VirtuCast: Multicast and Aggregation with In-Network Processing

An Exact Single-Commodity Algorithm

Matthias Rost and Stefan Schmid

Telekom Innovation Laboratories (T-Labs) & TU Berlin, Germany
{mrost,stefan}@net.t-labs.tu-berlin.de

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 $NP \subseteq P$), 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 significantly. We rigorously prove VirtuCast's correctness and show its applicability to solve realistically sized instances close to optimality.

Keywords: Network Virtualization, Network Functions Virtualization, Multicast, In-Network Aggregation, Data-Center, Middleboxes, ISP, Integer Programming.

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, 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 [14] over sensor networks [10,11] to fiber-optical transport [14].

This paper is motivated by the virtualization trend in today's Internet, and in particular by network (functions) virtualization [9] and software-defined networking, e.g.,

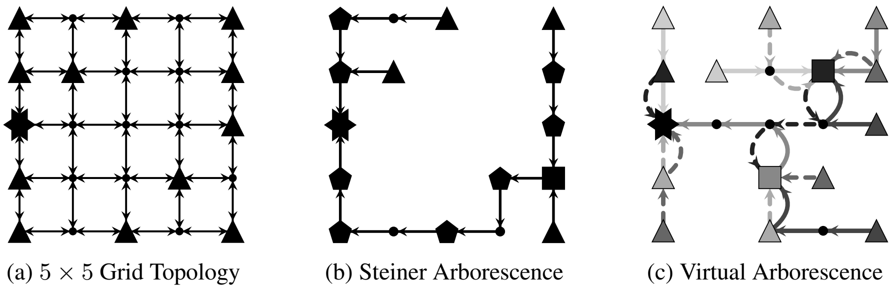

Fig. 1. An aggregation example on a 5×5 grid. Terminals are depicted as triangles and the receiver as a star. The terminals must establish a route towards the receiver, while multiple data streams may be aggregated by activated processing locations. Such processing locations are pictured as squares or,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

OpenFlow [23]. 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 [5], in universal nodes, or in distributed (micro-)datacenters in the wide-area network [7,20,29] can be used for in-network processing, e.g., to reduce traffic during the MapReduce shuffle phase [6]. Such distributed resource networks open new opportunities on how services can be deployed. Especially in the context of aggregation and multicasting 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 [28] are no longer applicable [19]. 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 single receiver (root), 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: the optimal solution, depicted in Figure 1b, uses 16 edges and 9 processing locations, i.e. nodes where data flows are merged. 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 overall: 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. We introduce 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 $NP \subseteq P$ holds 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, speeding up the branch-and-bound algorithm (see [26] for an in-depth discussion).

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 [19], forbidding directed acyclic graphs (DAGs) may yield suboptimal solutions only. Rather, we allow for the iterative processing of flows, such that processing nodes may be connected to other processing nodes.

To implement VirtuCast, we have developed a branch-and-cut framework, including a primal heuristic. Due to space constraints, we refer the interested reader to our technical report [26] for a detailed discussion of the implementation as well as for the full computational evaluation.

Overview. We formally introduce CVSAP and show its inapproximability in Section 2. We continue by presenting our VirtuCast algorithm in Section 3. In Section 4 we shortly outline the results of our computational study. We conclude this paper with summarizing related work in Section 5.

2 The Constrained Virtual Steiner Arborescence Problem

The Constrained Virtual Steiner Arborescence Problem (CVSAP) generalizes several in-network processing problems related with multicasting and aggregation of data where 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 [28], our model distinguishes between nodes that merely relay 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 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*. As will be discussed at the end of this section, the multicast case can be easily reduced to the aggregation case. Hence, in the following we only introduce the variant of CVSAP in which data flows are directed towards the root. 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 of $\mathcal{P}$ that contains 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 denote the set of outgoing and incoming edges, restricted on a subset $F \subseteq E_G$ in G, by $\delta^+_F(v) = \{(v,u) \in F\}$ and $\delta^-_F(v) = \{(u,v) \in F\}$ for $v \in V_G$. We naturally extend this definition to sets: $\delta^+_F(W) = \{(v,u) \in F | v \in W, u \notin W\}$ and $\delta^-_F(W) = \{(u,v) \in F | v \in W, u \notin W\}$ respectively. We abridge $\mathsf{f}((\mathsf{y},\mathsf{z}))$ to $\mathsf{f}(\mathsf{y},\mathsf{z})$ for functions defined on tuples.

