On the Fixed Cost k-Flow Problem and related problems*

MohammadTaghi Hajiaghayi[1]**, Rohit Khandekar[2], Guy Kortsarz[3]***, and Zeev Nutov[4]

[1] University of Maryland, College Park, MD. hajiagha@cs.umd.edu.
[2] Knight Capital Group, Jersey city, NJ. rkhandekar@gmail.com.
[3] Rutgers University–Camden, NJ. guyk@camden.rutgers.edu
[4] The Open University of Israel. nutov@openu.ac.il.

Abstract. In the FIXED COST k-FLOW problem, we are given a graph $G = (V, E)$ with edge-capacities $\{u_e \mid e \in E\}$ and edge-costs $\{c_e \mid e \in E\}$, source-sink pair $s, t \in V$, and an integer k. The goal is to find a minimum cost subgraph H of G such that the minimum capacity of an st-cut in H is at least k. We show that the GROUP STEINER ON TREES problem is a special case of FIXED COST k-FLOW. This implies the first non constant lower bound for FIXED COST k-FLOW and the first non constant lower bounds for problems that are more general than FIXED COST k-FLOW. In particular, the CAPACITATED MULTICOMMODITY FLOW and the CAPACITATED STEINER NETWORK and the Capacitated Buy at Bulk problem. A special case of both FIXED COST k-FLOW and the related NODE-WEIGHTED k-FLOW problem is the NODE-MINIMUM BIBARTITE k-FLOW problem: given a bipartite graph $G = (A \cup B, E)$ with edge capacities and an integer $k > 0$, find a node subset $S \subseteq A \cup B$ of minimum size $|S|$ such that the minimum capacity of an $(S \cap A, S \cap B)$-cut is at least k. The NODE-WEIGHTED k-FLOW problem admits an easy $O(k)$-approximation algorithm, and in [18] is posed an open question whether it admits ratio $o(k)$. We give an $O(\sqrt{k \log k})$ approximation for NODE-MINIMUM BIBARTITE k-FLOW, which could be a step toward resolving this open question. We also show the following bicriteria result: we can compute a solution of *optimum value* and deliver $\Omega(k/\text{polylog}|V|)$ flow. We also give an $O(n^{0.172})$ ratio for the case of capacities 1 by showing that it is *equivalent* to the minimization version of the Dense k-subgraph problem. It is widely believed that the minimization version of the Dense k-subgraph problem admits only polynomial ratio. If this is true, then FIXED COST k-FLOW also admits only a polynomial ratio. The final special case of FIXED COST k-FLOW that we study is called the GENERALIZED-P2P

* Part of this work was done at DIMACS. We thank DIMACS for their hospitality. See also an archive version of the paper: MohammadTaghi Hajiaghayi, Rohit Khandekar, Guy Kortsarz, Zeev Nutov: Combinatorial Algorithms for Capacitated Network Design. CoRR abs/1108.1176 (2011)

** Supported in part by NSF CAREER award 1053605, ONR YIP award N000141110662, DARPA/AFRL award FA8650-11-1-7162, and University of Maryland Research and Scholarship Award (RASA). The author is also with AT&T Labs–Research, Florham Park, NJ.

*** Supported in part by NSF grant number 434923.

problem. Besides its practical applications to shift design problems [10], it generalizes many problems such as k-Steiner Tree, Steiner Forest, and Point to Point Connection. We give a logarithmic approximation algorithm for this problem. Finally, we consider a problem related to Buy at Bulk with capacities and with rooted requirements called Connected Rent or Buy Multicommodity Flow. We give a $\log^{3+\epsilon} n$ approximation scheme for it, using Group Steiner on trees techniques.

1 Introduction

1.1 Problems considered

Graphs in this paper are undirected and simple, unless stated otherwise. For a graph H with edge capacities u_e (a default capacity of an edge is 1) let $\lambda_H(A, B)$ denote the max-flow/min-cut value between A and B in H. We study variants of the following network design problems from [10].

