A note on two source location problems

Guy Kortsarz

Department of Computer Science, Rutgers, Camden, USA

guyk@crab.rutgers.edu.

Zeev Nutov

Computer Science Division, The Open University of Israel

nutov@openu.ac.il

Abstract

We consider *Source Location* (SL) problems: given a capacitated network $G = (V, E)$, cost $c(v)$ and a demand $d(v)$ for every $v \in V$, choose a min-cost $S \subseteq V$ so that $\lambda(v, S) \geq d(v)$ holds for every $v \in V$, where $\lambda(v, S)$ is the maximum flow value from v to S. In the directed variant, we have demands $d^{in}(v)$ and $d^{out}(v)$ and we require $\lambda(S, v) \geq d^{in}(v)$ and $\lambda(v, S) \geq d^{out}(v)$. Undirected SL is (weakly) NP-hard on stars with $r(v) = 0$ for all v except the center. But, it is known to be polynomially solvable for uniform costs and uniform demands. For general instances, both directed an undirected SL admit a $(\ln D+1)$-approximation algorithms, where D is the sum of the demands; up to constant this is tight, unless P=NP. We give a pseudopolynomial algorithm for undirected SL on trees with running time $O(|V|\Delta^3)$, where $\Delta = \max_{v \in V} d(v)$. This algorithm is used to derive a linear time algorithm for undirected SL with $\Delta \leq 3$. We also consider the *Single Assignment Source Location* (SASL) where every $v \in V$ should be assigned to a single node $s(v) \in S$. While the undirected SASL is in P, we give a $(\ln |V| + 1)$-approximation algorithm for the directed case, and show that this is tight, unless P=NP.

1 Introduction

Let $G = (V, E)$ be a simple (possibly directed) graph with integral capacities $\{u(e) : e \in E\}$; we refer to the pair (G, u) as a *network*. Let $n = |V|$ and $m = |E|$. Given a network, let $\lambda(v, S)$ denote the maximum flow value in the network from v to S, where $\lambda(v, S) = \infty$ for $v \in S$. We consider the following *Source Location* (SL) problem: given a network (G,u), integral node demands $\{d(v) : v \in V\}$ and costs $\{c(v) : v \in V\}$, choose a minimum-cost subset of sources $S \subseteq V$ so that $\lambda(v,S) \geq d(v)$ for all $v \in V$. In the directed variant, we have demands $d^+(v)$ and $d^-(v)$ and we require $\lambda(S,v) \geq d^+(v)$ and $\lambda(v,S) \geq d^-(v)$ for all $v \in V$. In the *Single Assignment Source Location* (SASL) every $v \in V$ should be assigned to a single node $s(v) \in S$ so that $\lambda(v,s(v)) \geq d(v)$ ($\lambda(s(v),v)) \geq d^+(v)$ and $\lambda(v,s(v)) \geq d^-(v)$ in the directed case) for all $v \in V$.

SL problems naturally arise in various applications. For example, given a network in which nodes represent users, determine a location of servers so that each user v can communicate with at least one server even if $d(v)-1$ link failures occur. If the cost of locating a server at v is $c(v)$, our goal is to find the cheapest location to ensure the required reliability of communication. This is a special case of SL where all edges have capacity 1.

A *ρ-approximation algorithm* for a minimization problem is a polynomial time algorithm that produces a solution of value no more than ρ times the value of an optimal solution. We say that an optimization problem is *ρ-hard* if, up to constants, an approximation ratio better than ρ for it is not possible, unless P=NP. For example, a problem is $\Omega(\ln n)$-hard if there exists a constant $B > 0$ such that the problem cannot have a $B \ln n$-approximation algorithm, unless P=NP. It is well known that the Set-Cover (SC) problem on a groundset of size n is $\Omega(\ln n)$-hard [10].

For SL problems the following results were known. Undirected SL is NP-hard even on stars [2], but is polynomially solvable for uniform requirements or for uniform costs [13, 2]. Both directed and undirected SL admit a $(1+\ln D)$-approximation algorithm [3] (see also [11]), where D is the sum of the demands. It is easy to show that the directed case is at least as hard as the Set-Cover problem (even for $0,1$ demands), and thus is $\Omega(\ln D)$-hard. In [11] it is shown in that the undirected SL is also $\Omega(\ln n)$-hard, and that similar approximation ratios and hardness results hold for the node-connectivity variant of the problem. A related problem on digraphs with both uniform requirements and uniform costs is considered in [6, 4]. A variant when the flow demands should be satisfied simultaneously is studied in [1]. For the case of node-connectivity demands see, c.f., [9, 11].

