Packing Strictly-Shortest Paths in a Tree for QoS-Aware Routing

Jose Yallouz*, János Tapolcai†, Attila Kőrösi‡, Kristóf Bérczi§, László Gyimóthi†, Ariel Orda*
*Dept. of Electrical Engineering, Technion Israel Institute of Technology, {jose@tx, ariel@ee}.technion.ac.il
‡MTA-BME Information Systems Research Group, Budapest University of Technology (BME), korosi@tmit.bme.hu
†MTA-BME Future Internet Research Group, BME, {tapolcai, gyimothi}@tmit.bme.hu
§MTA-ELTE Egerváry Research Group, Budapest, berkri@cs.elte.hu

***Abstract*—Spanning trees are a basic and important network design tool, which constitutes an efficient infrastructure for broadcasting and routing protocols. The number of shortest-paths covered by a spanning tree is a metric of major importance for evaluating the "quality" of the tree. However, typically, demanding that the connection would be precisely through a shortest path is essential only for a few source-destination pairs with strict communication requirements (*critical-demands*). Accordingly, we define the *covering effectiveness* of a spanning tree as the proportion of critical-demands whose paths in the spanning tree are indeed shortest in the network. We provide a rigorous study of this novel metric and classify several optimization problems. Specifically, we are interested in scenarios where the critical-demands originate at a few selected nodes. According to the tractability of the considered problems, we derive either optimal or heuristic solutions for finding a spanning tree with maximum covering effectiveness. Then, through extensive simulations, we demonstrate the effectiveness of our solutions. Most notably, we indicate that the quite common approach, in which a (spanning) shortest-paths tree from a single source node is selected, is often unsuitable for the scenario where critical-demands are associated with more than one pair of nodes.**

I. Introduction

A *spanning tree* is a well-known network design tool for providing connectivity to all nodes with a minimum number of links. Indeed, spanning trees are often employed in various networking environments, e.g. LANs [1], wireless [2] and optical networks [3], for several purposes such as broadcast, loop avoidance and compact routing.

The provision of Quality of Service (QoS) is an issue of major importance in the context of routing through a spanning tree. Accordingly, the selection of an efficient spanning tree has been widely studied in the context of weighted (additive) metrics. When efficient communication to a single "central" node is sought, an optimal solution is provided by a Shortest-Path (spanning) Tree (SPT) rooted at that node, and standard shortest path algorithms, e.g. Dijkstra or Bellman-Ford, provide such a solution. Another example is selecting an optimal spanning tree in terms of the overall weight of its links, i.e., a Minimum Spanning Tree (MST) [4]. Note that an MST does not provide any guarantee in terms of the quality (weight) of any of its paths. A compromise between these approaches has been considered in [5], which focuses on the creation of a spanning tree that balances between two approximation ratios, namely to the quality of minimum spanning tree and to that of the shortest path tree.

However, quite often efficient communication is sought to more than one "central" node. In this case, an SPT solution (rooted at any of the "central" nodes) would often fall short of providing efficient paths to the other "central" nodes. Generally, for the multiple "central" nodes case, the restriction to route through the paths of a unique tree affects the quality of the routing paths. Accordingly, *spanners* have been proposed as an efficient compromise between routing over a single spanning tree and employing per-destination shortest path routing. Specifically, a spanner is a spanning tree in which the cost of a path between any two nodes is at most k times worse than the shortest path between these nodes in the network [6], where the parameter k is called *stretch*. Several variants of this problem have been widely studied, e.g., the worst case [6], average case [7] and additive case [8] variants. In most of these cases, the problem has been shown to be NP-hard while approximation solutions exist for several problem variants [9]. However, the spanner approach typically evaluates the approximation to the optimal routing in terms of the total (aggregate) weight of all source-destination pairs in the network without providing any guarantee on selecting the shortest path of any source-destination pair. Yet, there are important scenarios in which one should focus only on a (possibly small) subset out of all possible source-destination pairs, while, on the other hand, due to strict QoS requirements, it is essential to insist on shortest path routing for members of this selected subset. For example, a real-time service requiring that the connection would be establish through a minimum-hop path.

Accordingly, we introduce an alternative metric, termed *covering effectiveness*, which quantifies the number of considered source-destination pairs that are connected by shortest paths in the selected spanning tree. We aim to provide zero stretch for a maximum number of required connections, in contrast to the spanner approach, where the goal is to provide decent service to every customer. To the best of our knowledge, the covering effectiveness problem has not been investigated previously. This

This work was partially supported by the Hungarian Scientific Research Fund (OTKA grant K108947, K109240, 108947) and by Ericsson.
ISBN 978-3-901882-94-4

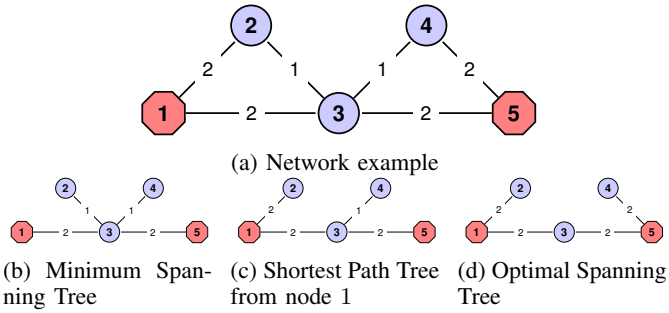
novel metric evaluates the sustainability of a shortest path rather than the gap between the sustained path and the optimal one, as done by the spanner approach. We provide a rigorous analytical study for employing spanning trees according to the covering effectiveness metric.

A different approach for providing network-wide QoS guarantees is to give a restriction on the diameter of the Steiner tree (also known as shallow Steiner tree problem [10], [11]). The main drawback with this approach is that the resulting tree has no guarantees on the per-destination connections, like spanners.

As mentioned, spanning trees have been widely employed in various networking environments. In the Ethernet [1], the Spanning Tree Protocol (STP) ensures a loop-free topology that overcomes undesirable effects such as "broadcast storms" [12]. Furthermore, the spanning tree solution simplifies the routing mechanism of the Ethernet architecture. Particulary, in [13], a variant of STP protocol have been shown to be extremely efficient for data center networks in terms of configurability, scalability and bandwidth utilization. It is interesting to note that the designers of the STP protocol attempted to consider the efficiency of the routing paths by selecting a shortest path (spanning) tree rooted at a selected node. This approach is beneficial in case that most of the network traffic is designated to a single node, but, as will be shown, might be very inefficient for other scenarios.

Spanning trees have been employed also in the context of wireless networks and optical networks as well as in the context of Multiprotocol Label Switching (MPLS). For example, in sensor [2] and ad-hoc [14] networks, spanning trees are utilized to optimize energy consumption. In multi-protocol lambda switching (MPλS) optical networks [3], shared backup trees can be employed to protect a set of working light-paths towards the same destination. In the MPLS architecture [15], a tree-based routing protocol is utilized in order to handle the trade-off between label size and stack depths. Accordingly, the concept of a *tree cover*, i.e. a small set of subtrees such that, for each pair of nodes, one of the trees contains a shortest path between them, was evaluated. Moreover, in [16], a Steiner tree structure is employed to connect VPN endpoints through MPLS, resulting in efficient utilization of network bandwidth. Furthermore, distributed algorithmic schemes for computing optimal spanning trees between arbitrary communication pairs of nodes in the network are suggested in [17].

