The Constrained Virtual Steiner Arborescence Problem:
Formal Definition, Single-Commodity Integer Programming Formulation and Computational Evaluation

Matthias Rost, Stefan Schmid
{mrost,stefan}@net.t-labs.tu-berlin.de

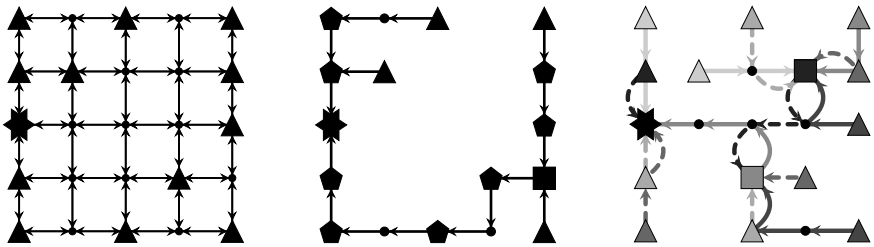
Telekom Innovation Laboratories (T-Labs) & TU Berlin, Germany

Abstract. As the Internet becomes more virtualized and software-defined, new functionality is introduced in the network core: the distributed resources available in ISP central offices, universal nodes, or datacenter middleboxes can be used to process (e.g., filter, aggregate or duplicate) data. Based on this new networking paradigm, we formulate the Constrained Virtual Steiner Arborescence Problem (CVSAP) which asks for optimal locations to perform in-network processing, in order to jointly minimize processing costs and network traffic while respecting link and node capacities.
We prove that CVSAP cannot be approximated (unless $NP = P$), and accordingly, develop the exact algorithm VirtuCast to compute optimal solutions to CVSAP. VirtuCast consists of: (1) a compact single-commodity flow Integer Programming (IP) formulation; (2) a flow decomposition algorithm to reconstruct individual routes from the IP solution. The compactness of the IP formulation allows for computing lower bounds even on large instances quickly, speeding up the algorithm. We rigorously prove VirtuCast's correctness.
To complement our theoretical findings, we have implemented VirtuCast and present an extensive computational evaluation, showing that VirtuCast can solve realistically sized instances (close to) optimality. We show that VirtuCast significantly improves upon naive multi-commodity formulations and also initiate the study of primal heuristics to generate feasible solutions during the branch-and-bound process.

1 Introduction

Multicast and aggregation are two fundamental functionalities offered by many communication networks. In order to efficiently distribute content (e.g., live TV) to multiple receivers, a multicast solution should duplicate the content as close to the receivers as possible. Analogously, in aggregation applications such as distributed network monitoring, the monitoring data may be filtered or aggregated along the path to the observer, to avoid redundant transmissions over physical links. Efficient multicasting and aggregation is a mature research field, and many important theoretical and practical results have been obtained over the last decades. Applications range from IPTV [17] over sensor networks [12,13] to fiber-optical transport [17].

This paper is motivated by the virtualization trend in today's Internet, and in particular by network (function) virtualization [11] and software-defined networking, e.g.,



(a) 5×5 Grid Topology (b) Steiner Arborescence (c) Virtual Arborescence

Fig. 1: An aggregation example on a 5×5 grid. Terminals are depicted as triangles while the receiver is depicted as star. The terminals must establish a path towards the receiver, while multiple data streams may be aggregated by activated processing locations. Such processing locations are pictured as squares or, in case that a processing location is collocated with a terminal, pentagons. In Figure (c), equally colored and dashed edges represent paths, originating at the node with the same color.

OpenFlow [29]. In virtualized environments, resources can be allocated or leased flexibly at the locations where they are most useful or cost-effective: Computational and storage resources available at middleboxes in datacenters [6] or in distributed (micro-)datacenters in the wide-area network [9,25,38] can for example be used for in-network processing, e.g., to reduce traffic during the MapReduce shuffle phase [7].

Such distributed resource networks open new opportunities on how services can be deployed. In the context of aggregation and multicasting, for example, a new degree of freedom arises: the sites (i.e., the number and locations) used for the data processing, becomes subject to optimization.

This paper initiates the study of how to efficiently allocate in-network processing functionality in order to jointly minimize network traffic and computational resources. Importantly, for many of these problem variants, classic Steiner Tree models [36] are no longer applicable [24]. Accordingly, we coin our problem the *Constrained Virtual Steiner Arborescence Problem (CVSAP)*, as the goal is to install a set of processing nodes and to connect all terminals via them to a single root.

Example. To illustrate our model, consider the aggregation example depicted in Figure 1. The terminals must connect to the receiver, while processing functionality can be placed on nodes to aggregate any number of incoming data flows into a single one. Assuming no costs for placing processing functionality, the problem reduces to the Steiner Arborescence Problem and the optimal solution, depicted in Figure (1b), uses 16 edges and 9 processing locations. However, assuming unit edge costs and activation costs of 5 for processing locations, this solution is suboptimal. Figure (1c) depicts a solution which only uses 2 processing locations and 26 edges: Terminals in the first column directly connect to the receiver, while the remaining terminals use one of the two processing nodes. Note that we allow for nested processing of flows: the upper processing node forwards its aggregation result to the lower processing node, from where the result is then forwarded to the receiver.

Contribution. This paper presents the first concise graph-theoretic formulation of the *Constrained Virtual Steiner Arborescence Problem (CVSAP)* which captures the trade-off between traffic optimization benefits and in-network processing costs arising in virtualized environments, and which also generalizes many classic in-network processing problems related to multicasting and aggregation. We prove that CVSAP cannot be approximated unless $NP \subseteq P$ and therefore focus on obtaining provably good solutions for CVSAP in non-polynomial time. To this end we introduce the algorithm VirtuCast, which is based on Integer Programming (IP) and allows to obtain optimal solutions. The advantage of VirtuCast lies in the fact that even for large problem instances, when optimal solutions cannot be computed in reasonable time, our approach bounds the gap to optimality as lower bounds are computed on the fly.

VirtuCast consists of two components: a single-commodity IP formulation which can be solved by branch-and-cut methods and a decomposition algorithm to construct the routing scheme. Our IP formulation not only uses a smaller number of variables compared to alternative multi-commodity IP formulations, but also yields good linear relaxations in practice.

Our main contribution is the constructive proof that any solution to our IP formulation can be decomposed to yield a valid routing scheme connecting all terminals via processing nodes to the root. This is intriguing, as the single-commodity flow in the network is not restricted to directed acyclic graphs (cf. Figure 1c). In fact, as already shown in [24], forbidding directed acyclic graphs (DAGs) may yield suboptimal solutions. Rather, we allow for the iterative processing of flows, such that processing nodes may be connected to other processing nodes.

To complement our theoretical findings, we present an extensive computational study. As our results indicate, using VirtuCast realistically sized instances can be solved (close to) optimality. Adapting well-known separation techniques from Steiner Tree literature, we improve the runtime of VirtuCast by the order of a magnitude. To compute feasible solutions to CVSAP in the course of the branch-and-bound process, we developed a new primal heuristic and investigate its performance. To demonstrate the advantages of our single-commodity formulation over multi-commodity formulations, we have implemented a simple multi-comodity formulation and show its computational inferiority.

Overview. In Section 2 we formally introduce CVSAP and its aggregation and multicast versions and show its general inapproximability. We continue by present the algorithm VirtuCast, that relies on a single-commodity IP formulation, in Section 3. For a later computational comparison, we introduce a multi-commodity Mixed Integer Program for CVSAP in Section 4. We present our implementation of VirtuCast in Section 4 and importantly introduce a primal heuristic to generate feasible solutions in the course of the branch-and-bound process. In Section 6 we present our extensive computational evaluation, containing a validation of our implementation choices, an analysis of VirtuCast's performance on realistically sized instances, an analysis of our primal heuristic's performance as well as the computational comparison with the multi-commodity flow formulation introduced earlier. We conclude this paper with summarizing related work in Section 7.

2 The Constrained Virtual Steiner Arborescence Problem

The Constrained Virtual Steiner Arborescence Problem (CVSAP) considers multicast routing and optimal in-network aggregation problems in which processing locations can be *chosen* to reduce traffic. As using (or leasing) in-network processing capabilities comes at a certain cost (e.g., the corresponding resources cannot be used by other applications), there is a trade-off between additional processing and traffic reduction. In contrast to the classic Steiner Tree Problems [36], our model distinguishes between nodes that merely forward traffic and nodes that may actively *process flows*. Informally, the task is to construct a minimal cost spanning arborescence on the set of active processing nodes, sender(s) and receiver(s), such that edges in the arborescence correspond to paths in the original graph. As edges in the arborescence represent logical links (i.e., routes) between nodes, we refer to the problem as *Virtual* Steiner Arborescence Problem. Based on the notion of virtual edges, the underlying paths may overlap and may use both the (resource-constrained) nodes and edges in the original graph multiple times (cf. Figure 1c). We naturally adopt the notion of *Steiner nodes* in our model, and refer to processing nodes contained in the virtual arborescence as *active Steiner nodes*. The following notations will be used throughout this paper.

Notation. In a directed graph $G = (V_G, E_G)$ we denote by $\mathcal{P}_G$ the set of all simple, directed paths in G. Given a set of simple paths $\mathcal{P}$, we denote by $\mathcal{P}[e]$ the subset of paths contained in $\mathcal{P}$ that contain edge e. We use the notation $P = \langle v_1, v_2, \ldots, v_n \rangle$ to denote the directed path P of length $|P| = n$ where $P_i \triangleq v_i \in V_G$ for $1 \leq i \leq n$ and $(v_i, v_{i+1}) \in E_G$ for $1 \leq i < n$. We use $\delta_F^+(v) = \{(v,u) \in F\}$ and $\delta_F^-(v) = \{(u,v) \in F\}$ to denote the set of outgoing and incoming edges of node $v \in V$ restricted on a subset $F \subseteq E$ in G and set $\delta_F^+(W) = \{(v,u) \in F | v \in W, u \notin W\}$ and respectively $\delta_F^-(W) = \{(u,v) \in F | v \in W, u \notin W\}$. We abridge $\mathsf{f}((\mathsf{y},\mathsf{z}))$ to $\mathsf{f}(\mathsf{y},\mathsf{z})$ for functions defined on tuples.

Formal Problem Statement. Our general problem definition presented henceforth captures both the multicast and the aggregation scenarios. We model the physical infrastructure as capacitated, directed network $G = (V_G, E_G, c_E, u_E)$ with integral capacities on the edges $u_E : E_G \to \mathbb{N}$ and real-valued, positive edge costs $c_E : E_G \to \mathbb{R}^+$. On top of this network, we define an abstract request $R_G = (r, S, T, u_r, c_S, u_S)$, where $T \subseteq V_G$ defines the set of terminals that need to be connected with the root $r \in V_G \setminus T$, for which an integral capacity $u_r \in \mathbb{N}$ is given. The set $S \subseteq V_G \setminus (\{r\} \cup T)$ denotes the set of possible *Steiner sites*, i.e. nodes at which Steiner nodes may be *activated*. Steiner sites are attributed with a positive cost that is incurred upon using it as active Steiner node $c_S : S \to \mathbb{R}^+$, and an integral capacity $u_S : S \to \mathbb{N}$. It should be noted that we require the sets S and T to be disjoint for terminological reasons. A node $v \in S \cup T$ can easily be modeled by introducing a new node $v_T \in T$ and letting $v \in S$ such that v_T is only connected to v with $c_E(v, v_T) = 0$ and $u_E(v, v_T) = 1$.

In the aggregation scenario the terminals represent nodes holding data that needs to be forwarded to the root (the single receiver) while data may be aggregated at active Steiner nodes. Contrary, in the multicast scenario the root represents the single sender

that must stream (the same) data to each of the terminals, while active Steiner nodes may duplicate and reroute the stream. The capacities on the root and on the Steiner sites limit the degree in the Virtual Arborescence, formally introduced next.

Definition 1 (Virtual Arborescence). *Given a directed graph $G = (V_G, E_G)$ and a root $r \in V_G$, a* Virtual Arborescence (VA) *on G is defined as $\mathcal{T}_G = (V_\mathcal{T}, E_\mathcal{T}, r, \pi)$ where $\{r\} \subseteq V_\mathcal{T} \subseteq V_G$, $E_\mathcal{T} \subseteq V_\mathcal{T} \times V_\mathcal{T}$, r is the root and $\pi : E_\mathcal{T} \to \mathcal{P}_G$ maps each edge in the arborescence on a simple directed path $P \in \mathcal{P}_G$ such that*

(VA-1) *for all $(u, v) \in E_\mathcal{T}$ the directed path $\pi(u, v)$ connects u to v in G and*
(VA-2) *all edges in $E_\mathcal{T}$ are either directed away or towards r.* □

A link $(v, w) \in E_\mathcal{T}$ represents a logical connection between nodes v and w while the function $\pi(v, w) = P$ defines the route taken to establish this link. Note that the directed path P must, *pursuant* to the orientation (v, w) of the logical link in the arborescence, start with v and end at w. Figure 1c illustrates our definition of the VA: equally colored and dashed paths represent edges of the Virtual Arborescence. Using the concept of Virtual Arborescence, we can concisely state the problem we are attending to.

Definition 2 (Constrained Virtual Steiner Arborescence Problem). *Given a directed capacitated network $G = (V_G, E_G, c_E, u_E)$ and a request $R_G = (r, S, T, u_r, c_S, u_S)$ as above, the* Constrained Virtual Steiner Arborescence Problem (CVSAP) *asks for a minimal cost Virtual Arborescence $\mathcal{T}_G = (V_\mathcal{T}, E_\mathcal{T}, r, \pi)$ satisfying the following conditions:*

(CVSAP-1) *$\{r\} \cup T \subseteq V_\mathcal{T}$ and $V_\mathcal{T} \subseteq \{r\} \cup S \cup T$,*
(CVSAP-2) *for all $t \in T$ holds $\delta^+_{E_\mathcal{T}}(t) + \delta^-_{E_\mathcal{T}}(t) = 1$,*
(CVSAP-3) *for the root $\delta^+_{E_\mathcal{T}}(r) + \delta^-_{E_\mathcal{T}}(r) \leq u_r$ holds,*
(CVSAP-4) *for all $s \in S \cap V_\mathcal{T}$ holds $\delta^-_{E_\mathcal{T}}(s) + \delta^+_{E_\mathcal{T}}(s) \leq u_S(s) + 1$ and*
(CVSAP-5) *for all $e \in E_G$ holds $|\,(\pi(E_\mathcal{T}))\,[e]| \leq u_E(e)$.*

Any VA $\mathcal{T}_G$ satisfying CVSAP-1 - CVSAP-5 is said to be a feasible solution. The cost of a Virtual Arborescence is defined to be

$$C_{\text{CVSAP}}(\mathcal{T}_G) = \sum_{e \in E_G} c_E(e) \cdot |\,(\pi(E_\mathcal{T}))\,[e]| + \sum_{s \in S \cap V_\mathcal{T}} c_S(s)\,.$$

where $|\,(\pi(E_\mathcal{T}))\,[e]|$ is the number of times an edge is used in different paths □

In the above definition, CVSAP-1 states that terminals and the root must be included in $V_\mathcal{T}$, whereas non Steiner sites are excluded. We identify $V_\mathcal{T} \setminus (\{r\} \cup T)$ with the set of active Steiner nodes. Condition CVSAP-2 states that terminals must be leaves in $\mathcal{T}_G$ and CVSAP-3 and CVSAP-4 enforce degree constraints in $\mathcal{T}_G$. The term $\pi(E_\mathcal{T})$ in Condition CVSAP-5 determines the set of all used paths and consequently $\pi(E_\mathcal{T})[e]$ yields the set of paths that use $e \in E_\mathcal{T}$. As π is injective and maps on simple paths, Condition CVSAP-5 enforces that edge capacities are not violated.

Definition 3 (Multicast / Aggregation CVSAP). *The CVSAP constitutes the* Constrained Virtual Steiner Aggregation Problem (A-CVSAP) *in case that the edges of* $E_{\mathcal{T}}$ *are oriented towards* r*, or conversely constitutes the* Constrained Virtual Steiner Multicast Problem (M-CVSAP) *if edges are oriented away from* r*. We denote the set of feasible solutions by* $\mathcal{F}_{\text{A-CVSAP}}$ *and* $\mathcal{F}_{\text{M-CVSAP}}$ *respectively.*

It is important to note that Definitions 1 and 3 together imply that in A-CVSAP each terminal is connected to the root and conversely that in M-CVSAP the root is connected to all terminals. Furthermore, the following remark establishes that the degree constraints (see CVSAP-3 and CVSAP-4) effectively constrain the number of incoming connections (A-CVSAP) and respectively constrains the number of outgoing connections (M-CVSAP).

