	gap	time	#solved	gap	time
VIENNA	60	20	0.19	2556	21	0.16	2514	**22**	**0.14**	**2490**
ACTMOD	24	10	2.30	2158	10	2.19	2140	**10**	**2.11**	**2136**

Table 3: Parameter comparison for parallel solution of formulation (B). For each tested number of threads and dataset, the columns list the number of solved instances, the average optimality gap in percent and the average runtime in seconds, given a one-hour timelimit. The best results are marked in bold.

columns "p.gap" (as well as %). Note that the performance on the real-world dataset BIO and the bioinformatics-based ACTMOD instances is very similar. This shows that also ACTMOD instances capture the difficulty of the problem, even though the labels were assigned randomly. Formulation (SCF) performs worst for all instances. One observes that for a small number of labels, there is no computational advantage in decomposition, what results in (CUT) outperforming (B) for $|L|= 2$ (ACTMOD) and for both BIO instances. However, by increasing the number of labels, the gaps of (B) are getting closer to those returned by (CUT). Furthermore, for the largest ACTMOD instances (drosophila001, drosophila005), gaps obtained by (B) are the best ones, which marks the point at which the high number of variables in (CUT) becomes too much of a burden. At this point formulation (B) becomes the more favorable choice.

Table 6 shows detailed computational results on infrastructure networks from GEO and VIENNA. In comparison with the biological networks, the associated graphs are much sparser. Again, formulation (SCF) performs worst on all instances. (B) outperforms (CUT) in most of the cases, already for $|L|= 3$. On the largest instances, gaps obtained by (B) after one hour of computation are up to one order of magnitude smaller than for (CUT). This trend continues for higher values of $|L|$, for which decomposition enables (B) to scale much better. Note that after reaching the one-hour time limit, (B) has always computed near-optimal bounds, but fails to prove optimality for most instances. This behavior is attributable to the strong tailing-off effect of the Benders decomposition, which remains noticeable despite using the in-out separation.

The results in Table 6 show that for the large-scale GEO instances, both formulation (CUT) and (SCF) are much less effective due to their extremely high number of variables. Formulation (SCF) fails to solve the root relaxation within the time limit for most instances. Formulation (CUT) produces gaps ranging between 35% and 70%. In contrast, (B) scales much better, and computes bounds with gaps within 5-10%, even for $|L|= 9$.

Figure 3 shows a graphical performance summary on datasets VIENNA and GEO. For each number of labels, a plot compares the number of instances solved within a certain gap per formulation. The plots clearly show that the proposed stabilized Benders decomposition approach outperforms the other two algorithmic alternatives, specially for the GEO instances. Take, for instance the plot for GEO instances with $|L|= 6$. While the decomposi-

tion approach is able to solve all 7 instances with gaps less or equal than 10%, the (CUT) approach produces 6 solutions with more than 40%, and the (SCF) approach is simply unable to provide a primal bound. A similar situation repeats for the other group and for other values of $|L|$.

Tables 7 and 8 compare the quality of upper bounds computed within a time limit of one hour for datasets `ACTMOD`, `VIENNA` and `GEO`. The number of nodes $|V|$ and arcs $|A|$ after preprocessing is given for $|L|= 6$ (`ACTMOD`) and $|L|= 9$ (`VIENNA`, `GEO`). For each number of labels, column "best" lists the objective values of the best solution found. The next three columns list for each algorithm the gap between the best solution and the algorithm's best found solution. Results show that the quality of upper bounds is generally much better than the quality of lower bounds for (CUT) and (SCF), and that formulation (B) achieves the best upper bounds on average. This highlights another advantage of the Benders reformulation versus the other two MIP models: for the SStA, Benders decomposition is capable of delivering high-quality solutions within a short computing time, and thus, may be a competitive tool against heuristics, with the clear advantage of computing good lower bounds as a guarantee for the solution quality.