An edge from x to y is denoted by xy. For $X \subseteq V$ let $\delta(X) = \{xy \in E : x \in X, y \in V - X\}$ be the *cut induced by* X in G, and let $u(\delta(X)) = \sum_{e \in \delta(X)} u(e)$ denote its capacity. SL problems can be formulated as a covering problem. For $X \subseteq V$ let $d(X) = \max_{v \in X} d(v)$ be the *demand* of X (where $d(\emptyset) = 0$). For undirected SL, we say that $X \subseteq V$ is *deficient* if $d(X) > u(\delta(X))$. By the Max-Flow-Min-Cut Theorem, S is a feasible solution to SL if, and only if, S covers the family $\mathcal{F}$ of minimal deficient sets; $|\mathcal{F}|$ might be exponential in n even if G is a star (see

[2]). We prove:

Theorem 1.1 *There is an* $O(n\Delta^3)$ *time algorithm for undirected* SL *on trees, where* $\Delta = \max_{v \in V} d(v)$.

A similar result was independently obtained in [11].

In practical applications the connectivity demands are usually rather small. While the directed SL is $\Omega(\ln n)$-hard even for $\Delta = 1$, we use Theorem 1.1 to prove:

Theorem 1.2 *Undirected* SL *with* $\Delta \leq 3$ *can be solved in linear time.*

Undirected SASL is polynomially solvable [12]. We consider the directed case and prove:

Theorem 1.3 *Directed* SASL *admits a* $(\ln n + 1)$*-approximation algorithm, and it is* $\Omega(\ln n)$*-hard even if* $\Delta = 1$.

Theorems 1.1, 1.2, and 1.3 are proved in Sections 2, 3, and 4, respectively.

2 Proof of Theorem 1.1

To prove Theorem 1.1 we use dynamic programming. Throughout this section, assume that $G = T$ is a tree. Let $s \in V$ be an arbitrary node of T designated as a root. The choice of s induces a parent-child relation on V. Let T_v denote the subtree of T induced by the descendants of v. Let $ch(v)$ denote the number of children of v. A node v is a *leaf* if $ch(v) = 0$. The height $h(v)$ of v is the number of edges in the longest path from v to a leaf in T_v. The leaves have height 0. We will assume some fixed order $a_1, \ldots, a_{ch(v)}$ of the children of every node v in the tree. For a node v of T with children $a_1, \ldots, a_{ch(v)}$ and $0 \leq i \leq ch(v)$ let $T_v^i = T_v - \cup_{j>i} T_{a_j}$ denote the subtree of T_v induced by v and the subtrees of its first i children $a_1, \ldots, a_i$ (where T_v^0 is the trivial tree containing only v).

The algorithm fills a 5-dimensional array $C[v, i, q, f, b]$ where $v \in V$, $0 \leq i \leq ch(v)$, $0 \leq q, f \leq R$ integers, and $b \in \{0, 1\}$. The interpretation is as follows. Let $S' = S \cap V(T_v^i)$ be the sources in T_v^i. If $T_v^i \neq T$, then flow can reach T_v^i for "free" from $T \setminus T_v^i$. Given q, f and b, we look for the best feasible set S of sources under the following additional restrictions:
(i) $\lambda(v, S') = q$ and $\lambda(v, S - S') = f$; namely, at least f flow units should arrive to v from $T \setminus T_v^i$ and S' should be able to provide q flow units to v.
(ii) If $b = 1$ then $v \in S$, and if $b = 0$ then $v \notin S$.

Formally, to model f flow units arriving from "outside" of T_v^i into v let $T_v^i(f)$ be obtained by adding to T_v^i a new node a and edge av with $r(a) = c(a) = 0$ and $u(av) = f$.