The following example, depicted in Fig. 1, demonstrates the concept of covering effectiveness with spanning trees. Consider the network depicted in Fig. 1a, where the octagonal red nodes and the circular blue nodes represent premium and regular nodes, respectively. Moreover, each link is associated with a weight. Assume that we wish to establish an efficient communication through a spanning tree designated to the two premium nodes represented by nodes 1 and 5 by demanding a shortest path connection between the following critical demands: $\langle 2,1\rangle$, $\langle 3,1\rangle$, $\langle 3,5\rangle$, $\langle 4,5\rangle$. Clearly, the establishment of the MST depicted in Fig. 1b results in a poor covering effectiveness of $\frac{2}{4}$, as it covers only the requirements $\langle 3,1\rangle$ and $\langle 3,5\rangle$. Considering now the SPT from the controller in node 1, depicted in Fig. 1c, it covers the additional requirement of $\langle 2,1\rangle$, resulting in a covering effectiveness of $\frac{3}{4}$. Note that, due to symmetry, the SPT from the controller in node 5 also leads to the same result. Finally, the tree depicted in Fig. 1d provides the optimal covering effectiveness of 1 by covering all the required demands.



(a) Network example

(b) Minimum Spanning Tree (c) Shortest Path Tree from node 1 (d) Optimal Spanning Tree

Fig. 1: Covering Effectiveness Metric Example

To summarize, this study aims to provide a generic framework for evaluating routing through a spanning tree and it differs from previous studies in two major aspects. First, we do not consider neither a single "central" node nor all possible source-destination pairs, but rather focus on a subset of *critical demands*; second, for each critical demand, we aim at establishing a connection precisely through a path that is shortest.

The remainder of the paper is organized as follows. Section II formalizes the problem of employing a spanning tree that connects a maximum number of critical demands through shortest paths. In Section III, we classify several variants of the problem and establish their tractability. Specifically, we present a polynomial algorithmic scheme for two special cases, namely for a fixed number of critical demands (Section III-A) and for critical demands from two central nodes to all other nodes (Section III-B). We then establish that the problem is in general NP-hard (Section III-C). Consequently, in Section IV, we propose three heuristic approaches, the first based on a greedy method (Section IV-A), the second on shortest path trees (Section IV-B), and the third on Kruskal's algorithm (Section IV-C). We then establish upper bounds on the maximum number of demands that can be covered by a single tree (Section IV-D). Through extensive simulations, in Section V, we demonstrate the effect of several parameters, e.g. network density and number of demands, on the covering effectiveness metric and the efficiency of the proposed heuristics. Furthermore, we demonstrate the weakness of the quite common shortest path spanning tree approach in addressing the considered problem. Due to space limits, some proofs and details are omitted from this version and can be found (online) in [18].

II. Problem Formulation

A *network* is represented by an undirected graph $G(V,E)$, where V is the set of nodes and E is the set of links. We denote the size of these sets by $n = |V|$ and $m = |E|$, respectively.

Each link $e \in E$ is associated with a positive *weight* $w_e \in \mathbb{R}^+$ that represents an additive QoS target such as delay, cost, jitter, etc. In the unweighted case w_e is assumed to be 1 on each link.

A *path* is a finite sequence of nodes $\pi = \{v_0, v_1, \ldots, v_h\}$ such that $(v_i, v_{i+1}) \in E$ $(0 \leq i \leq h-1)$. Alternatively, a path can be represented by the sequence of its links. The *length* of a path π is the number of its links and is denoted by $|\pi|$. A path is *simple* if all of its nodes are distinct. The *weight* of path π is denoted by $w(\pi)$ and is defined as the sum of the weight of its links, i.e., $w(\pi) = \sum_{e \in \pi} w_e$. A *shortest path* between two nodes $s, t \in V$ is a path with minimum weight. Let $G'(V, E')$ be a subgraph of G. Note that between two nodes in the network several shortest paths might exist. Accordingly, the *set of shortest paths* between s and t in G' is denoted by $\Pi_{G'}(s,t)$ or $\Pi_{E'}(s,t)$. The *set of all shortest paths links* in $\Pi_{G'}(s,t)$, i.e. $\bigcup_{\pi \in \Pi_{G'}(s,t)} \pi$, is denoted by $\Lambda_{G'}(s,t)$. In all cases, we will omit the subscript when G' is identical to G. A member of $\Pi_{G'}(s,t)$ will be usually denoted by $\pi_{G'}(s,t)$. The *common weight* of paths in $\Pi_{G'}(s,t)$ is denoted by $l_{G'}(s,t)$. Note that, in the unweighted case, $l_{G'}(s,t)$ denotes the length (that is, the number of links) of a shortest path between s and t in G'.

A *tree* is an undirected graph in which any two nodes are connected by exactly one simple path. A *spanning tree* T of $G(V, E)$ is a tree that is a subgraph of G composed of all network nodes V and some of the links in E. Furthermore, we will also use the abbreviation T for specifying the spanning tree link set. Given a node $v \in V$, a *shortest path tree rooted at* s is a spanning tree T of G such that $l_T(s,v) = l_G(s,v)$ for every $v \in V$, that is, the unique path in T between s and any other node v is a shortest path in G. Note that such a tree can be determined in polynomial time by using a shortest path algorithm, e.g., Dijkstra.

As explained, often we seek shortest paths between several specific pairs of nodes, giving rise to the following definition.

Definition 2.1: Given a network $G(V, E)$, a *critical demand* $\langle s, t \rangle$ represents a demand to communicate along a shortest path between the nodes $s, t \in V$. Accordingly, the *critical demands set* D is defined as the set of all critical demands (out of the $\binom{|V|}{2}$ pairs of nodes). The number of critical demands is denoted by $|D|$.

For ease of presentation, the critical demands set is also represented by an undirected graph $H(V, D)$, termed *demand graph*, where a link $\langle s, t \rangle \in D$ constitutes a demand.

A critical demand $\langle s, t \rangle \in D$ is *covered* by a tree T if $l_T(s,t) = l_G(s,t)$; in other words, the unique path between s and t in T is a shortest path in $G(V, E)$. The *number* of critical demands covered by a tree T is denoted by $\theta_D(T)$. We omit the subscript D if the demand graph is complete.

Definition 2.2: The *covering effectiveness* of a tree T is denoted by $\eta(T)$ and is defined as

$$\eta(T) = \frac{\theta_D(T)}{|D|}.$$

The goal of this study is to find a tree that covers a maximum number of critical demands, formally:

Definition 2.3: ***Single Tree Effectiveness problem (STE)***: Given a network $G(V, E)$ with positive link weights $w : E \to \mathbb{R}^+$ and critical demands D, find a spanning tree T that maximizes $\eta(T)$.

For complexity analysis, we also define the following decision version of the problem. Given are a network $G(V, E)$ with positive link weights $w : E \to \mathbb{R}^+$, critical demands D and a positive integer k. Is there a spanning tree T such that $\theta_D(T) \geq k$?