	$\|L\|=2$						$\|L\|=4$						$\|L\|=6$					
	B		CUT		SCF		B		CUT		SCF		B		CUT		SCF	
instance	gap	time	gap	time	gap	time	gap	time	gap	time	gap	time	gap	time	gap	time	gap	time
HCMV	2.14	*TL*	**0.00**	**1877**	38.81	*TL*	1.41	*TL*	**0.29**	***TL***	36.69	*TL*	1.91	*TL*	**0.89**	***TL***	38.49	*TL*
drosophila001	1.14	*TL*	**0.00**	**847**	32.54	*ML*	**2.68**	***TL***	3.86	*TL*	34.08	*ML*	**8.36**	***TL***	11.49	*TL*	39.24	*TL*
drosophila005	1.74	*TL*	**0.00**	**1605**	30.48	*TL*	4.10	*TL*	**3.20**	***TL***	31.99	*TL*	**7.64**	***TL***	9.96	*TL*	35.64	*TL*
drosophila0075	1.22	*TL*	**0.54**	***TL***	24.87	*ML*	4.49	*TL*	**3.73**	***TL***	26.14	*TL*	7.31	*TL*	**6.96**	***TL***	82.30	*TL*
lymphoma	0.00	339	**0.00**	**12**	23.66	*TL*	1.47	*TL*	**1.07**	***TL***	21.09	*TL*	5.04	*TL*	**2.33**	***TL***	27.00	*TL*
metabol_expr_mice_1	0.00	1	**0.00**	**1**	10.03	*TL*	0.00	19	**0.00**	**5**	18.73	*TL*	0.00	305	**0.00**	**54**	23.75	*TL*
metabol_expr_mice_2	0.00	1	**0.00**	**1**	3.89	*TL*	0.00	116	**0.00**	**17**	24.55	*TL*	0.00	46	**0.00**	**43**	24.92	*TL*
metabol_expr_mice_3	0.00	4	**0.00**	**1**	16.59	*TL*	0.00	8	**0.00**	**3**	19.01	*TL*	0.00	25	**0.00**	**15**	19.97	*TL*
(average)	0.78	1843	0.07	993	22.61	3143	1.77	2268	1.52	2253	26.53	3406	3.78	2297	3.95	2264	36.41	3600

Table 4: Computational results for dataset `ACTMOD`. A time limit of one hour and memory limit of 16GB has been used.

		B			CUT			SCF		
instance	best	gap	p.gap	time	gap	p.gap	time	gap	p.gap	time
`ER`	144.5	2.56	0.01	*TL*	**0.77**	**0.00**	***TL***	33.73	0.08	*TL*
`Influenza`	136.5	4.71	0.02	*TL*	**1.85**	**0.00**	***TL***	35.36	0.06	*TL*

Table 5: Computational results for dataset `BIO`. A time limit of one hour and memory limit of 16 GB has been used.