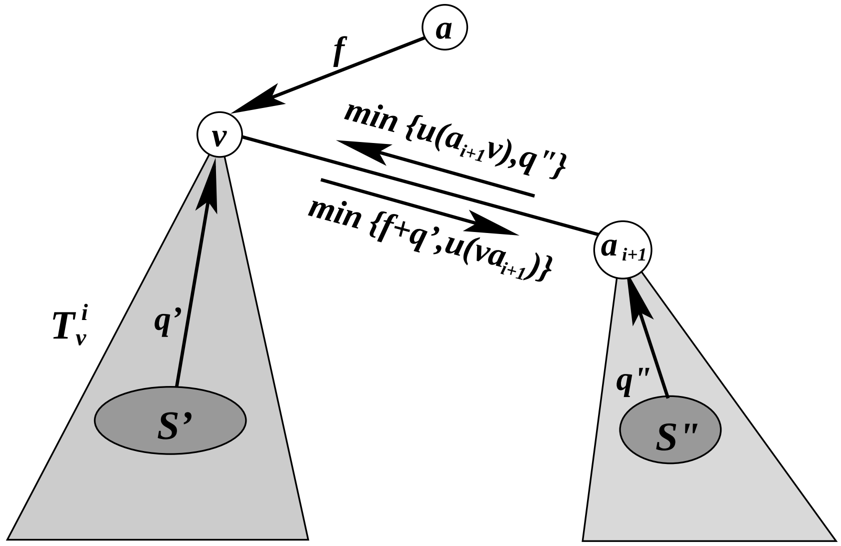

Figure 1: Decomposition of flow contributions.

The entry $C[v,i,q,f,b]$ should store the optimum cost of a solution S to the problem on $T_v^i(f)$, so that:
(i) $\lambda(S-a,v)=q$, and (ii) $b=1$ if $v \in S$ and $b=0$ otherwise.
If such S does not exist, then $C[v,i,q,f,b]=\infty$. Clearly, the optimal solution value on T is:

$$\min_{q,b}\{\ C[s,ch(s),q,0,b]\ \}.$$

The f entry is 0 since when $i=ch[s]$ then $T_s^i=T$, and so the root can not get "outside flow".

The array C is filled by increasing height of nodes, starting from leaves. We have:
$C[v,0,0,f,1]=c(v)$ if $f<d(v)$ (v becomes a source);
$C[v,0,0,f,0]=0$ if $f \geq d(v)$ (a is always a source);
$C[v,0,0,f,0]=\infty$ otherwise ($v \notin S$, a cannot satisfy the demand of v).

In particular, the rule above applies for leaves, since they have no children.

Assume now that the entries $C[v,j,q,f,b]$ have been computed for all $0 \leq j \leq i \leq ch(v)-1$. We show how to fill the $C[v,i+1,q,f,b]$ entries. We have (see Fig. 1):

$$C[v,i+1,q,f,b]=\min\ \{C[v,i,q',f',b]+C[a_{i+1},ch(a_{i+1}),q'',f'',b'']\ \}, \tag{1}$$

where the minimum is taken over $b'' \in \{0,1\}$ and all $0 \leq q',q'' \leq R$ such that:

$$q=q'+\min\{q'',u(a_{i+1}v)\}; \tag{2}$$

$$f'=f+\min\{q'',u(a_{i+1}v)\}; \tag{3}$$

$$f'' = \min\{f + q', u(va_{i+1})\}. \tag{4}$$

The total flow reaching from outside T_v^{i+1} into the root v is f. Let $S' = S \cap T_v^i$ and $S'' = S \cap T_{a_{i+1}}$. We enumerate over all possible q': the flow from S' to v, and all the possible flow q'' from S'' to a_{i+1}. Given q', q'', then the cost over T_v^i is $C[v, i, q', f', b]$ with $f' = f + \min\{q'', u(a_{i+1}v)\}$. This is because the tree rooted at a_{i+1} is "external" to T_v^i. The cost over $T_{a_{i+1}}$ is $C[a_{i+1}, ch(a_{i+1}), q'', f'', b'']$ with $f'' = \min\{f + q', u(va_{i+1})\}$. Indeed, the number of external flow units that can reach a_{i+1} is the f external flow units plus q'' from S', unless it exceeds the $u(a_{i+1}v)$ capacity.

Hence, every entry is computable using previously computed entries. Once all the C entries are computed, it is easy to recover S. Given C, we use the following recursive algorithm. We pick the smallest cost $C[s, ch(s), q, f, b]$ over all q, f, b. Let q, f, b be the optimum triplet. If $b = 1$ then $s \in S$, and $s \notin S$ otherwise.

We then use Equalities (2), (3) and (4) to define q', f', f'', b''. Then, recursively extend S by running the algorithm on $C[v, i, q', f', b]$ and $C[a_{i+1}, ch(a_{i+1}), q'', f'', b'']$. This ends the description of the algorithm.