While there might be several shortest paths between any two nodes, we are interested in covering just one (any) of them. This freedom introduces some additional complexity and sometimes we might be interested in guaranteeing that the shortest path is unique (e.g., for uniformity when the algorithm is run at different nodes). This uniqueness can be achieved for example by introducing a "tie breaker" such as the nodes' id's. Thus, we also study the particular instance of STE where $|\Pi(s,t)| = 1$, that is, the shortest path is unique for every demand $\langle s, t \rangle \in D$, which we term the *Unique Path Single Tree Effectiveness (UPSTE) problem*.

For each of the two problems, STE and UPSTE, we consider the four cases, listed by increasing order of difficulty:

- The *fixed number of demands case* where $|D|$ is bounded by a constant.
- The 2-*to-all case* where H is the union of two star-graphs.
- The *all-to-all case* where H is a complete graph, i.e. $|D| = \binom{|V|}{2}$.
- The *general case* where H is an arbitrary graph.

We proceed to consider the tractability of the above problem variants.

III. Analysis of the STE Problem

In this section, we study several variants of the STE problem. First, we consider the case of a fixed number of demands, which is proved to be strongly related to a well known open problem of finding node-disjoint shortest paths. This relation provides a polynomial-time algorithm for the special case where the network is planar. Then, we examine the 2-to-all problem variant and establish a polynomial-time algorithm. We show that the STE problem is NP-hard for all-to-all demands. Concerning the case of general demands, we prove that UPSTE, hence also STE, are NP-hard problems. Table I summarizes the main results of this section.

TABLE I: Summary of the results on STE problem

Type of demands	STE	UPSTE
Fixed number of demands	Open, Equiv. to k-DSP, Thm. 3.1 (P for planar graphs)	P, Sec. III-A
2-to-all demands	P, Sec. III-B	
All-to-all demands	NP-hard, Thm. 3.2	Open
General demands	NP-hard, Thm. 3.3	

A. Fixed Number of Demands

Consider first the UPSTE problem with a fixed (i.e., $O(1)$) number of demands $|D| = d$. In this case, we can consider every possible subset of the demands and check if it can be

covered by a single tree. To check whether a subset $D' \subseteq D$ of demands can be covered, we just have to consider the union of the unique paths $\pi \in \Pi(s,t)$ for $\langle s,t\rangle \in D'$. Clearly, D' can be covered by a single tree if and only if this link set does not contain a cycle. Note that there are 2^d possible choices for D', hence, all of these subsets can be checked in polynomial time as d is not part of the input.

Now, we show that the STE problem for a fixed number of demands is equivalent to a long-standing open problem, namely the *k-Disjoint Shortest Paths (k-DSP) problem*. Given a network $G(V,E)$, a weight function $w : E \to \mathbb{R}^+$ and a set D of demands consisting of k distinct pairs of nodes $\langle s_1,t_1\rangle,\dots,\langle s_k,t_k\rangle$, the k-DSP problem aims to find k pairwise node-disjoint paths $\pi_1,\dots,\pi_k$ such that $\pi_i \in \Pi_G(s_i,t_i)$ for $i=1,\dots,k$. This problem is NP-complete when k is part of the input, and can be solved in polynomial time for $k=2$ [19]. However, in case the network $G(V,E)$ is planar and k is fixed then the problem becomes tractable [20]. Hence our reduction shows that the STE problem can be solved in polynomial time for a fixed number of demands and planar topologies.

Theorem 3.1: The STE problem for fixed number of demands and the k-DSP problem for fixed k are polynomially equivalent.

Proof: First, we show that the k-DSP problem for a fixed k is polynomially reducible to the STE problem for a fixed number of demands. Consider an instance $G(V,E)$ and $D = \{\langle s_1,t_1\rangle,\dots,\langle s_k,t_k\rangle\}$ of the k-DSP problem. Add a new node v_0 and links (v_0,s_i) to the network with $w(v_0,s_i) = L$ for $i=1,\dots,k$, where L is large, say, $L = |E|\cdot\max\{w(e) : e\in E\}+1$. Let $G'(V',E')$ denote the extended network and $D' = D \cup \{\langle v_0,s_i\rangle : \ i=1,\dots,k\}$. Now consider the STE problem on this new network $G'(V',E')$ and demand set D'. As the weights of the extra links are large, there is no shortest path from s_i to t_i through v_0 for $i=1,\dots,k$. Hence, there is a tree T in G' with $\theta_l(T) = |D'| = 2k$ if and only if there are k pairwise node-disjoint paths $\pi_1,\dots,\pi_k$ in the original network such that $\pi_i \in \Pi_G(s_i,t_i)$ for $i=1,\dots,k$.

Next we show that the STE problem for a fixed number of demands is polynomially reducible to the k-DSP problem for fixed k. Consider an instance $G(V,E)$ of the STE problem with a fixed number of demands $|D| = d$. It suffices to show that the problem of deciding whether a subset $D' \subseteq D$ of demands can be covered by a single tree is polynomially reducible to the k-DSP problem for fixed k. Indeed, in this case we can apply the reduction to each subsets D' of D. As there are 2^d subsets of D where d is fixed, we get a polynomial reduction of the original problem.

Let $D' = \{\langle s_1,t_1\rangle,\dots,\langle s_l,t_l\rangle\}$ and assume temporarily that we have a tree T covering each demand in D'. We may assume that every leaf v of T is one of the nodes $s_1,\dots,s_l,t_1,\dots,t_l$ as otherwise we could simply delete v from T. Thus the number of leafs in T is at most $2l$. We call a node v of T a *demand node* if $v \in \{s_1,\dots,s_l,t_1,\dots,t_l\}$, and a *hub* if v is not a demand node and its degree in T is at least 3. The number of hub nodes is denoted by h, and it is at most the number of leafs, thus $h \le 2l$. A path π in T is called *branch* if both of its end nodes are among the demand and hub nodes and π does not contain a demand or a hub node as an intermediate node. Let T' denote the tree obtained from T by substituting its branches by a single link. In other words we merge every 2-degree node other than hubs and demands. We will call T' the *topology* of T. Observe that T' has at most $4l$ nodes.

By the above, the set of hub nodes in the network can be chosen in $\binom{n}{h} = O(n^{2l})$ possible ways. After a set of hub nodes is selected, the number of possible topologies with the given hubs is bounded by $4^{4l-2}l^{4l-2}$, the number of labelled trees on $4l$ nodes. Hence the number of possible topologies is $O(n^{2l}4^{4l}l^{4l})$, which is polynomial in n as $l \le d$ and d is fixed.

Summing up the above, for a given subset $D' \subseteq D$ of demands with $|D'| = l$ we can check whether D' can be covered by a single tree as follows: for each possible topology T', we try to embed T' into G in such a way that each link e of T' corresponds to a shortest path in G between the end nodes of e. Recall that these paths should be disjoint in order to obtain a tree T. That is, if the number of links in the topology T' is k then we have to solve a k-DSP problem in G where each demand corresponds to a link in T'. Note that there are at most $4l-1$ links in the tree T', i.e. $k \le 4l-1$.