	$\|L\|=3$						$\|L\|=6$						$\|L\|=9$					
	B		CUT		SCF		B		CUT		SCF		B		CUT		SCF	
instance	gap	time	gap	time	gap	time	gap	time	gap	time	gap	time	gap	time	gap	time	gap	time
G101	**6.63**	**TL**	32.88	*TL*	60.45	*TL*	**7.50**	***TL***	41.12	*TL*	-	*TL*	**8.65**	***TL***	50.81	*TL*	-	*TL*
G102	**7.44**	***TL***	52.36	*TL*	76.75	*TL*	**8.74**	***TL***	60.67	*TL*	-	*TL*	**9.33**	***TL***	66.75	*TL*	-	*TL*
G103	**6.91**	***TL***	57.69	*TL*	-	*TL*	**8.66**	***TL***	65.49	*TL*	-	*TL*	**9.36**	***TL***	69.06	*TL*	-	*TL*
G104	**6.16**	***TL***	59.70	*TL*	-	*TL*	**8.62**	***TL***	63.47	*TL*	-	*TL*	**8.86**	***TL***	68.58	*TL*	-	*TL*
G105	**6.92**	***TL***	26.87	*TL*	63.48	*TL*	**7.98**	***TL***	36.45	*TL*	-	*TL*	**8.33**	***TL***	45.43	*TL*	-	*TL*
G106	**32.45**	***TL***	56.91	*TL*	-	*TL*	**7.62**	***TL***	60.80	*TL*	-	*TL*	**21.55**	***TL***	61.96	*TL*	-	*TL*
G107	**8.09**	***TL***	45.89	*TL*	82.37	*TL*	**9.49**	***TL***	60.00	*TL*	-	*TL*	**9.99**	***TL***	70.07	*TL*	-	*TL*
(average)	10.66	3600	47.47	3600	83.29	3600	8.37	3600	55.43	3600	-	3600	10.87	3600	61.81	3600	-	3600
I004a	0.00	95	**0.00**	**5**	15.23	*TL*	0.01	*TL*	**0.00**	**116**	14.73	*TL*	0.01	*TL*	**0.00**	**420**	18.51	*TL*
I005a	0.00	478	**0.00**	**31**	17.46	*TL*	0.00	*TL*	**0.00**	**422**	21.64	*TL*	0.00	*TL*	**0.00**	**2098**	20.82	*TL*
I012a	0.00	*TL*	**0.00**	**1343**	14.24	*TL*	**0.02**	***TL***	3.29	*TL*	16.52	*TL*	**0.06**	***TL***	5.14	*TL*	16.72	*TL*
I014a	0.00	1284	**0.00**	**311**	12.74	*TL*	**0.00**	**1148**	1.27	*TL*	16.41	*ML*	**0.00**	***TL***	4.53	*TL*	14.71	*TL*
I021a	**0.31**	***TL***	3.67	*TL*	21.98	*TL*	**0.61**	***TL***	11.52	*TL*	23.38	*TL*	**1.05**	***TL***	15.48	*TL*	24.61	*TL*
I039a	0.00	*TL*	**0.00**	**1290**	8.08	*TL*	**0.04**	***TL***	3.04	*TL*	8.17	*TL*	**0.10**	***TL***	3.60	*TL*	8.83	*TL*
I043a	**0.05**	***TL***	0.13	*TL*	10.64	*TL*	**0.27**	***TL***	3.27	*TL*	12.43	*TL*	**0.45**	***TL***	5.01	*TL*	12.45	*ML*
I048a	0.15	*TL*	**0.00**	**1888**	15.23	*TL*	**0.59**	***TL***	3.80	*TL*	14.67	*ML*	**0.44**	***TL***	5.55	*TL*	16.62	*TL*
I052a	**0.00**	**1**	**0.00**	**1**	0.85	*TL*	**0.00**	**1**	**0.00**	**1**	5.61	*TL*	**0.00**	**1**	0.00	1	9.93	*ML*
I053a	**0.00**	**3**	0.00	10	6.42	*TL*	**0.00**	**16**	0.00	33	8.06	*TL*	**0.00**	**23**	0.00	103	8.99	*TL*
I054a	**0.00**	**1**	0.00	5	19.18	*TL*	**0.00**	**4**	0.00	86	22.62	*TL*	**0.00**	**10**	0.00	240	28.25	*TL*
I055a	0.08	*TL*	**0.00**	**1339**	14.15	*TL*	**0.33**	***TL***	2.18	*TL*	17.20	*TL*	**0.58**	***TL***	4.68	*TL*	16.41	*TL*
I056a	**0.00**	**1**	**0.00**	**1**	9.25	*TL*	**0.00**	**1**	0.00	3	11.99	*TL*	**0.00**	**1**	0.00	17	14.47	*TL*
I059a	0.00	285	**0.00**	**106**	17.01	*TL*	0.00	*TL*	**0.00**	**1678**	19.01	*TL*	**0.00**	***TL***	4.37	*TL*	18.42	*TL*
I065a	0.00	112	**0.00**	**57**	9.57	*TL*	0.00	1161	**0.00**	**863**	10.61	*TL*	**0.00**	**3232**	0.28	*TL*	11.99	*TL*
I066a	**0.23**	***TL***	4.43	*TL*	18.21	*TL*	**0.09**	***TL***	10.26	*TL*	21.61	*ML*	**0.16**	***TL***	13.46	*TL*	24.85	*TL*
I069a	0.10	*TL*	**0.00**	**2211**	8.46	*TL*	**0.21**	***TL***	1.79	*TL*	10.75	*ML*	**0.51**	***TL***	4.00	*TL*	11.15	*TL*
I074a	0.00	*TL*	**0.00**	**1421**	9.67	*TL*	**0.07**	***TL***	2.01	*TL*	11.99	*TL*	**0.11**	***TL***	4.18	*TL*	11.61	*TL*
I076a	**0.22**	***TL***	1.58	*TL*	17.13	*TL*	**0.63**	***TL***	6.21	*TL*	17.50	*TL*	**0.95**	***TL***	9.45	*TL*	19.03	*TL*
I085a	0.00	1160	**0.00**	**794**	15.37	*ML*	**0.00**	**3572**	1.71	*TL*	17.39	*TL*	**0.00**	***TL***	3.78	*TL*	18.37	*TL*
(average)	0.06	1971	0.49	1261	13.04	3560	0.14	2635	2.52	2320	15.11	3178	0.22	2863	4.18	2664	16.34	3457

Table 6: Computational results for datasets `GEO` and `VIENNA`. For each number of labels, column pairs B, CUT and SCF list optimality gaps in percent and runtimes in seconds, given a one-hour time limit, a 16 GB memory limit for `VIENNA` and 32 GB for `GEO`. Runs which reached the time limit are denoted by *TL*. Runs which reached the memory limit are denoted by *ML*. If the formulation's root relaxation could not been solved within the time limit, then "-" is written instead of a gap value. For each instance and label configuration, the best results are marked in bold.