Formal Problem Statement. We model the physical infrastructure as capacitated, directed network $G = (V_G, E_G, c_E, u_E)$, where $u_E : E_G \to \mathbb{N}$ defines integral capacities and $c_E : E_G \to \mathbb{R}^+$ defines real-valued, positive costs on the edges. 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 processing nodes may be *activated*. Such Steiner sites are attributed with a positive cost $c_S : S \to \mathbb{R}^+$ that is incurred upon using it, 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 $c_E(v_T, v) = 0$ and $u_E(v_T, v) = 1$.

In the aggregation scenario considered henceforth the terminals hold data that needs to be forwarded to the root (the single receiver) while data may be aggregated at active Steiner nodes. The capacities on the Steiner sites (and on the root) limit the number of flows that can be *actively* processed: any number of incoming flows less than or equal to $u_S(s)$ can be merged into a single flow by $s \in S$ upon activation. To model both routing decisions and paths taken, we introduce the concept of Virtual Arborescences:

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 virtual arborescence on a simple directed path $P \in \mathcal{P}_G$ such that*

(VA-1) $(V_{\mathcal{T}}, E_{\mathcal{T}}, r)$ *is an arborescence rooted at r with edges directed towards r,*
(VA-2) *for all* $(u, v) \in E_{\mathcal{T}}$ *the directed path* $\pi(u, v)$ *connects u to v in G.* □

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: in Figure 1c equally colored and dashed paths represent edges of the Virtual Arborescence. 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. Using the concept of Virtual Arborescences, 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 terminals* $t \in T$ *holds* $\delta^+_{E_{\mathcal{T}}}(t) = 1$,
(CVSAP-3) *for the root* $\delta^-_{E_{\mathcal{T}}}(r) \leq u_r$ *holds,*
(CVSAP-4) *for all activated Steiner sites* $s \in S \cap V_{\mathcal{T}}$ *holds* $\delta^-_{E_{\mathcal{T}}}(s) \leq u_S(s)$ *and*
(CVSAP-5) *for all edges* $e \in E_G$ *holds* $|\left(\pi(E_{\mathcal{T}})\right)[e]| \leq u_E(e)$.

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*

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

where $|\left(\pi(E_{\mathcal{T}})\right)[e]|$ *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.

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. □

Similarly to the above definitions, CVSAP can be defined for multicasting applications in which the task is to distribute a single data item from the root (single sender) to terminals (receivers) via processing nodes (with routing capability) that may duplicate the data and route it to several different destinations. To obtain a formal definition for this scenario, edges in the VA must be oriented away from the root and $\delta^-(\cdot)$ must be replaced by $\delta^+(\cdot)$ and vice versa in Definition 2. Subject to this slight adaption, the root and active Steiner nodes can reproduce an incoming stream, such that terminals must receive this stream. By essentially reversing the direction of edges, the multicasting version of CVSAP can be reduced to the aggregation version presented above.

3 VirtuCast Algorithm

In this section we present the Algorithm VirtuCast to solve CVSAP. VirtuCast first computes a solution for a single-commodity flow Integer Programming formulation and then constructs the corresponding Virtual Arborescence. Even though our IP formulation can be used to compute the optimal solution for any CVSAP instance, feasible solutions to our IP formulation already yield feasible solutions to CVSAP. This allows to derive near-optimal solutions *during* the solution process. Our single-commodity approach improves dramatically upon naive multi-commodity flow formulations and enables us to solve realistically sized instances in the first place (see [26] for a discussion).

3.1 IP Formulation

Our IP (see IP-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 3). While o_r^- may only receive flow from the root r, all possible Steiner sites $s \in S$ are connected 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 ensure connectivity.

Definition 3 (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 $E_{\text{ext}}^{R} \triangleq E_{\text{ext}} \setminus E_{\text{ext}}^{S^-}$, such that edges towards o_S^- are excluded in E_{ext}^{R}. □

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 a Steiner site. 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 $G^f_\text{ext} \triangleq (V^f_\text{ext}, V^f_\text{ext})$ with $V^f_\text{ext} \triangleq V_\text{ext}$ and $V^f_\text{ext} \triangleq \{e | e \in E_\text{ext} \wedge f(e) \geq 1\}$.

