Simultaneous Source Location

KONSTANTIN ANDREEV
Chatham Financial
and
CHARLES GARROD
Carnegie Mellon University
and
DANIEL GOLOVIN
Carnegie Mellon University
and
BRUCE MAGGS
Carnegie Mellon University and Akamai Technologies
and
ADAM MEYERSON
UCLA

We consider the problem of Simultaneous Source Location – selecting locations for sources in a capacitated graph such that a given set of demands can be satisfied simultaneously, with the goal of minimizing the number of locations chosen. For general directed and undirected graphs we give an $O(\log D)$ approximation algorithm, where D is the sum of demands, and prove matching $\Omega(\log D)$ hardness results assuming $\mathbf{P} \neq \mathbf{NP}$. For undirected trees, we give an exact algorithm and show how this can be combined with a result of Räcke to give a solution that exceeds edge capacities by at most $O(\log^2 n \log\log n)$, where n is the number of nodes. For undirected graphs of bounded treewidth we show that the problem is still **NP**-Hard, but we are able to give a PTAS with at most $(1+\epsilon)$ violation of the capacities for arbitrarily small ϵ, or a $(k+1)$-approximation with exact capacities, where k is the treewidth.

Categories and Subject Descriptors: F.2.2 [**Nonnumerical Algorithms and Problems**]: Routing and layout; G.2.1 [**Combinatorics**]: Combinatorial algorithms; G.2.2 [**Graph Theory**]: Network problems, Graph algorithms, Trees

General Terms: Algorithms, Theory

Authors' physical and email addresses: K. Andreev, Chatham Financial, 235 Whitehorse Lane, Kennett Square, PA 19348, kandreev@chathamfinancial.com, C. Garrod, D. Golovin, and B. Maggs, Carnegie Mellon University, Pittsburgh, PA 15213, {charlie,dgolovin,bmm}@cs.cmu.edu, A. Meyerson, UCLA, 4732 Boelter Hall, Los Angeles, CA 90095, awm@cs.ucla.edu.
A preliminary version of this paper appeared as an extended abstract in the proceedings of APPROX 2004 (LNCS 3122), Cambridge, MA, USA, August 2004.
The authors want to thank the Aladdin Center. This work was in part supported by the NSF under grants CCR-0085982, CCR-0122581, and CNF-0435382. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Additional Key Words and Phrases: Approximation Algorithms, Submodular Cover

1. INTRODUCTION

Suppose we are given a capacitated network and we have various demands for service within this network. We would like to select locations for servers in order to satisfy this demand. Such problems arise naturally in a variety of scenarios. One example would be the placement of web caches in the Internet, or file servers in an intranet. Another example would be choosing the locations of warehouses in a distribution network.

There are several possible ways to model the servicing of demands. Most previous work has assumed that each server can service each demand for some cost (typically this cost is linear in some underlying distance between server and demand) [Arya et al. 2004; Mahdian et al. 2002; Shmoys et al. 1997]. In some cases the servers have been considered to be capacitated (each one can provide for only some number of demands) [Pál et al. 2001]. Still, the primary goal can be considered as minimizing the aggregate distance of demands to servers.

In many natural applications there is no meaningful notion of distance. Consider serving a web page across the Internet. The latency (travel time of a single small packet) under low-congestion conditions tends not to be noticeable to the end users. The real difficulty here is the underlying capacity of the network. If links become congested, then latency will increase and throughput will suffer. In the case of a distribution network, there may be some relation (typically nonlinear) of costs to distance traveled. But we will definitely have to consider the available throughput. The transportation network has capacities as well as costs, and previous work (assuming costs only) tends to ignore this constraint except at the warehouses themselves.

We consider the problem of *Simultaneous Source Location* (SSL) – selecting locations for sources in a capacitated network in order to simultaneously satisfy all the given demands. More formally, each vertex v has a demand $d(v)$ for (single commodity) flow, and we must select some vertices S to act as sources. A set $S \subseteq V$ is feasible if there is a single commodity flow from sources S that simultaneously routes $d(v)$ flow to each vertex v while obeying the capacity constraints. Our goal is to minimize the number of sources used, or in the more general case in which vertices have costs associated with them, to minimize the total cost of the sources used.

Simultaneous Source Location is closely related to the *Source Location* problem (SL) introduced by Tamura et al. [Tamura et al. 1990; 1992], in which sufficiently many sources must be selected to be able to satisfy *any single* demand in an edge-capacitated undirected graph. Arata et al. [Arata et al. 2002] gave an exact algorithm for the undirected SL with uniform vertex costs. They also show that the (undirected) Source Location problem is **NP**-hard with arbitrary vertex costs. Polynomial time algorithms for several variants of SL with uniform vertex costs have been given [Bárász et al. 2005; Nagamochi et al. 2001; Tamura et al. 1996; van den Heuvel and Johnson 2008]. Since the preliminary version of this paper,

there has been much additional progress on many SL variants. Kortsarz and Nutov [Kortsarz and Nutov 2006] give a pseudo-polynomial time algorithm for SL on trees, a linear time algorithm for undirected SL when the maximum node demand is at most three (all demands are assumed to be integers), and proved that undirected SL is **APX**-hard. Note that it is easy to show that directed SL is $\Omega(\ln D)$-hard via a reduction to *Set Cover*. Sakashita et al. [Sakashita et al. 2006] improved on the hardness result of [Kortsarz and Nutov 2006] and proved that undirected SL is in fact $\Omega(\ln D)$ hard, give an $O(\ln D)$ approximation for a class of SL variants that simultaneously allow edge connectivity constraints and two types of vertex connectivity constraints, and give a pseudo-polynomial time algorithm for this SL variant on trees. In another paper, Sakashita et al. [Sakashita et al. 2006] give polynomial time algorithms for a class of problems that generalizes SL in hypergraphs with uniform vertex costs.

All of the $O(\ln D)$ approximations discussed above are achieved via greedy algorithms, and exploit submodularity in their analyses along the lines of Wolsey [Wolsey 1982]. This is no accident. Bar-Ilan *et. al.* [Bar-Ilan et al. 2001] extended the approach of Wolsey and considered so-called *Generalized Submodular Cover Problems*. One such problem Bar-Ilan *et. al.* considered is the *Fault Tolerant Center Selection Problem*, which is a facility location problem that strongly resembles directed SL. Informally, one must open a minimum cost set of facilities such that each client node v has at least k arc-disjoint (or vertex-disjoint) paths from facilities to v, and ensure that the total flow cost is below some given threshold M. Bar-Ilan *et. al.* obtained a greedy algorithm for the Generalized Submodular Cover Problem whose approximation ratio is logarithmic in the problem parameters. This yields a $O(\log(knM))$ approximation for Fault Tolerant Center Selection. Moreover, with some work one can show that *all* of the SL variants described above are Generalized Submodular Cover Problems (specifically, they can be written as programs in the class ILP^f defined in [Bar-Ilan et al. 2001]). When specialized to any of the SL variants described above, their algorithm obtains an approximation guarantee of $O(\log(nD))$.