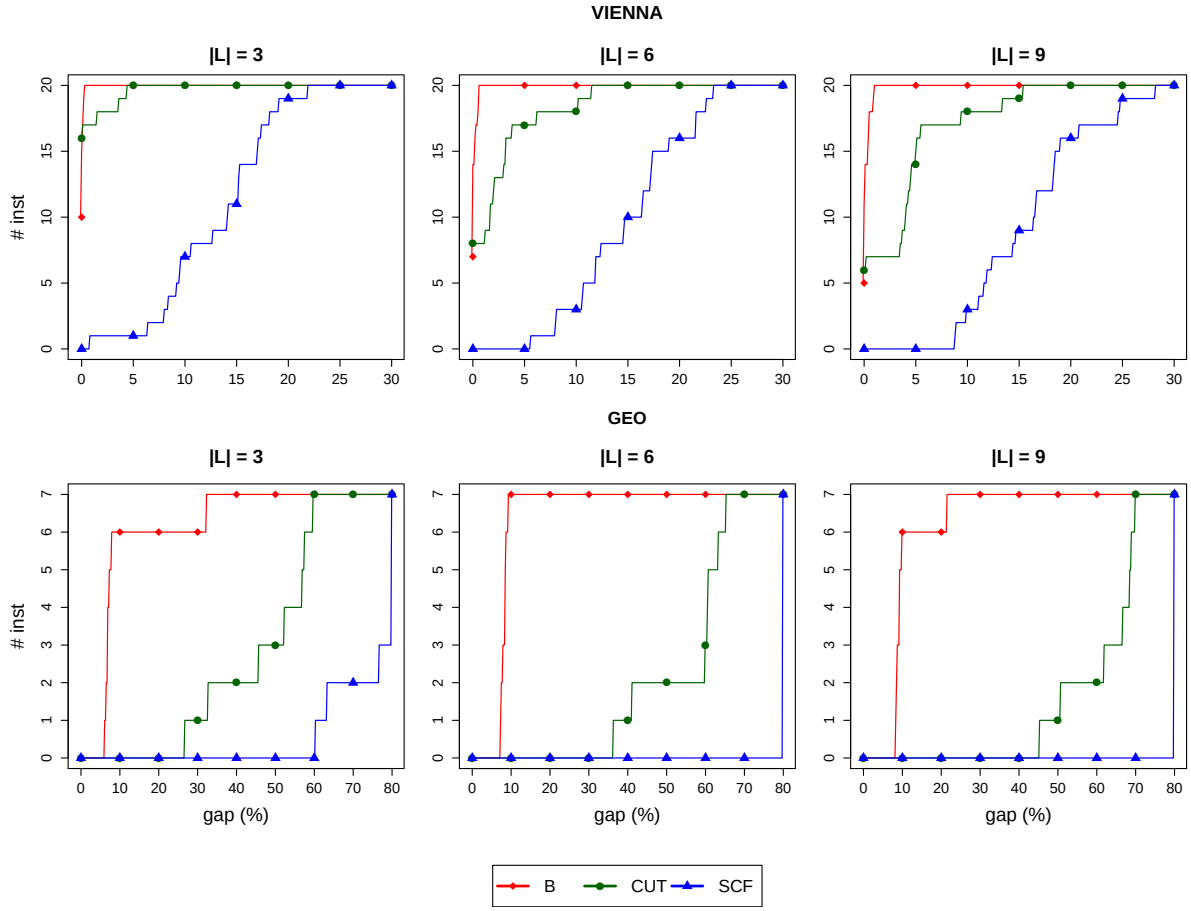


Figure 3: Performance comparison of formulations on datasets `ACTMOD` and `VIENNA`. For each number of labels and dataset, the number of instances solved below a given gap within a one-hour time limit is shown.

			$\|L\|=2$				$\|L\|=4$				$\|L\|=6$			
instance	$\|V\|$	$\|A\|$	best	B	CUT	SCF	best	B	CUT	SCF	best	B	CUT	SCF
HCMV	2753	55622	69.5	0.71	**0.00**	3.47	174.5	0.29	**0.00**	2.79	168.0	**0.00**	**0.00**	4.00
drosophila001	3911	183638	112.5	0.88	**0.00**	7.41	261.5	**0.00**	1.51	5.94	387.5	**0.00**	2.52	3.49
drosophila005	3925	183678	199.0	1.00	**0.00**	7.23	439.0	0.79	**0.00**	4.98	696.0	**0.00**	2.32	3.27
drosophila0075	3947	183740	355.0	0.28	**0.00**	6.21	934.5	0.05	**0.00**	3.26	1230.0	**0.00**	0.20	1.13
lymphoma	1260	13824	41.0	**0.00**	**0.00**	5.75	130.0	**0.00**	0.38	5.11	205.5	1.44	**0.00**	6.16
metabol_expr_mice_1	701	2852	118.0	**0.00**	**0.00**	**0.00**	252.0	**0.00**	**0.00**	1.18	518.0	**0.00**	**0.00**	1.99
metabol_expr_mice_2	643	2686	39.5	**0.00**	**0.00**	**0.00**	259.5	**0.00**	**0.00**	1.70	353.5	**0.00**	**0.00**	1.53
metabol_expr_mice_3	444	1882	163.5	**0.00**	**0.00**	0.91	140.5	**0.00**	**0.00**	2.09	354.5	**0.00**	**0.00**	2.34

Table 7: Comparsion of upper bounds for dataset `ACTMOD`.