Remark 1. Note that in case of M-CVSAP the Condition CVSAP-2 amd CVSAP-3 reduce to $\forall s \in S \cap V_{\mathcal{T}}.\delta^{+}_{E_{\mathcal{T}}}(s) \leq u_S(s)$ and $\delta^{+}_{E_{\mathcal{T}}}(r) \leq u_r$ respectively. Analogously, Conditions CVSAP-2 and CVSAP-3 simplify to $\forall s \in S \cap V_{\mathcal{T}}.\delta^{-}_{E_{\mathcal{T}}}(s) \leq u_S(s)$ and $\delta^{-}_{E_{\mathcal{T}}}(r) \leq u_r$ in A-CVSAP.

Obviously, as the single difference between A-CVSAP and M-CVSAP is the orientation of the edges, the problems can be reduced on each other.

Remark 2 (Equivalence of A-CVSAP and M-CVSAP). The problems A-CVSAP and M-CVSAP can be reduced on each other by inverting the orientation of the edges in the underlying network G and adapting c_E and u_E accordingly.

Based on the above fact, it suffices to give an algorithm for either one of the two problems. As our IP approach presented in Section 3 can be more intuitively understood in the aggregation scenario, we restrict our further discussion to the case of A-CVSAP. Lastly, we give the following remark relating the directed CVSAP to its undirected counterpart, namely the Constrained Virtual Steiner Tree Problem.

Remark 3 (Constrained Virtual Steiner Tree Problem). Analogously to the definition of VA the concept of a (rooted) Virtual Tree can be introduced, in which (undirected) virtual edges are mapped on undirected simple paths (see VA-1), and the orientation constraint (see VA-2) is dropped. Substituting $\delta^{+}_{E_{\mathcal{T}}}(\cdot) + \delta^{-}_{E_{\mathcal{T}}}(\cdot)$ by its undirected counterpart $\delta_{E_{\mathcal{T}}}(\cdot)$ in Definition 2 of CVSAP then directly translates to the definition of the Constrained Virtual Steiner Tree Problem (CVSTP).

The following theorem motivates our approach in Section 3, namely to search for provably good solutions in non-polynomial time.

Theorem 1. *Checking whether a feasible solution for CVSAP exists is NP-complete. Thus, unless* $NP \subseteq P$ *holds, there cannot exist an (approximation) algorithm yielding a feasible solution in polynomial time.*

Proof. We give a reduction on the decision variant of set cover. Let U denote the universe of elements and let $\mathcal{S} \subseteq 2^U$ denote a family of sets covering U. To check whether a set cover using at most k many sets exists, we construct the following CVSAP instance. We introduce a terminal t_u for each element $u \in U$ and a Steiner site s_S for

each $S \in \mathcal{S}$. A terminal t_u is connected by a directed link to each Steiner site s_S iff. $u \in S$. Each Steiner site s_S is connected to the root r. We set the capacity of the root to k and capacities of Steiner sites to $|U|$. It is easy to check that there exists a feasible solution to this CVSAP instance iff. there exists a set cover of less than k elements. □

3 Compact Integer Programming Formulation for CVSAP

We describe a compact, single-commodity flow Integer Programming (IP) formulation to solve (A-)CVSAP. Our IP (see IP-A-CVSAP) is based on an *extended graph* containing a single super source o^+ and two distinct super sinks o_S^- and o_r^- (see Definition 4). While o_r^- may only receive flow from the root r, all possible Steiner sites $s \in S$ connect to o_S^-. Distinguishing between these two super sinks is necessary, as we will require activated Steiner nodes to not *absorb all* incoming flow, but forward at least one unit of flow towards o_r^-, which will indeed ensure connectivity.

Definition 4 (Extended Graph). *Given a directed network* $G = (V_G, E_G, c_E, u_E)$ *and a request* $R_G = (r, S, T, u_r, c_S, u_S)$ *as introduced in Section 2, we define the* extended graph $G_{\text{ext}} = (V_{\text{ext}}, E_{\text{ext}})$ *as follows*

(EXT-1) $V_{\text{ext}} \triangleq V_G \cup \{\mathrm{o}^+, \mathrm{o}_S^-, \mathrm{o}_r^-\}$,
(EXT-2) $E_{\text{ext}} \triangleq E_G \cup \{(r, \mathrm{o}_r^-)\} \cup E_{\text{ext}}^{S^-} \cup E_{\text{ext}}^{S^+} \cup E_{\text{ext}}^{T^+}$,

where $E_{\text{ext}}^{S^-} \triangleq S \times \{\mathrm{o}_S^-\}$, $E_{\text{ext}}^{S^+} \triangleq \{\mathrm{o}^+\} \times S$ *and* $E_{\text{ext}}^{T^+} \triangleq \{\mathrm{o}^+\} \times T$. *We define* $E_{\text{ext}}^{R} \triangleq E_{\text{ext}} \setminus E_{\text{ext}}^{S^-}$. □

Further Notation. To clearly distinguish between variables and constants, we typeset constants in bold font: instead of referring to c_E, c_S and u_E, u_r, u_S we use $\mathbf{c}_{\mathsf{y}}$ and $\mathbf{u}_{\mathsf{y}}$, where y may either refer to an edge or a Steiner site. Similarly, we use $\mathbf{u}_{\mathsf{y}}$ where y may either refer to an edge, the root or Steiner node. We abbreviate $\sum_{\mathsf{y} \in \mathsf{Y}} f_{\mathsf{y}}$ by $f(\mathsf{Y})$. We use $\mathsf{Y} + \mathsf{y}$ to denote $\mathsf{Y} \cup \{\mathsf{y}\}$ and $\mathsf{Y} - \mathsf{y}$ to denote $\mathsf{Y} \setminus \{\mathsf{y}\}$ for a set Y and a singleton y. For $f \in \mathbb{Z}_{\geq 0}^{E_{\text{ext}}}$ we define the flow-carrying subgraph $G_{\text{ext}}^f \triangleq (V_{\text{ext}}^f, V_{\text{ext}}^f)$ with $V_{\text{ext}}^f \triangleq V_{\text{ext}}$ and $V_{\text{ext}}^f \triangleq \{e | e \in E_{\text{ext}} \wedge f(e) \geq 1\}$.

3.1 The IP Model

Our IP formulation (see IP-A-CVSAP) uses an integral single-commodity flow and we define a flow variable $f_e \in \mathbb{Z}_{\geq 0}$ for each edge $e \in E_{\text{ext}}$ in the extended graph (see IP-11). As we use an aggregated flow formulation, that does not model routing decisions explicitly, we show in Section 3.2 how this single-commodity flow can be decomposed into paths for constructing an actual solution for CVSAP.

Whether a Steiner site $s \in S$ is activated is decided by the binary variable $x_s \in \{0, 1\}$ (see IP-10). Constraint IP-8 forces each terminal $t \in T$ to send a single unit of flow. As flow conservation is enforced on all original nodes $v \in V_G$ (see IP-1), all flow originating at o^+ must be forwarded to one of the super sinks o_r^- or o_S^-, while not violating link capacities (see IP-7).

Integer Program IP-A-CVSAP

$$\text{minimize} \quad C_{\text{IP}}(x,f) = \sum_{e \in E_G} \mathbf{c}_e f_e + \sum_{s \in S} \mathbf{c}_s x_s \qquad \text{(IP-OBJ)}$$

$$\text{subject to} \quad f(\delta^+_{E_{\text{ext}}}(v)) = f(\delta^-_{E_{\text{ext}}}(v)) \qquad \forall\, v \in V_G \qquad \text{(IP-1)}$$

$$f(\delta^+_{E^R_{\text{ext}}}(W)) \geq x_s \qquad \forall\, W \subseteq V_G, s \in W \cap S \neq \emptyset \qquad \text{(IP-2)}$$

$$f(\delta^+_{E^R_{\text{ext}}}(W)) \geq 1 \qquad \forall\, W \subseteq V_G, T \cap W \neq \emptyset \qquad \text{(IP-3}^\star\text{)}$$

$$f_e \geq x_s \qquad \forall\, e = (s, \mathrm{o}^-_S) \in E^{S^-}_{\text{ext}} \qquad \text{(IP-4}^\star\text{)}$$

$$f_e \leq \mathbf{u}_s x_s \qquad \forall\, e = (s, \mathrm{o}^-_S) \in E^{S^-}_{\text{ext}} \qquad \text{(IP-5)}$$

$$f_{(r,\mathrm{o}^-_r)} \leq \mathbf{u}_r \qquad \text{(IP-6)}$$

$$f_e \leq \mathbf{u}_e \qquad \forall\, e \in E_G \qquad \text{(IP-7)}$$

$$f_e = 1 \qquad \forall\, e \in E^{T^+}_{\text{ext}} \qquad \text{(IP-8)}$$

$$f_e = x_s \qquad \forall\, e = (\mathrm{o}^+, s) \in E^{S^+}_{\text{ext}} \qquad \text{(IP-9)}$$

$$x_s \in \{0,1\} \qquad \forall\, s \in S \qquad \text{(IP-10)}$$

$$f_e \in \mathbb{Z}_{\geq 0} \qquad \forall\, e \in E_{\text{ext}} \qquad \text{(IP-11)}$$

As the definition of A-CVSAP requires that each terminal $t \in T$ establishes a path to r, we need to enforce connectivity; otherwise active Steiner nodes would simply absorb flow by directing it towards o^-_S. To prohibit this, we adopt well-known *Connectivity Inequalities* [23] and *Directed Steiner Cuts* [21]. Our Connectivity Inequalities IP-2 state that each set of nodes containing a Steiner site $s \in S$ must emit at least one unit of flow in E^R_{ext}, if s is activated. As E^R_{ext} does not contain edges towards o^-_S, this constraint therefore enforces that there exists a path in G^f_{ext} from each activated Steiner node s to the root r.

Analogously, the Directed Steiner Cuts IP-3* enforce that there exists a path from each terminal $t \in T$ towards r in G^f_{ext}. These directed Steiner cuts constitute valid inequalities which are implied by IP-1 and IP-2 (see Lemma 1). These Directed Steiner Cuts can strengthen the model by improving the LP-relaxation during the branch-and-cut process (see Lemma 5 in Section A for the proof). As they are not needed for proving the correctness and could technically be removed, we mark them with a * (star).

As a Steiner node $s \in S$ is activated iff. $x_s = 1$, Constraint IP-9 requires activated Steiner nodes to receive one unit of flow while being able to maximally absorb $\mathbf{u}_s$ many units of flow by forwarding it to o^-_S (see IP-5). Furthermore, by IP-5 inactive Steiner sites may not absorb flow at all. The Constraint IP-4* requires active Steiner nodes to at least absorb one unit of flow. This is a valid inequality, as activating a Steiner site $s \in S$ incurs non-negative costs. We introduce this constraint here, as it specifies a condition that is used in a proof later on.

Constraint IP-6 defines an upper bound on the amount of flow that the root may receive and the objective function IP-OBJ mirrors the CVSAP cost function (see Definition 2).

We denote with $\mathcal{F}_{\text{IP}} = \{(x, f) \in \{0,1\}^S \times \mathbb{Z}_{\geq 0}^{E_{\text{ext}}} | \text{IP-1 - IP-11}\}$ the set of feasible solutions to IP-A-CVSAP.

3.2 Flow Decomposition

Given a feasible solution $(\hat{x}, \hat{f}) \in \mathcal{F}_{\text{IP}}$ for IP-A-CVSAP, Algorithm DecomposeFlow constructs a feasible solution $\hat{\mathcal{T}}_G \in \mathcal{F}_{\text{A-CVSAP}}$ for CVSAP. Similarly to well-known algorithms for computing flow decompositions for simple s-t flows (see e.g. [2]), our algorithm iteratively deconstructs the flow into paths from the super source o^+ to the super sinks o_S^- or o_r^-, which are successively removed from the network. However, as IP-A-CVSAP does not pose a simple flow problem, we constantly need to ensure that Connectivity Inequalities IP-2 hold after removing flow in $G_{\text{ext}}^{\hat{f}}$. We first present DecomposeFlow in more detail and then prove its correctness.

We first present DecomposeFlow in more detail and then prove its correctness and give the runtime in Section 3.3.

Synopsis of Algorithm. Algorithm DecomposeFlow constructs a feasible VA $\hat{\mathcal{T}}_G$ given a solution $(\hat{x}, \hat{f}) \in \mathcal{F}_{\text{IP}}$. In Line 2 $\hat{\mathcal{T}}_G$ is initialized without any edges but containing all the nodes the final solution will consist of, namely the root r, the terminals T and the activated Steiner nodes $\{s \in S | x_s \geq 1\}$. In Line 3 a terminal node $t \in \hat{T}$ is selected for which a path is constructed to either an active Steiner node or to the root itself (Lines 5-13). In Line 5 a path P connecting t to the root r in the flow network $G_{\text{ext}}^{\hat{f}}$ is chosen (see Lemma 1 for the proof of existence for such a path). Note that by definition of $G_{\text{ext}}^{\hat{f}}$ all edges contained in P carry at least one unit of flow. Within the loop beginning in Line 6, the flow on path P is iteratively decremented (see Line 7) as long as the Connectivity Inequality IP-2 is not violated. In case it is violated, we revert the reduction of flow (see Line 11) and select a path towards the super sink o_S^- starting at the current node P_j (see Line 10). Such a path must exist according to Lemma 3. The path P is accordingly redirected in Line 12 .

The path construction (in Lines 5 to 12) terminates once the flow from the second last node $P_{|P|-1}$ towards the last node $P_{|P|}$ has been reduced. By construction, the path P leads from the super source o^+ via the terminal $t \in \hat{T}$ towards the super sink o_r^- or o_S^-. If P terminates in o_S^- via Steiner node $s = P_{|P|-1} \in \hat{S}$ such that (s, o_S^-) carries no flow anymore, s itself becomes a terminal (see Lines 15 and 16). Otherwise, P terminates in o_r^- and $P_{|P|-1} = r$ holds. Lastly, in Line 18 the (virtual) edge $(t, P_{|P|-1})$ is added to $\hat{E}_{\mathcal{T}}$ and $\hat{\pi}(t, P_{|P|-1})$ is set accordingly to the truncated path P, where head, tail and any cycles are removed (function `simplify`).