Let us now discuss the running time of the algorithm. At every iteration we have six parameters $0 \le q, q', q'', f, f', f'' \le \Delta$ to determine for computing the minimum. However, three parameters e.g., q, f, q'' determine the others via equations (2), (3), and (4). We have one iteration per edge of T, thus $n - 1$ iterations. Thus the total time complexity is $O(n\Delta^3)$ as claimed.

3 Proof of Theorem 1.2

We can assume that G is connected, and focus on the more complicated case $\Delta = 3$. We will show a 2-stage reduction from the case $\Delta = 3$ to an equivalent problem on a tree with capacities in $\{1, 2\}$. It is known that for any integer k the relation $\mathcal{R}_k$ on nodes of a graph "$(x, y) \in \mathcal{R}_k$ if $\lambda(x, y) \ge k$" is an equivalence. Its equivalence classes are called *classes of k-(edge)-connectivity*, or *k-classes* for short. Recall that for SL a set $X \subseteq V$ is deficient if $d(X) > u(\delta(X))$.

Lemma 3.1 *For any $k \ge \Delta$, if a deficient set X intersects a k-class Y, then $Y \subseteq X$.*

Proof: Suppose to the contrary that there is $y \in Y - X$. Let $x \in Y \cap X$. Then

$$u(\delta(X)) \ge \lambda(x, y) \ge k \ge \Delta \ge d(X),$$

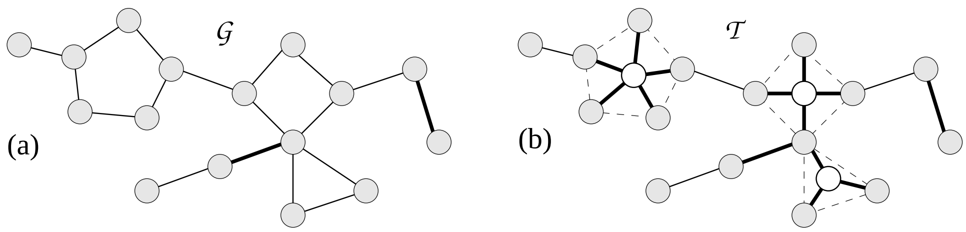

Figure 2: (a) $\mathcal{G}$ for $k = 3$ (bold edges have capacity 2); (b) $\mathcal{T}$ (dashed edges show removed cycles).

contradicting that X is deficient. □

Lemma 3.1 implies that for any $k \geq \Delta$, instead of considering the original network G, we can consider the network $\mathcal{G}$ obtained from G by shrinking every k-class X of G into a single node v_X and setting $d(v_X) = d(X)$ and $c(v_X) = \min_{v \in X} c(v)$. The corresponding quotient mapping $\psi(v) = v_X$ takes the nodes of a k-class X to the node v_X. For a set $\mathcal{S}$ of sources of $\mathcal{G}$, the corresponding set S of sources of G is defined by choosing for every $v_X \in \mathcal{S}$ a node $u \in X$ such that $c(u) = c(v_X)$. We summarize the first stage of our reduction as follows:

Corollary 3.2 *S is a feasible solution for G if, and only if, $\psi(S)$ is a feasible solution for $\mathcal{G}$. In particular, if $\mathcal{S}$ is an optimal solution for $\mathcal{G}$, then choosing the cheapest node from every k-class X with $v_X \in \mathcal{S}$ gives an optimal solution for G.*

A connected graph is a *cactus-tree* if any two cycles in it have at most one node in common (that is, every its block is an edge or a cycle). It is well known that for $k = 3$ $\mathcal{G}$ is a cactus tree, such that each its bridge has capacity in $\{1, 2\}$, and any its edge belonging to a cycle has capacity 1 (see Fig. 2a). We note that the k-classes (and thus the corresponding graph $\mathcal{G}$) can be computed in $n - 1$ k-flow computations (thus in $O(knm)$ time) using the Gomory-Hu cut tree [5]; the complexity can be further reduced to $O(k^2 n^2)$ using sparse certificates. But for $k = 3$, $\mathcal{G}$ can be computed in linear time [7, Theorem 7.3.3]. The other parts of our reduction can be also implemented in linear time.

We now describe how to solve the problem for the particular case when the input graph is a cactus-tree as above and $k = 3$, by establishing a reduction to the tree case considered in Section 2.