			$\|L\|=3$				$\|L\|=6$				$\|L\|=9$			
instance	$\|V\|$	$\|A\|$	best	B	CUT	SCF	best	B	CUT	SCF	best	B	CUT	SCF
G101	22874	71220	5105588.5	**0.00**	2.02	1.64	8391072.0	**0.00**	3.03	13.14	11416759.5	**0.00**	2.30	11.54
G102	62288	220304	21786805.5	0.01	2.85	**0.00**	36218123.5	**0.00**	2.66	12.06	48512857.0	**0.00**	2.78	11.85
G103	82369	295890	26776824.0	**0.00**	2.76	7.00	46714297.0	**0.00**	3.13	12.06	64348279.5	**0.00**	3.32	11.61
G104	100866	364156	36272316.5	**0.00**	3.19	7.48	59588363.0	**0.00**	3.62	12.02	83926883.5	**0.00**	3.67	11.92
G105	31824	104266	15815768.0	0.37	2.46	**0.00**	28027163.0	**0.00**	2.44	10.40	39863061.0	**0.00**	2.87	10.88
G106	137526	501138	58301945.0	**0.00**	0.25	1.83	104908497.5	**0.00**	4.49	11.26	140736008.5	**0.00**	0.23	8.38
G107	39120	132838	10828742.0	**0.00**	1.62	0.67	18646401.5	**0.00**	2.69	13.27	26406929.5	**0.00**	2.27	11.76
I004a	867	2476	55852413.0	**0.00**	**0.00**	2.11	87131485.0	0.01	**0.00**	2.38	116976915.5	0.00	**0.00**	5.48
I005a	1677	4860	72857405.0	**0.00**	**0.00**	3.50	117034442.5	0.00	**0.00**	5.14	155166293.0	0.00	**0.00**	5.23
I012a	3500	10428	28743567.5	0.00	**0.00**	0.79	45757352.5	**0.00**	0.65	1.04	60100451.5	**0.00**	0.92	0.89
I014a	3577	10622	24977019.5	**0.00**	**0.00**	0.45	39299654.0	**0.00**	0.40	0.84	55503382.5	**0.00**	0.94	0.77
I021a	5195	15722	23164442.5	**0.00**	0.42	1.05	36818149.5	**0.00**	1.05	1.02	51903560.5	**0.00**	1.07	0.73
I039a	3719	11066	23746901.5	0.00	**0.00**	0.38	42890149.5	**0.00**	0.42	0.40	61563684.0	**0.00**	0.57	0.42
I043a	4511	13480	28329402.0	0.01	**0.00**	0.43	48107646.5	**0.00**	0.44	0.37	67962103.5	**0.00**	0.54	0.40
I048a	4920	14712	16451422.5	0.09	**0.00**	0.84	32073780.5	**0.00**	0.41	0.72	41657940.5	**0.00**	0.80	0.72
I052a	160	474	2742172.0	**0.00**	**0.00**	0.01	4595363.0	**0.00**	**0.00**	**0.00**	6115941.0	**0.00**	**0.00**	0.05
I053a	693	2046	10345270.0	**0.00**	**0.00**	0.14	14261180.5	**0.00**	**0.00**	0.28	20833823.0	**0.00**	**0.00**	0.27
I054a	540	1634	5297628.5	**0.00**	**0.00**	0.08	9908443.0	**0.00**	**0.00**	0.12	13717847.0	**0.00**	**0.00**	0.10
I055a	4701	13958	17845784.0	0.04	**0.00**	0.84	25207862.0	**0.00**	0.59	0.97	35610520.5	**0.00**	0.68	0.73
I056a	290	878	4689298.5	**0.00**	**0.00**	0.09	10104636.5	**0.00**	**0.00**	0.09	11852180.0	**0.00**	**0.00**	0.03
I059a	2800	8314	18704918.0	**0.00**	**0.00**	0.73	27415545.5	**0.00**	**0.00**	0.68	37887414.0	**0.00**	0.65	0.71
I065a	1185	3512	5944593.0	**0.00**	**0.00**	0.55	11228197.5	**0.00**	**0.00**	0.88	13267573.0	**0.00**	0.20	0.91
I066a	4551	13642	41687203.0	**0.00**	0.62	0.64	61969175.0	**0.00**	0.74	0.88	84193036.5	**0.00**	1.21	1.11
I069a	3508	10312	23233334.5	0.06	**0.00**	0.53	32910704.0	**0.00**	0.51	0.67	49106774.0	**0.00**	0.58	0.55
I074a	4441	13124	52389470.0	0.00	**0.00**	0.84	79616273.5	**0.00**	0.42	0.98	108915470.5	**0.00**	0.63	0.87
I076a	4909	14536	41906321.5	**0.00**	0.41	1.67	69129431.5	**0.00**	0.87	0.76	93747265.5	**0.00**	1.02	0.53
I085a	2780	8246	20782753.0	**0.00**	**0.00**	0.45	28672185.5	**0.00**	0.39	0.76	41758859.0	**0.00**	0.56	0.72

Table 8: Comparsion of upper bounds for datasets `GEO` and `VIENNA`.