The IP Model. The IP formulation IP-CVSAP uses an integral single-commodity flow. We define flow variables $f_e \in \mathbb{Z}_{\geq 0}$ 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.

The binary variable $x_s \in \{0,1\}$ (see IP-10) decides, whether a Steiner site $s \in S$ is activated. By Constraint IP-8, each terminal $t \in T$ is forced to send a single unit of flow, as flow conservation is enforced on all original nodes $v \in V_G$ (see IP-1). Therefore, 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).

As the definition of 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* IP-2 [18] and *Directed Steiner Cuts* IP-3* [16]. Our connectivity inequalities (see IP-2) state that each set of nodes containing a Steiner site $s \in S$ must emit at

Integer Program IP-CVSAP

$$\begin{array}{llrr}
\text{minimize} & C_\text{IP}(x,f) = \sum\limits_{e\in E_G} \mathbf{c}_e f_e + \sum\limits_{s\in S} \mathbf{c}_s x_s & & \text{(IP-OBJ)}\\
\text{subject to} & f(\delta^+_{E_\text{ext}}(v)) = f(\delta^-_{E_\text{ext}}(v)) & \forall\, v \in V_G & \text{(IP-1)}\\
& f(\delta^+_{E^R_\text{ext}}(W)) \geq x_s & \forall\, W \subseteq V_G, s \in W \cap S \neq \emptyset & \text{(IP-2)}\\
& f(\delta^+_{E^R_\text{ext}}(W)) \geq 1 & \forall\, W \subseteq V_G, T \cap W \neq \emptyset & \text{(IP-3}^\star\text{)}\\
& f_e \geq x_s & \forall\, e = (s, \mathrm{o}^-_S) \in E^{S-}_\text{ext} & \text{(IP-4}^\star\text{)}\\
& f_e \leq \mathbf{u}_s x_s & \forall\, e = (s, \mathrm{o}^-_S) \in E^{S-}_\text{ext} & \text{(IP-5)}\\
& f_{(r,\mathrm{o}^-_r)} \leq \mathbf{u}_r & & \text{(IP-6)}\\
& f_e \leq \mathbf{u}_e & \forall\, e \in E_G & \text{(IP-7)}\\
& f_e = 1 & \forall\, e \in E^{T+}_\text{ext} & \text{(IP-8)}\\
& f_e = x_s & \forall\, e = (\mathrm{o}^+, s) \in E^{S+}_\text{ext} & \text{(IP-9)}\\
& x_s \in \{0,1\} & \forall\, s \in S & \text{(IP-10)}\\
& f_e \in \mathbb{Z}_{\geq 0} & \forall\, e \in E_\text{ext} & \text{(IP-11)}
\end{array}$$

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, Constraint IP-3* enforces that there exists a path from each terminal $t \in T$ towards r in G^f_{ext}. The directed Steiner cuts constitute valid inequalities which are implied by IP-1 and IP-2 (see [26] for the proof). However, these cuts can strengthen the model by improving the LP-relaxation during the branch-and-cut process. As discussed in [26], including these constraints substantially improved the quality of lower bounds in our computational evaluation. 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 a non-negative cost. We introduce this constraint here, as it specifies a condition needed in the proof of correctness 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 $\mathcal{F}_{\text{IP}} = \{(x, f) \in \{0,1\}^S \times \mathbb{Z}^{E_{\text{ext}}}_{\geq 0} | \text{IP-1 - IP-11}\}$ the set of feasible solutions to IP-CVSAP.

3.2 Decomposition Algorithm

Given a feasible solution $(\hat{x}, \hat{f}) \in \mathcal{F}_{\text{IP}}$ for IP-CVSAP, Algorithm Decompose constructs a feasible solution $\hat{\mathcal{T}}_G \in \mathcal{F}_{\text{CVSAP}}$ 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 one of the super sinks o^-_S or o^-_r and reduces flow along the found paths to yield a solution to a subproblem. However, as IP-CVSAP does not pose a simple flow problem, we constantly need to ensure that Connectivity Inequalities IP-2 hold after removing flow in $G^{\hat{f}}_{\text{ext}}$. We first present Algorithm Decompose in more detail and then prove its correctness.

Synopsis of Algorithm. Algorithm Decompose 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 $\{s \in S | x_s = 1\}$.