In our problem we must satisfy all demands simultaneously (thus the name Simultaneous Source Location). We believe this is a better model of the natural problems described above. Simultaneous Source Location is actually a special case of the Source Location problem with arbitrary vertex costs explored by Arata et al. – our version of the problem can be reduced to theirs using a high-cost super-sink that is satisfied if and only if all other demands are met. However, we show that even with uniform vertex costs our version of the problem cannot be approximated to a $o(\ln D)$ factor unless $\mathbf{P} = \mathbf{NP}$, even if we restrict ourselves to the case that D is polynomial in n. Thus, whereas previous work demonstrated $\Omega(\ln D)$ hardness of undirected SL with *arbitrary* vertex costs we demonstrate $\Omega(\ln D)$ hardness of a natural special case of undirected SL, namely Simultaneous Source Location with *uniform* vertex costs. In contrast to these hardness results, we describe various approximations for both the general SSL problem and on special graphs.

1.1 Our Contributions

Our results take several forms. We show $\Omega(\log D)$ hardness for both general undirected and directed graphs even in the case that all vertices have unit cost. In the directed case, we show that the problem remains $\Omega(\log D)$ hard even if we allow

the algorithm to violate edge capacities by arbitrary factors. These results are optimal up to constant factors, because SSL can be reduced to SL in such a way that preserves the $(\ln D + 1)$-approximation factor of the greedy algorithm, even in the case with arbitrary vertex weights.

We also describe techniques for achieving better approximation guarantees for Simultaneous Source Location on a variety of simple graphs. We give an exact algorithm on trees and a $(k + 1)$-approximation for graphs of treewidth k. We observe that, in contrast to many other **NP**-Hard problems (for example vertex cover), Simultaneous Source Location is still **NP**-Hard even on graphs of treewidth two. Finally, we give bicriteria results in which we are allowed to use $\beta > 1$ times the capacity of each edge, but compare the number of sources we use to the optimal solution respecting the capacity of each edge. If we use α times the number of sources that the optimal solution does, we call this an (α, β)-approximation. For graphs of bounded treewidth, we give a $(1, 1 + \epsilon)$-approximation for arbitrarily small ϵ. Using our algorithm for trees combined with a result of Räcke et al. [Räcke 2002; Bienkowski et al. 2003] and later Harrelson et al. [Harrelson et al. 2003], we obtain an $(1, O(\log^2 n \log \log n))$-approximation for general undirected graphs. By translating this approach to the directed case and combining it with the hardness result for directed graphs, we can show that no tree decomposition similar to Räcke's decomposition can be found in the directed case in polynomial time, unless $\mathbf{P} = \mathbf{NP}$.

2. PROBLEM STATEMENT

An instance of Simultaneous Source Location (SSL) consists of a graph $G = (V, E)$ along with a capacity function $\text{cap} : E \to R^+$ on the edges and a demand function $d : V \to R^+$ on the vertices. We extend $\text{cap}(\cdot)$ and $d(\cdot)$ to sets by defining $\text{cap}(A) := \sum_{e \in A} \text{cap}(e)$ and $d(S) := \sum_{v \in S} d(v)$. We must select some subset of the vertices $S \subseteq V$ to act as sources. A source set is considered to be feasible if there exists a flow originating from the nodes S that simultaneously supplies demand $d(v)$ to each node $v \in V$ without violating the capacity constraints. This is single-commodity flow – we can imagine adding a single "super-source" s that is connected to each node of S with an infinite capacity edge, and a single "super-sink" t such that each node of V is connected to t with an edge of capacity $d(v)$, and asking whether the maximum s-t flow equals the sum of the demands. Our goal is to select such a source set S of the smallest size.

An alternate characterization of the problem is to select a minimum sized set of sources S such that $S \cap Q \neq \emptyset$ for each subset of vertices Q such that $d(Q)$ is greater than the capacity of all edges from $V \setminus Q$ to Q. This is a straightforward corollary of the Max-Flow/Min-Cut Theorem [Ford and Fulkerson 1956], and essentially says that cuts are only barrier to satisfying the demands. A similar observation for SL was made by Sakashita *et. al.* (see Lemma 2.1 of [Sakashita et al. 2006]).

From the above description, it is clear that the problem is in **NP**. A source set S can be checked for feasibility by simply solving a flow problem. Unfortunately, finding the set S of minimum size is **NP**-Hard, as we will prove.

At times we consider various generalizations of this problem. For example, we might have certain nodes in V that are not permitted to act as sources, or have costs

associated with making various nodes sources and seek to find a set S of minimum cost. We will also consider simplifications of the problem that place restrictions on the graph G (for example by bounding the treewidth).

3. SIMULTANEOUS SOURCE LOCATION ON GENERAL GRAPHS

Suppose $G = (V, E)$ is a graph on n nodes with integer edge capacities for which we would like to solve the SSL problem. It is possible to obtain an $O(\log D)$ approximation by reducing the SSL problem to an SL instance and applying previously known algorithms [Sakashita et al. 2006]. In this section we first show that a simple greedy algorithm yields an $O(\log D)$ approximation to the optimal solution, where D is sum of demands in G.

In fact, this greedy algorithm yields an $O(\log D)$ approximation for the generalization in which each vertex v has a cost $\mathrm{c}(v)$ to locate a source there and we wish to minimize the total cost. We then show that SSL is inapproximable to better than a $O(\log n)$ factor unless $\mathbf{P} = \mathbf{NP}$.

3.1 An $O(\log D)$ Approximation

We propose a simple greedy algorithm, described in Figure 1.

Let $f(S)$ equal the maximum possible demand that can be satisfied using S as sources. That is, if we add a super-source s and sink t, along with edges (v, t) of capacity $d(v)$ for each $v \in V$, and edges (s, v) of infinite capacity for all $v \in S$, then $f(S)$ is the value of the maximum $s - t$ flow. We let $\partial_{\text{in}} S := E \cap ((V \setminus S) \times S)$ denote the edges entering S.

Algorithm Greedy-SSL(G, d, c)

(1) Initialize $S = \emptyset$.
(2) While $f(S) < d(V)$:
    (a) Find $v \in V \setminus S$ minimizing $\mathrm{c}(v)/\left(f(S \cup \{v\}) - f(S)\right)$ and add it to S.
(3) Output S.