3.3 Proof of Correctness

We will now formally prove the correctness of Algorithm DecomposeFlow. We use an inductive argument similar to the one used for proving the existence of flow decomposi-

Algorithm DecomposeFlow

Input : Network $G = (V_G, E_G, c_E, u_E)$, Request $R_G = (r, S, T, u_r, c_S, u_S)$, Solution $(\hat{x}, \hat{f}) \in \mathcal{F}_{\text{IP}}$ to IP-A-CVSAP
Output: Feasible Virtual Arborescence $\hat{\mathcal{T}}_G$ for CVSAP

1 **set** $\hat{S} \triangleq \{s \in S | x_s \geq 1\}$ **and** $\hat{T} \triangleq T$
2 **set** $\hat{\mathcal{T}}_G \triangleq (\hat{V}_{\mathcal{T}}, \hat{E}_{\mathcal{T}}, r, \hat{\pi})$ **where** $\hat{V}_{\mathcal{T}} \triangleq \{r\} \cup \hat{S} \cup \hat{T}$, $\hat{E}_{\mathcal{T}} \triangleq \emptyset$ **and** $\hat{\pi} : \hat{E}_{\mathcal{T}} \to \mathcal{P}_G$
3 **while** $\hat{T} \neq \emptyset$ **do**
4 **let** $t \in \hat{T}$ **and** $\hat{T} \leftarrow \hat{T} - t$
5 **choose** $P \triangleq \langle \mathrm{o}^+, t, \ldots, \mathrm{o}_r^- \rangle \in G_{\text{ext}}^{\hat{f}}$
6 **for** $j = 1$ **to** $|P| - 1$ **do**
7 **set** $\hat{f}(P_j, P_{j+1}) \leftarrow \hat{f}(P_j, P_{j+1}) - 1$
8 **if** Constraint IP-2 is violated *with respect to* $\hat{f}$ and $\hat{S}$ **then**
9 **choose** $W \subseteq V_G$ **such that** $W \cap \hat{S} \neq \emptyset$ **and** $\hat{f}(\delta^+_{E^R_{\text{ext}}}(W)) = 0$
10 **choose** $P' \triangleq \langle P_j, \ldots, \mathrm{o}_S^- \rangle \in G_{\text{ext}}^{\hat{f}}$ **such that** $P_i \in W$ for $1 \leq i < m$
11 **set** $\hat{f}(P_j, P_{j+1}) \leftarrow \hat{f}(P_j, P_{j+1}) + 1$ **and** $\hat{f}(P'_1, P'_2) \leftarrow \hat{f}(P'_1, P'_2) - 1$
12 **set** $P \leftarrow \langle P_1, \ldots, P_{j-1}, P_j = P'_1, P'_2, \ldots, P'_m \rangle$
13 **end**
14 **end**
15 **if** $P_{|P|} = \mathrm{o}_S^-$ **and** $\hat{f}(P_{|P|-1}, P_{|P|}) = 0$ **then**
16 **set** $\hat{S} \leftarrow \hat{S} - P_{|P|-1}$ **and** $\hat{x}(P_{|P|-1}) \leftarrow 0$ **and** $\hat{T} \leftarrow \hat{T} + P_{|P|-1}$
17 **end**
18 **set** $\hat{E}_{\mathcal{T}} \leftarrow \hat{E}_{\mathcal{T}} + (t, P_{|P|-1})$ **and** $\hat{\pi}(t, P_{|P|-1}) \triangleq \texttt{simplify}(\langle P_2, \ldots, P_{|P|-1} \rangle)$
19 **end**

tions (see [2])We assume that all constraints of IP-A-CVSAP hold and show that for any terminal $t \in T$ a path towards the root or to an active Steiner node can be constructed, such that decrementing the flow along the path by one unit does again yield a feasible solution to IP-A-CVSAP, in which t has been removed from the set of terminals (see Theorem 2 below). During this induction step we also show the well-definedness of the **choose** operations.

Theorem 2. *Assuming that the constraints of DecomposeFlow hold with respect to $\hat{S}, \hat{T}, \hat{f}, \hat{x}$ before executing Line 4, then the constraints of DecomposeFlow will also hold in Line 18 with respect to then reduced problem $\hat{S}, \hat{T}, \hat{f}, \hat{x}$.*

To prove the above theorem, we use the following three Lemmas 1 through 3.

Lemma 1. *Assuming that IP-1 and IP-2 hold, there exists a path $P = \langle \mathrm{o}^+, t, \ldots, \mathrm{o}_r^- \rangle \in G_{\text{ext}}^{\hat{f}}$ in Line 5.*

Proof. Note that initially (i.e. in Line 1) $\hat{f}(\mathrm{o}^+, v) = 1$ holds for $v \in \hat{S} \cup \hat{T}$ by IP-8 and IP-9. This flow will only be reduced once, as a node $t \in \hat{T}$ will only be handled once when it is removed from $\hat{T}$ in Line 4, and similarly, a node $s \in \hat{S}$ will only be moved once into $\hat{T}$ in Line 16. By flow conservation (see IP-1), there must exist a path

from t to either o_r^- or o_S^-. However, as we assume IP-2 to hold, there exists a path from each $s \in \hat{S}$ to o_r^- and we conclude that such a path $P = \langle o^+, t, \ldots, o_r^- \rangle \in G_{\text{ext}}^{\hat{f}}$ must exist. □

Lemma 2. *Assuming that IP-1 has held in Line 5, $f(\delta^+_{E_{\text{ext}}}(v)) - f(\delta^-_{E_{\text{ext}}}(v)) = \delta_{v,P_{j+1}}$ holds for all $v \in V_G$ during construction of P (Lines 8-13), where $\delta_{x,y} \in \{0,1\}$ and $\delta_{x,y} = 1$ iff. $x = y$.*

Proof. We prove this statement by an inductive argument assuming for now that **choose** operations in Lines 9 and 10 are well-defined.

After the first execution of Line 7, $f(\delta^+_{E_{\text{ext}}}(P_2 = t)) - f(\delta^-_{E_{\text{ext}}}(P_2 = t)) = 1$ holds, while for no other node $v \in V_G$ flow on adjacent edges were changed, and therefore $f(\delta^+_{E_{\text{ext}}}(v)) - f(\delta^-_{E_{\text{ext}}}(v)) = 1$ holds. Furthermore, the reduction of flow on edge $(o^+, P_2 = t)$ cannot violate IP-2, such that our claim holds until Line 13 and therefore for the base case $j = 1$.

Assuming that $f(\delta^+_{E_{\text{ext}}}(v)) - f(\delta^-_{E_{\text{ext}}}(v)) = \delta_{v,P_{j+1}}$ has held for $j = n$, it is easy to check that it will continue to hold for $j' = n + 1$, as either in Line 7 or in Line 11 the outgoing flow from node $P_{j'}$ towards node $P_{j'+1}$ is reduced such that $f(\delta^+_{E_{\text{ext}}}(v)) - f(\delta^-_{E_{\text{ext}}}(v)) = \delta_{v,P_{j'+1}}$ indeed holds for all $v \in V_G$. □

Lemma 3. *Assuming that connectivity inequalities IP-2 have held before executing Line 7, these inequalities will hold again at Line 13.*

Proof. We only have to consider the case in which the Constraint IP-2 was violated after executing Line 7. Assume therefore that IP-2 is violated in Line 8. The **choose** operation in Line 9 is well-defined, as IP-2 is violated. Let $W \subseteq V_G$ be any violated set with $\hat{S} \cap W \neq \emptyset$. To prove this lemma, we prove the following four statements:

(a) P_j is contained in W while P_{j+1} is not contained in W.
(b) $\hat{f}(P_j, P_{j+1}) = 0$ holds in Lines 9-10.
(c) Before flow reduction in Line 7, there existed a path $P'' = \langle s, \ldots, P_j, P_{j+1}, \ldots, o_r^- \rangle \in G_{\text{ext}}^{\hat{f}}$ for $s \in \hat{S} \cap W$.
(d) There exists a path $P' = \langle P_j, \ldots, o_S^- \rangle$ with $P'_i \in W$ for $1 \leq i < |P'|$ in $G_{\text{ext}}^{\hat{f}}$.

Considering (a), note that edge (P_j, P_{j+1}) is by definition only included in $\delta^+_{E_{\text{ext}}}(W)$ if $P_j \in W$ and $P_j \notin W$. Thus, assuming that either P_j is not contained in W or assuming that P_{j+1} is contained in W, we can conclude that edge (P_j, P_{j+1}) is not contained in $\delta^+_{E_{\text{ext}}}(W)$. However, in this case the connectivity inequality IP-2 must have been violated even before flow was reduced. This contradicts our assumption that connectivity inequalities IP-2 have held beforehand, therefore proving (a).

The correctness of (b) directly follows from (a), as by (a) $(P_j, P_{j+1}) \in \delta^+_{E_{\text{ext}}}(W)$ holds. As $\hat{f}(\delta^+_{E_{\text{ext}}}(W)) = 0$ holds by definition of W and flow may not be negative, we derive the second statement.

We now prove the statement (c). As connectivity inequalities IP-2 are assumed to have held *before* the flow reduction in Line 7, for each activated Steiner node $s \in \hat{S}$ there existed a path from s to o_r^- in $G_{\text{ext}}^{\hat{f}}$. By the second statement, (P_j, P_{j+1}) is the only

edge in $G^{\hat{f}}_{\text{ext}}$ leaving W showing that indeed a path $P'' = \langle s, \dots, v, P_j, P_{j+1}, \dots, \mathrm{o}_r^- \rangle \in G^{\hat{f}}_{\text{ext}}$ for $s \in \hat{S}$ existed *before* reduction of flow on (P_j, P_{j+1}).

By statement (c), the prefix $\langle s, \dots, P_j \rangle$ of path P'' still exists in $G^{\hat{f}}_{\text{ext}}$ inducing that P_j is reached by a positive flow. By Lemma 2 flow conservation holds for all nodes $w \in W$, since by statement (a) P_{j+1} is not included in W. As o_r^- is not included in W, there must exist a path $P' = \langle P_j, \dots, \mathrm{o}_S^- \rangle \in G^{\hat{f}}_{\text{ext}}$ with $P_i \in W$ for $1 \leq i < m$. This shows the fourth statement (d) and shows that the **choose** operation in Line 10 is well-defined.

We will now prove the main statement of this lemma, namely that in Line 13 the connectivity inequalities IP-2 hold (again). In Line 11, the flow along edge (P_j, P_{j+1}) is incremented again. Assume for the sake of contradiction, that the reduction of flow along (P'_1, P'_2) violates a connectivity inequality with node set W' such that $\hat{f}(\delta^+_{E_{\text{ext}}}(W')) = 0$ holds. By the same argument as used for proving statement (a), it is easy to see that $P'_1 \in W'$ and $P'_2 \notin W'$ must hold. However, by statement (c), after having reverted the flow reduction along (P_j, P_{j+1}), the path $\langle P_j, P_{j+1}, \dots, \mathrm{o}_r^- \rangle$ was re-established in $G^{\hat{f}}_{\text{ext}}$. As flow along any of the edges contained in this path is greater or equal to one, W' cannot possibly violate IP-2 and contain $P_j \in W'$ as the super sink for the root $\mathrm{o}_r^- \notin W' \subseteq V_G$ may never be contained in W'. □

Using the above lemma, we can now prove Theorem 2.

Proof (Theorem 2)**.** Assume that the constraints of IP-A-CVSAP hold with respect to $\hat{S}, \hat{T}, \hat{f}, \hat{x}$ before executing Line 4. By Lemma 1 the **choose** operation in Line 5 is well-defined as IP-1 and IP-2 hold by our assumption. By Lemma 3 the path construction process in Lines 7 through 13 is well-defined as initially IP-2 holds. The execution of Lines 4-18 is therefore well-defined.

To distinguish the state of the variables $\hat{S}, \hat{T}, \hat{f}, \hat{x}$ at Lines 4 and 18 we will use primed variables $\hat{S}', \hat{T}', \hat{f}', \hat{x}'$ to denote the latter state. First note that IP-1 holds by Lemma 2: As path P must terminate in either o_S^- or o_r^- (see Lines 5,10), Lemma 2 reduces to $f(\delta^+_{E_{\text{ext}}}(v)) - f(\delta^-_{E_{\text{ext}}}(v)) = 0$ for all $v \in V_G$ for $j = |P| - 1$ as neither of the super sinks are included in V_G. The connectivity inequalities IP-2 will also hold with respect to $\hat{S}'$ and $\hat{f}'$ as these are preserved by Lemma 3 and $\hat{S}' \subseteq \hat{S}$ holds. Constraint IP-9 holds with respect to $\hat{S}'$ as $\hat{S}' \subseteq \hat{S}$ and the flow along edges in $E^{S^+}_{\text{ext}}$ is never reduced. As similarly flow along edges in $E^{T^+}_{\text{ext}}$ is only reduced for the terminal being connected, Constraint IP-8 could only be violated by a node satisfying $t' \in \hat{T}'$ but $t \notin \hat{T}$. If such a node exists, then it must have been added in Line 16 and as IP-9 has held for $\hat{S}$, constraint IP-3* will hold for $t' \in \hat{S} \cap \hat{T}'$. Analogously, constraint IP-5 is not violated as setting $\hat{x}(s)$ to zero for $s \in \hat{S}$ implies that $s \notin \hat{S}'$ (see Line 16). Constraint IP-4* holds for $\hat{S}'$ as the variable $\hat{x}(s)$ is set to zero whenever the flow along an edge (s, o_S^-) is reduced to zero. Lastly, it is easy to observe that the capacity constraints IP-5, IP-7 cannot be violated as the flow is only reduced. □

Using Theorem 2 we can now prove that Algorithm DecomposeFlow terminates.

Theorem 3. *Algorithm DecomposeFlow terminates.*

Proof. By iteratively applying Theorem 2 the **choose** operations of Algorithm DecomposeFlow are well-defined. Note that by construction of the path $|P|$ (see Lines 5,10) flow variables which values are decremented must have been greater or equal to one before the reduction took place. Since the flow $\hat{f} \in \mathbb{Z}_{\geq 0}$ is finite and is successively reduced during the process of path construction, the inner loop (see Lines 6-14) must terminate. The outer loop must eventually terminate as well, because each node in $\hat{T}$ (see Line 4) is handled exactly once and as a node $s \in \hat{S}$ may be only moved only once into $\hat{T}$ (see Line 16). □

Using Theorem 2 and 3 we can finally prove that Algorithm DecomposeFlow indeed constructs a feasible solution for A-CVSAP.

Theorem 4. *Algorithm DecomposeFlow constructs a feasible solution $\hat{\mathcal{T}}_G \in \mathcal{F}_{\text{A-CVSAP}}$ for A-CVSAP given a solution $(\hat{x}, \hat{f}) \in \mathcal{F}_{\text{IP}}$. Additionally, $C_{\text{CVSAP}}(\hat{\mathcal{T}}_G) \leq C_{\text{IP}}(\hat{x}, \hat{f})$ holds.*

Proof. To show that for $\hat{\mathcal{T}}_G$ constructed by Algorithm DecomposeFlow $\hat{\mathcal{T}}_G \in \mathcal{F}_{\text{M-CVSAP}}$ holds, we need to check CVSAP-1-CVSAP-5 as well as VA-1 and VA-2.We first give short arguments why in fact the conditions CVSAP-1-CVSAP-5 hold:

CVSAP-1	This constraint naturally holds due to Line 2.
CVSAP-2	Algorithm DecomposeFlow does not allow for connecting nodes to terminals. Thereby terminals are indeed leaves in $\hat{\mathcal{T}}_G$ and CVSAP-2 holds.
CVSAP-3	Each time another node is connected to the root r the flow along (r, o_r^-) is decremented (see IP-6). As the flow along this edge is bounded by u_r, the degree constraint CVSAP-3 is satisfied by $\hat{\mathcal{T}}_G$
CVSAP-4	An analogue argument as for CVSAP-3 applies.
CVSAP-5	As paths are constructed according to the flow variables $\hat{f}$ that initially respect capacity constraints on edges IP-7, and as $\hat{f}$ is appropriately reduced on used edges, $\hat{\mathcal{T}}_G$ satisfies the edge capacity constraint CVSAP-5.

It remains to prove that $\hat{\mathcal{T}}_G$ satisfies the conditions VA-1 and VA-2 given by in Definition 1. VA-1 follows directly from Line 18. It remains to prove that $\hat{\mathcal{T}}_G$ satisfies the connectivity requirements VA-2.

First note that $\hat{T} = \emptyset$ holds when DecomposeFlow terminates. We prove that $\hat{S} = \emptyset$ equally holds, thereby showing that each node in $\hat{V}_{\mathcal{T}} \setminus \{r\}$ is connected to another node in $\hat{V}_{\mathcal{T}}$ in Line 15. Assume that $\hat{S} \neq \emptyset$ but $\hat{T} = \emptyset$ holds. We show that this can never be the case using the invariant $s \in \hat{S} \Rightarrow \hat{f}(s, \mathrm{o}_S^-) \geq 1$ which directly follows from Theorem 2 as IP-4* holds. As this holds for all Steiner nodes, $\hat{f}(\delta^+_{E^S_{\text{ext}}}(\hat{S})) \geq |\hat{S}|$ follows. On the other hand, the amount of flow emitted by o^+ equals $|\hat{S}|$ as we assume $\hat{T} = \emptyset$ to hold and by Theorem 2 the constraints IP-9 and IP-8 must hold. Due to the flow conservation constraint IP-1, this implies $\hat{f}(r, \mathrm{o}_r^-) \leq 0$ which immediately violates Constraint IP-2 by considering the node set $W = V_G$. As this contradicts the statement of Theorem 2, we conclude that $\hat{S} = \hat{T} = \emptyset$ must hold when terminating, implying that for all included nodes (see Line 2) an edge was introduced in $\hat{E}_{\mathcal{T}}$ (see Line 18).

As each node (except for the root) has one outgoing edge, it remains to show that $\hat{\mathcal{T}}_G$ does not contain cycles. This follows immediately from the order in which nodes are extracted from $\hat{T}$. This order in fact defines a topological ordering on $\hat{V}_{\mathcal{T}}$ as a cycle containing nodes u and v would imply that u was connected before v and vice versa, that v was connected before u. As this can never be the case, this concludes the proof that $\hat{\mathcal{T}}_G \in \mathcal{F}_{\text{A-CVSAP}}$ holds.