Unconnected terminals in $\hat{T}$ are connected iteratively. For an unconnected terminal $t \in \hat{T}$ the path generation procedure from Line 6 to 14 computes a path P from o^+ via t to o^-_S or o^-_r. If the path P terminates in o^-_r then t is connected to r. Otherwise, if P terminates in o^-_S, then the second last node of P is an active Steiner node and t is connected to it (see Line 18). During the path generation procedure the flow variables $\hat{f}$ are decremented. If the second last node of P was indeed an active Steiner node $s \in \hat{S}$ and s does not forward any flow towards o^-_S anymore, s itself is added to the set of unconnected terminals (see Line 16). Note that in Line 18 the (virtual) edge $(t, P_{|P|-1})$

Algorithm Decompose

```
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-CVSAP
Output: Feasible Virtual Arborescence $\hat{\mathcal{T}}_G$ for CVSAP
set $\hat{S} \triangleq \{s \in S | x_s \geq 1\}$ and $\hat{T} \triangleq T$
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 $\hat{\pi} : \hat{E}_\mathcal{T} \to \mathcal{P}_G$
while $\hat{T} \neq \emptyset$ do
    let $t \in \hat{T}$ and $\hat{T} \leftarrow \hat{T} - t$
    choose $P \triangleq \langle \text{o}^+, t, \dots, \text{o}_r^- \rangle \in G_{\text{ext}}^{\hat{f}}$
    for $j = 1$ to $|P| - 1$ do
        set $\hat{f}(P_j, P_{j+1}) \leftarrow \hat{f}(P_j, P_{j+1}) - 1$
        if Constraint IP-2 is violated with respect to $\hat{f}$ and $\hat{S}$ then
            choose $W \subseteq V_G$ such that $W \cap \hat{S} \neq \emptyset$ and $\hat{f}(\delta^+_{E^R_{\text{ext}}}(W)) = 0$
            choose $P' \triangleq \langle P_j, \dots, \text{o}_S^- \rangle \in G_{\text{ext}}^{\hat{f}}$ such that $P_i \in W$ for $1 \leq i < m$
            set $\hat{f}(P_j, P_{j+1}) \leftarrow \hat{f}(P_j, P_{j+1}) + 1$ and $\hat{f}(P'_1, P'_2) \leftarrow \hat{f}(P'_1, P'_2) - 1$
            set $P \leftarrow \langle P_1, \dots, P_{j-1}, P_j = P'_1, P'_2, \dots, P'_m \rangle$
        end
    end
    if $P_{|P|} = \text{o}_S^-$ and $\hat{f}(P_{|P|-1}, P_{|P|}) = 0$ then
        set $\hat{S} \leftarrow \hat{S} - P_{|P|-1}$ and $\hat{x}(P_{|P|-1}) \leftarrow 0$ and $\hat{T} \leftarrow \hat{T} + P_{|P|-1}$
    end
    set $\hat{E}_\mathcal{T} \leftarrow \hat{E}_\mathcal{T} + (t, P_{|P|-1})$ and $\hat{\pi}(t, P_{|P|-1}) \triangleq$ simplify($\langle P_2, \dots, P_{|P|-1} \rangle$)
end
```

is added to $\hat{E}_\mathcal{T}$ and $\hat{\pi}(t, P_{|P|-1})$ is set accordingly to the truncated path P, where any cycles are removed (function `simplify`).

Proof of Correctness. We will now prove the correctness of Algorithm Decompose, thereby showing that IP-CVSAP can be used to compute (optimal) solutions to CVSAP. Our proof relies on an inductive argument similar to the one used for proving the existence of flow decompositions (see [2]): we assume that all constraints of IP-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-CVSAP, in which t has been removed from the set of terminals (see Theorem 2 below). During the course of this induction, the well-definedness of the **choose** operations is shown. As the complete proof is included in [26], we allow us to mainly sketch the proofs.

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

To prove the above theorem, we use the following Lemmas 1 through 3 of which we only prove the essential third one; the proofs for Lemmas 1 and 2 are included in [26]. Lemma 1 shows the well-definedness of choosing the path in Line 5 and is easy to check. Lemma 2 states that flow conservation (IP-1) holds during the execution of Decompose except at node P_{j+1} at which the outgoing flow exceeds the incoming flow by exactly one unit.

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.*

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.* $x = y$.

