Single-Sink Fractionally Subadditive Network Design*

Guru Guruganesh[†1], **Jennifer Iglesias**[‡2], **R. Ravi**[3], **and Laura Sanità**[4]

1 **Carnegie Mellon University, Pittsburgh, PA, USA**
ggurugan@andrew.cmu.edu
2 **Carnegie Mellon University, Pittsburgh, PA, USA**
jiglesia@andrew.cmu.edu
3 **Carnegie Mellon University, Pittsburgh, PA, USA**
ravi@andrew.cmu.edu
4 **University of Waterloo, Waterloo, Canada**
laura.sanita@uwaterloo.ca

Abstract

We study a generalization of the Steiner tree problem, where we are given a weighted network G together with a collection of k subsets of its vertices and a root r. We wish to construct a minimum cost network such that the network supports one unit of flow to the root from every node in a subset simultaneously. The network constructed does not need to support flows from all the subsets simultaneously.

We settle an open question regarding the complexity of this problem for $k = 2$, and give a $\frac{3}{2}$-approximation algorithm that improves over a (trivial) known 2-approximation. Furthermore, we prove some structural results that prevent many well-known techniques from doing better than the known $O(\log n)$-approximation. Despite these obstacles, we conjecture that this problem should have an $O(1)$-approximation. We also give an approximation result for a variant of the problem where the solution is required to be a path.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, G.2.2 Graph Theory

Keywords and phrases Network design, single-commodity flow, approximation algorithms, Steiner tree

Digital Object Identifier 10.4230/LIPIcs.ESA.2017.46

1 Introduction

We study a robust version of a single-sink network design problem that we call the *Single-sink fractionally-subadditive network design* (f-SAND) problem. In an instance of f-SAND, we are given an undirected graph $G = (V, E)$ with edge costs $w_e \geq 0$ for all $e \in E$, a root node $r \in V$, and k *colors* represented as vertex subsets $C_i \subseteq V \setminus \{r\}$ for all $i \in [k]$, that wish to send flow to r. A feasible solution is an integer capacity installation on the edges of G, such that for every $i \in [k]$, each node in C_i can *simultaneously* send one unit of flow to r. Thus,

* Full version available at: https://arxiv.org/abs/1707.01487.

† This material is based upon work supported in part by National Science Foundation awards CCF-1319811,CCF-1536002, and CCF-1617790.

‡ This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. 2013170941.

the total flow sent by color i nodes is is $|C_i|$ while the flows sent from nodes of different colors are instead *non-simultaneous* and can share capacity. An optimal solution is a feasible one that minimizes the total cost of the installation.

The single-sink nature of the problem suggests a natural *cut*-covering formulation, namely:

$$\begin{aligned} \min \sum_{e\in E} w_e x_e \quad & \text{s.t.} \\ \sum_{e\in\delta(S)} x_e \geq f(S) \qquad & \forall S \subset V \setminus \{r\} \\ x \geq 0\,, \quad x \in \mathbb{Z}\,, & \end{aligned} \tag{IP}$$

where $\delta(S)$ denotes the set of edges with exactly one endpoint in S, and

$$f(S) := \max_{i\in[k]}\{|C_i \cap S|\} \tag{1}$$

for all $S \subseteq V \setminus \{r\}$. Despite having exponentially many constraints, the LP-relaxation of (IP) can be solved in polynomial-time because the separation problem reduces to performing k max-flow computations. The main challenge is to round the resulting solution into an integer solution.

Rounding algorithms for the LP relaxation of (IP) have been investigated by many authors, under certain assumptions of the function $f(S)$. Prominent examples are some classes of 0/1-functions (such as *uncrossable* functions), or integer-valued functions such as *proper* functions, or *weakly supermodular* functions [12, 19]; however, these papers consider arbitrary cut requirements rather than the single-sink connectivity requirements we study.

Our single-sink problem is a special case of a broader class of subadditive network design problems where the function f is allowed to be a general subadditive function. Despite their generality, the single-sink network design problem for general subadditive functions can be approximated within an $O(\log |V|)$ factor by using a tree drawn from the probabilistic tree decomposition of the metric induced by G using the results of Fakcharoenphol, Rao, and Talwar [8], and installing the required capacity on the tree edges. Hence, a natural direction is to consider special cases of such subadditive cut requirement functions.

Our function $f(S)$ defined in (1) is an interesting and important special case of subadditive functions. It was introduced as XOS-functions (max-of-sum functions) in the context of combinatorial auctions by Lehman et al. [20]. Feige [9] proved that this function is equivalent to fractionally-subadditive functions which are a strict generalization of submodular functions (hence the title). These functions have been extensively studied in the context of learning theory and algorithmic game theory [1, 3, 20]. Our work is an attempt to understand their behavior as single-sink network design requirement functions.

f-SAND was first studied by Oriolo et al. [21] in the context of *robust* network design, where the goal is to install minimum cost capacity on a network in order to satisfy a given set of (non-simultaneous) traffic demands among terminal nodes. Each subset C_i can in fact be seen as a way to specify a distinct traffic demand that the network would like to support. They observed that f-SAND generalizes the Steiner tree problem: an instance of the Steiner tree problem with $k+1$ terminals $t_1, \ldots, t_{k+1}$ is equivalent to the f-SAND instance with $r := t_{k+1}$ and $C_i := \{t_i\}$ for all $i \in [k]$. This immediately shows that f-SAND is NP-hard (in fact, APX-hard [6]) when k is part of the input. The authors in [21] strengthened the hardness result by proving that f-SAND is NP-hard even if k is not part of the input, and in particular for $k = 3$ (if $k = 1$ the problem is trivially solvable in polynomial-time by computing a shortest path tree rooted at r). From the positive side, they observed that there is a trivial

k-approximation algorithm, that relies on routing via shortest paths, and an $O(\log |\cup_i C_i|)$-approximation algorithm using metric embeddings [8, 17]. The authors conclude their paper mentioning two open questions, namely whether the problem is polynomial-time solvable for $k = 2$, and whether there exists an $O(1)$-approximation algorithm.