Fig. 1. Greedy Algorithm for SSL

THEOREM 3.1. *The greedy algorithm is a* $\ln(D)+1$ *approximation for undirected and directed SSL instances with vertex costs.*

PROOF. We will first prove that $f(\cdot)$ is submodular, using the submodularity of the cut capacity function $z(S) := \mathrm{cap}(\partial_{\text{out}} S)$, where $\partial_{\text{out}} S := E \cap (S \times (V \setminus S))$ denotes the edges exiting S (see e.g. [Schrijver 2003]). Let G' be the original SSL graph G to which we've added t and edges (v, t) of capacity $d(v)$ for each $v \in V$. Fix $A, B \subseteq V[G]$ and note $f(S)$ is the capacity of the minimum (S, t) cut. Let Q_A, Q_B be the source sides of minimum (A, t) and (B, t) cuts, respectively. Then $\partial(Q_A \cap Q_B)$ is a $(A \cap B, t)$ cut, and $\partial(Q_A \cup Q_B)$ is a $(A \cup B, t)$ cut. With the submodularity of $z(\cdot)$, this yields

$$\begin{aligned} f(A \cap B) + f(A \cup B) &\leq z(Q_A \cap Q_B) + z(Q_A \cup Q_B) \\ &\leq z(Q_A) + z(Q_B) \\ &= f(A) + f(B) \end{aligned}$$

We then apply the the analysis of Wolsey [Wolsey 1982] to the greedy algorithm to complete the proof. □

3.2 Inapproximability Results for General Graphs

We show that $O(\log D)$ is the best approximation factor we can expect to obtain for the directed SSL problem in polynomial time, even if we allow edge capacities to be inflated by arbitrary factors. The reduction is from Set Cover. Interestingly, SSL on undirected graphs is similarly inapproximable. In both cases, we prove hardness in the version of SSL without vertex costs.

THEOREM 3.2. *For some positive constants λ, λ' it is* **NP***-Hard to approximate SSL on directed graphs to better than $\lambda \ln D$ or to better than $\lambda' \ln n$. Furthermore, these approximation ratios hold even for algorithms that may violate the edge capacities by arbitrary factors relative to the optimal solution.*

PROOF. We will first show $\Omega(\ln D)$ hardness. Fix a Set Cover instance $(U, \mathcal{F})$, where $\mathcal{F}$ is a collection of subsets of U. Construct a directed SSL instance by adding a node v_u for each $u \in U$ and a node v_S for each set $S \in \mathcal{F}$. Additionally, add edges (v_S, v_u) for all pairs (S, u) such that $u \in S$. All edges have unit capacity, all nodes of the form v_u for some $u \in U$ have unit demand, and all nodes of the form v_S for some $S \in \mathcal{F}$ have zero demand. Note that $D = |U|$. We now claim a $\rho(D)$-approximation algorithm $\mathcal{A}$ for directed SSL yields a $\rho(|U|)$ approximation algorithm for Set Cover. First note that without loss of generality on these instances $\mathcal{A}$ outputs solutions $Q \subseteq \{v_S \mid S \in \mathcal{F}\}$. If not simply postprocess the solution by replacing v_u with some v_S such that $u \in S$. It is easy to see that this maintains feasibility and does not increase the solution cost. It is also easy to see that the correspondence $\{v_{S_1}, \ldots, v_{S_k}\} \longleftrightarrow \{S_1, \ldots, S_k\}$ maps SSL solutions $Q \subseteq \{v_S \mid S \in \mathcal{F}\}$ to set cover solutions of the same size and vice versa. Combining these facts with the known $\Theta(\log |U|)$ hardness of Set Cover under the assumption $\mathbf{P} \neq \mathbf{NP}$ as shown by Raz and Safra [Raz and Safra 1997] yields the claimed hardness result. Note there exist $\Omega(\log |U|)$-hard instances of Set Cover $(U, \mathcal{F})$ with $|\mathcal{F}| = \text{poly}(|U|)$. Thus in our constructed SSL instance, if there are $n = |U| + |\mathcal{F}|$ vertices then $D = \Omega(n^\epsilon)$ for some $\epsilon > 0$, and so we obtain $\Omega(\ln n)$ hardness as well. Finally, note that inflating edge capacities has no effect on the set of solutions, which proves that this hardness results holds even for algorithms that can relax the edge capacities. □

THEOREM 3.3. *For some positive constants λ, λ' it is* **NP***-Hard to approximate SSL on undirected graphs to better than $\lambda \ln D$ or to better than $\lambda' \ln n$, even in the case of uniform vertex costs.*

To show hardness of undirected SSL, we first consider a variant called SSL with restricted sources in which we may place sources only at certain locations specified in the input. In Lemma 3.4 we show how to reduce SSL with restricted sources to regular SSL and in Lemma 3.5 we show $\Omega(\log n)$ hardness for SSL with restricted sources. Combining the two results and noting that $D = \text{poly}(n)$ in the constructed instances then proves Theorem 3.3.

LEMMA 3.4. *For directed (resp. undirected) SSL, if there exists an $\alpha(n)$ approximation algorithm, where $\alpha(n) = O(\text{polylog}(n))$, then there exists an $O(\alpha(n))$*

approximation algorithm for directed (resp. undirected) SSL with restricted sources on instances with polynomially bounded edge capacities.

PROOF. We reduce SSL with restricted sources to general SSL as follows. We assume $\text{cap}(e) \leq n^w$ for each edge, where $n = |V[G]|$ and w is any constant. Given an SSL instance G with allowed sources $\hat{V}$, make $k := n^{w+3}$ copies of G, called $G_1, G_2, \ldots, G_k$. In each copy, the edge capacities and node demands are the same. Let node v in G correspond to node v^i in G_i. For each $v \in \hat{V}$, add infinite capacity edges (v^i, v^j) and (v^j, v^i) for each $i \neq j \in \{1, 2, \ldots, k\}$. Call the resulting graph G'. In G' any vertex may be selected as a source. For $S \subseteq V$, let $S^i := \{v^i \mid v \in S\}$. Clearly, any solution $S \subseteq \hat{V}$ in G yields a solution S^1 in G', in which we can use S^1 and the infinite capacity edges to treat S^j as sources for all j without actually including them in the solution. Now consider any solution to SSL on G', denoted by W. Without loss of generality, if $v^i \in W$ then $v^j \notin W$ for $i \neq j$. Furthermore, we can assume without loss of generality that each $v^i \in S^i \cap W$ is replaced with v^1, so that $W \cap \left(\bigcup_{i=1}^{k} \hat{V}^i\right) \subseteq \hat{V}^1$. Now consider the vertices in W not in $\hat{V}^1$, say Q. If $|Q| \leq n^2$, then as sources Q can deliver total flow at most $|Q|(n-1)n^w < n^{w+3} = k$. Since all edge capacities and demands are integers and a solution induces a maximum flow, we can assume that the flow is integral [Ford and Fulkerson 1956], and so if the flow from Q helps satisfy the demand of some v^i, it delivers at least one unit of flow to v^i. Since Q delivers less than $n^{w+3} = k$ flow, there is some subgraph G^i in G' that gets no flow from Q. Thus $W \cap \hat{V}^1$ must service all demand in G^i. However in this case, $W \cap \hat{V}^1$ can service all demand at all levels, so $W \cap \hat{V}^1$ is an SSL solution to G'. Furthermore, $W \cap \hat{V}^1$ is clearly a solution to the original restricted SSL instance on G. Now, since $\mathsf{OPT}(G') \leq n$, any $\alpha(n) \leq \text{polylog}(n)$ approximation for SSL yields a solution W with at most $n \,\text{polylog}(kn) = O(n \,\text{polylog}(n))$ sources from $\bigcup_i (V^i - \hat{V}^i)$. Thus $|Q| = O(n \,\text{polylog}(n)) < n^2$ and so $\{v \mid \exists i : v^i \in W \cap \hat{V}^i\}$ is a solution of size at most $\mathsf{OPT}(G) \cdot \alpha(kn) = \Theta(\mathsf{OPT}(G) \cdot \alpha(n))$ (since we have assumed $\alpha(n) = O(\text{polylog}(n))$). The end result is an $O(\alpha(n))$-approximation to the original restricted SSL instance. □