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

Proof Sketch. We only need 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 $\hat{S} \cap W \neq \emptyset$. Our proof relies on 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 $P'' = \langle s, \ldots, P_j, P_{j+1}, \ldots, \mathrm{o}_r^- \rangle \in G_{\text{ext}}^{\hat{f}}$ for $s \in \hat{S} \cap W$.
(d) There exists a path $P' = \langle P_j, \ldots, \mathrm{o}_S^- \rangle$ with $P'_i \in W$ for $1 \leq i < |P'|$ in $G_{\text{ext}}^{\hat{f}}$ after reduction of flow.

To see that statement (a) holds, consider the following. Before the reduction of flow on (P_j, P_{j+1}) all inequalities IP-2 held. For the expression $\hat{f}(\delta^+_{E_{\text{ext}}}(W)) = 0$ to hold after reduction of flow, the edge (P_j, P_{j+1}) must be contained in $\delta^+_{E_{\text{ext}}}(W)$.

The correctness of (b) directly follows from (a), as $(P_j, P_{j+1}) \in \delta^+_{E_{\text{ext}}}(W)$ holds.

We now prove statement (c). As connectivity inequalities IP-2 have held *before* the flow reduction in Line 7, for each activated Steiner node $s \in \hat{S} \cap W$ there existed a path from s to o_r^- in $G_{\text{ext}}^{\hat{f}}$. By (b), (P_j, P_{j+1}) was the only edge in $G_{\text{ext}}^{\hat{f}}$ leaving W before reduction of flow. Therefore a path as claimed in (c) must have existed before reduction of flow.

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

To see that the main statement of this lemma holds, consider the case that after Line 11 any connectivity inequality of IP-2 is violated. Let $W' \subseteq V_G$ with $W' \cap \hat{S} \neq \emptyset$ be a violated node set 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 three, after having reverted the flow reduction along (P_j, P_{j+1}), the path $\langle P_j, P_{j+1}, \ldots, \mathrm{o}_r^- \rangle$ was re-established in $G_{\text{ext}}^{\hat{f}}$. As flow along any of the edges contained in this path is greater or equal to one, W' cannot possibly violate IP-2. Therefore all Connectivity Inequalities IP-2 hold. □

Using the above lemmas, we now outline the proof of Theorem 2. By Lemmas 1 and 3 the algorithm is well-defined. Lemma 2 implies that flow preservation holds at Line 18 as node P_{j+1} is one of the super sinks. Lemma 3 directly ensures that connectivity constraints IP-2 hold. As capacity related constraints trivially hold as flow was only decreased, it only remains to check that placing a former active Steiner node into the set of terminals in Line 16 does not violate the terminal related constraint IP-8. This however is easy to check as constraint IP-9 ensured that this node received one unit of flow from the super source.

Using Theorem 2 it is easy to check that Algorithm Decompose terminates: Since no constraint is violated during the execution of the path generation and as flow is only reduced, the inner loop must eventually terminate

Theorem 3. *Algorithm Decompose terminates.*

We can now turn to proving that Algorithm Decompose indeed constructs a feasible solution for CVSAP. As the proof is of a rather technical nature, we again only sketch the proof and refer the interested reader to [26] for the complete argument.

Theorem 4. *Algorithm Decompose constructs a feasible solution $\hat{\mathcal{T}}_G \in \mathcal{F}_{\text{CVSAP}}$ for 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.*

By Theorem 3 the algorithm terminates such that we only need to check feasibility of the solution. First note that, as the VA $\hat{\mathcal{T}}_G$ is constructed using only resources accounted for in IP-CVSAP, $C_{\text{CVSAP}}(\hat{\mathcal{T}}_G) \leq C_{\text{IP}}(\hat{x}, \hat{f})$ must hold. Clearly, as capacity constraints of CVSAP are modeled explicitly in IP-CVSAP, it must only be checked whether indeed $\hat{\mathcal{T}}_G$ is a Virtual Arborescence. For proving that, first note that at the end of executing Decompose both sets $\hat{S}$ and $\hat{T}$ are empty and therefore all terminals and active Steiner nodes have been connected. While this holds by definition of the outer loop for $\hat{T}$, proving $\hat{S} = \emptyset$ requires the following argument. Assume that $\hat{S} \neq \emptyset$ while $\hat{T} = \emptyset$. By constraint IP-9 each active Steiner node $s \in \hat{S}$ receives one unit of flow from the super source. On the other hand constraint IP-4* safeguards that each active Steiner node absorbs at least a single unit of flow. This implies that no flow reaches o_r^-, violateing Constraint IP-2. Lastly, to check that $\hat{\mathcal{T}}_G$ defines an arborescence note that the order in which terminals (or former active Steiner sites) are removed from $\hat{T}$ defines a topological order on $(\hat{V}_{\mathcal{T}}, \hat{E}_{\mathcal{T}}, r)$.