The second stage of our reduction is: construct from $\mathcal{G}$ a tree $\mathcal{T}$ by "implanting" instead every cycle a star with edges having capacity 2 (see Fig. 2b); the center of each star is "empty", and has cost infinity and requirement 0. Let $\mathcal{O}$ denote the centers of the stars implanted. Note that the nodes that are not in $\mathcal{O}$ and edges that are not incident to nodes in $\mathcal{O}$ are common to $\mathcal{G}$ and to $\mathcal{T}$.

Lemma 3.3 *Let S be a set of nodes of $\mathcal{G}$ and let v be a node of $\mathcal{G}$ that is not in S. Then*

$$\lambda_{\mathcal{G}}(v,S) = \lambda_{\mathcal{T}}(v,S).$$

Proof: Consider the connected components $\mathcal{G}_1,\ldots,\mathcal{G}_q$ of $\mathcal{G}-v$ that intersect S and the corresponding connected components $\mathcal{T}_1,\ldots,\mathcal{T}_q$ of $\mathcal{T}-v$. Let $S_i = \mathcal{G}_i \cap S = \mathcal{T}_i \cap S$, $i=1,\ldots,q$, (the second inequality follows from the fact that $S \cap \mathcal{O} = \emptyset$). It is not hard to see that there is a bridge (with capacity 1) that separates S_i from v in $\mathcal{G}$ if and only if there is such a bridge in $\mathcal{T}$; thus in this case we must have $\lambda_{\mathcal{G}}(v,S_i) = \lambda_{\mathcal{T}}(v,S_i) = 1$. Otherwise, $\lambda_{\mathcal{G}}(v,S_i) = \lambda_{\mathcal{T}}(v,S_i) = 2$. Hence

$$\lambda_{\mathcal{G}}(v,S_i) = \lambda_{\mathcal{T}}(v,S_i), \qquad i=1,\ldots,q.$$

The claim follows, since clearly

$$\lambda_{\mathcal{G}}(v,S) = \sum_{i=1}^{q} \lambda_{\mathcal{G}}(v,S_i), \qquad \lambda_{\mathcal{T}}(v,S) = \sum_{i=1}^{q} \lambda_{\mathcal{T}}(v,S_i).$$

□

Corollary 3.4 *$\mathcal{S}$ is a feasible solution for $\mathcal{G}$ if, and only if, $\mathcal{S}$ is a feasible solution for $\mathcal{T}$ not containing any center of a star implanted. Thus $\mathcal{S}$ is an optimal solution for $\mathcal{G}$ if, and only if, $\mathcal{S}$ is an optimal solution for $\mathcal{T}$.*

Corollary 3.4 implies that instead of solving the problem on G we can solve the problem on $\mathcal{T}$. By Theorem 1.1, this can be done in $O(n)$ time. Since the 3-classes can be found in linear time, $\mathcal{T}$ can be constructed in linear time. Thus the overall time complexity is linear. This finishes the proof of Theorem 1.2.

4 Proof of Theorem 1.3

Note that $S \subseteq V$ is a feasible solution for directed SASL if, and only if, for every $w \in V$ there is $s \in S$ so that: if $d^{in}(w) > 0$ then $\lambda(s,w) \geq d^{in}(w)$ and if $d^{out}(w) > 0$ then $\lambda(w,s) \geq d^{out}(w)$. That is, for every $w \in V$ with $\max\{d^{in}(w), d^{out}(w)\} > 0$, S intersects the set D_w defined as follows. Let $D_w^{in} = \{v \in V : \lambda(v,w) \geq d^{in}(w)\}$, $D_w^{out} = \{v \in V : \lambda(w,v) \geq d^{out}(w)\}$. Then

$$D_w = \begin{cases} D_w^{in} & d^{in}(w) > 0, d^{out}(w) = 0 \\ D_w^{out} & d^{in}(w) = 0, d^{out}(w) > 0 \\ D_w^{in} \cap D_w^{out} & d^{in}(w) > 0, d^{out}(w) > 0 \end{cases}$$

Thus for directed SASL the deficient sets are $\{D_w : w \in V, \max\{d^{in}(w), d^{out}(w)\} > 0\}\}$. Clearly, the number of deficient sets is at most n, and they all can be computed using $O(n^2)$ max-flow computations, hence in polynomial time.