LEMMA 3.5. *([Swamy and Pál 2006]) Undirected SSL with restricted sources is* **NP***-hard to* $o(\log n)$ *approximate.*

PROOF. Start with Set Cover instance $(U, \mathcal{F})$, where $\mathcal{F} \subseteq 2^U$. We build a graph G as follows. Start with nodes for each $u \in U$ and $S \in \mathcal{F}$. Add a node v_{uS} for each pair $(u, S) \in U \times \mathcal{F}$ such that $u \in S$, and a node t. Add unit capacity edges $\{(u, v_{uS}) \mid u \in S\}$ and $\{(v_{uS}, S) \mid u \in S\}$. Finally, add edges (t, u) of capacity $|\{S \mid u \in S\}| - 1$ for each u. The demands are as follows: $d(v_{uS}) = 1$ for all $u \in S$, and the demand of all other nodes is zero. The allowed sources $\hat{V} = \mathcal{F} \cup \{t\}$. Now, we claim that given any set cover $\mathcal{C}$, $\mathcal{C} \cup \{t\}$ is an SSL solution, and given any SSL solution W, $W \setminus t$ is a set cover. Fix a set cover $\mathcal{C}$, and use the sources in $\mathcal{C}$ to send one unit of flow to some vertex in $\{v_{uS} \mid u \in S\}$ for each $u \in U$ using only edges of the form (S, v_{uS}). Add source t to the solution and for each u send $|\{S \mid u \in S\}| - 1$ flow from t to u and then from u to the remaining $|\{S \mid u \in S\}| - 1$ elements of $\{v_{uS} \mid u \in S\}$. Next, fix an SSL solution W and let $W' = W \setminus t$. Suppose W' is not a set cover, and $u \in U$ is not covered by it. Then consider $Q := \{v_{uS} \mid u \in S\}$.

Since $d(Q) = |Q|$ and $\text{cap}(t, u) = |Q| - 1$, at least one unit of flow entering Q comes from some $S \in \mathcal{F}$. By assumption, u is not covered so $S \notin W'$ and without loss of generality we can assume that the solution flow is integral, so S gets a unit of flow from some $v_{u'S}$. This flow must come from u' and travel over edge $(u', v_{u'S})$. However, this means we have sent one unit of flow along path $(u', v_{u'S}, S)$ and have no leftover capacity to service the demand at $v_{u'S}$, contradicting our assumption that W is a solution. It follows that an SSL solution of size k or $k+1$ exists iff a set cover of size k exists. Thus an α-approximation for SSL with restricted sources yields an $\alpha \cdot \left(1 + \frac{1}{\mathsf{OPT}}\right) \leq \alpha \cdot (1 + \epsilon)$ approximation for Set Cover on instances with $\mathsf{OPT} \geq 1/\epsilon$. We complete the proof by noting that there is some positive constant η such that Set Cover is **NP**-hard to $\eta \ln n$ approximate [Raz and Safra 1997]. □

Lemmas 3.4 and 3.5 together imply $\Omega(\ln n)$ hardness of undirected SSL with uniform vertex costs. Note that in the proof of Lemma 3.5, the SSL instances constructed satisfy $n = D + |\mathcal{F}| + |U| + 1 > D$. Since we have $\lambda' \ln n$ hardness for these instances for some $\lambda' > 0$ and $n > D$ we immediately have $\lambda' \ln D$ hardness as well.

4. SOLVING SIMULTANEOUS SOURCE LOCATION ON TREES

Suppose our graph $G = (V, E)$ is a tree. This special case allows us to solve SSL exactly by dynamic programming. For each vertex v and number of sources i, we define $f(v, i)$ to be the amount of flow that must be sent to v by its parent in order to satisfy all demands in the subtree of v, assuming this subtree contains i sources. More formally, $f(v, i) = \min_{S \subseteq T_v : |S| = i} f(v, S)$ where T_v is the subtree rooted at v and $f(v, S)$ is the flow needed by T_v with sources placed on S. If $f(v, i)$ is negative then T_v with i sources can send flow to v's parent. Our algorithm assumes that the tree is rooted and binary; in general we can create an equivalent binary tree by creating virtual nodes and connecting them by edges of infinite capacity, as shown in Figure 2. For convenience, for each node v we define $\text{cap}(v)$ to be the capacity of the edge from v to its parent.

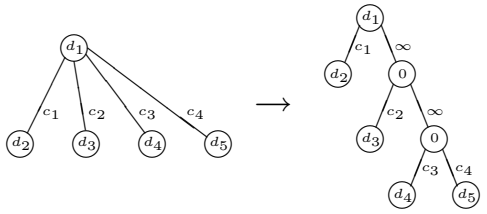

Fig. 2. By adding additional nodes linked by infinite capacity, we can convert any non-binary tree into an equivalent binary tree while at most doubling the number of nodes in the tree. Above, nodes are labeled by their demands and edges by their capacities.

Our algorithm is described in Figure 3.

Assuming that the above algorithm computes $f(v, i)$ correctly for each vertex, the correctness of the result is immediate. We need to show that $f(v, i)$ correctly

Algorithm BinaryTree(G, d)

(1) Initialize $f(v,i) = \infty$ for all $v \in V$ and $0 \le i \le |V|$
(2) For each leaf vertex $v \in V$:
  (a) Set $f(v,0) = d(v)$.
  (b) Set $f(v,i) = -\operatorname{cap}(v)$ for all $i \ge 1$.
(3) Consider any vertex v with children v_1, v_2 for whom f has been computed:
  (a) Loop over all values of i_1 and i_2 with $0 \le i_1, i_2 \le |V|$.
  (b) If $f(v_1, i_1) \le \operatorname{cap}(v_1)$ and $f(v_2, i_2) \le \operatorname{cap}(v_2)$ then:
    i. Set $f(v, i_1 + i_2) =$
      $\min\{f(v, i_1 + i_2), \max\{f(v_1, i_1) + f(v_2, i_2) + d(v), -\operatorname{cap}(v)\}\}$.
    ii. Also set $f(v, i_1 + i_2 + 1) = -\operatorname{cap}(v)$.
(4) Continue until f is defined at all vertices.
(5) Return the minimum k such that $f(r,k) \le 0$ where r is the root.

Fig. 3. Algorithm for SSL on a Binary Tree

represents the minimum amount of flow that must be sent from the parent of v provided the number of sources in the subtree is i.

PROPOSITION 4.1. *The algorithm described produces an exact solution to the source location problem on a binary tree.*

PROOF. The proof will be by induction. Our base case is at the leaves. Either a leaf is a source or it is not. If the leaf v is a source, then it requires no flow from its parent, and can send at most $\operatorname{cap}(v)$ flow upwards along its edge. This yields $f(v,1) = -\operatorname{cap}(v)$. On the other hand, if the leaf is not a source it requires flow $d(v)$ (its demand) from the parent, so $f(v,0) = d(v)$. Of course, it might not be feasible to send this amount of demand if $d(v) > \operatorname{cap}(v)$.