To prove that formulation IP-CVSAP indeed computes optimal solutions, we need the following technical lemma showing that each solution to CVSAP can be mapped on a solution of IP-CVSAP with equal cost. As this mapping is straightforward, we refer the reader to [26] for the construction and only state the following lemma.

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 CVSAP. There exists a solution $(\hat{x}, \hat{f}) \in \mathcal{F}_{\text{IP}}$ with $C_{\text{CVSAP}}(\hat{\mathcal{T}}_G) = C_{\text{IP}}(\hat{x}, \hat{f})$.*

The above lemma fills the last gap in our proof to show that algorithm VirtuCast, which first computes an optimal solution to IP-CVSAP and then constructs a corresponding Virtual Arborescence using Decompose, solves CVSAP to optimality.

Theorem 5. *Algorithm VirtuCast solves CVSAP to optimality.*

We conclude this section with stating that each **choose** operation in Algorithm Decompose and checking whether connectivity inequalities IP-2 hold can be implemented using depth-first search. Implementing Decompose in this way and assuming that an optimal solution for IP-CVSAP is given and that G does not contain zero-cost cycles, we can bound the runtime of Decompose by $\mathcal{O}\left(|V_G|^2 \cdot |E_G| \cdot (|V_G| + |E_G|)\right)$ [26].

4 Computational Evaluation

We have implemented VirtuCast using SCIP [1] as underlying branch-and-cut framework. As the separation procedures employed to enforce Constraints IP-2 and IP-3* are well-known [16], we only shortly outline the results of our computational evaluation. A detailed discussion of all our results can be found in [26]. Furthermore, our solver as well as the test instances are obtainable from [25]. 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.

In our computational evaluation we consider two complementary graph topologies with varying sizes: symmetric $n \times n$ grid graphs and ISP topologies generated by IGen [24]. We report only on results obtained for the largest topology sizes, namely on a 20×20 grid and an IGen topology with 3200 nodes (further refered to by IGen.3200). The IGen.3200 topology is created by populating a world map with 3200 nodes, applying a local clustering and then connecting these clusters, yielding 19410 edges.

For each of the both test sets we generated 25 instances according to the following parameters. The receiver as well as the Steiner sites and the terminals are picked uniformly at random. For the grid instances we selected 80 Steiner sites and 100 terminals. For the IGen.3200 topology we chose 400 Steiner sites and 600 terminals. Common to both test sets, we set the edge capacities to 3 and the capacity of Steiner sites and the root to 5. On the grid topology, we set edge costs to 1 and activation costs for Steiner sites to 20. For IGen.3200 instances, edge costs are defined by the euclidean distance and activation costs are distributed uniformly according to $\mu(c_E) \cdot \mathcal{U}(25, 75)$, where $\mu(c_E)$ denotes the average edge length.

Figure 2 shows the objective gap, i.e. the relative quality guarantee, over time for both test sets, consisting of 25 instances each. Independent of the test set, the objective gap stabilizes after one hour of computation. For the highly symmetric grid instances as well as for the IGen.3200 instances, a median gap of less than 4% is achieved. As documented in [26], the lower bound improves by less than 2% for IGen.3200 instances and by less than 12% for grid instances. Hence, the lower bounds obtained initially are already reasonably accurate and the progress of the objective gap (cf. Figure 2) is driven by the quality of the solutions found. Based on this observation, we have implemented a primal heuristic to generate feasible solutions based on the linear relaxations during the branch-and-bound search (see [26]).