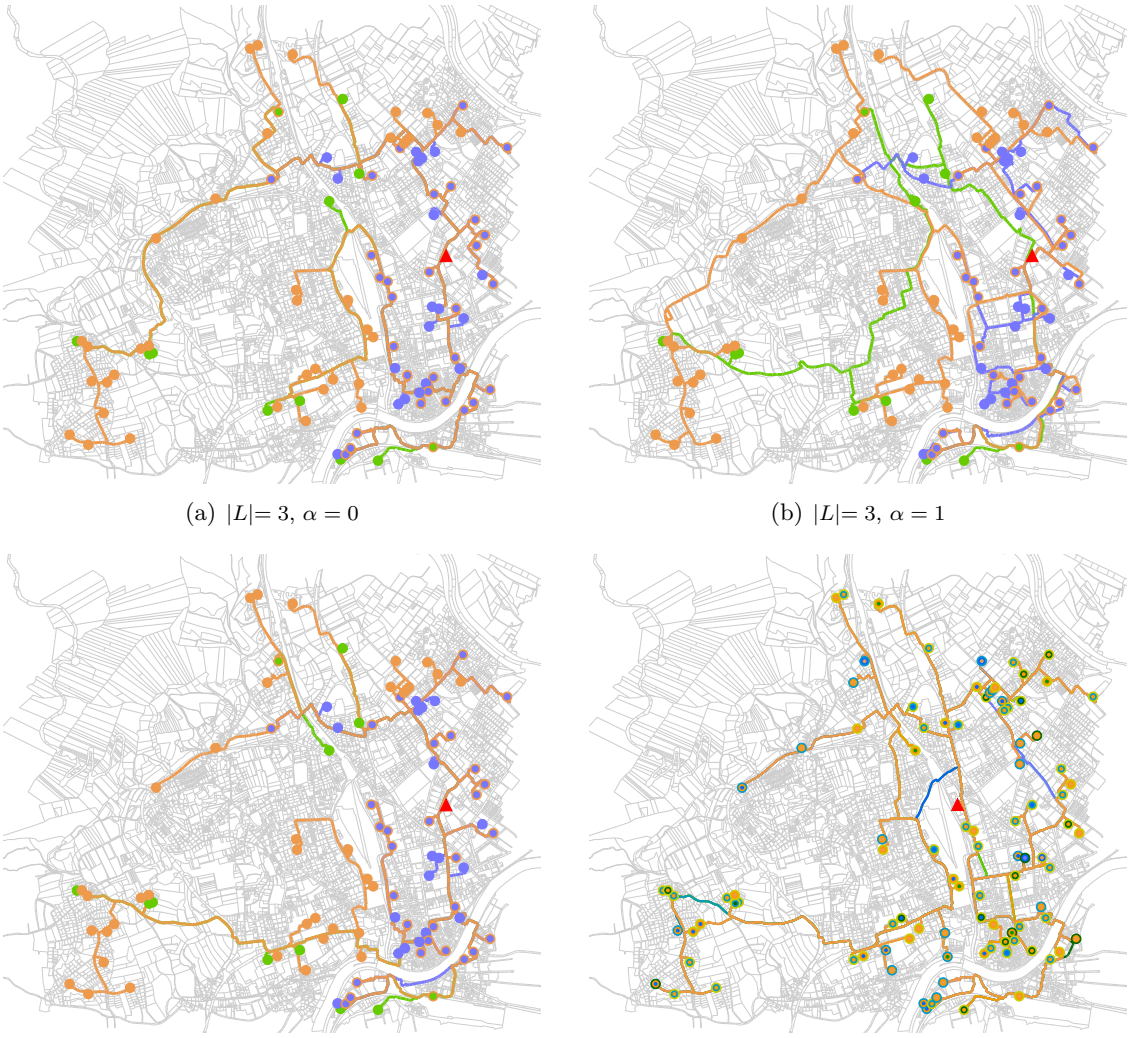
Analyzing GEO Solutions Figure 4 compares the structure of feasible solutions computed on instance GEO/G101 after a one-hour time limit. In Plots 4(a)-4(c), one has $|L|= 3$ and $\alpha = \{0, 0.5, 1\}$. Note that for $\alpha = 0$, an optimal solution to SStA corresponds to an optimal Steiner tree on the union of all terminal sets T_l, and for $\alpha = 1$, an optimal solution is the union of optimal Steiner arborescence for each T_l. In Plot 4(a), where $\alpha = 0$, the computed solution, although not optimal, already takes the form of a tree. In Plot 4(b), where $\alpha = 1$, the computed SStA solution contains numerous cycles due to the overlapping Steiner trees for each label. In Plot 4(c), where $\alpha = 0.5$, the computed solution combines features present in the solutions shown in Plots 4(a) and 4(b), as now both parts of the objective function are of equal importance. Aside from a single cycle, the solution is a tree. Cycles will only appear when their inclusion brings an advantage, which for $|L|= 3$ does not seem to be the case very often. In Plot 4(d), a different root node is chosen, $\alpha = 0.5$ and $|L|= 9$. In comparison with Plot 4(c), most parts of the solution remain unchanged, especially in areas of the graph where only relatively few terminals exist. However, due to the inclusion of additional labels several "shortcuts" have been included into the solution. The inclusion of these paths form alternative connections between subtrees, such that for some labels a cheaper connection can be realized.