1.1 Our results

1. In this paper, we answer the first open question in [21] by showing that f-SAND is NP-hard for $k = 2$ via a reduction from SAT.
2. We give a $\frac{3}{2}$-approximation algorithm for this case ($k = 2$). This is the first improvement over the (trivial) k-approximation obtained using shortest paths for any k. Our approximation algorithm is based on pairing terminals of different groups together, and therefore reducing to a suitable minimum cost matching problem. While the idea behind the algorithm is natural, its analysis requires a deeper understanding of the structure of the optimal solution.
3. We also introduce an interesting variant of f-SAND, which we call the Latency-f-SAND problem, where the network built is restricted to being a *path* with the root r being one of the endpoints (f-SAND-path). We show a $O(\log^2 k \log n)$-approximation using a new reformulation of the problem that allows us to exploit techniques recently developed for *latency* problems [4].
4. While being a generalization of well-studied problems, f-SAND does not seem to admit an easy $O(1)$-approximation via standard LP-rounding techniques for arbitrary values of k. We prove some structural results that highlight the difficulty of the general problem (see full version [18]). In particular, we show a family of a instances providing a super-constant gap between an optimal f-SAND solution and an optimal *tree*-solution, i.e., a solution whose support is a tree – this rules out many methods that output a solution with a tree structure. The bulk of the construction was shown in [13] and we amend it to our problem using a simple observation. Furthermore, we give some evidence that an iterative rounding approach (as in Jain's fundamental work [19]) is unlikely to work. This follows by considering a special class of Kneser Graphs, where the LP seems to put low fractional weight on each edge in an extreme point.
5. **Open Questions.** We offer the following conjecture as our main open question.[1]
   ▶ **Conjecture 1.** *There exists an $O(1)$-approximation algorithm for the* f-SAND *problem.*
   Although standard LP-based approaches seem to fail in providing a constant factor approximation, the worst known integrality gap example we are aware of yields a (trivial) lower bound of 2 on the integrality gap of (IP) for f-SAND. A related open question is if there is an instance of f-SAND for which the integrality gap of (IP) is greater than 2.

1.2 Related work

Network design problems where the goal is to build a minimum cost network in order to support a given set of flow demands, have been extensively studied in the literature (we refer to the survey [5]). There has been a huge amount of research focusing on the case the set of demands is described via a polyhedron (see e.g. [2]). In this context a very popular model is the *Virtual private network* [7, 11], for which many approximation results have been developed (see e.g. [14, 15, 16] and the references therein). For the case where the set of

[1] Although the problem is known in some circles, it has not been explicitly stated as a conjecture. We do so here, in the hopes that it will encourage others to work on this problem.

demands is instead given as a (finite) discrete list, the authors in [21] developed a constant factor approximation algorithm on ring networks, and proved that f-SAND is polynomial-time solvable on ring networks.
Regarding the formulation (IP), Goemans and Williamson [12] gave a $O(\log(f_{max}))$-approximation algorithm for solving (IP) whenever $f(S)$ is an integer-valued proper function that can take values up to f_{max}, based on a primal-dual approach. Subsequently, Jain [19] improved this result by giving a 2-approximation algorithm using iterative rounding of the LP-relaxation. Recently, a strongly-polynomial time FPTAS to solve the LP-relaxation of (IP) with proper functions has been given in [10].

2 3/2-approximation for the two color case

The goal of this section is to give a $\frac{3}{2}$-approximation algorithm for SAND with two colors. We remark that our algorithm bypasses the difficulties mentioned in the previous section. In particular, the final output is not a tree.

2.1 Simplifying Assumptions

We will refer to the two colors as *green* and *blue*, and let $C_G \subset V$ denote the set of green terminals, and $C_B \subset V$ denote the set of blue terminals. Without loss of generality, we will assume that $|C_G| = |C_B|$, i.e., the cardinality of green terminals is equal to the cardinality of blue terminals (if not, we could easily add dummy nodes at distance 0 from the root). Furthermore, by replacing each edge in the original graph with $|C_G|$ parallel edges of the same cost, we can assume that in a feasible solution the capacity installed on each edge must be either 0 or 1. This means that each edge is used by *at most* one terminal of C_G (resp. C_B) to carry flow to the root. Lastly, we assume that every terminal in C_G shares at least one edge with some terminal in C_B in the optimal solution.[2]

Let OPT denote an optimal solution to a given instance of SAND with two colors. We start by developing some results on the structure of OPT, that will be crucial to analyze our approximation algorithm later.

2.2 Understanding the structure of OPT

A feasible solution of a SAND instance consists of a (integer valued) capacity installation on the edges that allows for a flow from the terminals to the root. Given a feasible solution, each terminal will send its unit of flow to t on a single path. Let us call the collection of such paths a *routing* associated with the feasible solution. The first important concept we need is the concept of splits.

2.2.1 Shared Edges and Splits

Given a routing, for each terminal $g \in C_G$ (and $b \in C_B$ respectively) let P_g (P_b) denote the path along which g (b) sends flow to the root; i.e. $P_g := \{g = x_0, x_1, \ldots, x_{|P_g|} = r\}$. We say that an edge e is **shared** if the paths of two terminals of different color contain the edge. We say that $g \in C_G$ and $b \in C_B$ are **partners** with respect to a shared edge $e = uv$, if their respective paths use the edge e; i.e. $e \in P_g \cap P_b$.

[2] We can easily ensure this e.g. by modifying our instance as follows: we add a dummy node r' which is only connected to r with $|C_G|$ parallel edges of 0 cost, and we make r' be the new root. In this way, all terminals will use one copy of the edge (r, r').