This approach requires solving polynomial number of instances of the k-DSP problem, thus concluding the proof. ∎

B. 2-to-all Demands Case

When all demands share a common end-node s, a shortest path tree rooted at s constitutes an optimal solution for the STE problem. Next, we extend this observation to the STE problem for 2-to-all demands with center nodes s_1 and s_2 and provide a polynomial algorithmic scheme for finding the optimal solution.

For simplicity, first we consider the critical demand set consisting of pairs $\langle s_i,v\rangle$ for $i=1,2$ and $v \in V\setminus\{s_1,s_2\}$, that is, we are not interested in covering a shortest path between s_1 and s_2. Further, we will show how the algorithm can be modified to also include the demand $\langle s_1,s_2\rangle$.

An arbitrary spanning tree T determines a partition of $V \setminus \{s_1,s_2\}$ into three disjoint sets A_T, B_T, C_T as follows:

- $v \in C_T$ if T covers both $\langle s_1,v\rangle$ and $\langle s_2,v\rangle$,
- $v \in B_T$ if T covers only one of $\langle s_1,v\rangle$ or $\langle s_2,v\rangle$,
- $v \in A_T$ if T covers none of $\langle s_1,v\rangle$ and $\langle s_2,v\rangle$.

We will show that there exists a spanning tree for which $|C_T|$ is maximal among all possible spanning trees, and in addition $A_T = \emptyset$. Such a tree clearly maximizes $|B_T|+2|C_T|$ and hence is optimal.

Consider an arbitrary spanning tree T. We first show that C_T spans a connected subtree of T.

Lemma 3.1: C_T induces a connected subgraph in T.

Proof: Let $u,v \in C_T$, and let $\pi_T(u,v)$ be the path in T between them. We will show that any node $w \in \pi_T(u,v)$ is also in C_T. Let z be the node in $\pi_T(u,v)$ closest to s_1, that is, one for which $\pi_T(s_1,z)$ has the least number of links. Note that z is uniquely determined. By symmetry, we may assume that w is part of the segment between u and z of $\pi_T(u,v)$. Now the path $\pi_T(s_1,u)$ goes through w, and since T covers $\langle s_1,u\rangle$

it also covers $\langle s_1, w\rangle$ (the subpath $\pi_T(s_1, w)$ of the shortest path $\pi_T(s_1, u)$ is also a shortest path between its end-nodes). A similar argument shows that T covers $\langle s_2, w\rangle$. ■

Now, we provide an upper bound on the size of C_T. Let $t_1, t_2 \in C_T$ denote the closest nodes to s_1 and s_2, respectively. Observe that for any node $v \in C_T$, $\pi_T(s_i, v)$ is a shortest path from s_i to v through node t_i $(i = 1, 2)$. We define the *chain* of T to be $\pi_T(t_1, t_2)$, the path between t_1 and t_2 in T. Given a node $v \in V \setminus \{s_1, s_2\}$, let Z_v denote the set of nodes that can be reached both from s_1 and s_2 on a shortest path containing v. We define the *value of the chain* as $\sum[|Z_v| : \ v \in \pi_T(t_1, t_2)]$.

Lemma 3.2: The size of C_T is at most the value of the chain of T, i.e. $|C_T| \leq \sum_{v \in \pi_T(t_1,t_2)} |Z_v|$.

Proof: By Lemma 3.1, the node-set C_T can be partitioned into the sub-trees hanging on the nodes of the chain, and the size of sub-tree corresponding to node v is bounded by $|Z_v|$ from above. ■

In order to show that the upper bound for $|C_T|$ can be achieved, we will need the following lemma.

Lemma 3.3: $Z_{v_1} \cap Z_{v_2} = \emptyset$ for any $v_1, v_2 \in \pi_T(t_1, t_2)$.

Proof: Suppose that $v_1, v_2 \in \pi_T(t_1, t_2)$ are distinct nodes on the chain such that $z \in Z_{v_1} \cap Z_{v_2}$. We may assume that nodes t_1, v_1, v_2 and t_2 come in this order on the chain of T. Recall that $l(u, v)$ denotes the length of a shortest path between u and v in G.

As $z \in Z_{v_1} \cap Z_{v_2}$, we have $l(s_1, v_1) + l(v_1, z) = l(s_1, v_2) + l(v_2, z)$ and $l(s_2, v_2) + l(v_2, z) = l(s_2, v_1) + l(v_1, z)$. As v_1 and v_2 lie on the chain of T, we have $l(s_1, v_2) = l(s_1, v_1) + l(v_1, v_2)$ and $l(s_2, v_1) = l(s_2, v_2) + l(v_2, v_1)$. By summing up the left- and right-hand sides of these equalities we get $2l(v_1, v_2) = 0$, contradicting the positivity of the link weights. ■

Lemma 3.3 suggests the following algorithm for finding an optimal spanning tree. Define a subgraph $G'(V, E')$ of G in which an link uv is kept if and only if u can be reached from s_1 on a shortest path containing v, and vice versa, v can be reached from s_2 on a shortest path containing u. Observe that E' consists exactly of those links of G that may appear in the chain of a tree. Now assign the value $|Z_v|$ to each node $v \in V$. Note that a node u belongs to Z_v *iff* $l(s_1, v) + l(v, u) == d(s_1, u)$ and $l(s_2, v) + l(v, u) == d(s_2, u)$. Accordingly, an an All-Pairs Shortest Paths algorithm should be employed for ensuring these conditions. Moreover, orient an link $uv \in E'$ from u to v if $l(s_1, u) < l(s_1, v)$. The resulting node-weighted graph is acyclic, hence one can find a path maximizing the total weight of its nodes employing Program Evaluation and Review Technique (PERT) algorithm, which can be solved in linear time [4]. This path will be our candidate as the chain of the solution. Let t_1 and t_2 denote the end-nodes of the path and assume that $l(s_1, t_1) \leq l(s_1, t_2)$.

Lemma 3.3 shows that if we assume a shortest-path tree rooted at v and spanning Z_v for each node v of the chain, then the chain together with these sub-trees form a tree T'. If we could show that T' can be extended to a spanning tree T of G in such a way that $A_T = \emptyset$ and C_T contains the node-set of T' then the optimality of T would follow from $A_T = \emptyset$, the choice of the starting chain and the upper bound given in Lemma 3.2.

The following lemma shows that T' can indeed be extended in a proper way.

Lemma 3.4: There exists a tree T with $A_T = \emptyset$ and C_T containing all nodes of T'.

Proof: We provide sketch of the proof and further details can be found in [18]. At this point T' contains a chain between t_1 and t_2 and the proper sub-trees of the chain. Next, we aim to achieve $A_T = \emptyset$. First we extend T' by adding to it the shortest paths $\pi_G(s_1, t_1) \in \Pi_G(s_1, t_1)$ and $\pi_G(s_2, t_2) \in \Pi_G(s_2, t_2)$ to it thus obtaining a larger tree. Clearly, this step does not create a cycle in T'. Our plan is roughly as follows. We consider the nodes $\{v\}$ not yet contained in the tree iteratively and connect v to the tree by adding a path such that either $\langle s_1, v\rangle$ or $\langle s_2, v\rangle$ is covered. We explain why this step is (garanteed to be) possible through an example depicted in Fig. 2.