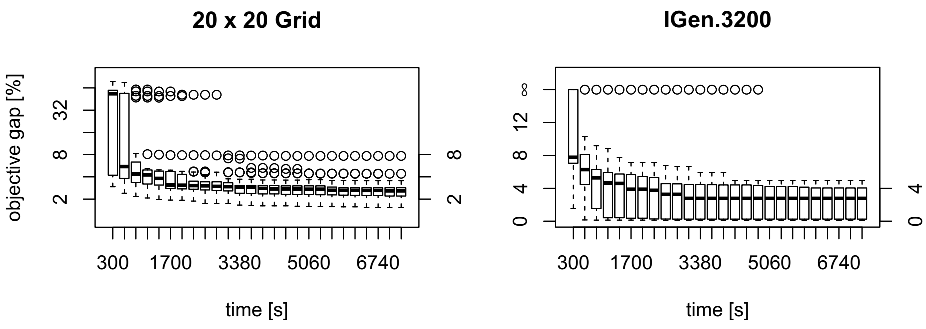

Fig. 2. Objective gap over time for the 20 × 20 grid and IGen.3200 test sets. Note the logarithmic y-axis for the grid instances. A gap of ∞ indicates that no primal solution has been found.

5 Related Work

The CVSAP problem differs from many models studied in the context of IPTV [14], sensor networks [10,11], fiber-optical transport [14], or Active Networking [3], to just name a few, in that the number and placement of processing locations is subject to optimization as well. The problem is complicated further by the fact that the communication between sender and receiver may be processed *repeatedly* within the network. The result from [19] 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 [21] and [27]. While [27] already showed the applicability of selecting only a few processing nodes for multicasting, no concise formalization is given and the described heuristic does not provide performance guarantees. In a series of publications, Oliviera and Pardalos consider the Flow Streaming Cache Placement Problem (FSCPP) [21]. Unfortunately, their FSCPP definition is inherently flawed as it does not guarantee connectivity (see [26] for a discussion). Interestingly, the authors also provide a correct approximation algorithm, which however only considers the rather 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 [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. [22] 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 [19] who studies the structure of the so-called multi-constrained multicast routing problem. However, unlike in CVSAP, an edge may be only used 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 the classic Steiner Tree Problem [12] and its degree-bounded variants [17]. A closer look shows that CVSAP can be easily modified to generalize the standard formulation of prize-collecting Steiner trees [15] where used edges entail costs, and visited nodes may come with a benefit. However, CVSAP does not generalize other STP variants where disconnected nodes yield penalties [15] or which need to support anycasts [8]. Lastly, CVSAP generalizes the standard facility location problem [13].

6 Conclusion

This paper presented VirtuCast to optimally solve CVSAP. 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. Using VirtuCast, we solved realistically sized instances to within 4% of optimality. Since CVSAP is related to several classical optimization problems, we believe that our approach is of interest beyond the specific model studied here.

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 may be violated slightly.

Acknowledgement. We would like to thank Marten Schönherr from Deutsche Telekom. This research was supported by the EU projects BigFoot and UNIFY.

References

1. Achterberg, T.: SCIP: Solving Constraint Integer Programs. Mathematical Programming Computation 1(1), 1–41 (2009)
2. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network Flows: Theory, Algorithms and Applications. Prentice Hall (1993)
3. Banchs, A., Effelsberg, W., Tschudin, C., Turau, V.: Multicasting Multimedia Streams with Active Networks. In: Proc. Local Computer Network Conference (LCN). IEEE (1998)
4. Cai, Z., Lin, G., Xue, G.: Improved Approximation Algorithms for the Capacitated Multicast Routing Problem. In: Wang, L. (ed.) COCOON 2005. LNCS, vol. 3595, pp. 136–145. Springer, Heidelberg (2005)
5. Costa, P., Donnelly, A., Rowstron, A., Shea, G.O.: Camdoop: Exploiting In-network Aggregation for Big Data Applications. In: Proc. USENIX Symposium on Networked Systems Design and Implementation (NSDI) (2012)
6. Costa, P., Migliavacca, M., Pietzuch, P., Wolf, A.L.: NaaS: Network-as-a-Service in the Cloud. In: Proc. USENIX Hot-ICE Workshop (2012)
7. Cranor, C., Johnson, T., Spataschek, O., Shkapenyuk, V.: Gigascope: A Stream Database for Network Applications. In: Proc. ACM SIGMOD International Conference on Management of Data, pp. 647–651 (2003)
8. Demaine, E.D., Hajiaghayi, M., Klein, P.N.: Node-weighted steiner tree and group steiner tree in planar graphs. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S., Thomas, W. (eds.) ICALP 2009, Part I. LNCS, vol. 5555, pp. 328–340. Springer, Heidelberg (2009)