▶ **Definition 2.** A **split** in the path P_g is a maximal set of consecutive edges of the path such that g is partnered with some b on all the edges of this set.

If $\{(x_i, x_{i+1}), (x_{i+1}, x_{i+2}), \ldots, (x_{i+j-1}, x_{i+j})\}$ is a split in the path $P_g = \{g = x_0, x_1, \ldots, x_{|P_g|} = r\}$ for $g \in C_G$, then there exists a unique terminal $b \in C_B$ such that P_b contains the edges $\{(x_i, x_{i+1}), (x_{i+1}, x_{i+2}), \ldots, (x_{i+j-1}, x_{i+j})\}$, P_b does not contain the edge (x_{i-1}, x_i), and if $x_{i+j} \neq r$ then P_b does not contain the edge (x_{i+j}, x_{i+j+1}). By our assumptions, the terminal b is unique as each edge is used by at most one terminal of each color.

Since the flow is going from a terminal g to r, the path P_g naturally induces an orientation on its edges given by the direction of the flow, even though the edges are undirected. Of course, the paths of different terminals could potentially induce opposite orientations on (some of) the shared edges (see Figure 1).

▶ **Definition 3.** A split is **wide**, if the paths of the two terminals that are partners on the edges of the split induce opposite orientations on the edges. A split is **thin**, if the paths of the two terminals that are partners on the edges of the split induce the same orientation on the edges.

The above notions are well defined for any routing with respect to a feasible solution. Now, we focus on the structure of an optimal routing, i.e., a routing with respect to an optimal solution. For the rest of this section, we let $\{P_g\}_{g \in C_G}$ and $\{P_b\}_{b \in C_B}$ be an optimal routing. The following lemma is immediate.

▶ **Lemma 4.** *Let $\{(x_i, x_{i+1}), (x_{i+1}, x_{i+2}), \ldots, (x_{i+j-1}, x_{i+j})\}$ be a split in the path P_g (for some $g \in C_G$). The edges of the split form a shortest path from x_i to x_{i+j}.*

Proof. If not, we could replace this set of edges with the set of edges of a shortest path from x_i to x_{i+j}, in both P_g and P_b, where b is the partner of g on the split. Therefore, we can install one unit of capacity on these edges, and remove the unit of capacity from the edges of the split. We get another feasible solution with smaller cost, a contradiction to the optimality of our initial solution. ◀

2.2.2 Split Graph

A consequence of Lemma 4 is each split is entirely characterized by the endpoints of the split and the terminals that share them. We denote each split by a tuple (u, v, g, b) which states that there is a shortest path between u and v whose edges are shared by g and b.

Let $\mathbb{S}$ denote the set of all splits in the optimal routing. We construct a directed graph $G^{\mathbb{S}}$ whose vertex set corresponds to $V = \mathbb{S} \cup C_G \cup C_B$ (i.e. the vertex set contains one vertex for each split and one vertex for each terminal). For each $g \in C_G$, we place a directed *green* edge going between two consecutive splits in P_g. Specifically, if $\{(x_i, x_{i+1}), \ldots, (x_{i+j-1}, x_{i+j})\}$ and $\{(x_{i'}, x_{i'+1}), \ldots, (x_{i'+j'-1}, x_{i'+j'})\}$ are two splits in P_g with $i < i'$, we say that they are consecutive if the subpath from x_{i+j} to $x_{i'}$ does not contain any split. In this case, we place a directed edge in $G^{\mathbb{S}}$ whose tail is the vertex corresponding to the first split, and whose head is the vertex corresponding to the second one. Similarly, for each $b \in C_B$ we place a directed *blue* edge between vertices of consecutive splits that appear in P_b. Furthermore, for each $g \in C_G$ (resp. $b \in C_B$) we place a directed green (resp. blue) edge from g (resp. b) to the vertex corresponding to the first split on the path P_g (resp. P_b), if any. This graph is denoted as the **Split Graph** (see Figure 1).

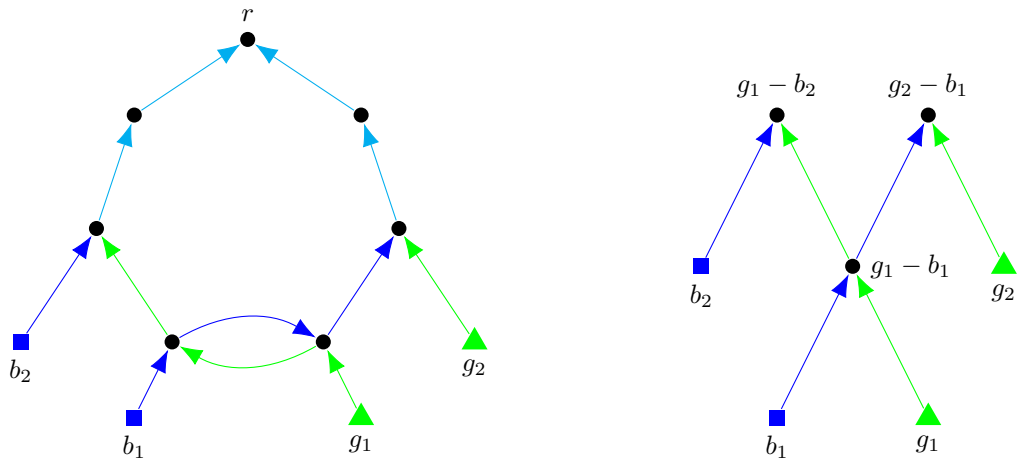

Figure 1 The above left graph (where each undirected edge is supposed to have unit capacity) shows an optimal routing for some f-SAND instance. Note that b_1 and g_2 (resp. b_2 and g_1) send flow to r going counterclockwise (resp. clockwise) on the edges of the cycle. The path P_{b_1} contains two splits: the first is wide (b_1 is partnered with g_1), the second is thin (b_1 is partnered with g_2). The graph on the right is the Split Graph for the optimal solution on the left. The pair of vertices g_1, b_1 and the pair of vertices g_2, b_2 constitute the fresh pairs.