Remark In the undirected case, the deficient sets are $\{D_w : w \in V, d(w) > 0\}$, where $D_w = \{v \in V : \lambda(w,v) \geq d(w)\}$, and they can be computed using $n-1$ max-flow computations via the Gomory-Hu cut-tree [5]. Moreover, for undirected SASL the deficient sets are disjoint [12]. This immediately implies a polynomial time algorithm: choose the cheapest source from every deficient set.

For directed SASL the algorithm is as follows. We compute the the family $\mathcal{F}$ of the deficient sets. Let τ^* denote the optimal value of the LP-relaxation $\min\{\sum_{v \in V} c(v)x_v : \sum_{v \in X} x_v \geq 1 \ \forall X \in \mathcal{F}\}$. By a well known result of Lovász [8], the greedy algorithm (which repeatedly removes the node that covers the maximum number of sets, together with these sets, until no sets remain) computes a feasible solution S of size at most $H(|\mathcal{F}|)\tau^* \leq (\ln|\mathcal{F}|+1)\tau^*$, where $H(k)$ denotes the kth Harmonic number. Since $|\mathcal{F}| \leq n$, this gives an $H(n)$-approximation algorithm for directed SASL.

Let $\Gamma_J(X)$ denote the set of *neighbors* of a node subset X in a graph J. To show that directed SASL is $O(\ln n)$-hard, we use the following well known formulation of the Set-Cover problem:

Set-Cover (SC):
Input: A bipartite graph $J = (A+B, I)$ without isolated nodes.
Output: A minimum size subset $S \subseteq A$ such that $\Gamma_J(S) = B$.

In this formulation, J is the incidence graph of sets and elements, where A is the family of sets and B is the universe. Given an instance $J = (A+B, I)$ for the SC, construct an instance for directed SASL by directing the edges in J from B to A, and setting $d^{out}(b) = 1$ and $d^{in}(b) = 0$ for every $b \in B$, and $d^{in}(a), d^{out}(a) = 0$ for every $a \in A$. The cost of every node is 1. It is straightforward to see that:
(i) for any feasible solution S' , there exists a feasible solution $S \subseteq A$ with $|S| \leq |S'|$, and
(ii) $S \subseteq A$ is a feasible solution for G if, and only if, S is a feasible solution for SC on J.
Since SC is $\Omega(\ln n)$-hard [10], the result follows.

Acknowledgments. We thank two anonymous referees for useful comments on a preliminary version of this paper. The first author thank Joseph Cheriyan for helpful discussions.

References

[1] K. Andreev, C. Garrod, B. Maggs, and A. Meyerson. Simultaneous source location. In *APPROX 2004*, pages 13–26, 2004.

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

[3] J. Bar-Ilan, G. Kortsarz, and D. Peleg. Generalized submodular cover problems and applications. *Theoretical Computer Science*, 250:179–200, 2001.

[4] M. Bárász, J. Becker, and A. Frank. An algorithm for source location in directed graphs. *EGRES TR-2004-06*, 2004.

[5] R. Gomory and T. Hu. Multi-terminal network flows. *SIAM J. Appl. Math.*, 9:551–570, 1961.

[6] J. Heuvel and M. Johnson. Transversals of subtree hypergraphs and the source location problem in digraph. *CDAM Research Report LSE-CDAM-2004-10*, 2004.

[7] T.-H. Hsu. Graph augmentation and related problems: theory and practice. *Ph. D. Thesis, The Univ. of Texas at Austin*, 1993.

[8] L. Lovász. On the ratio of optimal integral and fractional covers. *Discrete Math.*, 13:383–390, 1975.

[9] H. Nagamochi, T. Ishii, and H. Ito. Minimum cost source location problem with vertex-connectivity requirements in digraphs. *Information Processing Letters*, 80(6):287–294, 2001.

[10] R. Raz and S. Safra. A sub constant error probability low degree test, and a sub constant error probability PCP characterization of NP. In *Proc. 29th ACM Symp. on Theory of Computing*, pages 475–484, 1997.

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

[12] H. Tamura, M. Sengoku, S. Shinoda, and T. Abe. Some covering problems in location theory on flow networks. *IEICE Trans. Fund.*, E75-A:678–683, 1992.

[13] H. Tamura, H. Sugawara, M. Sengoku, and S. Shinoda. Plural cover problem on undirected flow networks. *IEICE Trans. Fund.*, (J81-A):863–869, 1998. (In Japanese).