Lastly, $C_{\text{CVSAP}}(\hat{\mathcal{T}}_G) \leq C_{\text{IP}}(\hat{x}, \hat{f})$ is valid as costs associated with activating Steiner nodes are incurred in both objectives and DecomposeFlow uses only edges already accounted for in $C_{\text{IP}}(\hat{x}, \hat{f})$. In fact $C_{\text{CVSAP}}(\hat{\mathcal{T}}_G) < C_{\text{IP}}(\hat{x}, \hat{f})$ may only be the case if the function `simplify` (see Line 18) truncated a path. □

To prove that our formulation IP-A-CVSAP indeed computes an optimal solution, we need the following lemma showing that each solution to A-CVSAP can be mapped on a solution of IP-A-CVSAP with equal cost:

Lemma 4. *Given a network $G = (V_G, E_G, c_E, u_E)$, a request $R_G = (r, S, T, u_r, c_S, u_S)$ and a feasible solution $\hat{\mathcal{T}}_G = (\hat{V}_{\mathcal{T}}, \hat{E}_{\mathcal{T}}, r, \hat{\pi})$ to the corresponding A-CVSAP. There exists a solution $(\hat{x}, \hat{f}) \in \mathcal{F}_{\text{IP}}$ with $C_{\text{CVSAP}}(\hat{\mathcal{T}}_G) = C_{\text{IP}}(\hat{x}, \hat{f})$.*

Proof. We define the solution $(\hat{x}, \hat{f}) \in \{0,1\}^S \times \mathbb{Z}_{\geq 0}^{E_{\text{ext}}}$ in the following way

- $\hat{x}_s = 1$ iff. $s \in \hat{V}_{\mathcal{T}}$ for all $s \in S$,
- $\hat{f}_e \triangleq |(\hat{\pi}(\hat{E}_{\mathcal{T}}))[e]|$ for all $e \in E_G$,
- $\hat{f}_e = 1$ if $v \in \hat{V}_{\mathcal{T}} \setminus \{r\}$ for all $e = (\text{o}^+, v) \in E_{\text{ext}}^{S^+} \cup E_{\text{ext}}^{T^+}$ and $\hat{f}_e = 0$ otherwise,
- $\hat{f}_e \triangleq \delta^-_{\hat{E}_{\mathcal{T}}}(s)$ for all $e = (s, \text{o}_S^-) \in E_{\text{ext}}^{S^-}$ and $\hat{f}(r, \text{o}_r^-) \triangleq \delta^-_{\hat{E}_{\mathcal{T}}}(r)$.

Checking that $(\hat{x}, \hat{f}) \in \mathcal{F}_{\text{IP}}$ and $C_{\text{CVSAP}}(\hat{\mathcal{T}}_G) = C_{\text{IP}}(\hat{x}, \hat{f})$ holds is straightforward. □

Theorem 5. *Using IP-A-CVSAP and DecomposeFlow we can construct a Virtual Arborescence $\hat{\mathcal{T}}_G \in \mathcal{F}_{\text{A-CVSAP}}$ that solves A-CVSAP to optimality.*

Proof. We use IP-A-CVSAP to compute an optimal solution $(\hat{x}, \hat{f}) \in \mathcal{F}_{\text{IP}}$ and afterwards construct the corresponding $\hat{\mathcal{T}}_G \in \mathcal{F}_{\text{A-CVSAP}}$ via DecomposeFlow. Assume for the sake of deriving a contradiction that $\hat{\mathcal{T}}_G$ is not optimal and there exists $\tilde{\mathcal{T}} \in \mathcal{F}_{\text{A-CVSAP}}$ with $C_{\text{CVSAP}}(\tilde{\mathcal{T}}) < C_{\text{CVSAP}}(\hat{\mathcal{T}}_G)$. By Lemma 4 any solution for A-CVSAP can be mapped on a feasible solution of IP-A-CVSAP of the same objective value. This contradicts the optimality of $(\hat{x}, \hat{f}) \in \mathcal{F}_{\text{IP}}$ and $\hat{\mathcal{T}}_G$ must therefore be optimal. □

3.4 Runtime Analysis for DecomposeFlow

We conclude this section with stating that each **choose** operation in DecomposeFlow and checking whether connectivity inequalities IP-2 hold can be implemented using depth-first search. Implementing DecomposeFlow in this way and assuming that an optimal solution for IP-A-CVSAP is given and that G does not contain zero-cost cycles, we can bound the runtime from above as follows.

Theorem 6. *Using depth-first search for choosing paths in Algorithm IP-A-CVSAP and for determining whether connectivity inequalities IP-2 are violated, we can bound the runtime by* $\mathcal{O}\left(|V_G|^2 \cdot |E_G| \cdot (|V_G| + |E_G|)\right)$*, given an optimal solution* $(\hat{x}, \hat{f}) \in \mathcal{F}_{\text{IP}}$ *and assuming that graph* G *does not contain zero-cost cycles.*

Proof. We use depth-first search to separate the connectivity inequalities IP-2 in a canonical manner which we only explain briefly. Given an activated Steiner node $s \in \hat{S}$, we compute the set of all reachable nodes R via depth-first search. If o_r^- is contained in R, then no set of nodes $W \subseteq V_G$ containing s can violate IP-2. On the other hand, if o_r^- is not contained in R, then obviously $W \triangleq R$ violates IP-2. Checking the connectivity inequalities in Line 8 can therefore be performed in time $\mathcal{O}(|\hat{S}| \cdot (|V_{\text{ext}}| + |E_{\text{ext}}|))$. The runtime for choosing a path in Line 10 is clearly dominated by the runtime for checking the connectivity inequalities, and as the previous depth-first search provides a node set W, we do not consider these operations.

The length of any used path P is bounded by $|E_{\text{ext}}|$ as otherwise P would contain a cycle with positive cost. As this cycle can be removed (see function `simplify` in Line 18) yielding a better objective value while remaining feasible, this may never occur by the assumption that our solution is optimal.

Thus, the runtime for the inner loop (Lines 6-14) amounts to $\mathcal{O}(|E_{\text{ext}}| \cdot |\hat{S}| \cdot (|V_{\text{ext}}| + |E_{\text{ext}}|))$ Lastly, the outer loop is performed at most $|\hat{S}| + |\hat{T}|$ many times and the runtime for choosing path P in Line 5 is clearly dominated by the runtime of the inner loop. As $|E_{\text{ext}}| \in \Theta(E_G)$ (assuming E_G to be connected), $|V_{\text{ext}}| \in \Theta(E_G)$ and $|\hat{S}| + |\hat{T}| \in \Theta(V_G)$ holds, the runtime of DecomposeFlow is bounded by

$$\mathcal{O}\left(|V_G|^2 \cdot |E_G| \cdot (|V_G| + |E_G|)\right)$$

and our claim follows. □

4 A Multi-Commodity Flow Formulation

This section introduces a naive multi-commodity flow (MCF) formulation (see MIP-A-CVSAP-MCF) to solve A-CVSAP. The formulation MIP-A-CVSAP-MCF models the virtual arborescence searched for rather directly, as it uniquely determines virtual links and paths for active Steiner nodes. This explicit representation comes at the price of a substantially larger model. In Section 6.4 we provide a computational comparison showing the superiority of our compact formulation IP-A-CVSAP.

4.1 Notation

For ease of representation of MIP-A-CVSAP-MCF we use a modified extended graph, which does not contain a super source but a single super sink.

Definition 5 (Extended Graph for MIP-A-CVSAP-MCF). *Given a directed network* $G = (V_G, E_G, c_E, u_E)$ *and a request* $R_G = (r, S, T, u_r, c_S, u_S)$ *as introduced in Section 2, we define the* extended graph *for the MIP-A-CVSAP-MCF formulation* $G_{\text{MCF}} = (V_{\text{MCF}}, E_{\text{MCF}})$ *as follows*

(EXT-1-MCF) $V_{\text{MCF}} \triangleq V_G \cup \{\text{o}^-\}$,
(EXT-2-MCF) $E_{\text{MCF}} \triangleq E_G \cup \{(r, \text{o}^-)\} \cup E_{\text{MCF}}^{S^-}$,

where we define $E_{\text{MCF}}^{S^-} \triangleq S \times \{\text{o}^-\}$. *We denote by* E_{MCF}^{S} *the set of edges not containing the edges from the super source to the terminals:* $E_{\text{MCF}}^{S} \triangleq E_{\text{MCF}} \setminus E_{\text{MCF}}^{T^+}$. □

As already introduced in Lemma 2 we use the Kronecker-Delta $\delta_{\text{x,y}} \in \{0, 1\}$, where $\delta_{\text{x,y}} = 1$ holds iff. $\text{x} = \text{y}$.

Flow variables corresponding to different commodities are distinguished by superscripts and we use $f^{\text{x}}(\text{Y})$ to denote $\sum_{y \in \text{Y}} f^{\text{x}}(\text{y})$.

We denote the set of feasible solution for MIP-A-CVSAP-MCF by $\mathcal{F}_{\text{MCF}}$.

Mixed Integer Program MIP-A-CVSAP-MCF

$$\text{minimize} \quad C_{\text{MCF}} = \sum_{e \in E_G} \mathbf{c}_e (f_e + \sum_{s \in S} f_{s,e}) + \sum_{s \in S} \mathbf{c}_s \cdot x_s \tag{MCF-OBJ}$$

$$\text{subject to} \quad f^T(\delta^+_{E_{\text{MCF}}}(v)) = f^T(\delta^-_{E_{\text{MCF}}}(v)) + |\{v\} \cap T| \qquad \forall\, v \in V_G \tag{MCF-1}$$

$$f^s(\delta^+_{E^S_{\text{MCF}}}(v)) = f^s(\delta^-_{E^S_{\text{MCF}}}(v)) + \delta_{s,v} \cdot x_s \qquad \forall\, s \in S, v \in V_G \tag{MCF-2}$$

$$f_e^T + \sum_{s \in S} f_e^s \leq \begin{cases} \mathbf{u}_s x_s, & e = (s, \text{o}^-), s \in S \\ \mathbf{u}_r, & e = (r, \text{o}^-) \\ \mathbf{u}_e, & e \in E_G \end{cases} \qquad \forall e \in E_{\text{MCF}} \tag{MCF-3}$$

$$-|S|(1 - f^s_{\bar{s},\text{o}^-}) \leq p_s - p_{\bar{s}} - 1 \qquad \forall\, s, \bar{s} \in S \tag{MCF-4}$$

$$f^s_{(\bar{s},\text{o}^-)} \leq x_{\bar{s}} \qquad \forall\, s \in S, \bar{s} \in S - s \tag{MCF-5*}$$

$$f^s_{s,\text{o}^-} = 0 \qquad \forall\, s \in S \tag{MCF-6*}$$

$$f^s_{\bar{s},\text{o}^-} + f^{\bar{s}}_{s,\text{o}^-} \leq 1 \qquad \forall\, s, \bar{s} \in S \tag{MCF-7*}$$

$$x_s \in \{0, 1\} \qquad \forall\, s \in S \tag{MCF-8}$$

$$f_e^T \in \mathbb{Z}_{\geq 0} \qquad \forall\, e \in E_{\text{MCF}} \tag{MCF-9}$$

$$f_e^s \in \{0, 1\} \qquad \forall\, s \in S, e \in E^S_{\text{MCF}} \tag{MCF-10}$$

$$p \in [0, |S| - 1] \qquad \forall\, s \in S \tag{MCF-11}$$

4.2 The MIP Model

The formulation MIP-A-CVSAP-MCF uses one commodity for each Steiner site (see MCF-10) and a single commodity for the flow originating at the terminals (see MCF-9). Note that while f^s defines a flow variable for each Steiner site $s \in S$ we use the superscript f^T to denote a single commodity. Furthermore note that flow variables f^s

corresponding to Steiner sites are binary whereas the aggregated flow variables f^T from the terminals are defined to be integers.

We now briefly describe how a solution $(x, p, f^s, f^T) \in \mathcal{F}_{\text{MCF}}$ relates to a virtual arborescence $\hat{\mathcal{T}}_G = (\hat{V}_\mathcal{T}, \hat{E}_\mathcal{T}, \hat{r}, \hat{\pi}) \in \mathcal{F}_{\text{A-CVSAP}}$. We naturally set $\hat{V}_\mathcal{T} \triangleq \{r\} \cup \{s \in S | \hat{x}_s \geq 1\} \cup T$ and $\hat{r} = r$. We continue by showing how $\hat{E}_\mathcal{T}$ and $\hat{\pi}$ can be retrieved.

Constraints MCF-1 and MCF-2 specify flow preservation for the commodities such that terminal nodes emit one unit of flow in f^T and activated Steiner nodes emit one unit of flow in f^s. Note that in MCF-2 $\delta_{s,v}$ is a constant. As these constraints are specified for nodes $v \in V_G$, flows in f^T and f^s must terminate in o^- via edges in $E^{S^-}_{\text{MCF}}$ or via (r, o^-).

If a Steiner node $s \in S$ is activated, f^s defines a path P^s from s to o^-. We therefore include $e = (s, P^s_{|P^s|-1})$ in $\hat{E}_\mathcal{T}$ and set $\hat{\pi}(e) = \langle P^s_1, \dots, P^s_{|P^s|-1} \rangle$. As we use a single commodity for flow originating at the terminals, we have to first decompose f^T into paths $\{P^t | t \in T\}$ such that P^t originates at t and terminates in o^-. Due to the single destination, this can always be done using the standard $s - t$ flow decomposition [2].

As the capacity constraints MCF-3 are defined analogously to IP-5-IP-7, we only need to establish the validity of connectivity condition VA-2 to show that $\hat{\mathcal{T}}_G \in \mathcal{F}_{\text{A-CVSAP}}$ holds. As terminals and active Steiner nodes must be connected as discussed above, VA-2 may only be violated by $\hat{\mathcal{T}}_G$ if a cycle exists in $\hat{E}_\mathcal{T}$. To forbid such cycles, we adapt the well-known Miller-Tucker-Zemlim (MTZ) constraints [5] using continuous priority variables $p_s \in [0, |S| - 1]$ in MCF-4. The MTZ constraint MCF-4 enforces $f^s_{\bar{s},\text{o}^-} = 1 \Rightarrow p_s \geq p_{\bar{s}} + 1$, forbidding cyclic assignments containing only Steiner nodes. As terminals may not receive flow and the root may not send flow, this suffices to forbid cycles in $\hat{E}_\mathcal{T}$ overall and thus $\hat{\mathcal{T}}_G \in \mathcal{F}_{\text{A-CVSAP}}$ holds.

As formulations relying on MTZ constraints are comparatively weak [28], we introduce additional valid inequalities MCF-5*, MCF-6* and MCF-7* to strengthen the formulation. Constraint MCF-6* disallows Steiner node $s \in S$ to absorb its own flow and MCF-7* explicitly forbids cycles of length 2. Lastly, Constraint MCF-5* forces Steiner nodes receiving flow from another Steiner node to be activated.

5 Branch-and-Cut Solver

We have developed a branch-and-cut solver for CVSAP based on IP-A-CVSAP and DecomposeFlow, which can be obtained from [33]. Our solver uses SCIP [1] as underlying Branch-and-Cut framework with SoPlex [37] as LP solver. In Section 5.1 we shortly discuss our implementation of the separation procedures for Constraints IP-2 and IP-3*. Afterwards we present in Section 5.2 a primal heuristic to generate feasible solutions during the branch-and-bound search.

5.1 Separation

Our solver generally follows the comprehensive work by Koch et al. [21] and we assume the reader's familiarity with separation procedures (see e.g. [34]). As the separation techniques used are well-known, we only sketch the most important features.

Instead of using a sophisticated maximal flow algorithm as [21] proposes, we implemented the algorithm of Edmonds and Karp (see e.g. [2]). As choosing this simple algorithm only allows for constructing $s - t$ flows, we perform a single maximal flow computation for each $s \in S$ when separating connectivity inequalities IP-2 and analogously perform $|T|$ many maximal flow computations when the valid inequalities of IP-3* are to be separated. To improve performance for executing the maximal flow computations at each node, we use multithreading to speed up the computation.