We now consider any node v. Suppose we have correctly computed $f(v_1, i_1)$ and $f(v_2, i_2)$ for all values i_1, i_2 for the children of node v. Suppose we would like to compute $f(v,i)$. There are i sources to be placed in this subtree. If v is not a source itself, then all the sources are in the child trees. The total demand sent into v will have to be enough to satisfy the demand $d(v)$ and additionally to satisfy any residual demand on the children. This means $f(v,i) = \min(\max(d(v) + f(v_1, i_1) + f(v_2, i - i_1), -\operatorname{cap}(v)))$ where the minimum is over choices of i_1 and the "max" term ensures that a child node cannot provide more flow to its parent than the capacity of their connecting edge. This is exactly what will be computed in the algorithm, in step 3b(i). Notice that if satisfying either child tree in this way would require overflowing a capacity $\operatorname{cap}(v_1)$ or $\operatorname{cap}(v_2)$ then this allocation of sources to subtrees is not feasible and so should not be considered; this is resolved in step 3b. However, it is also possible that v is itself a source. In this case, provided there is some choice of $i_1 \le i - 1$ such that $f(v_1, i_1) \le \operatorname{cap}(v_1)$ and $f(v_2, i - i_1 - 1) \le \operatorname{cap}(v_2)$, we will be able to produce a solution. This solution can send $\operatorname{cap}(v)$ upwards since v itself is a source. This is dealt with in step 3b(ii). It follows that the algorithm computes exactly the correct values $f(v,i)$ and the algorithm solves SSL. □

PROPOSITION 4.2. *Simultaneous Source Location can be solved in time $O(n^3)$ on a tree, even if some nodes are disallowed as sources.*

If our tree is non-binary we can replace any node with more than two children

by multiple nodes as shown in Figure 2. This increases the number of nodes by at most a constant factor.

We can also modify our algorithm to disallow certain nodes as sources. If a leaf v cannot be a source, then we ignore step 2b and instead set $f(v,i) = d(v)$ for all i. If some higher node cannot be a source, then we remove step 3b(ii) for that node (this step considers the case where v is a source). The correctness proof is the same as before.

5. USING RÄCKE'S TREE DECOMPOSITION

Harald Räcke showed that for any undirected graph, it is possible to construct a tree which approximately captures the flow properties of the original graph[Räcke 2002]. Later work [Bienkowski et al. 2003; Harrelson et al. 2003] improved the result and described a polynomial-time algorithm to construct such a tree. The most recent result is stated more precisely in the following theorem:

THEOREM 5.1. *Given any capacitated, undirected graph $G = (V, E, \text{cap} : E \to R^+)$, there exists a capacitated tree $T = (V_T, E_T, \text{cap} : E_T \to R^+)$ with the following properties:*

(*1*) *The vertices of V are the leaves of T.*

(*2*) *For any multicommodity flow F which is feasible on G, there exists a flow of equal value between the corresponding leaves on T.*

(*3*) *For any flow F_T feasible between the leaves of T, there is a feasible $\frac{1}{\rho}F_T$ flow on G for some $\rho = O(\log^2 n \log\log n)$.*

This gives us an approximation for the source location problem. We first construct a tree T as described above. We then solve SSL on the tree, permitting only the leaves to act as sources, using the algorithm of Section 4. We consider using the sources we have obtained on the original graph. We know there exists a flow F_T on the tree from our selected source nodes which satisfies all the demands. It follows that $\frac{1}{\rho}F_T$ is feasible on the graph. We conclude that if we violate the capacities by a factor of $\rho = O(\log^2 n \log\log n)$, then we have a feasible solution to source location. On the other hand, any feasible solution on the graph must also be feasible on the tree. This allows us to produce an exact solution (in terms of the number of sources) while increasing the capacities by $O(\log^2 n \log\log n)$.

THEOREM 5.2. *We can produce an optimum (in terms of number of sources) solution to SSL in polynomial time, if we permit $O(\log^2 n \log\log n)$ stretch on the capacities.*

Our results also have interesting implications for directed graphs. Consider the possibility of a Räcke-like representation of a directed graph by a bidirection of a tree, where the capacity on an edge when routing "upwards" might differ from the "downwards" capacity. Suppose such a thing existed, and could be computed in polynomial time, for some factor ρ. This would enable us to solve SSL exactly on directed graphs, while exceeding capacities by a factor of ρ. But this is **NP**-Hard, as was proved in Theorem 3.2. Thus we have the following:

THEOREM 5.3. *No Räcke-like decomposition of a directed graph into an ρ-approximately flow-conserving tree can be computed in polynomial time, for any value of ρ polynomial in n.*

Note that our hardness result is a computational hardness result and assumes a tree-like structure decomposition. Azar et al. [Azar et al. 2003] have found an existential hardness result which is true for any decomposition, but their bound is weaker: $O(\sqrt{n})$.

6. SIMULTANEOUS SOURCE LOCATION WITH BOUNDED TREEWIDTH

6.1 Defining Treewidth

The notion of treewidth was introduced by Robertson and Seymour [Robertson and Seymour 1986]. Many problems that are in general intractable become polynomial-time solvable when restricted to graphs of bounded treewidth. Furthermore, many graphs arising from natural applications have bounded treewidth. A good survey on the topic is given by Bodlaender [Bodlaender 1997]. Here is one of the many equivalent definitions of treewidth:

Definition 6.1. A graph $G = (V, E)$ has treewidth k if there exists a tree $\tau = (V_\tau, E_\tau)$ along with a mapping $f : V_\tau \to 2^V$ with the following properties:

(1) For all $\alpha \in V_\tau$, $|f(\alpha)| \leq k + 1$.
(2) For any $(u, v) \in E$ there exists some $\alpha \in V_\tau$ such that $u, v \in f(\alpha)$.
(3) For any $\alpha, \beta, \gamma \in V_\tau$ where β lies along the path from α to γ, if for some $x \in V$ we have $x \in f(\alpha)$ and $x \in f(\gamma)$, then we must also have $x \in f(\beta)$.

The above conditions essentially state that each tree vertex represents some subset of at most $k + 1$ graph vertices, each edge in the graph has its endpoints represented together in at least one of the tree vertices, and the set of tree vertices which represent sets containing any fixed vertex in V must form a contiguous subtree.

We observe that it is possible to produce such a tree decomposition for a graph of treewidth k in time linear in the number of edges and vertices (but exponential in k)[Bodlaender 1997]. Assuming k is constant, we are therefore able to produce a tree decomposition in polynomial time. In general, however, computing the treewidth of an arbitrary graph is **NP**-Hard [Robertson and Seymour 1986].