9. European Telecommunications Standards Institute. Network Functions Virtualisation - Introductory White Paper. SDN and OpenFlow World Congress, Darmstadt-Germany (2012)
10. Eyal, I., Keidar, I., Patterson, S., Rom, R.: In-Network Analytics for Ubiquitous Sensing. In: Afek, Y. (ed.) DISC 2013. LNCS, vol. 8205, pp. 507–521. Springer, Heidelberg (2013)
11. Fasolo, E., Rossi, M., Widmer, J., Zorzi, M.: In-Network Aggregation Techniques for Wireless Sensor Networks: A Survey. IEEE Wireless Communications 14, 70–87 (2007)
12. Goemans, M.X., Myung, Y.-S.: A catalog of Steiner tree formulations. Networks 23(1), 19–28 (1993)
13. Gollowitzer, S., Ljubić, I.: MIP models for Connected Facility Location: A theoretical and computational study. Computers & Operations Research 38(2), 435–449 (2011)
14. Hermsmeyer, C., Hernandez-Valencia, E., Stoll, D., Tamm, O.: Ethernet aggregation and core network models for effcient and reliable IPTV services. Bell Labs Technical Journal 12(1), 57–76 (2007)
15. Johnson, D.S., Minkoff, M., Phillips, S.: The prize collecting Steiner tree problem: theory and practice. In: Proc. 11th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 760–769. Society for Industrial and Applied Mathematics (2000)
16. Koch, T., Martin, A.: Solving Steiner tree problems in graphs to optimality. Networks 32(3), 207–232 (1998)
17. Lee, Y., Lu, L., Qiu, Y., Glover, F.: Strong formulations and cutting planes for designing digital data service networks. Telecommunication Systems 2(1), 261–274 (1993)
18. Lucena, A., Resende, M.G.: Strong lower bounds for the prize collecting Steiner problem in graphs. Discrete Applied Mathematics 141(1), 277–294 (2004)
19. Molnár, M.: Hierarchies to Solve Constrained Connected Spanning Problems. Technical Report lrimm-00619806, University Montpellier 2, LIRMM (2011)
20. Narayana, S., Jiang, W., Rexford, J., Chiang, M.: Joint Server Selection and Routing for Geo-Replicated Services. In: Proc. Workshop on Distributed Cloud Computing (DCC) (2013)
21. Oliveira, C., Pardalos, P.: Streaming Cache Placement. In: Mathematical Aspects of Network Routing Optimization. Springer Optimization and Its Applications, pp. 117–133. Springer, New York (2011)
22. Park, J.-W., Lim, H., Kim, J.: Virtual-node-based multicast routing and wavelength assignment in sparse-splitting optical networks. Photonic Network Communications 19(2), 182–191 (2010)
23. Qazi, Z., Tu, C.-C., Chiang, L., Miao, R., Sekar, V., Yu, M.: SIMPLE-fying Middlebox Policy Enforcement Using SDN. In: Proc. ACM SIGCOMM (2013)
24. Quoitin, B., Van den Schrieck, V., Franois, P., Bonaventure, O.: IGen: Generation of router-level Internet topologies through network design heuristics. In: Proc. 21st International Teletraffic Congress (ITC), pp. 1–8 (2009)
25. Rost, M., Schmid, S.: CVSAP-Project Website (2013), `http://www.net.t-labs.tu-berlin.de/~stefan/cvsap.html`
26. Rost, M., Schmid, S.: The Constrained Virtual Steiner Arborescence Problem: Formal Definition, Single-Commodity Integer Programming Formulation and Computational Evaluation. Technical report, arXiv: 1310.0346 (2013)
27. Shi, S.: A Proposal for A Scalable Internet Multicast Architecture. Technical Report WUCS-01-03, Washington University (2001)
28. Voß, S.: Steiner Tree Problems in Telecommunications. In: Handbook of optimization in telecommunications, ch. 18. Spinger Science + Business Media, New York (2006)
29. Zhang, Z., Zhang, M., Greenberg, A., Hu, Y.C., Mahajan, R., Christian, B.: Optimizing cost and performance in online service provider networks. In: Proc. 7th USENIX Conference on Networked Systems Design and Implementation (NSDI) (2010)