Furthermore, we have implemented techniques that (empirically) *improve* the *quality* of found violated inequalities for IP-2 or IP-3*. Following [21] we implemented *creep-flow* and *nested-cuts*. We opted not to implement *back cuts*, as in our formulation of IP-2 violated node sets with respect to a given Steiner site $s \in S$ would probably be violated for other Steiner sites too. Adding back cuts for each of the violated node sets with respect to many $s \in S$ would in turn probably lead to many redundant constraints.

5.2 Primal Heuristic

In Section 6 we will show that the dual bound using the formulation IP-A-CVSAP comes close to the optimal value within minutes of execution. Even though the branch-and-cut framework SCIP implements many primal heuristics [1], the heuristics of SCIP found effective for CVSAP are generally computationally expensive as they e.g. perform dive operations in the branch-and-bound tree (see 6.3). Additionally, some of the heuristics implemented in SCIP, which are based on local search, already need a feasible solution as input.

Notational Remark. We use the function `FlowDecomposition` with parameters (G, f, v, t, D) to calculate a flow decomposition in graph G from $t \in V_G$ to one of the nodes in $D \subseteq V_G$. The flow is given by $f : E_G \to \mathbb{R}_{\geq 0}$ and $v \in \mathbb{R}_{>0}$ specifies the amount of flow to decompose. The result of this function is a set of paths with a value specifying the amount of flow carried by it $\{(P_i, f_i)\} \in \mathcal{P}_G \times \mathbb{R}_{>0}$ such that all paths start in t, terminate at one of the nodes of D, the sum of the carried flow amounts to v and the sum of carried flow on each edge does not exceed the original flow f on any edge. We furthermore use the function $\texttt{ShortestPath}(G, c, t, D)$ to calculate the shortest paths in graph G from $t \in V_G$ to one of the nodes in $D \subseteq V_G$ with respect to the partial cost function $c : E_G \to \mathbb{R}_{\geq 0}$. An edge for which no cost is specified, is assumed to be of zero cost.

Synopsis of FlowDecoRound. Our primal heuristic FlowDecoRound uses the LP relaxation at the current node in the branch-and-bound tree as input. We denote by $\mathcal{F}_{\text{LP}}$ the set of feasible solutions to IP-A-CVSAP where the integrality restrictions on the flow (see IP-11) and the decision variables for activating Steiner sites (see IP-10) are relaxed to $f_e \in \mathbb{R}_{\geq 0}$ and $x_s \in [0, 1]$ for all $e \in E_{\text{ext}}$ and $s \in S$ respectively. Our heuristic works in the following three phases:

1. In the first phase for each terminal a flow decomposition is perfomed based on the flow values $\hat{f}$ of the current LP solution $(\hat{x}, \hat{f}) \in \mathcal{F}_{\text{LP}}$ such that the path may either

Algorithm FlowDecoRound

Input : Network $G = (V_G, E_G, c_E, u_E)$, Request $R_G = (r, S, T, u_r, c_S, u_S)$, LP relaxation solution $(\hat{x}, \hat{f}) \in \mathcal{F}_{\text{LP}}$ to IP-A-CVSAP

Output: Potentially a feasible Virtual Arborescence $\hat{\mathcal{T}}_G$ for CVSAP

set $\hat{S} \triangleq \emptyset$ **and** $\hat{T} \triangleq \emptyset$ **and** $U = T$

set $\hat{V}_{\mathcal{T}} \triangleq \{r\}$, $\hat{E}_{\mathcal{T}} \triangleq \emptyset$ **and** $\hat{\pi} : \hat{E}_{\mathcal{T}} \to \mathcal{P}_{G_{\text{ext}}}$

set $u(e) \triangleq \begin{cases} u_E(e) & \text{, if } e \in E_G \\ u_r(r) & \text{, if } e = (r, \text{o}_r^-) \\ u_S(s) & \text{, if } e = (s, \text{o}_S^-) \in E_{\text{ext}}^{S^-} \\ 1 & \text{, else} \end{cases}$ **for all** $e \in E_{\text{ext}}$

while $U \neq \emptyset$ **do**

  choose $t \in U$ uniformly at random **and set** $U \leftarrow U - t$

  set $\Gamma_t \triangleq \texttt{FlowDecomposition}\left(G_{\text{ext}}, \hat{f}, \hat{f}(\text{o}^+, t), t, \{\text{o}_S^-, \text{o}_r^-\}\right)$

  set $\hat{f} \leftarrow \hat{f} - \sum_{(P,f) \in \Gamma_t, e \in P} f$

  set $\Gamma_t \leftarrow \Gamma_t \setminus \{(P, f) \in \Gamma_t | \exists e \in P . u(e) = 0\}$

  set $\Gamma_t \leftarrow \Gamma_t \setminus \{(P, f) \in \Gamma_t | (\hat{V}_{\mathcal{T}} + t, \hat{E}_{\mathcal{T}} + (t, P_{|P|-1}))$ is not acyclic $\}$

  if $\Gamma_t \neq \emptyset$ **then**

    choose $(P, f) \in \Gamma_t$ with probability $f / \left(\sum_{(P_j, f_j) \in \Gamma_t} f_j\right)$

    if $P_{|P|-1} \notin \hat{V}_{\mathcal{T}}$ **then**

      set $U \leftarrow U + P_{|P|-1}$ **and** $\hat{V}_{\mathcal{T}} \leftarrow \hat{V}_{\mathcal{T}} + P_{|P|-1}$

    set $\hat{V}_{\mathcal{T}} \leftarrow \hat{V}_{\mathcal{T}} + t$ **and** $\hat{E}_{\mathcal{T}} \leftarrow \hat{E}_{\mathcal{T}} + (t, P_{|P|-1})$ **and** $\hat{\pi}(t, P_{|P|-1}) \triangleq P$

    set $u(e) \leftarrow u(e) - 1$ **for all** $e \in P$

set $u(e) \leftarrow 0$ **for all** $e = (s, \text{o}_S^-) \in E_{\text{ext}}^{S^-}$ **with** $s \in S \wedge s \notin \hat{V}_{\mathcal{T}}$

set $\bar{T} \triangleq (T \setminus \hat{V}_{\mathcal{T}}) \cup (\{s \in S \cap \hat{V}_{\mathcal{T}} | \delta^+_{\hat{E}_{\mathcal{T}}}(s) = 0\})$

for $t \in \bar{T}$ **do**

  choose $P \leftarrow \texttt{ShortestPath}\left(G^u_{\text{ext}}, c_E, t, \{\text{o}_S^-, \text{o}_r^-\}\right)$
  such that $(\hat{V}_{\mathcal{T}} + t, \hat{E}_{\mathcal{T}} + (t, P_{|P|-1}))$ is acyclic

  if $P = \emptyset$ **then**

    return `null`

  set $\hat{V}_{\mathcal{T}} \leftarrow \hat{V}_{\mathcal{T}} + t$ **and** $\hat{E}_{\mathcal{T}} \leftarrow \hat{E}_{\mathcal{T}} + (t, P_{|P|-1})$ **and** $\hat{\pi}(t, P_{|P|-1}) \triangleq P$

  set $u(e) \leftarrow u(e) - 1$ **for all** $e \in P$

for $e \in \hat{E}_{\mathcal{T}}$ **do**

  set $P \triangleq \hat{\pi}(e)$

  set $\hat{\pi}(e) \leftarrow \langle P_1, \ldots, P_{|P|-1} \rangle$

set $\hat{\mathcal{T}}_G \triangleq$ Virtual Arborescence $(\hat{V}_{\mathcal{T}}, \hat{E}_{\mathcal{T}}, r, \hat{\pi})$

return $\texttt{PruneSolution}(\hat{\mathcal{T}}_G)$

Algorithm PruneSteinerNodes

Input : Network $G = (V_G, E_G, c_E, u_E)$, Request $R_G = (r, S, T, u_r, c_S, u_S)$,
Solution $\hat{\mathcal{T}}_G \in \mathcal{F}_{\text{A-CVSAP}}$ for A-CVSAP
Output: Feasible Virtual Arborescence $\hat{\mathcal{T}}'_G \in \mathcal{F}_{\text{A-CVSAP}}$ with $C_{\text{CVSAP}}(\hat{\mathcal{T}}'_G) \leq C_{\text{CVSAP}}(\hat{\mathcal{T}}_G)$

set $O \triangleq S \cap \hat{V}_{\mathcal{T}}$
while $O \neq \emptyset$ **do**
  choose $s \in O$ **maximizing** $c_S(s)/|\delta^-_{\hat{E}_{\mathcal{T}}}(s)|$
  set $O \leftarrow O - s$
  set $U \triangleq \{t|(t,s) \in \delta^-_{\hat{E}_{\mathcal{T}}}(s)\}$ `//set of disconnected nodes`
  set $R \triangleq \delta^+_{\hat{E}_{\mathcal{T}}}(s) \cup \delta^-_{\hat{E}_{\mathcal{T}}}(s)$ `//set of removed virtual edges`
  set $\mathcal{P}_R \triangleq \{\hat{\pi}(e)|e \in R\}$ `//set of removed paths`
  set $b \triangleq c_S(s) + \sum_{P \in \mathcal{P}_R} c_E(P)$ `//budget`
  set $\hat{V}'_{\mathcal{T}} \triangleq \hat{V}_{\mathcal{T}} \setminus (U \cup \{s\})$
  set $\hat{E}'_{\mathcal{T}} \triangleq \hat{E}_{\mathcal{T}} \setminus (\delta^-_{\hat{E}_{\mathcal{T}}}(s) \cup \delta^+_{\hat{E}_{\mathcal{T}}}(s))$ **and** $\hat{\pi}' : \hat{E}'_{\mathcal{T}} \to \mathcal{P}_G$
    such that $\hat{\pi}'(e) = \hat{\pi}(e)$ **for all** $e \in \hat{E}'_{\mathcal{T}}$
  set $u'(e) \triangleq \begin{cases} u_E(e) - |\hat{\pi}(\hat{E}'_{\mathcal{T}})[e]| & \text{, if } e \in E_G \\ u_r(r) - |\delta^-_{\hat{E}'_{\mathcal{T}}}(r)| & \text{, if } e = (r, \text{o}_r^-) \\ u_S(s) - |\delta^-_{\hat{E}'_{\mathcal{T}}}(s')| & \text{, if } e = (s', \text{o}_S^-) \in E^{S-}_{\text{ext}} \\ 1 & \text{, else} \end{cases}$ **for all** $e \in E_{\text{ext}}$
  set $u'(s, \text{o}_S^-) \leftarrow 0$
  for $e = (t, s) \in \delta^-_{\hat{E}_{\mathcal{T}}}(s)$ **do**
    choose $P \triangleq \texttt{ShortestPath}\left(G^{u'}_{\text{ext}}, c_E, t, \{\text{o}_S^-, \text{o}_r^-\}\right)$
      such that $(\hat{V}'_{\mathcal{T}} + t, \hat{E}'_{\mathcal{T}} + (t, P_{|P|-1}))$ is acyclic
    if $P = \emptyset \vee b - c_E(P) \leq 0$ **then**
      goto 4;
    set $b \leftarrow b - c_E(P)$
    set $\hat{V}'_{\mathcal{T}} \leftarrow \hat{V}'_{\mathcal{T}} + t$
    set $\hat{E}_{\mathcal{T}} \leftarrow \hat{E}_{\mathcal{T}} + (t, P_{|P|-1})$ **and** $\hat{\pi}(t, P_{|P|-1}) \triangleq \langle P_1, \ldots, P_{|P|-1} \rangle$
    set $u(e) \leftarrow u(e) - 1$ **for all** $e \in P$
  set $\hat{\mathcal{T}}_G \leftarrow$ Virtual Arborescence $(\hat{V}'_{\mathcal{T}}, \hat{E}'_{\mathcal{T}}, r, \hat{\pi}')$
  set $O \leftarrow S \cap \hat{V}_{\mathcal{T}}$
return $\hat{\mathcal{T}}_G$

terminate in o_S^- or o_r^-. The flow decomposition returns a set of paths paired with an amount of flow carried by them. After discarding paths for which no capacity is left and paths that would lead to a cycle in the solution, one of the remaining paths is chosen uniformly according to the flow amount carried by it. If the path leads to an (inactive) Steiner site, then the aggregation node is opened and becomes itself a terminal to be connected during the first phase. If none of the paths returned by the flow decomposition is feasible, the terminal is not connected.

2. In the second phase the terminals (including Steiner nodes) that are still disconnected are connected using shortest paths under the restriction that these paths may not yield a cycle in the solution.
3. If all terminals (including Steiner nodes) have been connected in the second phase, then a feasible solution has been constructed. Since in the first phase any Steiner node is activated if a path to it was selected, we try to reduce the costs of the solution by removing activated Steiner nodes from the solution. This procedure is shown in Algorithm PruneSteinerNodes.
   The activated Steiner nodes are ordered according to the ratio of cost for installing it divided by the number of nodes connected to it. The Steiner node maximizing this ratio is selected and together with all its incoming and outgoing edges removed from the solution. Thus, the objective value is decreased, giving an budget b for reconnecting the disconnected nodes. Reconnecting the disconnected nodes is again done using shortest paths under the constraint that no cycles may be introduced to the solution. If a node cannot be connected or using the shortest path would exceed the budget, the algorithm selects another activated aggregation nodes and tries to remove it. If however all nodes could be reconnected and the budget was not exceeded, then a cheaper virtual arborescence has been found and the process is restarted with all opened aggregation nodes.

The idea to use a flow decomposition to generate a set of possible paths and afterwards selecting one of the paths at random (according to the carried flow) was first proposed by Raghavan and Thomposon [31]. Our incentive to apply this scheme to CVSAP is twofold. Firstly, using this scheme only paths will be selected which have been (partially) accounted for in the objective of the LP relaxation. Secondly, it allows for an easy mechanism for deciding which Steiner nodes to activate. A Steiner node with much incoming flow is more likely to be opened than some Steiner node absorbing few flow. Furthermore we thereby circumvent the problem of deciding a priori which Steiner nodes should be activated or not. Section 6.3 contains an evaluation of the performance of FlowDecoRound.

6 Computational Results & Evaluation

Since CVSAP is NP-hard (see Section 2), we investigate the applicability of our approach with an empirical computational study on different problem instances using the solver that was introduced in Section 5. We consider two different classes of instances, one being based on grid graphs and the other one being based on Internet topologies (see Section 6.1). In Section 6.2 we validate the choice of including the directed cut constraints IP-3* as well as separation related optimizations in our implementation.

While the main results considering our implementation are presented in Section 6.3 we present a computational comparison with the multi-commodity flow formulation MIP-A-CVSAP-MCF (see Section 4) in Section 6.4. We conclude our computational study in Section 6.5 with analyzing the runtime allocated by the different components of our solver to devise possible optimizations.

Note that all problem instances used for our evaluation are available from [33].

Technical Notes. All our experiments were conducted on machines equipped with an 8-core Intel Xeon L5420 processor running at 2.5 Ghz and 16 GB RAM. As we use 25 instances for each problem class, we mainly use box plots to present our results. Note that all boxplots presented in this section use the standard $1.5 * IQR$ whiskers of R (and not the 95th and 5th percentiles).

6.1 Problem Classes

We use two classes of problems for our experimal evaluation. One class is based on $n \times n$ grid graphs while the other is based on router-level Internet topologies [30]. Based on the inherent symmetry and supported by our computational results, problem instances based on grid graphs present hard instances that already for $n = 20$ cannot be solved to optimality within reasonable time. On the other hand, we use Internet topologies to show the applicability to solve realistically sized instances close to optimality.

Grid Graphs. All grid instances were generated according to the following parameters. Edge capacities are set to 3 while the capacity of the root and Steiner sites is set to 5. We use unit costs on edges and a cost of 20 for activating Steiner nodes. The locations of terminals, Steiner sites and the root are chosen in a uniformly distributed fashion, such that $|S| \propto 20\%|V_G|$ and $|T| \propto 25\%|V_G|$ holds. We consider problems based on $n \times n$ grids for $n = 12, 16, 20$ and generated 25 instances for each of these sizes. Table 2 summarizes the resulting number of graph sizes and the number of generated Steiner sites and terminals.

Internet Topologies. We have used the tool IGen [30] to generate two Internet alike topologies, one having 1600 and the other having 3200 nodes. Nodes are distributed uniformly on a world map. Topologies are created by clustering nodes in a 20 (horizontal) by 6 (vertical) grid. Each of these clusters can be understood as a single autonomous

n	$\|V\|$	$\|E\|$	$\|S\|$	$\|T\|$
12	144	528	29	36
16	256	960	51	64
20	400	1520	80	100

Table 2: Size of graph, number of Steiner sites $|S|$ and number of terminals $|T|$ for the different sizes of $n \times n$ grid graph.