7. Conclusions and Future Work

In this work the MCSN problem has been introduced, and a special case where the subproblems are Steiner arborescences, the SStA, has been computationally explored. A cut-based MIP model and a Benders decomposition thereof are introduced. An algorithmic framework with a specialized stabilization method based on in-out separation has been implemented, together with problem-specific preprocessing and a primal heuristic.

Results show that the algorithmic approach based on Benders decomposition outperforms the proposed cut-model for large-scale instances, i.e., when the graphs or the number of labels are large. For small-scale instances, the cut-formulation is slightly more successful at solving instances to optimality. Both variants significantly outperform the previous MIP formulation initially proposed by Mazza et al. [see 26].

As future work, it would be interesting to establish a connection between the MCSN and other applications, such as the design of bus rapid transit networks within cities [see, e.g. 19]. In this more complex setting, the network operator would aim at finding a minimum cost network of exclusive lanes (shared network), on which a set of bus routes (entities) will provide transportation service (which shall be also economically efficient).



(a) $|L|= 3$, $\alpha = 0$

(b) $|L|= 3$, $\alpha = 1$

(c) $|L|= 3$, $\alpha = 0.5$

(d) $|L|= 9$, $\alpha = 0.5$

Figure 4: Plots of GEO/G101. Each label is marked in a different color, i.e., its terminals and the arcs connecting them to the root (marked red) in the best feasible solution computed after a one-hour time limit. Multi-colored nodes represent terminals that belong to multiple labels. Gray lines represent arcs that do not belong to the solution. Figure (a)-(c) show the best feasible solutions for L=3 and $\alpha = \{0, 0.5, 1\}$. Figure (d) shows the best feasible solution for L=9 and $\alpha = 0.5$. For each solution, the relative gap to the computed lower bound is within 3-10%.

Acknowledgment The authors want to thank to M. Fischetti for useful inputs concerning the stabilization of the Benders approach. E. Álvarez-Miranda acknowledges the support of the Chilean Council of Scientific and Technological Research, CONICYT, through the

grant FONDECYT N.11140060 and through the Complex Engineering Systems Institute (ICM:P-05-004-F, CONICYT:FBO16). The research of M. Sinnl was supported by the Austrian Research Fund (FWF, Project P 26755-N19). M. Luipersbeck acknowledges the support of the University of Vienna through the uni:docs fellowship programme.

[1] 11th DIMACS Implementation Challenge. Steiner tree problems, 2014. URL `http://dimacs11.cs.princeton.edu/home.html`.

[2] E. Álvarez-Miranda, I. Ljubić, and E. Fernández. The recoverable robust facility location problem. *Transportation Research Part B*, 79:93–120, 2015.

[3] C. Backes, A. Rurainski, G. Klau, O. Müller, D. Stöckel, A. Gerasch, J. Küntzer, D. Maisel, N. Ludwig, M. Hein, A. Keller, H. Burtscher, M. Kaufmann, E. Meese, and H. Lenhof. An integer linear programming approach for finding deregulated subgraphs in regulatory networks. *Nucleic Acids Research*, 1:1–13, 2011.

[4] R. Battiti, R. Lo Cigno, F. Orava, and B. Pehrson. Global growth of Open Access networks: From warchalking and connection sharing to sustainable business. In *Proceedings of the 1st ACM International Workshop on Wireless Mobile Applications and Services on WLAN Hotspots*, WMASH, pages 19–28. 2003.

[5] W. Ben-Ameur and J. Neto. Acceleration of cutting-plane and column generation algorithms: Applications to network design. *Networks*, 49(1):3–17, 2007.

[6] I. Bomze, M. Chimani, M. Jünger, I. Ljubić, P. Mutzel, and B. Zey. Solving two-stage stochastic Steiner tree problems by two-stage branch-and-cut. In O. Cheong, K. Chwa, and K. Park, editors, *Proceedings of ISAAC 2010*, volume 6506 of *LNCS*, pages 427–439. Springer, 2010.

[7] B. Cherkassky and A. Goldberg. On implementing push-relabel method for the maximum flow problem. In E. Balas and J. Clausen, editors, *Proceedings of IPCO IV*, volume 920 of *LNCS*, pages 157–171. Springer, 1995.

[8] M. Chimani, C. Gutwenger, M. Jünger, G. Klau, K. Klein, and P. Mutzel. The open graph drawing framework (OGDF). *Handbook of Graph Drawing and Visualization*, pages 543–569, 2011.