<u>Fixed Cost k-Flow</u>
Instance: A graph $G = (V, E)$ with edge-capacities $\{u_e \mid e \in E\}$ and edge-costs $\{c_e \mid e \in E\}$, source-sink pair $s, t \in V$, and an integer k.
Objective: Find a minimum cost subgraph H of G such that $\lambda_H(s, t) \geq k$.

The Fixed Cost k-Flow problem is a special case of Capacitated Steiner Network In this problem we are given an edge weighted graph $G(V, E)$ with capacities over the edges and demands d_{uv} for every pairs in $V \times V$. The goal is to choose the minimum cost set of edges so that for a pair uv we can send (separately) d_{uv} flow units from u to v. The Fixed Cost k-Flow problem is also a special case of Capacitated Multicommodity Flow which is like Capacitated Steiner Network except that that the pairs need to be able to send flow simultaneously. As we give a lower bound of $\Omega(\log^2 n)$ Fixed Cost k-Flow, it implies the first non constant lower bound for Capacitated Steiner Network,Capacitated Multicommodity Flow. Indeed, because in Fixed Cost k-Flow there is a single source and sink, there is no difference between Capacitated Steiner Network and Capacitated Multicommodity Flow in this problem.

Fixed Cost k-Flow is closely related to the following problem defined in [18].

<u>Node-Weighted k-Flow</u>
Instance: A graph $G = (V, E)$ with node-costs $\{c_v \mid v \in V\}$ and integral edge-capacities $\{u_e : e \in E\}$, a source-sink pair $s, t \in V$, and an integer k.
Objective: Find a minimum cost set $V' \subseteq V$ such that the subgraph H of G induced by $V' \cup \{s, t\}$ satisfies $\lambda_H(s, t) \geq k$.

We study a particular case of Node-Weighted k-Flow on bipartite graphs which is also a particular case of Fixed Cost k-Flow.

<u>Node-Minimum Bibartite k-Flow</u>
Instance: A bipartite graph $G = (A \cup B, E)$ with integral edge-capacities $\{u_e : e \in E\}$, uniform costs on the vertices and an integer $k > 0$.
Objective: Find a minimum cost set $S \subseteq A \cup B$ such that $\lambda_G(S \cap A, S \cap B) \geq k$.

In [18] it is shown that if the uncapacitated version of this problem admits ratio ρ, then the Densest ℓ-Subgraph problem admits ratio $1/2\rho^2$. Independently, this problem was suggested to us by Deeparnab Chakrabarty in a personal communication.

The following special case of Fixed Cost k-Flow, that includes the Steiner Forest and k-Steiner Tree problems, was defined in [10].

<u>Generalized Point to Point Connection</u> (Generalized-P2P)
Instance: An undirected graph $G = (V, E)$ with edge-costs $\{c_e \mid e \in E\}$, a sub-partition V^+, V^- of V, and integer *charges* $\{b_v > 0 : v \in V^+\}$ and $\{b_v < 0 : v \in V^-\}$.
Objective: Find a minimum-cost spanning subgraph H of G such that $b(H') := \sum_{v \in H'} b_v \geq 0$ holds for every connected component H' of H.

It is easy to see that the above problem is equivalent to the following problem. Given an edge weighted graph with a special set $S \subseteq V$ of sources with each $s \in S$ associated with a number a_s, and a collection of sinks $T \subseteq V$, $T \cap S = \emptyset$, with numbers a_t for every $t \in T$, so that $\sum_{t \in T} a_t \geq \sum_{s \in S} a_s$, find a min cost subgraph so that in any connected component C, $\sum_{t \in C \cap T} a_t \geq sum_{s \in S \cap C} a_s$.

To transform this to a Fixed Cost k-Flow instance, add a source r and connect it to all the vertices $s \in S$ with an edge of capacity a_s. Add a sink z and connect every vertex of $t \in T$ to z with an edge of capacity a_t. The edges of the graph get infinite capacity. The goal is to find minimum fixed cost flow of value $\sum_{s \in S} a_s$. As the capacities not touching the source or the sink are infinite, clearly for delivering a flow of $\sum_{s \in S} a_s$, we just need that in any connected component $\sum_{t \in T} a_t \geq \sum_{s \in S} a_s$, hence the problem is a special case of *fixed cost flow*.