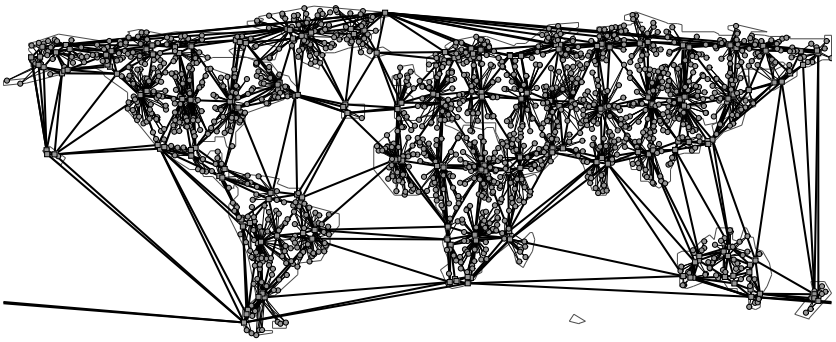


Fig. 2: Topology generated by IGen with 1600 nodes. Nodes depicted as squares are PoPs while node depicted as disks are internal nodes.

system (AS). Within each AS a certain number of nodes are selected to become Points-of-Presences (PoP).

While nodes within an AS are solely connected to a certain number of internal PoPs[1], PoPs are interconnected to provide global connectivity[2]. In Figure 2 the generated instance with 1600 nodes is depicted. Table 3 gives an overview over the characteristics for the two topologies. Note that for IGen.3200 the number of PoPs per cluster as well as the number of links between internal nodes and PoPs has been increased. For each of the topologies we have again generated 25 different instances according to the following parameters.

Steiner sites are only located at PoPs and Terminals may not be PoPs. The cost of using edges is given by the euclidean distance. Inter-PoP links have a capacity of 10 while intra-AS links have a capacity of 2. Activation costs for Steiner sites are chosen according to $\mu(c_E) \cdot \mathcal{U}(25, 75)$, where $\mu(c_E)$ denotes the average edge length. Steiner sites as well as the root have a capacity of 5.

Name	$\|V\|$	$\|E\|$	$\|P\|$	$\|I \to P\|$	$\|P \to P\|$	$\|S\|$	$\|T\|$
IGen.1600	1600	6816	3	2	2	200	300
IGen.3200	3200	19410	4	3	2	400	600

Table 3: Generation parameters for instances IGen.1600 and IGen.3200. $|V|$ denotes the number of nodes and $|E|$ the number of edges. $|P|$ denotes the number of PoPs per cluster. $|I \to P|$ denotes the number of edges with which each internal node is connected to a PoP[1]. $|P \to P|$ denotes the number of peers for each PoP[2].

[1] using the sprint heuristic, see [30]
[2] using a delaunay triangulation, see [30]

6.2 Validation of Implementation Choices

As shown in Section 6.3 instances based on grid graphs are even for smaller instances hard to solve. As we believe these instances to be hard for CVSAP due to their inherent symmetry (see [32] for a discussion of hypercube graphs as hard instances for STP), we validate the choice of implementation parameters on this problem class. For this purpose we haven chosen instances with $n = 12$ as they are still throughout solvable to optimality within reasonable time.

Following the generation parameters as described in Section 6.1, our 12×12 grid instances contain 29 possible Steiner sites and 36 terminals that need to be connected. We consider the following four parameter settings to evaluate whether optimizations for the separation procedure and the separation of the directed Steiner cuts IP-3* improve performance.

(T&S) Constraints IP-3* are separated and nested-cuts and creep-flow are used.
(T) Constraints IP-3* are separated but neither nested-cuts nor creep-flow are used.
(S) Constraints IP-3* are not separated but nested-cuts and creep-flow are used.
(-) Neither constraints IP-3* are separated, nor are nested-cuts or creep-flow used.

Figure 3 plots the total runtime as well as the number of cuts introduced, for solving each of the 25 instances to optimality. Note that while (T&S) provides the best runtime performance, the runtimes of (S) come close to (T) even though the number of cuts generated is substantially higher. Clearly, (-) provides the worst performance.

However, as one might favor an approach that generates *provably* good solutions rather quickly but takes longer to solve a problem to optimality, we investigate the objective gap over time. The objective gap is formally defined as $|P - D|/|D|$ where P denotes the value of the best (primal) solution found so far and D denotes the value of the (dual) lower bound. In Figure 4 the mean objective gap over the 25 instances is shown and Figure 5 provides detailed boxplots for each of the parameters. Again, (T&S) provides the best performance, while (T) comes close to it; Importantly, the mean objective gap for (S) is noticeably higher than for (T), even though the total runtimes are comparable.

We conclude by observing that CVSAP can be solved within minutes to optimality on 12×12 grids using parameter setting (S&T) and that the separation of the valid inequalities IP-3* dramatically improves performance. Furthermore, the usage of creep-flow and nested-cuts does improve performance slightly even though incurring further computational costs during the separation procedure. However, as the number of overall generated cuts is reduced, we have chosen to conduct all further experiments using the setting (T&S).

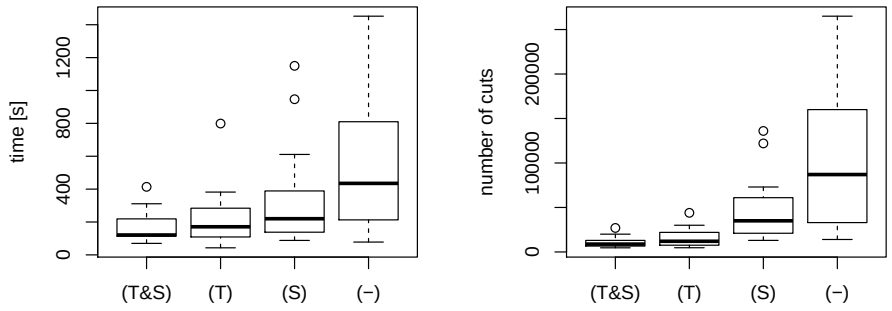


Fig. 3: Runtimes in seconds and number of generated cuts for the 25 runs on 12×12 grids using different parameters.

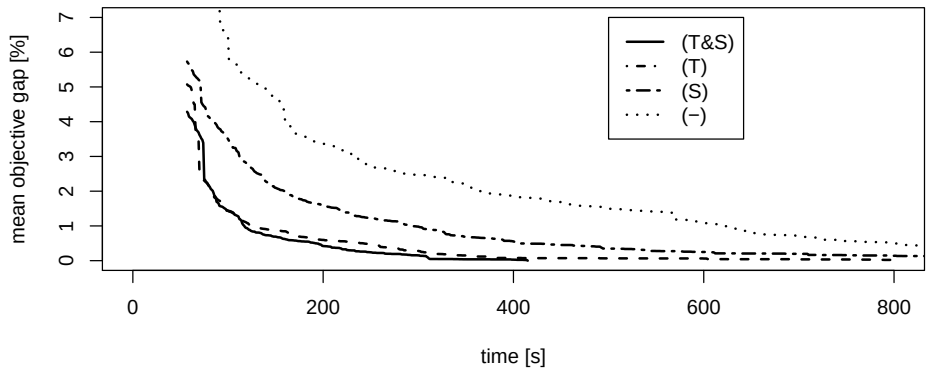


Fig. 4: The mean of the objective gap for the 25 instances using the parameter settings (T&S), (T), (S) and (-).

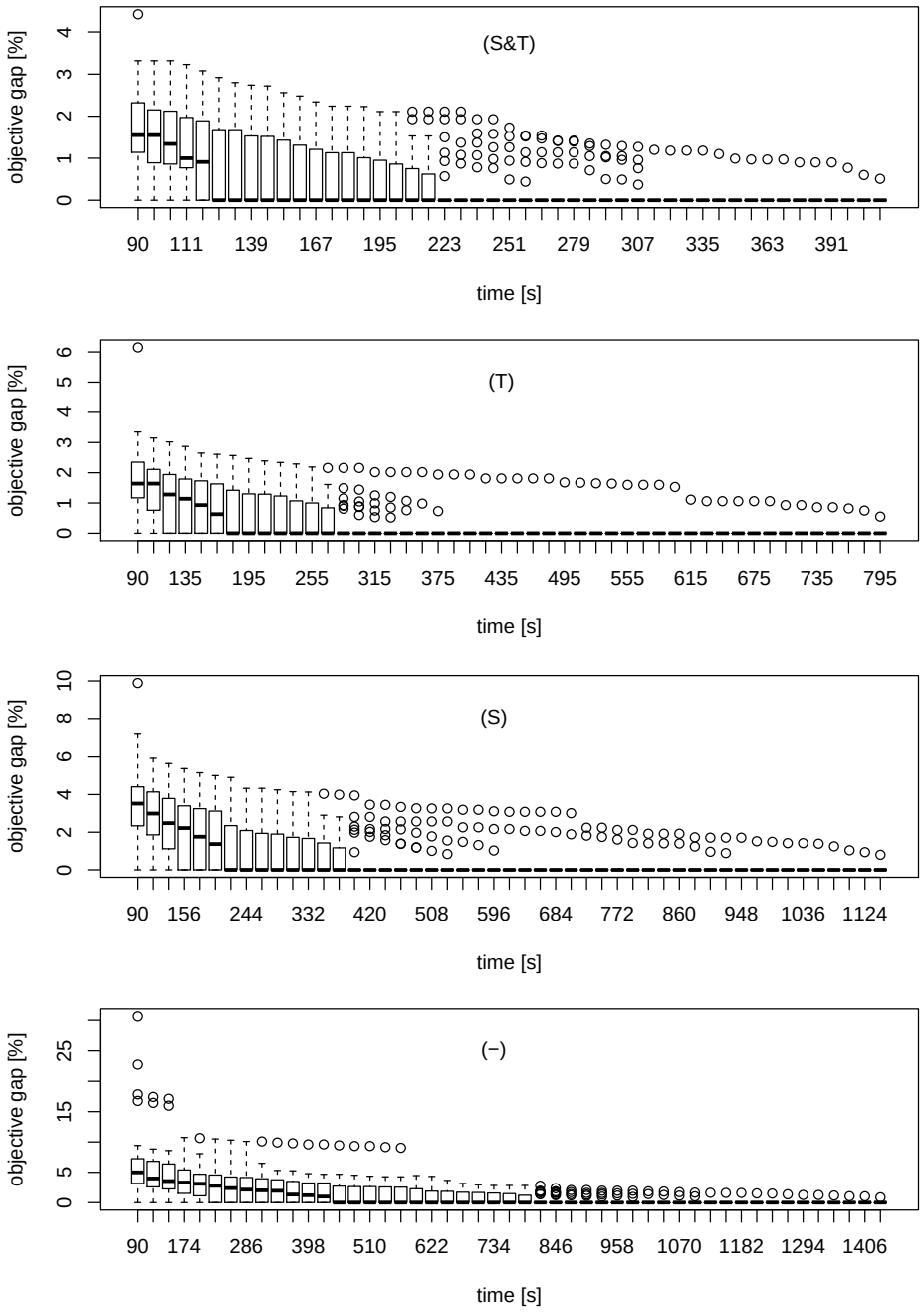


Fig. 5: The objective gap over time for the same 25 instances using 12×12 grids under the four different parameter settings (S&T), (T), (S) and (-) measured at regular time intervals. Note the different scales.

6.3 Main Computational Results

As shown in Section 6.2 instances based on grid graphs for $n = 12$ can be solved to optimality within minutes. We will now consider instances based on the larger $n \times n$ grids with $n = 16, 20$ and as well as instances based on Internet Topologies IGen.1600 and IGen.3200. For each of these problem classes we present results for 25 independently generated instances according to the parameters presented in Section 6.1. As many of the instances cannot be solved to optimality anymore, we terminate experiments after 2 hours.

Figure 6 depicts the objective gap over time. First note that, independent of the problem class, the gap decreases substantially during the first 30 minutes while decreasing only slightly during the last 90 minutes of execution. Furthermore, while for grid gaphs with $n = 16$ some solutions can be solved, this is not the case for $n = 20$. Considering the Internet topologies, IGen.1600 instances can be solved very close to optimality and notably, already after 10 minutes the median gap lies below 0.3%. While some of the IGen.3200 instances exhibit a similarly low gap, the median gap is generally a magnitude higher.

To better understand the progress in the objective gap, we will continue to investigate both the progress in the lower as well as the primal bound. In Figure 7 the progress of the lower bound during the first 3 minutes is depicted. As measure we use the relative lower bound, i.e. the lower bound with respect to the best lower bound achieved after 2 hours. Note that at point in time 0 the respective lower bound is the root relaxation without any separated cuts stemming from IP-2 and IP-3*. Across the board, the relative lower bound after 3 minutes is within a margin of less than 0.02%. Thus in the remaining 117 minutes of executions, the lower bound only improves sligthly.

To investigate the primal bound, i.e. the objective value of the best solution found so far, we depict in Figure 8 the solutions found for all grid instances as well as the Internet topology instances. Primal solutions may either be generated by our heuristic FlowDecoRound, a bundled heuristic of SCIP or integral solutions of the LP relaxation. Importantly to note, all but less than 10 solutions found by SCIP's heuristics were generated using LP diving, such that we only depict these.

Considering grid graphs, our heuristic FlowDecoRound does generally construct solutions very quickly with an objective gap of $30\% - 100\%$ whereas LP diving heuristics find solutions with a gap of an order less. The objective gap of solutions found by SCIP's diving heuristics are essentially the same for $n = 16$ and $n = 20$, while for $n = 20$ solution generation is clearly delayed. Considering the Internet topologies, the heuristic developed by us achieves a gap of $3\% - 13\%$, while LP diving heuristics provide again the best solutions within a gap of less than 0.8%. Finally, note that solutions found by integral linear relaxations yield the best objective values but are only found very late in the branch-and-cut process and only for the smaller instances of 16×16 grids and IGen.1600. We conclude the analysis of our results with the following observations.

1. Using formulation IP-A-CVSAP we can solve realistically sized instances on Internet topologies close to optimality. While for IGen.3200 instances the objective gap might be as high as 4% we believe this to be due to the lack of high quality solutions.

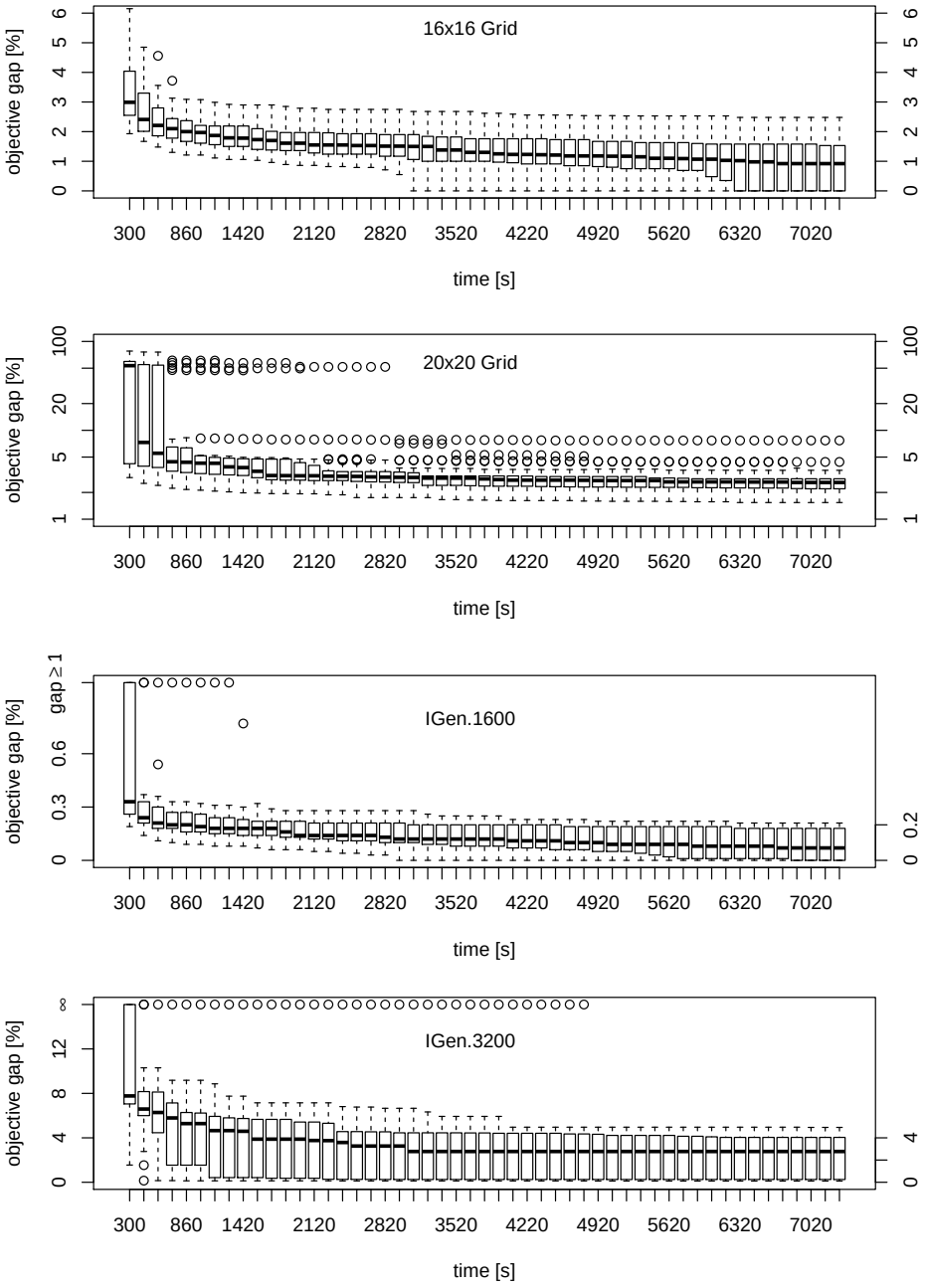


Fig. 6: The objective gap of 25 instances for $n \times n$ grids with $n = 16, 20$ as well as for IGen.1600 an IGen.3200 measured at regular time intervals. Note the logarithmic y-axis for $n = 20$. For IGen.1600, objective gaps above 1% are subsumed. For IGen.3200 an objective gap of ∞ expresses, that no primal solution has been found.