Each split indicates that two terminals of different colors are sharing the capacity on a set of edges in an optimal routing. Hence, each split-vertex in $G^{\mathbb{S}}$ has indegree 2 (in particular, one edge of each color). Furthermore, each split-vertex in $G^{\mathbb{S}}$ has outdegree either 0 or 2; if it has two outgoing edges, one is green and one is blue. Similarly, each terminal has indegree 0, and outdegree 1 (as we assume that each terminal shares at least one edge).

2.2.3 Fresh Pairs

We need one additional definition before proceeding to the algorithm.

▶ **Definition 5.** An **$\mathbb{S}$-alternating sequence** is a sequence of vertices of the Split Graph $\{v, s_1, s_2, \ldots, s_h, w\}$ with $h \geq 1$, that satisfies the following:

(i) (v, s_1) and (w, s_h) are directed edges in $G^{\mathbb{S}}$ and v, w are terminals of different color.

(ii) For all even $i \geq 2$, (s_i, s_{i-1}) and (s_i, s_{i+1}) are both directed edges in $G^{\mathbb{S}}$ with opposite colors.

We call the path obtained by taking the edges in (i) and (ii) an **$\mathbb{S}$-alternating path**. We call (v, w) a **fresh pair** if they are the endpoints of an $\mathbb{S}$-alternating path.

By definition, in an $\mathbb{S}$-alternating sequence the vertices $s_1, \ldots, s_h$ are all split-vertices, and h is odd. We remark here that an $\mathbb{S}$-alternating path is *not* a directed path. (See again Figure 1).

▶ **Lemma 6.** *We can find a set of edge-disjoint $\mathbb{S}$-alternating paths in the Split Graph such that each terminal is the endpoint of exactly one path in this set.*

Proof. We construct the desired set as follows. For each vertex $g \in C_G$, there is a unique outgoing edge to a split vertex $s \in \mathbb{S}$ (as we assume every terminal participates in a split). Since each split-vertex has indegree 2, s has another ingoing edge coming from a different vertex w. If $w \in C_B$, then (v, w) is a fresh pair and we have found an $\mathbb{S}$-alternating sequence $\{v, s, w\}$. If w is a split-vertex, then it has another outgoing edge to a different split-vertex

s', which in its turn has another incoming edge from a different vertex w'. We continue to build an alternating sequence (and a corresponding alternating path) in this way until it terminates in a terminal. Since the path is of even length and the colors alternate, we can conclude that this will terminate in a terminal of opposite color. We remove the edges of this path from the Split Graph, and iterate the process. Each terminal will belong to exactly one $\mathbb{S}$-alternating path, as it has outdegree exactly 1, and all the paths are edge-disjoint, proving the lemma. ◀

2.3 The Algorithm

We are now ready to present our *matching algorithm*. The algorithm has two steps. First, construct a complete bipartite graph $\mathcal{H}$ with the bipartitions C_G and C_B, where the weight on the edge $(g, b) \in C_G \times C_B$ is equal to the cost of the Steiner tree in G connecting g, b and the root. Note that the graph $\mathcal{H}$ can be computed in polynomial time, since a Steiner tree on 3 vertices can be easily computed in polynomial time.

Second, find a minimum-weight perfect matching $\mathcal{M}$ in $\mathcal{H}$, and for each edge $(g, b) \in \mathcal{M}$ install (cumulatively) one unit of capacity on each edge of G that is in the Steiner tree associated to the edge $(g, b) \in \mathcal{M}$. The capacity installation output by this procedure is a feasible solution to f-SAND, and has total cost equal to the weight of $\mathcal{M}$.

▶ **Lemma 7.** *The matching algorithm is a $\frac{3}{2}$-approximation algorithm.*

Proof. First, we partition OPT into four parts; let w_b (and w_g respectively) be the cost of the edges which are used only by blue (green respectively) terminals in OPT, and let w_t (w_d) be the cost of edges in thin (wide) splits in OPT. Thus, $w(OPT) = w_b + w_g + w_t + w_d$. By Lemma 6, we can extract from the Split Graph associated to OPT a set of $\mathbb{S}$-alternating paths such that each terminal is contained in exactly one fresh pair. Consider the matching $\mathcal{M}_1$ determined by the set of fresh pairs found by the aforementioned procedure. We will now bound the weight of $\mathcal{M}_1$.

▶ **Claim 8.** *The weight of the matching formed by connecting the fresh pairs is at most*

$$\frac{3}{2} \cdot w_b + \frac{3}{2} \cdot w_g + 1 \cdot w_t + 3 \cdot w_d.$$

Proof. Let (g, b) be a fresh pair and $(g, s_1, \ldots, s_h, b)$ be the corresponding $\mathbb{S}$-alternating sequence. The edges of the associated $\mathbb{S}$-alternating path naturally correspond to paths in G composed by non-shared edges (that connect either the endpoints of two different splits, or one terminal and one endpoint of a split). These paths together with the edges of the wide splits in the sequence, naturally yield a path $P(b, g)$ in G connecting g and b.

If we do this for all fresh pairs, we obtain that the total cost of the paths $P(b, g)$ is upper bounded by $1 \cdot w_b + 1 \cdot w_g + 2 \cdot w_d$. The reason for having a coefficient of 2 in front of w_d is because the $\mathbb{S}$-alternating paths of Lemma 6 are edge-disjoint, but not necessarily vertex-disjoint: however, since each split-vertex has at most 4 edges incident into it, it can be part of at most 2 $\mathbb{S}$-alternating paths.