Assume that T' consists of the path $\{s_1, v_2, t_1, t_2, v_1, s_2\}$. We would like to add p to the tree such that either $\langle s_1, p\rangle$ or $\langle s_2, p\rangle$ becomes covered. This is not possible only if the unique shortest path between s_i and p is $\{s_i, v_i, p\}$ for $i = 1, 2$. However, this, together with the fact that $\langle s_1, t_2\rangle$ and $\langle s_2, t_1\rangle$ are covered by T', implies the inequalities of Fig. 2. By adding up both sides of the inequalities we get $l(t_1, t_2) < 0$, a contradiction. Through a similar reasoning, we can show that T' can be extended to a spanning tree T with A_T being empty. ■

After finding the optimal chain and adding the shortest-path tree rooted at v and spanning Z_v for each node v of the chain, the tree-construction can be finished by using the algorithm of Lemma 3.4. Thus we get a spanning tree T of G with an empty A_T and maximal C_T, showing the optimality of T.

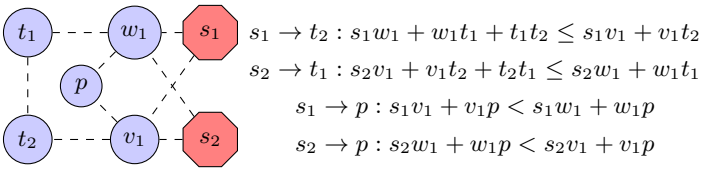


Fig. 2: Illustration of the proof of Lemma 3.4. Note that $l(a, b)$ is shortened as ab in the inequalities.

In the above discussion, we assumed that the demand set D does not include $\langle s_1, s_2\rangle$. However, the case where D does include $\langle s_1, s_2\rangle$ can be handled through a small modification of the algorithm. Specifically, s_2 should not be included in Z_{s_1}, while s_1 is contained by Z_{s_2}. This way the double counting of the demand $\langle s_1, s_2\rangle$ is avoided. The running time of the algorithm is dominated by an All-Pairs Shortest Paths algorithm employed for assigning the values of $|Z_v|$ to each node $v \in V$. The usage of Floyd-Warshall algorithm results in a running time of $O(|V|^3)$[4].

C. All-to-All and General Demands

Now we show that the STE problem is NP-hard for the general case and for particular all-to-all demands case, that is,

when the demand graph H is complete.

Theorem 3.2: The STE problem is NP-hard even if G is bipartite and uniform-weighted and H is complete.

Theorem 3.3: The Unique Single Tree Effectiveness (UPSTE) problem is NP-hard.

Proof: The proofs of Theorems 3.2 and 3.3 are both based on a reduction from a variant of the hypergraph perfect matching problem. The details appear in the Appendix of [18]. ■

IV. Heuristics and Bounds for the STE Problem

As the STE problem is intractable for general demands, we turn to focus on efficient heuristic approaches. First, we introduce a simple greedy iterative heuristic. Then, we propose a heuristic based on the establishment of a shortest path tree. Accordingly, we investigate when the optimal tree is a shortest path tree and present examples where the optimal solution for the STE ptoblem is not a shortest path tree. Next, we define a heuristic based on Kruskal's minimum-spanning tree algorithm, which also provides an upper bound on the covering effectiveness metric. Finally, we introduce a second upper bound, which can better cope with multiple shortest-path trees.

A. Iterative Greedy Heuristic

The first heuristic iteratively constructs a spanning tree T. Initially, the tree T does not contain any link. The heuristic selects a demand $\langle s,t\rangle$ from the critical demands set D iteratively, and checks if it can be covered by the tree by optionally extending the tree. A demand $\langle s,t\rangle$ can be covered if there is a shortest path $\pi \in \Pi(s,t)$ between s and t such that $\pi \cup T$ remains a tree, i.e. $\pi \cup T$ does not contain a cycle. For unique shortest paths, i.e. the UPSTE case, this property can be verified in linear time through a Depth First Search (DFS). This process is repeated until every demand is processed. The total running time of the heuristic is $O(|D| \cdot |E|^2)$.

In case the shortest path between s and t is not unique, this property verification is a bit more complex. Observe that if a shortest path π can be covered by T, $\pi \cup T$ cannot have a cycle. Therefore, π can be divided into three (possibly empty) segments: the first and third segment is disjoint with T, while the second segment is a part of T. Let i and j denote the intersection nodes at the end of the first segment and the second segment, respectively. The verification is done by searching for a proper node-pair i,j in the following way. Consider the all shortest paths links set $\Lambda(s,t)$ and search for a node pair i and j for which (1) there is a directed path from i to j in $\Lambda(s,t)$, (2) $l_{G\setminus T}(s,i) = l_G(s,i)$, (3) $l_T(i,j) = l_G(i,j)$, and (4) $l_{G\setminus T}(j,t) = l_G(j,t)$. If such i and j is found add $\pi_{G\setminus T}(s,i)$ and $\pi_{G\setminus T}(j,t)$ to tree T. The total running time of the heuristic is $O(|D| \cdot |V|^3 \log |V|)$.

The performance of the heuristic strongly depends on the order of the selected demands. Our experiments indicate that considering the demands in increasing order of their hop distance is typically a good choice; however the performance of the heuristic in terms of covering effectiveness and runtime can be improved. Specifically, it becomes computationally intensive if there are many demands in the network. Accordingly, we introduce an additional heuristic, based on shortest path trees.

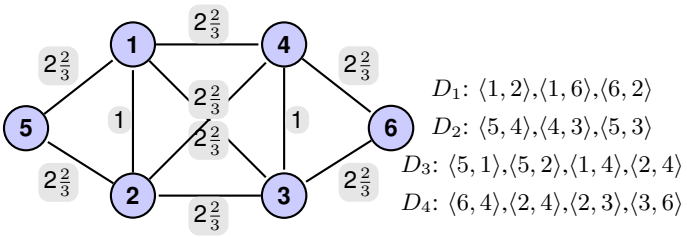


Fig. 3: Network example with unit link weights. The link labels correspond to the induced cost of ICKruskalHeur of Sec. IV-C. The demands on the right illustrate Thm. 4.2.

B. Shortest Path Tree Heuristic

In case the demand graph is a star with central node v, the optimal tree is the shortest path tree from v. Moreover, every subpath of a shortest path is a shortest path itself. Thus, a shortest path tree covers quite several demands, especially if the node v is in the "center" of the network, such as a node with high betweenness centrality value. This motivates the following simple heuristic: select a shortest path tree covering a large number of demands. In order to do that, first determine the all-pairs shortest paths by using the Floyd-Warshall algorithm [4]. For an arbitrary node v, consider a shortest path tree T_v. Then T_v covers a demand $\langle s,t\rangle \in D$ if and only if one of the following holds.

- $l_{T_v}(s,v) + l_{T_v}(v,t) = l_G(s,t)$ as in this case a shortest path between s and t goes through v,
- the path in T_v between v and s contains t, or
- the path in T_v between v and t contains s.