6.2 Nice Decompositions

Bodlaender [Bodlaender 1998] also introduced the notion of a nice decomposition and proved that any tree decomposition of a graph can be transformed into a nice decomposition still of polynomial size. In a nice decomposition, each node $\alpha \in V_\tau$ has one of the following types:

—A *leaf* node α has no children, and $|f(\alpha)| = 1$
—An *add* node α has one child β with $f(\alpha) = f(\beta) \bigcup \{v\}$ for some node $v \in V$
—A *subtract* node α has one child β with $f(\alpha) = f(\beta) - \{v\}$ for some node $v \in f(\beta)$
—A *merge* node α has two children β, γ with $f(\alpha) = f(\beta) = f(\gamma)$

In addition, the nice decomposition has a root node ρ (which is a *subtract* node) with $f(\rho)$ empty.

6.3 Approximation for Graphs of Bounded Treewidth

Suppose we are given a graph $G = (V, E)$ with treewidth k, for which we would like to approximate the SSL problem. Our algorithm (shown in Figure 4) takes as input a graph (V, E) along with some set of current sources S and returns a set of sources $S' \supseteq S$ which are feasible for the given graph.

Algorithm SL(S, V, E)

(1) Check whether the source set S is feasible for (V, E); if so return S
(2) If not, find sets $X \subseteq V$ and $B_X \subseteq V$ that have the following properties:
  (a) $|B_X| \leq k + 1$
  (b) For all $(x, y) \in E$ with $x \in X$ and $y \in V - X$, we have $x \in B_X$, i.e., B_X contains the "border" of X
  (c) $S \bigcup (V - X)$ is not a feasible source set
  (d) $S \bigcup (V - X) \bigcup B_X$ is a feasible source set
(3) Recursively solve SL($S \bigcup B_X, (V - X) \bigcup B_X, E$)

Fig. 4. Algorithm for SSL with treewidth k

We claim that the set S^R of returned sources has $|S^R| \leq (k + 1)|S^*|$ where $|S^*|$ is the smallest feasible set of sources for (V, E) which includes the given set S as a subset.

LEMMA 6.2. *Assuming we are always able to find sets X and B_X with the properties described and initializing $S = \emptyset$, algorithm SL is a $(k+1)$-approximation for the source location problem.*

PROOF. We will prove the lemma by induction on the number of sources required by the optimum solution. If the optimum requires no sources (in other words, S is feasible by itself) then the algorithm will not require any additional sources either (producing a trivial $k + 1$-approximation). Otherwise, we suppose the optimum requires at least t more sources (for some $t > 0$). We find a pair of sets X, B_X. The optimum must include at least one source from the set $X - S$, since $S \bigcup (V - X)$ is not feasible. Consider the flow structure of the optimum solution. Any flow that passes from a source in X to some node in $V - X$ must travel through B_X. If B_X are all sources, we can assume that no flow travels from sources in $X - B_X$ to nodes in $V - X$ (or vice versa), since such flow could equivalently originate from B_X. Consider adding all the sources of the optimum solution that are not in X to the set of sources $B_X \bigcup S$. This must be a feasible solution. It follows that the recursive SSL problem on SL($S \bigcup B_X, (V - X) \bigcup B_X, E$) requires at most $t - 1$ additional optimum sources, so by induction we will produce a solution which uses at most $(k + 1)(t - 1) + |B_X| \leq (k + 1)t$ sources in addition to S. □

We also prove that we can always find the sets X, B_X with the required properties in polynomial time. In a general graph such a pair of sets might not even exist, but we use the assumption of treewidth k to show that such sets exist and can be found in polynomial time.

LEMMA 6.3. *If the current set of sources S is not feasible for the graph $G = (V, E)$ with treewidth k, then there exists a pair of sets X, B_X with the required properties; furthermore, such a pair of sets can be found in polynomial time.*

PROOF. We produce a tree decomposition (τ, f) of G, with an arbitrary root. For each tree node α, we define τ_α to be the subtree of τ rooted at α. We define $f(\tau_\alpha) = \bigcup_{\beta \in \tau_\alpha} f(\beta)$. For each node α we will test whether $S \bigcup (V - f(\tau_\alpha))$ is a feasible set of sources. We find node α such that $S \bigcup (V - f(\tau_\alpha))$ is infeasible, but such that $S \bigcup (V - f(\tau_\beta))$ is feasible for each child β of α. Note that such an α must exist; we simply travel upwards from each leaf of the tree until we find one. We now consider returning $X = f(\tau_\alpha)$ and $B_X = f(\alpha)$. We will show that these sets satisfy the required properties.

Since the graph has treewidth k we know $|B_X| = |f(\alpha)| \leq k + 1$. Consider any $(x, y) \in E$ with $x \in X$ and $y \in V - X$. From the definition of treewidth, there must exist some node β with $x, y \in f(\beta)$. Since y is not in $f(\tau_\alpha)$ we conclude that β is not in τ_α. On the other hand, there is some node $\gamma \in \tau_\alpha$ such that $x \in f(\gamma)$ (this follows from $x \in X$). The path from γ to β must pass through α, so the treewidth definition implies $x \in f(\alpha)$ and therefore $x \in B_X$ as desired. The selection of α guarantees that $S \bigcup (V - X)$ is not a feasible source set. This leaves us to prove only the fourth condition, that $S \bigcup (V - X) \bigcup B_X$ is a feasible source set.

Consider the children of α. A pair of them γ_1, γ_2 must have $f(\gamma_1) \bigcap f(\gamma_2) \subseteq f(\alpha) = B_X$ because of the contiguity property of tree decomposition. By our selection criteria we know that for each child γ, the set of nodes $S \bigcup (V - f(\tau_\gamma))$ would be feasible. It follows that we can cover all the demand of $f(\tau_\gamma)$ using nodes of S and nodes external to $f(\tau_\gamma)$. Since the children sets are disjoint except for nodes which we have declared to be sources, the flows to satisfy each child subtree are edge-disjoint; any flow from external nodes must also pass through B_X, and we conclude that we can cover $f(\tau_\gamma)$ using $S \bigcup B_X$. It follows that we can cover all of X using the sources $S \bigcup B_X$, making $S \bigcup B_X \bigcup (V - X)$ a feasible source set for the entire graph. □

Theorem 6.4 follows from Lemma 6.2 and Lemma 6.3.

THEOREM 6.4. *Algorithm SL produces a $k+1$-approximation to the SSL problem on a graph of treewidth k.*

6.4 Bounded Treewidth with Capacity Stretch

In this section we describe an exact algorithm for the SSL problem on graphs of bounded treewidth. The running time of this algorithm is exponential in the treewidth, and also depends polynomially on the maximum edge capacity. In the general case where the capacities may be super-polynomial, we use the technique of section 6.5 to obtain a polynomial-time approximate solution with $1 + \epsilon$ stretch on the edge capacities. Our algorithm yields an exact solution without any capacity stretch for graphs of bounded treewidth if the capacities are already polynomially-bounded.

Suppose we have found a nice tree decomposition (τ, f). We will construct a set of vectors of dimension $k + 3$. Each vector has the following form:

$$(\alpha, i, f_1, f_2, ..., f_{k+1})$$

Here $\alpha \in V_\tau$, $0 \leq i \leq |V|$, and the f_i are feasible flow values (we assume these are from a polynomially-bounded range of integers). Let $S_\alpha = f(\tau_\alpha) - f(\alpha)$ represent

the nodes represented by the subtree rooted at α minus the nodes of its boundary (the nodes of $f(\alpha)$ itself). A feasible vector represents the excess flow needed to be sent directly from the nodes of $f(\alpha)$ to S_α to satisfy all the demand in S_α if S_α contains i sources.