Using the aforementioned connection, we can move all terminals in C_G to their partners in C_B. Subsequently, we connect them to the root using the P_b for all $b \in C_B$. This connection to the root will incur a cost of $1 \cdot w_b + 1 \cdot w_d + 1 \cdot w_t$. Combining this together, we get a total cost of $2 \cdot w_b + 1 \cdot w_g + 1 \cdot w_t + 3 \cdot w_d$. Analogously, if we connect the partners in C_G to the root using the the path P_g for all $g \in C_G$, we will incur a total cost of $1 \cdot w_b + 2 \cdot w_g + 1 \cdot w_t + 3 \cdot w_d$.

Since the sum of the cost of the Steiner trees connecting the fresh pairs to the root is no more than either of these two values, we can bound the weight of $\mathcal{M}_1$ by their average:

$$\frac{3}{2} \cdot w_b + \frac{3}{2} \cdot w_g + 1 \cdot w_t + 3 \cdot w_d. \qquad \blacktriangleleft$$

▶ **Claim 9.** *There exists a matching in $\mathcal{H}$ of weight at most $1 \cdot w_b + 1 \cdot w_g + 2 \cdot w_t$.*

Proof. Consider the flow routed on the optimal paths by the set of all terminals $C_G \cup C_B$. We modify the flow (and the corresponding routing) as follows. Whenever two terminals traverse a wide-split, re-route the flows so as to not use the wide-split. This is always possible as the two terminals traverse these edges in opposite directions (by definition of wide splits). This re-routing ensures that all the edges of wide-splits are not used anymore in the resulting paths. However, thin-splits which contained terminals of different colors passing in the same direction, might now contain two terminals of the same color passing through the edges. This means that these edges will be used twice (or must have twice the capacity installed). All other edges do not need to have their capacity changed. Thus, the resulting flow can be associated with a feasible solution of cost at most $1 \cdot w_b + 1 \cdot w_g + 2 \cdot w_t + 0 \cdot w_d$. This flow corresponds to all vertices directly connecting to the root as any shared edge is counted twice. Hence, this is a bound on any matching in $\mathcal{H}$. $\blacktriangleleft$

The average weight of the above matchings is an upper-bound on the minimum weight of a matching in $\mathcal{H}$. Hence, the weight of $\mathcal{M}$ is at most

$$\begin{aligned}&\frac{1}{2} \cdot \left(\frac{3}{2} \cdot w_b + \frac{3}{2} \cdot w_g + 1 \cdot w_t + 3 \cdot w_d\right) + \frac{1}{2} \cdot \left(1 \cdot w_b + 1 \cdot w_g + 2 \cdot w_t + 0 \cdot w_d\right) \\ &\leq \frac{3}{2} \cdot \left(w_b + w_g + w_t + w_d\right)\end{aligned}$$

Therefore, the matching algorithm is $\frac{3}{2}$-approximation algorithm. $\blacktriangleleft$

3 Hardness for two colors

We prove that the SAND problem is NP-hard even with just two colors.

▶ **Theorem 10.** *The SAND problem with 2 colors is NP-hard.*

Proof. We use a reduction from a variant of the Satisfiability (SAT) problem, where each variable can appear in at most 3 clauses, that is known to be NP-hard [22]. Formally, in a SAT instance we are given m clauses $K_1, \ldots, K_m$, and p variables $x_1, \ldots, x_p$. Each clause K_j is a disjunction of some *literals*, where a literal is either a variable x_i or its negation $\bar{x}_i$, for some i in $1, \ldots, p$. The goal is to find a truth assignment for the variables that satisfies all clauses, where a clause is satisfied if at least one of its literals takes value *true*. In the instances under consideration, each variable x_i appears in at most 3 clauses, either as a literal x_i, or as a literal $\bar{x}_i$. It is not difficult to see that, without loss of generality, we can assume that every variable appears in exactly 3 clauses. Furthermore, by possibly replacing all occurrences of x_j with $\bar{x}_j$ and vice versa, we can assume that each variable x_i appears in exactly one clause in its negated form ($\bar{x}_i$).

Given such a SAT instance, we define an instance of SAND as follows (see Fig. 2). We construct a graph $G = (V, E)$ by introducing one sink node r, one node k_j for each clause K_j, and 7 distinct nodes y_i^ℓ, ($\ell = 1, \ldots, 7$), for each variable x_i. That is,

$$V := \{r\} \cup \{k_1, \ldots, k_m\} \cup \left\{ \bigcup_{i=1}^{p} \{y_i^1, y_i^2, y_i^3, y_i^4, y_i^5, y_i^6, y_i^7\} \right\}$$

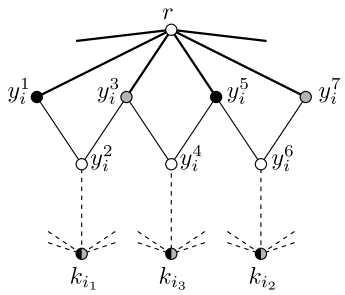

Figure 2 The picture shows the subgraph introduced for every variable x_i. Bold edges have cost 2, solid edges have cost 1, and dashed edges have cost M. Black circles indicate nodes in C_1, and grey circles indicate nodes in C_2. Nodes in $C_1 \cap C_2$ are colored half-black and half-grey.