We can compute a shortest path tree T_v for each $v \in V$ and choose the one with maximal $\eta(T_v)$. The running time of the heuristic is $O(|D| \cdot |V|^2 + |V|^3)$.

Intuitively, one could expect that there always exists an optimal solution of the STE problem with all-to-all demands consisting of a shortest path three rooted at some node of the network. Note that this does not contradict the NP-hardness of the problem as the number of possible shortest path trees may be exponential. However, in the Appendix of [18] we show a counter example that establishes that the intuition of the optimality of a shortest path tree is wrong. Next we introduce a more sophisticated approach, which also provides an upper bound on the number of demands.

C. Upper Bound and Heuristic Based on Kruskal's Algorithm

We start by introducing an upper bound for the UPSTE problem. We define a cost function $c : E \to \mathbb{R}^+$, as follows. For each $\langle s,t\rangle \in D$, consider the unique shortest path $\pi(s,t) \in \Pi(s,t)$ and increase the link costs along the path $\pi(s,t)$ by 1 unit in total. That is, if $c_e^{s,t}$ denotes the cost added to link e corresponding to the shortest path $\pi(s,t)$ then $\sum_{e\in\pi(s,t)} c_e^{s,t} = 1$. Let $c : E \to \mathbb{R}^+$ be the cost function defined as $c_e = \sum_{(s,t)\in D} c_e^{s,t}$ for $e \in E$. We call such

a cost function *induced*. For example, a possible choice for $c_e^{s,t}$ is setting $c_e^{s,t} = \frac{1}{|\pi(s,t)|}$. In this case c_e is set to be $\sum_{\pi(s,t) \ni e} \frac{1}{|\pi(s,t)|}$.

Theorem 4.1 (KruskalBound): In the UPSTE problem, the maximum number of demands covered by a single tree T is at most the cost of a maximum cost spanning tree of G having an induced cost function c,[1] formally:

$$\max\{\theta_D(T) : T \text{ is a tree}\} \leq$$
$$\lfloor \min\{c(T) : c \text{ is induced, } T \text{ is a maximum } c\text{-cost tree}\} \rfloor.$$

Proof: Consider an arbitrary tree T and an induced cost function c. Assume that T covers demands $\langle s_1, t_1\rangle, \ldots, \langle s_k, t_k\rangle$. Then the sum of the costs on the links of T is at least k as c is induced. Thus, the number of shortest paths covered by T is bounded by the cost of T. ∎

The above theorem can be generalized to the STE problem if $c_e^{s,t}$ is set such that for any shortest path $\pi(s,t) \in \Pi(s,t)$ we have $\sum_{e \in \pi(s,t)} c_e^{s,t} \geq 1$. However, it is possible to define $c_e^{s,t}$ in such a way that $\sum_{e \in \pi(s,t)} c_e^{s,t} = 1$ for each $\pi(s,t) \in \Pi(s,t)$. In order to do so, first determine the set $\Lambda(s,t)$ of links that lie on a shortest path between s and t by employing the In-All-Weight-Shortest-Paths Links algorithm described in [21]. It is well known that this way we get an acyclic digraph in which each directed path from s to t has the same weight, namely $l(s,t)$. Recall that $l(s,t)$ denotes the weight of a shortest path between s and t. Thus, setting $c_e^{s,t} = \frac{1}{l(s,t)} w_e$ for $e \in \Lambda(s,t)$ will give a proper cost function. Hence, we derive the following corollary.

Corollary 1: In the STE problem, the maximum number of demands covered by a single tree T is at most the floor of the cost of a maximum cost spanning tree G where the cost of an link e is defined as

$$c_e = \sum_{\substack{(s,t) \in D \\ e \in \Lambda(s,t)}} \frac{w_e}{l(s,t)}.$$

See Fig. 3 for an example of the above corollary, where the cost c_e of each link is depicted in the gray box. For example, link $(1,5)$ is contained in $\Lambda(1,5)$, $\Lambda(4,5)$, $\Lambda(3,5)$ and $\Lambda(5,6)$, hence its induced cost is $1 + \frac{1}{2} + \frac{1}{2} + \frac{1}{3} = 2\frac{2}{3}$. The induced costs of the remaining links can be computed analogously. A maximum cost tree is $(1,5),(2,5),(2,3),(3,6),(4,6)$ with a cost of $11\frac{2}{3}$ giving a bound of $\lfloor 11\frac{2}{3} \rfloor = 11$, while we have already seen that an optimal solution covers 10 demands.

The heuristic outputs the obtained maximum cost tree as the solution. The running time of the heuristic is $O(|D| \cdot |E| \log |V|)$ steps, see also Fig. 1 for the pseudocode of the heuristic algorithm.

D. Upper Bound Based on Conflict Cycles

We continue to present an additional upper bound, which focuses on the case where the shortest paths are not unique. Let $D' \subseteq D$ be a subset of demands consisting of node-pairs $\langle s_1, t_1\rangle, \ldots, \langle s_l, t_l\rangle$, where $l = |D'|$. The set D' is *conflicted* if the corresponding shortest paths cannot be covered by a single tree. In other words, D' is conflicted if there exist no paths $\pi_1, \ldots, \pi_l$ such that $\pi_i \in \Pi(s_i, t_i)$ and $\bigcup_{i=1}^{k} \pi_i$ is a forest.

[1]Since any induced cost function such that a $\langle s,t\rangle$ $\sum_{e \in \pi(s,t)} c_e^{s,t} = 1$ is valid, we consider one which results with the smallest upper bound.

Algorithm 1: Induced Cost Kruskal Heuristic

Data: $G(V,E)$-network, demand D, and link weights w
begin
  $c_e = 0$ for every link e
  for $\langle s,t\rangle \in D$ **do**
    for $e \in \Lambda_E(s,t)$ **do**
      $c_e := c_e + \frac{w_e}{l(s,t)}$
  return the maximum cost spanning tree with cost function c on links with Kruskal's algorithm

Theorem 4.2 (CCycleBound): Let $D_1, \ldots, D_k$ be k conflicted subsets of D that are pairwise disjoint (i.e. $D_i \cap D_j \equiv \emptyset$, $\forall i \neq j$). Then $\theta_D(T) \leq |D| - k$ for any spanning tree T of G.