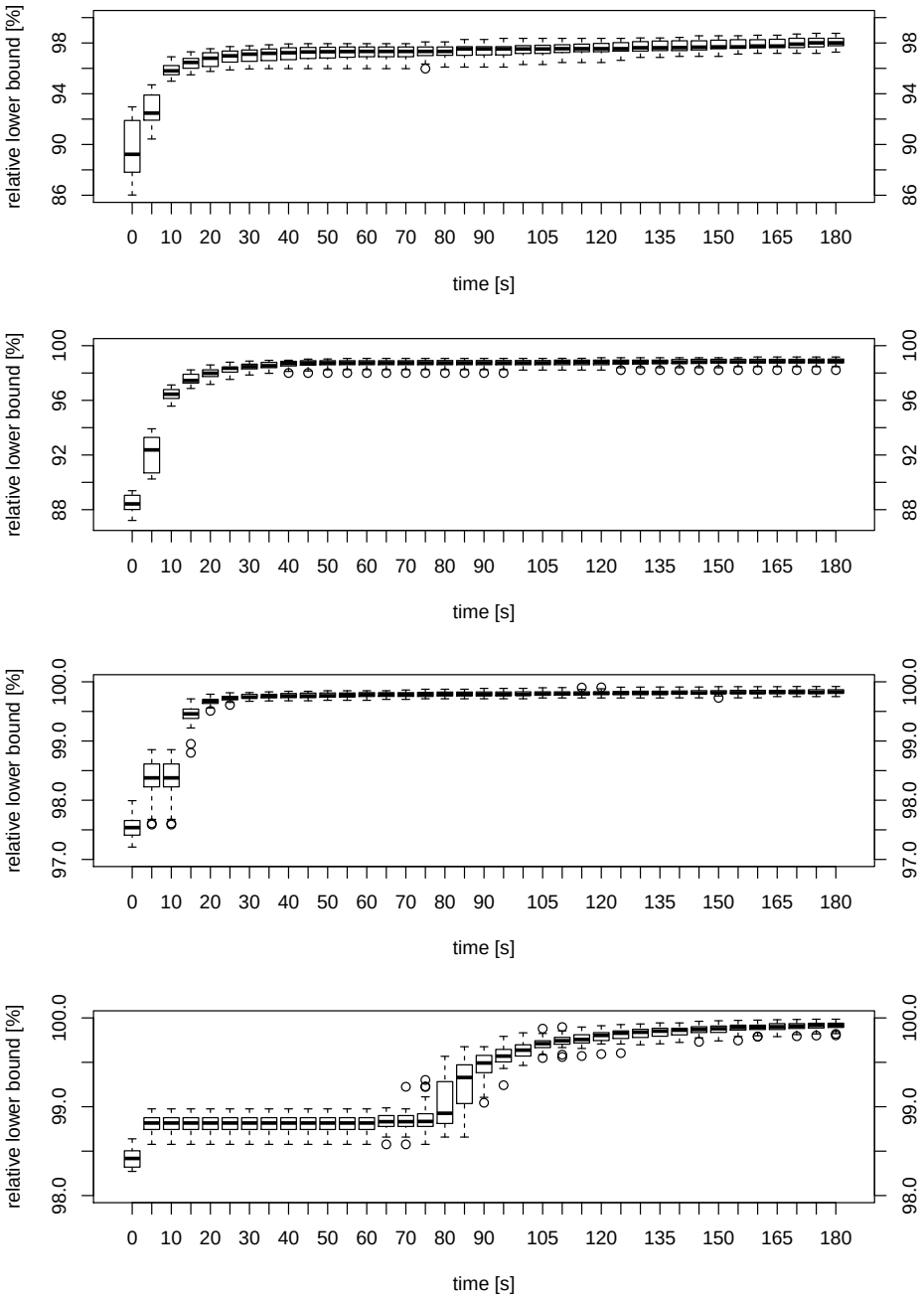


Fig. 7: The relative lower bound, i.e. the lower bound compared to the best lower bound found after 2 hours, of the $n \times n$ grid graphs for $n = 16, 20$ as well as for IGen.1600 and IGen.3200 measured at regular time intervals. Please note the log scale for $n = 20$.

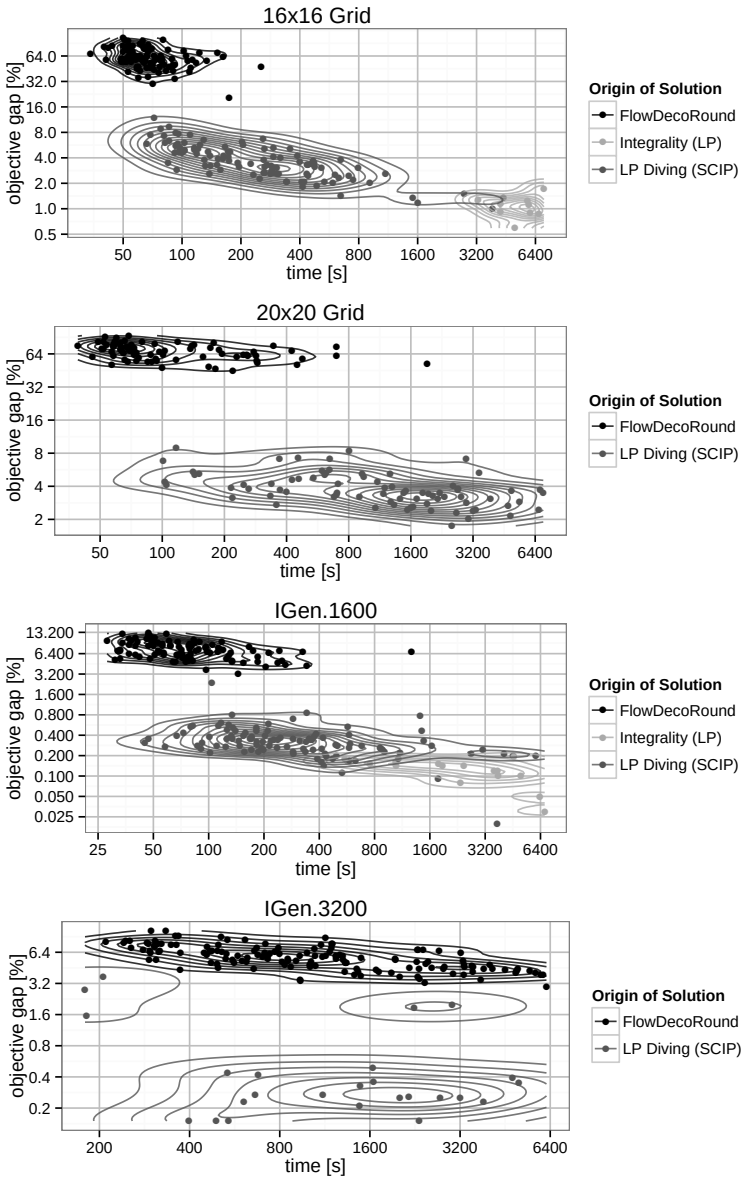


Fig. 8: Primal solutions found by our solver for $n \times n$ grids with $n = 16, 20$ and the IGen topologies IGen.1600 and IGen.3200 over time with a broad classification of the solution's origin. The depicted gap corresponds to the lower bound at the time the solution was found. Note the logarithmic x- and y-axis.

2. The heuristic FlowDecoRound presented in Section FlowDecoRound performs quite well on Internet topologies in which costs are not chosen uniformly.
3. $n \times n$ grid instances with uniform costs as presented in Section 6.1 seem to be hard to solve even for comparatively small sizes of $n = 16, 20$.

6.4 Comparison with MIP-A-CVSAP-MCF

Having presented the computational results for formulation IP-A-CVSAP in the above section, we will now examine the performance of MIP-A-CVSAP-MCF on 20×20 grid graph instances as well as on the smaller IGen.1600 instances. As already for IGen.1600 the formulation IP-A-CVSAP induces around one million binary variables for the IGen.1600 instances, we have chosen to use the commercial MIP solver CPLEX [8] instead of SCIP. To generate the instances for CPLEX we have modeled MIP-A-CVSAP-MCF in the GNU Mathematical Programming Language [14] and both the model files and the corresponding data files are available at [33]. We ran CPLEX with standard parameters making 12 GB of RAM available to it. As for the experiments running SCIP we limited the solution time to 2 hours. The logs of the experiments running CPLEX are also available at [33]. Note that CPLEX generally performs better than SCIP (see [20] for a comparison) in direct comparisons. The experiments are therefore a priori biased in favor of MIP-A-CVSAP-MCF.

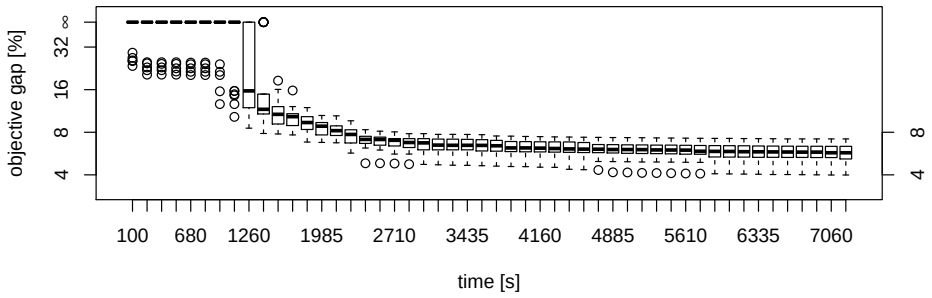


Fig. 9: Objective gap over time for the 20×20 grid instances using formulation MIP-A-CVSAP-MCF which is being solved by CPLEX. A gap of ∞ denotes that no primal solution has been found.

In Figure 9 the objective gap of the 20×20 grids using formulation MIP-A-CVSAP-MCF is shown over time. Compared to the corresponding plot for $n = 20$ in Figure 6, we notice that our approach yields solutions much quicker and that the final gap after 2 hours using IP-A-CVSAP lies approximately 3% below the gap found by CPLEX. CPLEX could only find solutions for 3 of the 25 instances of IGen.1600. We therefore do not provide a plot of the gap over time for these instances.

To computationally compare the strength of formulations IP-A-CVSAP and MIP-A-CVSAP-MCF, we will compare the lower bounds by the following metric.

Definition 6. *The relative improvement of formulation* A *over formulation* B *for minimization problems is defined as*

$$I^{\text{rel}}_{\text{dual}}(\mathsf{A},\mathsf{B}) = \frac{D_{\mathsf{A}} - D_{\mathsf{B}}}{P_{\text{best}} - D_{\mathsf{B}}} ,$$

where D_{A} *denotes the dual bound of* A*,* D_{B} *denotes the dual bound of* B *and* P_{best} *is the objective value of the best solution found overall.*

Note that $I^{\text{rel}}_{\text{dual}}(\mathsf{A},\mathsf{B}) = 1$ holds iff. $D_{\mathsf{A}} = P_{\text{best}}$ and therefore formulation A has found an optimal solution. Otherwise, $I^{\text{rel}}_{\text{dual}}(\mathsf{A},\mathsf{B})$ measures the improvement of the dual bound using formulation A compared to the absolute gap of formulation B. Furthermore note that this metric is independent of the process of determining primal solutions, since the best overall solution is chosen indepently of which formulation provided it and when it was found.

Figure 10 depicts $I^{\text{rel}}_{\text{dual}}$(IP-A-CVSAP, MIP-A-CVSAP-MCF) over time. As it takes CPLEX up to 1200 seconds to determine the root relaxation for IGen.1600 instances, we can only compare the relaxations after this point in time. Clearly, for both problem classes the relative improvement is substantial. We therefore conclude by stating that IP-A-CVSAP yields distinctively better lower bounds than MIP-A-CVSAP-MCF.

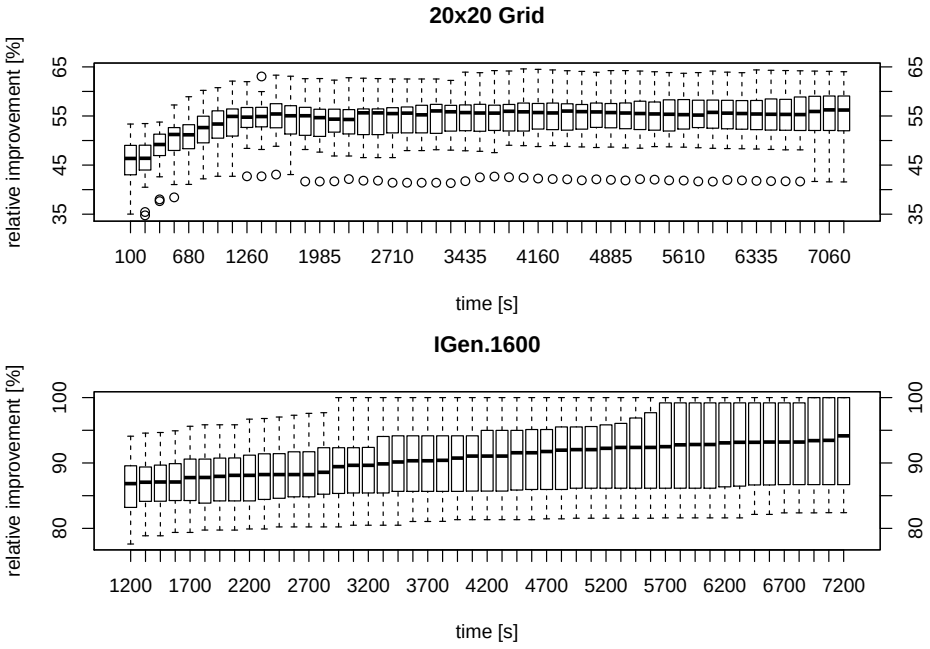


Fig. 10: $I^{\text{rel}}_{\text{dual}}$(IP-A-CVSAP, MIP-A-CVSAP-MCF) on IGen.1600 and 20×20 grid instances over time.

6.5 Analysis of Runtime Allocation

In this section we evaluate the computational effectiveness of our approach. To this end, we discuss the runtime allocation of different subroutines for experiments of the different problem classes to identify computational bottlenecks.