The set of edges E is the disjoint union of three different sets, $E := E_1 \cup E_2 \cup E_3$, where:

$$E_1 := \bigcup_{i=1}^{p} \left\{ \bigcup_{\ell=1}^{4} \{r, y_i^{2\ell-1}\} \right\}; \quad E_2 := \bigcup_{i=1}^{p} \left\{ \bigcup_{\ell=1}^{6} \{y_i^{\ell}, y_i^{\ell+1}\} \right\}.$$

To define the set E_3, we need to introduce some more notation. For a variable x_i, we let i_1 and i_2 be the two indices of the clauses containing the literal x_i, and we let i_3 be the index of the clause containing the literal $\bar{x}_i$. We then have

$$E_3 := \bigcup_{i=1}^{p} \left\{ \{y_i^2, k_{i_1}\}, \{y_i^4, k_{i_3}\}, \{y_i^6, k_{i_2}\} \right\}.$$

We assign cost 2 to the edges in E_1, unit cost to the edges in E_2, and a big cost $M >> 0$ to the edges of E_3 (in particular, $M > 2m + 8p$). Finally, we let the color classes[3] be defined as:

$$C_1 := \{k_1, \ldots, k_m\} \cup \left\{ \bigcup_{i=1}^{p} \{y_i^1, y_i^5\} \right\}; \quad C_2 := \{k_1, \ldots, k_m\} \cup \left\{ \bigcup_{i=1}^{p} \{y_i^3, y_i^7\} \right\}.$$

We claim that there exists an optimal solution to the SAND instance of cost at most $(M+2)m + 8p$ if and only if there is a truth assignment satisfying all clauses for the SAT instance.

3.1 Completeness

First, let us assume that the SAT instance is satisfiable. For each clause K_j, we select one literal that is set to *true* in the truth assignment. We define the paths for our terminal nodes in C_1 as follows. For each node $y \in \bigcup_{i=1}^{p} \{y_i^1, y_i^5\}$, we let the flow travel from y to r along the edge $\{y, r\}$. For each k_j, we let the flow travel to r on a path P_1^j, that we define based on the literal selected for K_j. Specifically, let x_i be the variable corresponding to the literal selected for the clause K_j. Then:

- if $K_j = K_{i_1}$, we let P_1^j be the path with nodes $\{k_j, y_i^2, y_i^3, r\}$,
- if $K_j = K_{i_2}$, we let P_1^j be the path with nodes $\{k_j, y_i^6, y_i^7, r\}$,
- if $K_j = K_{i_3}$, we let P_1^j be the path with nodes $\{k_j, y_i^4, y_i^3, r\}$.

[3] We here have $C_1 \cap C_2 \neq \emptyset$. However, the reduction can be easily modified to prove hardness of instances where $C_1 \cap C_2 = \emptyset$, by simply adding for all j two nodes k_j^1, k_j^2 adjacent to k_j with an edge of zero cost, and by letting k_j^1 (resp. k_j^2) be in C_1 (resp. C_2) instead of k_j.

We define the paths for our terminal nodes in C_2 similarly. For each node $y \in \bigcup_{i=1}^{p}\{y_i^3, y_i^7\}$, we let the flow travel from y to r along the edge $\{y, r\}$. For each k_j, we let the flow travel to r on a path P_2^j defined as follows. Let x_i be the variable corresponding to the literal selected for the clause K_j. Then:

- if $K_j = K_{i_1}$, we let P_2^j be the path with nodes $\{k_j, y_i^2, y_i^1, r\}$,
- if $K_j = K_{i_2}$, we let P_2^j be the path with nodes $\{k_j, y_i^6, y_i^5, r\}$,
- if $K_j = K_{i_3}$, we let P_2^j be the path with nodes $\{k_j, y_i^4, y_i^5, r\}$.

Note that the paths of terminals belonging to the same color set do not share edges. In fact, by construction, the paths of two terminals in C_1 could possibly share an edge only if for two distinct clauses $K_j \neq K_{j'}$ we selected a literal corresponding to the same variable x_i, and we have $K_j = K_{i_1}$ and $K_{j'} = K_{i_3}$, since in this case the paths P_1^j and $P_1^{j'}$ would share the edge $\{y_i^3, r\}$. However, selecting x_i for K_{i_1} means x_i takes value *true* in the truth assignment, while selecting x_i for K_{i_3} means x_i takes value *false* in the truth assignment, which is clearly a contradiction. A similar observation applies to paths of terminals in C_2. It follows that installing one unit of capacity on every edge that appears in (at least) one selected path is enough to support the flow of both color sets. The total installation cost is exactly $8p + (M + 2)m$.

3.2 Soundness

Suppose there is an optimal solution to the SAND instance of cost at most $(M + 2)m + 8p$. Let $\mathcal{S}$ denote such solution. Since the support of any feasible solution has to include at least one distinct edge of cost M for each node k_j, and $M > 2m + 8p$, it follows that $\mathcal{S}$ has exactly m edges of cost M in its support, each with one unit of capacity installed. Hence, if we denote by P_1^j (resp. P_2^j) the path used by k_j to send flow to r with terminals in C_1 (resp. C_2), we have the following fact.

Fact 1. For each $j = 1, \ldots, m$, the paths P_1^j and P_2^j from k_j to r share the first edge.

We use this insight to construct a truth assignment for the SAT variables. Specifically, let y_i^ℓ be the endpoint of the first edge of P_1^j and P_2^j. We set x_i to *true* if $y_i^\ell = y_i^2$ or if $y_i^\ell = y_i^6$, and we set x_i to *false* if $y_i^\ell = y_i^4$. We repeat this for all clauses $j = 1, \ldots, m$, and we assign an arbitrary truth value to all remaining variables, if any. In order to finish the proof, we have to show that this assignment is consistent for all $i = 1, \ldots, p$. To this aim, let us say that a variable x_i is *in conflict* if there is a node k_j sending flow to r on a path whose first edge has endpoint y_i^4, and there is node $k_{j'} \neq k_j$ sending flow to r on a path whose first edge has endpoint y_i^2 or y_i^6. Note that our assignment procedure is consistent and yields indeed a valid truth assignment if and only if there is no variable in conflict.