Proof: Consider an arbitrary spanning tree T of G. For each $i = 1, \ldots, k$, there must be a demand in D_i not covered by T as D_i is conflicted. Since the $D_i's$ are pairwise-disjoint, there are at least k demands not covered by T, thus concluding the proof. ∎

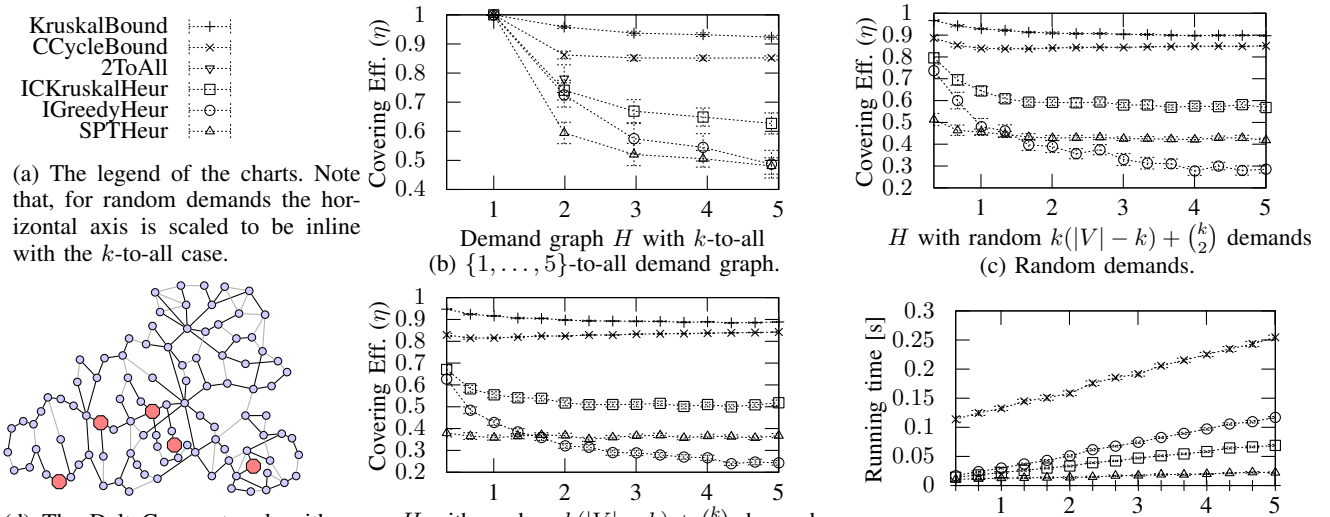
See Fig. 3 for an illustrative example, where the demands can be divided into disjoint conflicted subsets of D_1, D_2, D_3 and D_4.

Next, we define a simple brute-force heuristic to search for such pairwise disjoint conflicted sets. First, the demands with unique shortest paths are processed. In case of unique shortest paths, a set D_i of the demands can be evaluated by examining the 2-connected components of the subgraph that is determined by the union of the shortest paths in D_i. If the resulting subgraph is not a tree, i.e. there is a bi-connected component, then D_i can be removed from the original demand set, and the upper bound can be decreased by one, since D_i forms a conflicted set. Note that finding the 2-connected component requires a single run of Depth First Search (DFS).

If the demands have multiple shortest paths, we focus on sets of three demands only, i.e.$|D_i| = 3$. We select every possible triplet of nodes a, b and c. Let x be an arbitrary node, if $l_G(a,x) + l_G(x,b) = l_G(a,b)$, $l_G(a,x) + l_G(x,c) = l_G(a,c)$ and $l_G(b,x) + l_G(x,c) = l_G(b,c)$ holds, then the demands $\langle a,b\rangle$, $\langle b,c\rangle$ and $\langle a,c\rangle$ can be covered with a tree, where node x is a "central" node, see also the discussion at Section IV-B. Note that the opposite is also true, i.e. if there is no such $x \in V$ the three demands are in a conflict cycle.

V. Simulation Study

In Sections IV and III-B, we presented three heuristics (IGreedHeur, SPTHeur, ICKruskalHeur), two upper bounds (KruskalBound, CCycleBound) and an optimal algorithm for the case of 2-to-all demands (2ToAll). We proceed to evaluate the efficiency of these approaches in terms of covering effectiveness and running time in real-world and random network



(a) The legend of the charts. Note that, for random demands the horizontal axis is scaled to be inline with the k-to-all case.

(b) $\{1,\ldots,5\}$-to-all demand graph.

(c) Random demands.

(d) The DeltaCom network with a tree of ICKruskalHeur for 5-to-all demands from the large nodes.

(e) Random demands for UPSTE with transformed link weights

(f) The runtime corresponding to Fig. 4c.

Fig. 4: The results of DeltaCom with 103 nodes and 151 links.

topologies. Specifically, we examine the achieved covering effectiveness for two demand configurations, namely k-to-all and randomly chosen demands. It turns out that the spanning tree provided by the ICKruskalHeur heuristic offers the best covering effectiveness for all simulated scenarios. Furthermore, we demonstrate the negative effect of the density of the network links on the covering effectiveness of the spanning tree. Somewhat surprisingly, we show that the SPTHeur heuristic results in the worst covering effectiveness for almost all simulated scenarios. This finding is noteworthy in view of the quite common usage of shortest path spanning trees in networking scenarios.

We generated two classes of network topologies, namely the *real carrier topology* and the *random topology*.

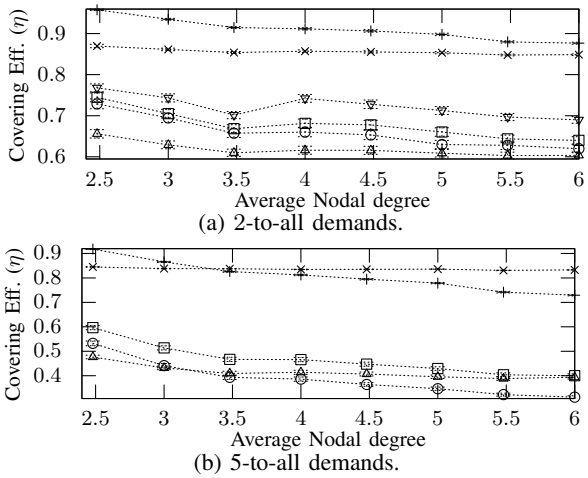
First, we evaluated twenty real-world carrier topologies [22]. For the sake of brevity, we present the results of the largest network only, namely the Deltacom, which is a relatively large network operated in the U.S.A region. This topology is depicted in Fig. 4d and contains 103 nodes and 151 links. In order to evaluate the performance of our heuristics in the context of the STE problem, the link weights were set to be uniform. Moreover, the link weights were modified with a very small value to have unique shortest path for every demand in the network, enabling the evaluation of the UPSTE problem.

Next, we consider the class of random topologies, based on the Waxman model [23]. We created 320 random networks containing 50 nodes and a range of $[62, 150]$ links. Note that these networks are very sparse and often planar. In all simulation instances of this class, we focused on the STE problem, therefore, the link weights were set to be uniform.

For each network topology class, we evaluated the three heuristics (ICKruskalHeur, IGreedHeur, SPTHeur), the two upper bounds (KruskalBound, CCycleBound) and the optimal algorithm for 2-to-all demands (2ToAll), where the legend for the charts of each case is depicted in Fig. 4d.

Fig. 4 demonstrates the simulation results of the DeltaCom carrier topology depicted in Fig. 4d. First, we evaluated the STE problem, i.e. the link weights were uniform. In Fig. 4b, we show the covering effectiveness for k-to-all demands, where $k = \{1,\ldots,5\}$ and the selected central nodes are depicted with diamonds in Fig. 4d. Each data point corresponds to an average of 60 executions and the 95% confidence interval is also drawn. As expected, for a 1-to-all demand graph the shortest path tree from the single central node provides 100% covering effectiveness for all evaluated cases. However, for 2-to-all demands the covering effectiveness dropped to 59%. Note that for 2-to-all demands, we have an optimal algorithm, which gives $\eta = 78\%$ (2ToAll point). For all cases, KruskalHeur resulted in better covering effectiveness than IGreedyHeur and SPTHeur and, moreover, the CCycleBound provided better bounds than KruskalBound. The KruskalHeur spanning tree with highest covering effectiveness for the 5-to-all demand graph is depicted in Fig. 4d.