We observe that if k is a constant and flow is polynomially-bounded, then the number of possible vectors is polynomial in size. We will determine which such vectors are feasible in polynomial time, then use this to solve SSL. Our algorithm is as follows:

(1) Start with the empty set of feasible vectors.

(2) Consider each node α from the bottom up:
   —If α is a leaf, then add vector $(\alpha, 0, 0)$.
   —If α is an add node with child β, then for each feasible vector for β, copy that vector for α, placing flow 0 in the new position corresponding to the additional node in $f(\alpha) - f(\beta)$.
   —If α is a merge node with children β, γ then for each pair of feasible vectors x_β, x_γ for the children, create a vector for α: $x_\alpha = x_\beta + x_\gamma$ (adding the number of sources and the flows while changing the choice of node from V_τ to α).
   —If α is a subtract node with child β, then consider each feasible vector x_β for the child. Let the subtracted node be $b \in f(\beta) - f(\alpha)$. This node requires some flow r_b which is the sum of the demand $d(b)$ and the flow value for b in x_β. We consider all feasible allocations of flow $F(a)$ to nodes $a \in f(\alpha)$ such that $|F(a)| \leq \text{cap}(b, a)$ and $\sum_{a \in f(\alpha)} F(a) \geq r_b$. For each such allocation we construct a vector x_α whose number of sources is equal to x_β and with flow value at a equal to the flow value in x_β plus $F(a)$. This corresponds to refusing to make b a source. We now consider making b a source. This corresponds to creating a vector x_α with one more source than the vector x_β. We set the flow value for a node $a \in f(\alpha)$ to be the flow value in x_β minus $\text{cap}(b, a)$.

(3) Now consider all vectors for the root node ρ. These are simply pairs (ρ, i) since $f(\rho)$ is empty. We return the minimum value of i such that (ρ, i) is in the feasible set.

THEOREM 6.5. *The above algorithm computes the optimal SSL solution in $O(kn^2NF^{2k+2})$ time, where the input graph G has n vertices, treewidth k, F possible flow values, and $N = |V_\tau|$ is the number of vertices in a nice tree decomposition of G.*

This running time is polynomial assuming that F is polynomial and k is a constant. If the number of possible flow values is not polynomial, we can use the result of section 6.5 to effectively reduce the number of flow values. This will cause us to exceed the graph capacities by a factor of $1 + \epsilon$.

6.5 Dealing with Super-Polynomial Capacities

Some of our analyses in section 6.4 assume that the capacities are all polynomially bounded. Here we present a method to extend those analyses to when the capacities

are super-polynomial, allowing us to satisfy demands while exceeding capacities by a $1+\epsilon$ factor in those cases.

When capacities are super-polynomial, we express the flow and capacity of each edge as αF^i for a fixed F which will be chosen later, an integer i, and an integer α between F and F^2 (i and α might vary from edge to edge). This can be done by rounding every flow up to the nearest value. Once this is done, we might no longer have a feasible flow. The new capacities (which might be larger than the old capacities by a $1+\frac{1}{F}$ factor) will not be exceeded. However, flow conservation may no longer hold. Consider any node. If the original inflow was f then the outflow was f also. But it is possible that after this rounding upwards, the inflow is still f and the outflow has increased to $f(1+\frac{1}{F})$. We consider such a flow to be feasible. This means that a node might effectively "generate" some amount of flow, but this amount is at most $\frac{1}{F}$ of the inflow.

Now consider any graph with a feasible flow under the representation described above. We would like to transform this into a real flow. If we consider splitting the "generated" flow equally among outgoing edges, we will see that as flow passes through a node the percentage of "real" flow might decrease by a factor of $\frac{F}{F+1}$. In the worst case, some edge might have a fraction of "real" flow equal to $(\frac{F}{F+1})^n \geq 1-\frac{n}{F}$.

We let $F=\frac{3n}{\epsilon}$. It follows that we can satisfy at least $1-\epsilon/3$ of each demand while exceeding capacities by a factor of $1+\frac{\epsilon}{3n}$. Since we can scale the flows and capacities, this means we can satisfy the demands while exceeding capacities by a $\frac{1+\epsilon/3n}{1-\epsilon/3} \leq 1+\epsilon$ factor.

This is useful in bounding the running time and approximation factor of various techniques for approximating SSL.

6.6 Lower Bound for Treewidth 2 Graphs

We show that the SSL problem is **NP**-Hard even on graphs with treewidth two.

THEOREM 6.6. *SSL is* **NP***-Hard even on graphs with treewidth two.*

PROOF. The proof is by reduction from subset sum. As input we are given a set of numbers $X=\{x_1,x_2,\ldots,x_n\}$ and a number A, and would like to determine whether some subset of X sums to A. Suppose the sum of X is S. We construct an SSL instance with $2n+2$ nodes as shown in Figure 5. This graph contains a vertex of demand A, a vertex of demand $S-A$, and for each x_i a gadget G_i of two vertices of demand S connected by an edge of capacity S. Each gadget G_i is connected to the vertices of demand A and $S-A$ with edges of capacity x_i as shown. We show that the instance of subset sum has a solution if and only if our reduction to SSL has a feasible solution using at most n sources.

Suppose there exists a subset $X' \subseteq X$ summing to A. For each gadget G_i in our reduction let the vertex adjacent to that of demand A be a source if $x_i \in X'$, and let the other vertex in G_i be a source otherwise. This chooses a total of n vertices as sources. It is straightforward to see that these sources are a feasible solution: each source in G_i satisfies the demand in G_i and sends flow x_i to its other adjacent vertex. Since X' sums to A these flows sum to A and $S-A$, satisfying the demand of final two vertices. On the other hand, consider a feasible SSL solution with n sources. Exactly one vertex of each G_i must be a source as otherwise there is not

enough input capacity to G_i to satisfy its total $2S$ demand, and thus neither of the vertices of demand A or $S - A$ are sources. Notice that each G_i has a net output capacity of only x_i entirely along a single exterior edge, because only one vertex in G_i is a source and the entire flow across the gadget's internal edge is needed to satisfy the demand of the internal non-source vertex. To satisfy the A and $S - A$ demand of the non-gadget vertices, the choice of source in each gadget therefore defines a partition of X with sums of A and $S - A$, respectively, corresponding to a subset sum solution.

Finally, notice that the graph given in Figure 5 has treewidth two; we can see this because if we remove the node of demand A, the remaining graph is a tree. We take the (treewidth one) tree decomposition and add the node of demand A to the subset $f(\alpha)$ for all α. This is a tree decomposition of width two. □

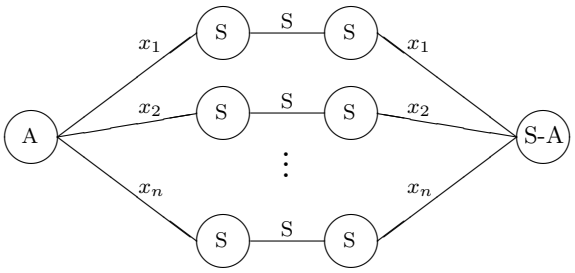

Fig. 5. Simultaneous Source Location is **NP**-Hard even on graphs of treewidth two. To satisfy each demand in this graph using only n sources we must find a partition of $\{x_1, x_2, \ldots, x_n\}$ whose sums are A and $S - A$.