We now make a few claims on the structure of $\mathcal{S}$, that will be useful to show that no variable can be in conflict. Next fact follows from basic flow theory.

Fact 2. Without loss of generality, we can assume that the flow sent from terminals in C_1 (resp. C_2) to r, does not induce directed cycles.

▶ **Claim 11.** *Without loss of generality, we can assume that every terminal sends flow to r on a path that contains exactly one node $y \in \bigcup_{i=1}^{p}\{y_i^1, y_i^3, y_i^5, y_i^7\}$.*

We defer the proof of this claim which is central to the remaining proof to the end. Let G_i be the subgraph of G induced by the nodes $\{r, y_i^1, \ldots, y_i^7\}$, and let χ_i be the total cost of the capacity that $\mathcal{S}$ installs on the subgraph G_i. Note that, by Fact 1, the cost of $\mathcal{S}$ is

$m \cdot M + \sum_{i=1}^{m} \chi_i$. We will use Claim 1 to give a bound on the value χ_i. To this aim, let n_i be the number of nodes k_j whose path P_1^j contains edges of G_i. Note that $0 \leq n_i \leq 3$, and each k_j contributes to exactly one n_i, for some $i = 1, \ldots, p$.

▶ **Claim 12.** *We have $\chi_i \geq 8 + 2n_i$, with the inequality being strict if the variable x_i is in conflict.*

Claim 12 finishes our proof, since it implies that the cost of $\mathcal{S}$ is at least

$$m \cdot M + \sum_{i=1}^{p} \chi_i \geq m \cdot M + \sum_{i=1}^{p} (8 + 2n_i) = m \cdot M + 8p + 2m,$$

with the inequality being tight if and only if there is no variable in conflict. ◀

4 Latency SAND

By adapting a construction from [13], there is a $\Omega(\log n)$ gap between the tree and graph version of f-SAND. This naturally raises the question of approximating f-SAND when the solution must be restricted to different topologies. In this section, we consider the f-SAND when the output topology must be a path. Since this variant of f-SANDis not easy to solve on a tree, it is not clear how to solve it using tree metrics.

▶ **Definition 13.** In the *latency*-f-SAND problem, we are given an instance of f-SAND, but require the output to be a path with the root r as one of its endpoints. Our goal is output a minimum cost path, where the cost of an edge is $w_e \cdot$ (load on e). The load on an edge is the maximum number of nodes of one color it separates from the root.

We assume that the lengths are integers and polynomially bounded in the input and give a time-indexed length formulation for this problem. This linear programming formulation was introduced by Chakrabarty and Swamy [4] for orienteering problems.

The Linear Programming Formulation for Latency-f-SAND.

$$\min \quad \sum_{j,t} t \cdot x_{j,t} \qquad (\mathrm{LP}_{\mathcal{P}}^{b})$$

$$\text{s.t.} \quad \sum_{t} x_{j,t} \geq 1 \qquad \forall j \in [m] \qquad (2)$$

$$\sum_{P \in \mathcal{P}_{b \cdot t}} z_{P,t} \leq 1 \qquad \forall t \in [T] \qquad (3)$$

$$\sum_{P \in \mathcal{P}_{b \cdot t} : j \in P} z_{P,t} \geq \sum_{t' \leq t} x_{j,t'} \qquad \forall j \in [m], t \in [T] \qquad (4)$$

$$x, z \geq 0$$

We assume without loss of generality, that $|C_i| = m$ for all $i \in [k]$. $\mathcal{P}_t$ denotes the set of paths of weight at most t starting from the root. Since the lengths are polynomially bounded, we can contain a variable for each possible length (we denote T to be the maximum possible length). We use $j \in P_t$ to indicate that the path P_t contains j terminals of each color. The variable $x_{j,t}$ indicates that we have seen j terminals of each color by time t and $z_{P,t}$ indicates that we use path P to visit the terminals at time t.

▶ **Lemma 14.** *The linear program $LP_{\mathcal{P}}^{b}$ is a relaxation of Latency-*f-SAND*for $b \geq 1$.*

Proof. We show that the contraints and objective are valid for any feasible solution to Latency-f-SAND.

- Constraint 2 ensures that j terminals of each color are covered at some given time period, for every $j \in [m]$.
- Constraint 3 ensures that only one path is (fractionally) picked for each time period t.
- Constraint 4 indicates that we must have picked a path P that covers j terminals by time t if $\sum_{t' \leq t} x_{j,t'} = 1$.
- The objective function correctly captures the cost of the path. For an integer solution, $x_{j,t} = 1$ indicates that time t is the first time j terminals of each color are present in the path. Thus the objective counts the prefix length t^1 corresponding to where $x_{1,t^1} = 1$ in all m of the terms, the next prefix of length $t^2 - t^1$ in $m - 1$ of them and so on. This accurately accounts for the loads in these segments of the path according to the objective function in f-SAND. Finally, $b \geq 1$ only allows the paths to be of lengths longer by a factor of b so keeps the optimal solution feasible. ◀

First, we can relax the above LP by replacing $\mathcal{P}_t$ with $\mathcal{T}_t$ which is the set of all trees of size at most t. This is a relaxation as $\mathcal{P}_t \subseteq \mathcal{T}_t$. Lemma 15, shows that we can round $\text{LP}^b_{\mathcal{T}}$ to get a $O(b)$ approximation to latency-f-SAND.

▶ **Lemma 15.** *Given a fractional solution (x, z) to $LP^b_{\mathcal{T}}$, we can round it to a solution to latency-*f-SAND *with cost at most $O(b)$ times the cost of $LP^b_{\mathcal{T}}$.*

We defer the proof to the full version [18] due to space constraints but briefly sketch the argument. Roughly, we sample the trees at geometric intervals and "eulerify" them to produce a solution whose cost is not too much larger than the LP-objective.