Fig. 4c shows the results when the demand graph is random. Note that the number of demands for the k-to-all case is $k(|V|-k)+\binom{k}{2}$. Here, we scaled the horizontal axis to have the same scale as of Fig. 4b, thus, the k-th tick on the horizontal axis corresponds to $k(|V|-k)+\binom{k}{2}$ randomly selected $\langle s,t\rangle$ demands for $k = \{1,\ldots,5\}$. In average, 10% less random demands can be covered compared to the k-to-all case. We obtained a larger difference when the number of demands is small. We have discussed the significant difference in the complexity of the STE and UPSTE versions of the problem. In order to analyze its effect in practice, we transformed the DeltaCom case to UPSTE by adding a very small random value to each link weight in the network. This will reduce the number



(a) 2-to-all demands.

(b) 5-to-all demands.

Fig. 5: The covering effectiveness respect the nodal degree of the random topology network. See Fig. 4a for the legend.

of shortest paths between each node-pair to one, and thus significantly reducing the search space of the algorithms. Fig. 4e shows the results in terms of average values over 30 UPSTE instances. Surprisingly, the covering efficiency in UPSTE was only 10% smaller in average compared to STE. Note that it is easy to construct networks where this difference is arbitrarily large. In Fig. 4f, we evaluate the running time of the proposed heuristics of the STE random demand graph and show that: (1) the CCycleBound incurs the highest computational penalty, (2) the SPTHeur is the fastest heuristic and less sensible to the number of demands and (3) the ICKrustalHeur is still reasonably fast, yet with better performance.

Finally, in Fig. 5, we evaluate the covering effectiveness with respect to the average nodal degree of the random topology class. Two demand scenarios were analysed, namely the 2-to-all and 5-to-all cases. For the 2-to-all case, we have an optimal algorithm, whose performance is close to the heuristics and far from the upper bounds. As expected, better coverage efficiency can be achieved for sparser networks. Indeed, our results clearly show that shortest paths in sparser networks can be better covered by a spanning tree due to their smaller diversity. In addition, we evaluated the 5-to-all demands, where the KruskalBound became better than the CycleBound compared to the 2-to-all case.

Notably, for almost all simulated cases, the SPTHeur heuristic results in a significantly small value of covering effectiveness, contradicting the quite common usage of shortest path spanning trees in networking scenarios.

VI. Conclusion

The *covering effectiveness* is a novel quantitative approach for evaluating the "quality" of a spanning tree to support QoS-aware communication between a restricted set of (several) critical pairs of nodes. This metric quantifies the number of source-destination pairs for which the path on the tree is shortest. Accordingly, we defined the STE optimization problem which, for a given set of critical pairs, aims to find a spanning tree that maximizes its covering effectiveness. We investigated four variants of the optimization problem according to the set of demands, namely: fixed number of critical demands, 2-to-all, all-to-all and the general case. For the first two cases we established optimal polynomial solutions while the other two were shown to be NP-hard. For the intractable cases, we derived theoretical bounds and presented heuristic solutions, all of which have been evaluated through comprehensive simulations. Most notably, we indicated that the quite common approach, in which a shortest path spanning tree from a single node is selected, often falls short of providing good solutions in the typical scenario where critical demands are associated with more than a single end-node. As an alternative, we proposed a simple yet typically efficient heuristic, which is based on Kruskal's algorithm.

References

[1] IEEE, "802.1d: Standard for local and metropolitan area networks: Media access control (MAC) bridges," 1990.

[2] J. Liang, J. Wang *et al.*, "An efficient algorithm for constructing maximum lifetime tree for data gathering without aggregation in wireless sensor networks," in *IEEE INFOCOM*, 2010.

[3] A. Groebbens, D. Colle *et al.*, "Efficient protection in mpλs networks using backup trees," *Photonic Network Communications*, 2003.

[4] T. H. Cormen, C. E. Leiserson *et al.*, *Introduction to Algorithms, Third Edition*, 3rd ed. MIT Press, 2009.

[5] S. Khuller, B. Raghavachari, and N. Young, "Balancing minimum spanning trees and shortest-path trees," *Algorithmica*, 1995.

[6] D. Peleg and A. A. Schäffer, "Graph spanners," *Journal of Graph Theory*, 1989.

[7] D. Peleg, "Low stretch spanning trees," in *Mathematical Foundations of Computer Science*, 2002.

[8] D. Kratsch, H.-O. Le *et al.*, "Additive tree spanners," *SIAM Journal on Discrete Mathematics*, 2003.

[9] C. Liebchen and G. Wünsch, "The zoo of tree spanner problems," *Elsevier Discrete Applied Mathematics*, 2008.

[10] G. Kortsarz and D. Peleg, "Approximating the weight of shallow steiner trees." *Discrete Applied Mathematics*, 1999.

[11] M. Chimani and J. Spoerhase, "Network design problems with bounded distances via shallow-light steiner trees." in *STACS*, 2015.

[12] Y.-C. Tseng, S.-Y. Ni *et al.*, "The broadcast storm problem in a mobile ad hoc network," *Wireless networks*, 2002.

[13] B. Stephens, A. L. Cox *et al.*, "PAST: scalable ethernet for data centers," in *CoNEXT*, 2012.

[14] H. Baala, O. Flauzac *et al.*, "A self-stabilizing distributed algorithm for spanning tree construction in wireless ad hoc networks," *Journal of Parallel and Distributed Computing*, 2003.

[15] A. Gupta, A. Kumar, and R. Rastogi, "Traveling with a pez dispenser (or, routing issues in MPLS)," *SIAM Journal on Computing*, 2005.

[16] A. Kumar, R. Rastogi *et al.*, "Algorithms for provisioning virtual private networks in the hose model," *IEEE/ACM TON*, 2002.

[17] S. Schmid, C. Avin *et al.*, "Splaynet: Towards locally self-adjusting networks," *IEEE/ACM Trans. Netw.*, 2016.

[18] "Packing strictly-shortest paths in a tree for QoS-aware routing," Tech. Rep., 2016. [Online]. Available: https://www.dropbox.com/s/487utisqbre3tfn/TR_PSSPTQAR.pdf?dl=0

[19] T. Eilam-Tzoreff, "The disjoint shortest paths problem," *Elsevier Discrete Applied Mathematics*, 1998.

[20] A. Schrijver, "Finding k disjoint paths in a directed planar graph," *SIAM Journal on Computing*, 1994.

[21] J. Yallouz and A. Orda, "Tunable QoS-aware network survivability," in *IEEE INFOCOM*, 2013.

[22] S. Knight, H. X. Nguyen *et al.*, "The Internet Topology Zoo," http://www.topology-zoo.org.

[23] B. M. Waxman, "Routing of multipoint connections," *IEEE Journal on Selected Areas in Communications*, 1988.