7. CONCLUSIONS

We define the Simultaneous Source Location problem and solve the problem exactly on trees. For general directed and undirected graphs with node costs, we give a $O(\log D)$ factor approximation with no violation of the capacities, where D is the sum of demands. We show a matching $\Omega(\log D)$ hardness of approximation for both directed and undirected graphs, in the special case that placing a source at any location costs the same amount. We also explore bicriteria (α, β)-approximations that can route up to $\beta \cdot \text{cap}(e)$ flow along each edge e and use at most $\alpha \cdot \mathsf{OPT}$ sources, where OPT is the number of sources used by an optimal solution that can only route $\text{cap}(e)$ flow along e. Specifically, we present a $(1, O(\log^2 n \log\log n))$-approximation for general undirected graphs, and a $(1, 1+\epsilon)$-approximation PTAS for graphs of bounded treewidth. We believe that many interesting applications of this problem involve graphs of low treewidth; many of the connectivity graphs of real networks have been observed to have low tree width [Bodlaender 1998].

Improving the approximation factors of our bicriteria approximation for general undirected graphs remains an interesting open problem. Our hardness result for undirected graphs does not rule out an $(1, O(1))$-approximation for this problem. There are several interesting variants that we do not touch upon in this work, including SSL with fixed routing paths provided as input, and multicommodity

versions in which each vertex has a demand for each commodity and for each set of commodities there is some cost associated with enabling a vertex to source exactly those commodities.

Acknowledgments

We thank Chaitanya Swamy and Martin Pál for the pointing out Lemma 3.5, and the anonymous referees for their helpful comments.

REFERENCES

Arata, K., Iwata, S., Makino, K., and Fujishige, S. 2002. Locating sources to meet flow demands in undirected networks. *J. Algorithms 42,* 1, 54–68.

Arya, V., Garg, N., Khandekar, R., Meyerson, A., Munagala, K., and Pandit, V. 2004. Local search heuristics for k-median and facility location problems. *SIAM J. Comput. 33,* 3, 544–562.

Azar, Y., Cohen, E., Fiat, A., Kaplan, H., and Racke, H. 2003. Optimal oblivious routing in polynomial time. In *STOC '03: Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*. ACM, New York, NY, USA, 383–388.

Bar-Ilan, J., Kortsarz, G., and Peleg, D. 2001. Generalized submodular cover problems and applications. *Theor. Comput. Sci. 250,* 1-2, 179–200.

Bárász, M., Becker, J., and Frank, A. 2005. An algorithm for source location in directed graphs. *Oper. Res. Lett. 33,* 3, 221–230.

Bienkowski, M., Korzeniowski, M., and Räcke, H. 2003. A practical algorithm for constructing oblivious routing schemes. In *SPAA '03: Proceedings of the fifteenth annual ACM symposium on Parallel algorithms and architectures*. ACM, New York, NY, USA, 24–33.

Bodlaender, H. L. 1997. Treewidth: Algorithmic techniques and results. In *MFCS '97: Proceedings of the 22nd International Symposium on Mathematical Foundations of Computer Science*. Springer-Verlag, London, UK, 19–36.

Bodlaender, H. L. 1998. A partial k-arboretum of graphs with bounded treewidth. *Theor. Comput. Sci. 209,* 1-2, 1–45.

Ford, L. R. and Fulkerson, D. R. 1956. Maximal flow through a network. *Canadian Journal of Mathematics 8*, 399–404.

Harrelson, C., Hildrum, K., and Rao, S. 2003. A polynomial-time tree decomposition to minimize congestion. In *SPAA '03: Proceedings of the fifteenth annual ACM symposium on Parallel algorithms and architectures*. ACM, New York, NY, USA, 34–43.

Kortsarz, G. and Nutov, Z. 2006. A note on two source location problems. Manuscript.

Mahdian, M., Ye, Y., and Zhang, J. 2002. Improved approximation algorithms for metric facility location problems. In *APPROX '02: Proceedings of the 5th International Workshop on Approximation Algorithms for Combinatorial Optimization*. Springer-Verlag, London, UK, 229–242.

Nagamochi, H., Ishii, T., and Ito, H. 2001. Minimum cost source location problem with vertex-connectivity requirements in digraphs. *Information Processing Letters 80,* 6, 287–293.

Pál, M., Éva Tardos, and Wexler, T. 2001. Facility location with nonuniform hard capacities. In *FOCS '01: Proceedings of the 42nd IEEE symposium on Foundations of Computer Science*. IEEE Computer Society, Washington, DC, USA, 329.

Räcke, H. 2002. Minimizing congestion in general networks. In *FOCS '02: Proceedings of the 43rd Symposium on Foundations of Computer Science*. IEEE Computer Society, Washington, DC, USA, 43–52.

Raz, R. and Safra, S. 1997. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *STOC '97: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*. ACM Press, New York, NY, USA, 475–484.

Robertson, N. and Seymour, P. D. 1986. Graph minors. II. algorithmic aspects of tree-width. *J. Algorithms 7,* 3, 309–322.

Sakashita, M., Makino, K., and Fujishige, S. 2006. Minimum cost source location problems with flow requirements. In *LATIN*. 769–780.

Sakashita, M., Makino, K., Nagamochi, H., and Fujishige, S. 2006. Minimum transversals in posi-modular systems. In *ESA'06: Proceedings of the 14th conference on Annual European Symposium*. Springer-Verlag, London, UK, 576–587.

Schrijver, A. 2003. *Combinatorial optimization : polyhedra and efficiency.* volume B, matroids, trees, stable sets, chapters 39-69. Springer. Schrijver.

Shmoys, D. B., Éva Tardos, and Aardal, K. 1997. Approximation algorithms for facility location problems. In *Proceedings of the 29th annual ACM Symposium on Theory of Computing*. 265–274.

Swamy, C. and Pál, M. 2006. Personal communication.

Tamura, H., Sengoku, M., Shinoda, S., and Abe, T. 1990. Location problems on undirected flow networks. *IEICE Transactions on Communication, Electronics, Information and Systems E73,* 12, 1989–1993.

Tamura, H., Sengoku, M., Shinoda, S., and Abe, T. 1992. Some covering problems in location theory on flow networks. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences E75-A,* 6, 678–684.

Tamura, H., Sugawara, H., Sengoku, M., and Shinoda, S. 1996. On a generalization of a covering problem on undirected flow networks. In *IEEE Asia Pacific Conference on Circuits and Systems*. 460–463.

van den Heuvel, J. and Johnson, M. 2008. Transversals of subtree hypergraphs and the source location problem in digraphs. *Networks 51,* 2, 113–119.

Wolsey, L. A. 1982. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica 2,* 4, 385–393.