[9] A. Costa. A survey on benders decomposition applied to fixed-charge network design problems. *Computers & operations research*, 32(6):1429–1450, 2005.

[10] M. de Aragão and R. Werneck. On the implementation of MST-based heuristics for the Steiner problem in graphs. In D. Mount and C Stein, editors, *Proceedings of ALENEX 2002*, volume 2409 of *LNCS*, pages 1–15. Springer, 2002.

[11] M. de Aragão, E. Uchoa, and R. Werneck. Dual heuristics on the exact solution of large Steiner problems. *Electronic Notes in Discrete Mathematics*, 7:150–153, 2001.

[12] M. Dittrich, G. Klau, A. Rosenwald, T. Dandekar, and T. Muller. Identifying functional modules in protein-protein interaction networks: an integrated exact approach. *Bioinformatics*, 24(13):i223–i231, 2008.

[13] C. Duin. *Steiner's problem in graphs*. PhD thesis, University of Amsterdam, 1993.

[14] C. Duin and A. Volgenant. Reduction tests for the Steiner problem in graphs. *Networks*, 19(5):549–567, 1989.

[15] M. Fischetti and D. Salvagnin. An in-out approach to disjunctive optimization. In A. Lodi, M. Milano, and P. Toth, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, volume 6140 of *Lecture Notes in Computer Science*, pages 136–140. Springer Berlin Heidelberg, 2010.

[16] M. Fischetti, M. Leitner, I. Ljubić, M. Luipersbeck, M. Monaci, M. Resch, D. Salvagnin, and M. Sinnl. Thinning out Steiner trees: A node-based model for uniform edge costs. *Workshop of the 11th DIMACS Implementation Challenge*, 2014.

[17] M. Fischetti, I. Ljubić, and M. Sinnl. Redesigning Benders Decomposition for Large Scale Facility Location. *Management Science*, 2016. to appear.

[18] M. Forzati, C. Larsen, and C. Mattsson. Open access networks, the swedish experience. In *Proceedings of the 12th IEEE International Conference on Transparent Optical Networks*, pages 1–4, 2010.

[19] K. Kepaptsoglou and M. Karlaftis. Transit route network design problem: Review. *Journal of Transportation Engineering*, 135(8):491–505, 2009.

[20] T. Koch and A. Martin. Solving Steiner tree problems in graphs to optimality. *Networks*, 32(3):207–232, 1998.

[21] G. Laporte and F. Louveaux. The integer l-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters*, 13(3):133–142, 1993.

[22] M. Leitner, I. Ljubić, M. Luipersbeck, M. Prossegger, and M. Resch. New real-world instances for the Steiner tree problem in graphs. Technical report, Technical report, ISOR, University of Vienna, 2014.

[23] 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 Progamming*, Series B(105):427–449, 2006.

[24] I. Ljubić, P. Mutzel, and B. Zey. Stochastic survivable network design problems. *Electronic Notes in Discrete Mathematics*, 41:245—-252, 2013.

[25] Thomas L Magnanti, Paul Mireault, and Richard T Wong. *Tailoring Benders decomposition for uncapacitated network design*. Springer, 1986.

[26] A. Mazza, I. Gat-Viks, H. Farhan, and R. Sharan. A minimum-labeling approach for reconstructing protein networks across multiple conditions. *Algorithms for Molecular Biology*, 9(1):1–8, 2014.

[27] T. Polzin and S. Daneshmand. Improved algorithms for the Steiner problem in networks. *Discrete Applied Mathematics*, 112(1):263–300, 2001.

[28] A. Quilliot. Network Design Problems: Fundamental Methods. In V. Paschos, editor, *Applications of Combinatorial Optimization*, chapter 9, pages 253–289. John Wiley & Sons, 2013.

[29] H. Takahashi and A. Matsuyama. An approximate solution for the Steiner problem in graphs. *Mathematica Japonica*, 24(6):573–577, 1980.

[30] Richard T. Wong. A dual ascent approach for Steiner tree problems on a directed graph. *Mathematical Programming*, 28(3):271–287, 1984. ISSN 0025-5610. doi: 10.1007/BF02612335.

[31] T. Yamamoto, H. Bannai, M. Nagasaki, and S. Miyano. Better decomposition heuristics for the maximum-weight connected graph problem using betweenness centrality. In J. Gama, V. Costa, A. Jorge, and P. Brazdil, editors, *Discovery Science*, volume 5808 of *LNCS*, pages 465–472. Springer, 2009.

[32] N. Yosef, E. Zalckvar, A. Rubinstein, M. Homilius, N. Atias, L. Vardi, I. Berman, H. Zur, A. Kimchi, E. Ruppin, and R. Sharan. ANAT: A tool for constructing and analyzing functional protein networks. *Science Signaling*, 4(196):pl1–pl1, 2011.