Despite, being able to round the LP, we cannot hope to solve it effeciently due to the exponential number of variables in the primal. We will use the dual to obtain a solution to a relaxed version of the primal.

$$\max \quad \sum_j \alpha_j - \sum_t \beta_t \tag{Dual$^b_{\mathcal{P}}$}$$

$$\text{s.t.} \quad \alpha_j \leq t + \sum_{t' \geq t} \theta_{j,t'} \qquad \forall j, t \tag{5}$$

$$\sum_{j \in P} \theta_{j,t} \leq \beta_t \qquad \forall t, P \in \mathcal{P}_{bt} \tag{6}$$

$$\alpha, \beta, \theta \geq 0. \tag{7}$$

Following [4] it is suffient that an "approximate separation oracle" in the sense of Lemma 16 is sufficient to compute an optimal solution to $\text{LP}^b_{\mathcal{T}}$.

▶ **Lemma 16.** *Given a solution (α, β, θ), we can show that either (α, β, θ) is a solution to $Dual^1_{\mathcal{T}}$ or find a violated inequality for (α, β, θ) for $Dual^b_{\mathcal{T}}$ for $b = O(\log^2 k \log n)$.*

Once again, we defer the proof to the full version [18], but sketch the argument. To efficiently separate, we observe that constraint 6 can be recast as a covering Steiner tree problem. Using approximation algorithms for this problem, we find a violated inequality for a (stronger) constraint. This results in the "approximate separation oracle".

▶ **Theorem 17.** *There exists a $O(\log^2 k \log n)$ approximation to the Latency-SAND problem.*

Proof. Combining Lemma 3.2 of [4] with Lemma 16, we can now compute an ϵ-additive optimal solution to $\text{LP}^b_{\mathcal{T}}$ for $b = O(\log^2 k \log n)$. Using Lemma 15, we then achieve an $O(b)$ approximation for our problem. ◀

References

1 M. Balcan, F. Constantin, S. Iwata, and L. Wang. Learning valuation functions. In *Conference on Learning Theory*, volume 23, pages 4–1, 2012.

2 W. Ben-Ameur and H. Kerivin. Routing of uncertain demands. *Optimization and Engineering*, 3:283–313, 2005.

3 K. Bhawalkar and T. Roughgarden. Welfare guarantees for combinatorial auctions with item bidding. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 700–709. Society for Industrial and Applied Mathematics, 2011.

4 D. Chakrabarty and C. Swamy. Facility location with client latencies: LP-based techniques for minimum-latency problems. *Mathematics of Operations Research*, 41(3):865–883, 2016.

5 C. Chekuri. Routing and network design with robustness to changing or uncertain traffic demands. *SIGACT News*, 38(3):106–128, 2007.

6 M. Chlebik and J. Chlebikova. Approximation hardness of the steiner tree problem on graphs. *Proc. of the Scandinavian Workshop on Algorithm Theory*, pages 170–179, 2002.

7 N. G. Duffield, P. Goyal, A. G. Greenberg, P. P. Mishra, K. K. Ramakrishnan, and J. E. van der Merwe. A flexible model for resource management in virtual private networks. *Proceedings of SIGCOMM*, 29:95–108, 1999.

8 J. Fakcharoenphol, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *Journal of Computer and System Sciences*, 69:485–497, 2004.

9 U. Feige. On maximizing welfare when utility functions are subadditive. *SIAM Journal on Computing*, 39(1):122–142, 2009.

10 A. E. Feldmann, J. Könemann, K. Pashkovich, and L. Sanità. Fast approximation algorithms for the generalized survivable network design problem. *Proceedings of ISAAC (International symposium on algorithms and computation)*, pages 33:1– 33:12, 2016.

11 J. Fingerhut, S. Suri, and J. Turner. Designing least-cost nonblocking broadband networks. *Journal of Algorithms*, 24(2):287–309, 1997.

12 M. X. Goemans and D. P. Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24(2):296–317, 1995.

13 N. Goyal, N. Olver, and F. B. Shepherd. Dynamic vs. oblivious routing in network design. *Algorithmica*, 61(1):161–173, 2011.

14 N. Goyal, N. Olver, and F. B. Shepherd. The VPN conjecture is true. *Journal of the ACM*, 60(3):17:1–17:17, June 2013.

15 F. Grandoni, T. Rothvoß, and L. Sanità. From uncertainty to non-linearity: Solving virtual private network via single-sink buy-at-bulk. *Mathematics of Operations Research*, 36(2):185–204, 2011.

16 A. Gupta, J. Kleingerg, R. Kumar, B. Rastogi, and B. Yener. Provisioning a virtual private network: A network design problem for multicommodity flow. *Proceedings of Symposium on Theory of Computing (STOC)*, pages 389–398, 2001.

17 A. Gupta, V. Nagarajan, and R. Ravi. An improved approximation algorithm for requirement cut. *Operations Research Letters*, 38(4):322–325, 2010.

18 G. Guruganesh, J. Iglesias, R. Ravi, and L. Sanità. Plane gossip: Approximating rumor spread in planar graphs. *arXiv preprint arXiv:1707.01487*, 2017.

19 K. Jain. A factor 2 approximation algorithm for the generalized steiner network problem. *Combinatorica*, 21(1):39–60, 2001.

20 B. Lehmann, D. Lehmann, and N. Nisan. Combinatorial auctions with decreasing marginal utilities. In *Proc. of the 3rd ACM Conf. on Electronic Commerce*, pages 18–28. ACM, 2001.

21 G. Oriolo, L. Sanità, and R. Zenklusen. Network design with a discrete set of traffic matrices. *Operations Research Letters*, 41(4):390–396, 2013.

22 M. Yannakakis. Node-and edge-deletion np-complete problems. In *Proceedings of the tenth annual ACM symposium on Theory of computing*, pages 253–264. ACM, 1978.