In Figure 11 the runtime allocation for the different problem classes is depicted according to the following classification:

- **B** Time spent in branching procedures. This represents the time needed for deciding which variables to branch on. This might, e.g. when strong-branching [1] is applied, be a computationally expensive subroutine.
- **H** Time spent in heuristics. While this includes the execution time of our heuristic FlowDecoRound its contribution is negligible compared to the time spent in diving heuristics. It must be noted that diving heuristics require the solving of linear relaxations and therefore trigger separation procedures. The runtime of separation procedures necessitated by heuristics are included in the runtime of the heuristics.
- **L** Time spent in solving linear relaxations.
- **S** Time spent in separation procedures to check the validity of IP-2 and IP-3* and to generate cuts where necessary.

The runtime allocation varies to a great extent between grid and Internet topology instances. Considering grid experiments, due to the rather small number of terminals and Steiner sites and as the graphs are comparatively small, the time spent in separation procedures is marginal. In fact, the time spent in solving linear relaxations and executing heuristics clearly dominates the runtime. In stark contrast, the runtime allocation for Internet topologies IGen.1600 and IGen.3200 shows the major influence of the separation procedures as the underlying gaphs are much larger and the number of terminals

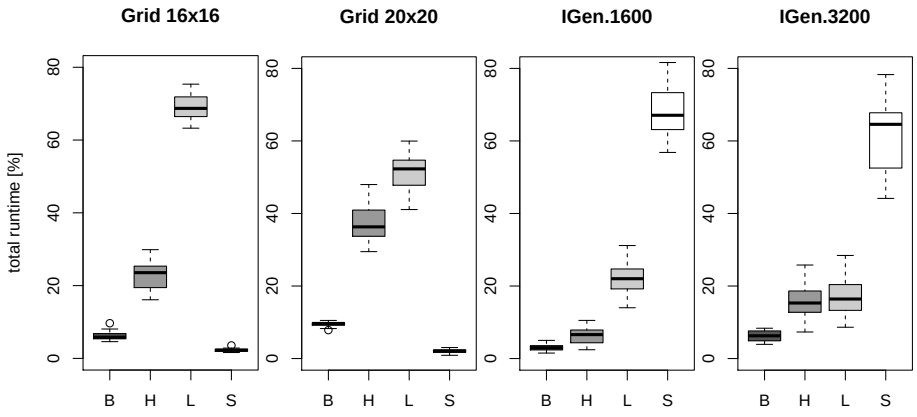


Fig. 11: Runtime allocation distinguished according to the following categories: execution time spent for branching (B), for heuristics (H), for solving of the linear relaxations (L) and the execution time needed for separating IP-2 and IP-3* (S).

and Steiner sites has increased by a magnitude. Note that for the larger instances, i.e. grids with $n = 20$ and IGen.3200, the time spent in heuristics and the time spent in solving linear relaxations almost equalizes.

The above observations allow the following two conclusions:

1. Our choice of using Edmonds-Karp to separate inequalities IP-2 and IP-3* limits performance for larger instances with many terminals and Steiner sites. It should therefore be either optimized or replaced by another algorithm, e.g. the one of Hao and Orlin (see [21] for a discussion).
2. While LP diving heuristics often find the best heuristic solutions (see Section 6.3), their overall runtime comes close to the time spent for solving LP relaxations. By devising more advanced combinatorial heuristics to replace these generic heuristics, performance could furthermore be improved.

7 Related Work

The CVSAP problem differs from many models studied in the context of IPTV [17], sensor networks [12,13], or fiber-optical transport [17], to just name a few, in that the number and placement of aggregation points is subject to optimization as well. The problem is complicated further by the fact that the communication between sender and receiver may be in-network processed *repeatedly*. The result from [24] on multi-constrained multicast routing also applies to CVSAP: any algorithm limited to (directed) acyclic graphs cannot solve the problem in general. Generally, while there exist many heuristic and approximate algorithms for related problem variants, we are the first to consider exact solutions.

The two closest models to CVSAP are studied in [35] and [26]. However, while [35] already showed the applicability of selecting only a few processing nodes for multicasting, no concise formalization is given and the described heuristics do not provide performance guarantees. In a series of publications, Oliviera and Pardalos consider the Flow Streaming Cache Placement Problem (FSCPP) [26]. Unfortunately, their FSCPP definition is inherently flawed as it does not guarantee connectivity see (Lemma 6 in Section A). Interestingly, the authors also provide a correct approximation algorithm, which however only considers the weak model which ignores traffic.

Other related problems and algorithms. The CVSAP is related to several classic problems. For example, CVSAP generalizes the *light-tree* concepts (e.g., [4]) in the sense that "light splitting" locations can be chosen depending on the repeatedly processed traffic; our approach can directly be used to optimally solve the light-tree problem. In the context of wave-length assignment, Park et al. [27] show that a small number of virtual splitters can be sufficient for efficient multicasting. Our formalism and the notion of hierarchy is based on the paper by Molnar [24] who studies the structure of the so-called multi-constrained multicast routing problem. Unlike in the CVSAP, an edge appears at most once in the solution. If the cost of in-network processing is zero and all nodes are possible Steiner sites, the CVSAP boils down to a classic Steiner Tree Problem [15] and its degree-bounded variants [22]. A closer look shows that CVSAP can

be easily modified to generalize the standard formulation of prize-collecting Steiner trees [18] where used edges entail costs, and visited nodes may come with a benefit. However, CVSAP does not directly generalize other STP variants where disconnected nodes yield penalties [18] or which need to support anycasts [10]. Lastly, CVSAP generalizes the standard facility location problem [16]. .

Mathematical programs. For a good overview of mathematical program optimization and avoiding inefficient relaxations that lead to high runtimes (e.g. [19]), we refer the reader to [3]. Our mathematical program builds upon the separation approach by Koch et al. [21] (see also [15]). Unlike [21], for CVSAP it is not sufficient to only compute the cuts from the senders, but also additional cuts are introduced depending on whether aggregation nodes are opened.

8 Conclusion

This paper presented a compact IP formulation based on a single-commodity flow to optimally solve the CVSAP problem. We rigorously proved that although the optimal IP solution may contain directed cyclic structures and flows may be merged repeatedly, there exists an algorithm to decompose the solution into individual routes. Since the CVSAP problem is related to several classical optimization problems, we believe that our approach is of interest beyond the specific model studied here.

Complementing our theoretical results, we have undertaken an extensive computational evaluation to study the performance of our solution approach. We have validated that the introduction of Directed Steiner Cuts, nested cuts and creep-flow can speed up computations significantly. Considering the generation of primal solutions, we have introduced a primal heuristic that finds feasible solutions far quicker than heuristics that are bundled together with SCIP. Our heuristic furthermore performs on the non-symmetric IGen instances quite well, yielding solutions with an objective gap below 14%. Lastly, our computational evaluation has demostrated that our algorithmic framework allows for solving realistically sized instances within an objective gap of 4%, while naive multi-commodity formulations perform significantly worse or cannot be used at all on large problem instances.

An interesting direction for future research regards the design of approximation algorithms as an efficient alternative to the rigorous optimization approach proposed in this paper. While in its general form CVSAP cannot be approximated, we believe that there exist good approximate solutions, e.g., for uncapacitated variants or bi-criteria models where capacities can be violated slightly.

References

1. T. Achterberg. SCIP: Solving Constraint Integer Programs. *Mathematical Programming Computation*, 1(1):1–41, 2009.
2. R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms and Applications*. Prentice Hall, 1993.

3. D. Bertsimas and R. Weismantel. *Optimization over Integers*. Dynamic Ideas, 2005.
4. Z. Cai, G. Lin, and G. Xue. Improved Approximation Algorithms for the Capacitated Multicast Routing Problem. In *Proc. 2nd International Conference on Combinatorial Optimization and Applications (COCOA)*, pages 136–145. Springer, 2005.
5. A. M. Costa, J.-F. Cordeau, and G. Laporte. Models and branch-and-cut algorithms for the steiner tree problem with revenues, budget and hop constraints. *Networks*, 53(2):141–159, 2009.
6. P. Costa, A. Donnelly, A. Rowstron, and G. O. Shea. Camdoop: Exploiting In-network Aggregation for Big Data Applications. In *Proc. USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2012.
7. P. Costa, M. Migliavacca, P. Pietzuch, and A. L. Wolf. NaaS: Network-as-a-Service in the Cloud. In *Proc. USENIX Hot-ICE Workshop*, 2012.
8. CPLEX. http://www.cplex.com/. In *Website*, 2012.
9. C. Cranor, T. Johnson, O. Spataschek, and V. Shkapenyuk. Gigascope: A Stream Database for Network Applications. In *Proc. ACM SIGMOD International Conference on Management of Data*, pages 647–651, 2003.
10. E. D. Demaine, M. Hajiaghayi, and P. N. Klein. Node-Weighted Steiner Tree and Group Steiner Tree in Planar Graphs. In *Proc. 36th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 328–340, 2009.
11. European Telecommunications Standards Institute. Network Functions Virtualisation - Introductory White Paper. *SDN and OpenFlow World Congress, Darmstadt-Germany*, 2012.
12. I. Eyal, I. Keidar, S. Patterson, and R. Rom. In-Network Analytics for Ubiquitous Sensing. In *Proc. International Symposium on Distributed Computing (DISC)*, 2013.
13. E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi. In-Network Aggregation Techniques for Wireless Sensor Networks: A Survey. *IEEE Wireless Communications*, 14:70–87, 2007.
14. GNU Linear Programming Kit. http://www.gnu.org/software/glpk/glpk.html. In *Website*, 2013.
15. M. X. Goemans and Y.-S. Myung. A catalog of Steiner tree formulations. *Networks*, 23(1):19–28, 1993.
16. S. Gollowitzer and I. Ljubić. MIP models for Connected Facility Location: A theoretical and computational study. *Computers & Operations Research*, 38(2):435–449, 2011.
17. C. Hermsmeyer, E. Hernandez-Valencia, D. Stoll, and O. Tamm. Ethernet aggregation and core network models for effcient and reliable IPTV services. *Bell Labs Technical Journal*, 12(1):57–76, 2007.
18. D. S. Johnson, M. Minkoff, and S. Phillips. The prize collecting Steiner tree problem: theory and practice. In *Proc. 11th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 760–769. Society for Industrial and Applied Mathematics, 2000.
19. D. Karget and M. Minkoff. Building steiner trees with incomplete global knowledge. In *Proc. 41st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 613–623, 2000.
20. T. Koch, T. Achterberg, E. Andersen, O. Bastert, T. Berthold, R. E. Bixby, E. Danna, G. Gamrath, A. M. Gleixner, S. Heinz, et al. Miplib 2010. *Mathematical Programming Computation*, 3(2):103–163, 2011.
21. T. Koch and A. Martin. Solving Steiner tree problems in graphs to optimality. *Networks*, 32(3):207–232, 1998.
22. Y. Lee, L. Lu, Y. Qiu, and F. Glover. Strong formulations and cutting planes for designing digital data service networks. *Telecommunication Systems*, 2(1):261–274, 1993.
23. A. Lucena and M. G. Resende. Strong lower bounds for the prize collecting Steiner problem in graphs. *Discrete Applied Mathematics*, 141(1):277–294, 2004.
24. M. Molnár. Hierarchies to Solve Constrained Connected Spanning Problems. Technical Report lrimm-00619806, University Montpellier 2, LIRMM, 2011.

25. S. Narayana, W. Jiang, J. Rexford, and M. Chiang. Joint Server Selection and Routing for Geo-Replicated Services. In *Proc. Workshop on Distributed Cloud Computing (DCC)*, 2013.
26. C. Oliveira and P. Pardalos. Streaming Cache Placement. In *Mathematical Aspects of Network Routing Optimization*, Springer Optimization and Its Applications, pages 117–133. Springer New York, 2011.
27. J.-W. Park, H. Lim, and J. Kim. Virtual-node-based multicast routing and wavelength assignment in sparse-splitting optical networks. *Photonic Network Communications*, 19(2):182–191, 2010.
28. T. Polzin and S. Vahdati Daneshmand. A comparison of steiner tree relaxations. *Discrete Applied Mathematics*, 112(1):241–261, 2001.
29. Z. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu. SIMPLE-fying Middlebox Policy Enforcement Using SDN. In *Proc. ACM SIGCOMM*, 2013.
30. B. Quoitin, V. Van den Schrieck, P. Franois, and O. Bonaventure. IGen: Generation of router-level Internet topologies through network design heuristics. In *Proc. 21st International Teletraffic Congress (ITC)*, pages 1–8, 2009.
31. P. Raghavan and C. D. Thompson. Provably good routing in graphs: regular arrays. In *Proc. 17th Annual ACM Symposium on Theory of Computing (STOC)*, pages 79–87, New York, NY, USA, 1985. ACM.
32. I. Rosseti, M. P. de Aragão, C. C. Ribeiro, E. Uchoa, and R. F. Werneck. New benchmark instances for the steiner problem in graphs. In *Metaheuristics*, pages 601–614. Kluwer Academic Publishers, 2004.
33. M. Rost and S. Schmid. *CVSAP-Project Website*. http://www.net.t-labs.tu-berlin.de/~stefan/cvsap.html, 2013.
34. A. Schrijver. *Theory of linear and integer programming*. Wiley, 1998.
35. S. Shi. A Proposal for A Scalable Internet Multicast Architecture. Technical Report WUCS-01-03, Washington University, 2001.
36. S. Voß. *Handbook of optimization in telecommunications*, chapter 18. Spinger Science + Business Media, New York, 2006.
37. R. Wunderling. *Paralleler und objektorientierter Simplex-Algorithmus*. PhD thesis, Technische Universität Berlin, 1996.
38. Z. Zhang, M. Zhang, A. Greenberg, Y. C. Hu, R. Mahajan, and B. Christian. Optimizing cost and performance in online service provider networks. In *Proc. 7th USENIX Conference on Networked Systems Design and Implementation (NSDI)*, 2010.

A Deferred Lemmas

Lemma 5. *The directed Steiner cuts (see IP-3*) can strengthen the formulation IP-A-CVSAP, i.e. improve the objective value of its LP relaxation.*

Proof. Consider the simple example in which the whole network $G = (V_G, E_G)$ consists only of three nodes on a line: $V_G = \{r, s, t\}$ and $E_G = \{(t, s), (s, r)\}$. We consider the following capacities and costs $u_S(s) = 10$, $u_r(r) = 1$, $c_E(t, s) = c_E(s, r) = 1$, $c_S(s) = 5$ and that r is the root, s is the single Steiner location and t the only terminal. The optimal solution of IP-A-CVSAP without IP-3* and relaxing the constraints IP-11 and IP-10 to $f \in \mathbb{R}_{\geq 0}^{E_{\text{ext}}}$ and $x \in [0, 1]^S$ is $f(t, s) = 1$, $x_s = 1/10$ and $f(s, r) = 1/10$ yielding an objective value of $5 \cdot x_s + f(t, s) + f(s, r) = 1.6$ By introducing Constraint IP-3* $f(s, r)$ must equal 1 and therefore, the solution obtained by introducing

this constraint yields the integral solution $f(t,s) = f(s,r) = 1$ and $x_s = 0$ with objective value 2, therefore strengthening the model. Note that we did not give values for the edges from and to the super sinks which are introduced in E_{ext} as these do not influence the objective value. □

Lemma 6. *The formulation for the Flow Cache Placement Problem (FCCP) given in [26] is incorrect, as it does not ensure connectivity as claimed by the authors.*

Proof. In a series of works, the latest being [26], Oliviera and Pardalos have introduced the Tree Streaming Cache Placement Problem (TSCPP) and the Flow Streaming Cache Placement Problem (FSCPP). The TSCPP can directly be understood as M-CVSAP in which flow must be routed along a tree. Using our notation, they require $G^f_{E_{\text{ext}}}$, which included each used edge, to be a tree.

They introduce FSCPP as a generalization of TSCPP in which the constraint that $G^f_{E_{\text{ext}}}$ must be a tree is simply removed. Their Integer Program therefore reduces to IP-A-CVSAP in which IP-2 (and IP-3*) are removed. However, it is easy to see that $G^f_{E_{\text{ext}}}$ might indeed be disconnected such that there exist caches (activated Steiner nodes) that do not receive any flow and we only informally describe a counterexample. Consider a graph with one source (i.e., root), one possible cache (a Steiner site) and two demand nodes (terminals). Then if one terminal sends flow to the root and the other sends flow to the Steiner node such that their paths do not cross, then all constraints of IP-A-CVSAP (except IP-2 and IP-3*) are satisfied (assuming appropriate capacities) and one of the terminals is not connected to the root. This contradicts the connectivity requirement. □