Our last problem is related to the Capacitated Buy at Bulk problem, that is a generalization of Fixed Cost k-Flow and Capacitated Multicommodity Flow. We are given an undirected graph $G = (V, E)$ with edge-capacities $\{u_e \mid e \in E\}$ and edge-costs $\{c_e \mid e \in E\}$, and connectivity/flow demands $\{d_v : v \in V\}$ to a single sink t. We have two options with regards to every edge e. First, we can *rent* $f(e) \leq u_e$ capacity units of e, and pay c_e per unit. The cost incurred in this case is $f(e) \cdot c_e$. The second possibility is to *buy* e, and then e can be assigned infinite capacity. Naturally, buying an edge is more expensive than renting a capacity unit of that edge. The cost incurred in this case is $M \cdot c_e$, where M is a large number called the *cost inflation factor*. (For simplicity we choose a uniform cost inflation factor, but in general there may be unrelated higher costs for buying then for renting, and our algorithms can handle this more general case as well.) Namely, we should determine a set E' of bought edges, which are assigned infinite capacity, and flow values $f(e)$ for edges in $E \setminus F'$ obeying the capacity constrains, so that d_v flow units are sent from every v to t (simultaneously). The overall cost is $M \cdot c(E') + \sum_{e \in E \setminus E'} f(e) \cdot c_e$.

Remark: There is a Capacitated Steiner Network version of the problem in which the flow should be sent separately and we wish to minimize the maximum cost of delivering d_v units over all v. A similar approximation follows for this version as well because of properties of submodular functions (details omitted). This version of the problem is a generalization of the Source Location problem (see [17]).

Capacitated Buy at Bulk seems very hard. It may be that Capacitated Buy At Bulk is significantly harder to approximate than Capacitated Multicommodity Flow, because Capacitated Multicommodity Flow is Capacitated Buy at Bulk edges were edges can only be bought.

But we add a very natural constraint. Then we show that Group Steiner like techniques allow us to get polylog approximation for the problem. The impose requirement is that the the set E' of bought edges forms a tree $T' = (V', E')$ which serves as a *backbone* tree. We can join t to the tree as we may assume that the distance between t to the tree is at most *opt*. A similar constraint appears in several other problems, such as Connected Dominating Set, Connected Facility Location, Connected Source Location, and other problems.

> Connected Rent or Buy Multicommodity Flow
> *Instance:* A graph $G = (V, E)$ with with edge-capacities $\{u_e \mid e \in E\}$ and edge-costs $\{c_e \mid e \in E\}$, a cost inflation number M, a single sink t, and demands $\{d_v : v \in V\}$.
> *Objective:* Find a subtree $T' = (V', E')$ of G containing t, and capacity values $\{f(e) \leq u_e : e \in E \setminus E'\}$, such that after the edges in E' are given infinite capacity, and each $e \in E \setminus E'$ is given capacity $f(e)$, every $v \in V \setminus V'$ can deliver d_v flow units to t, where the flows should be delivered simultaneously. Minimize $c(T) = M \cdot c(E') + \sum_{e \in E \setminus E'} f(e)c_e$.

Note that for any *given* tree T', after setting the capacities of the edges in e' to ∞, we can determine optimal $f(e)$ values by computing a min-cost max-flow in the network obtained as follows. Add a new source node s, connect s it to every $v \in V \setminus V'$ by an edge of capacity d_v and cost 0, and reset the cost of the edges in E' to 0.

1.2 Our results

In the Group Steiner problem we are given an undirected graph $G = (V, E)$ with edge-costs $\{c_e \mid e \in E\}$, and a collection of groups $g_1, g_2, \ldots, g_k \subseteq V$. The goal is to find a minimum cost subtree H of G that contains at least one node from every group. In the Group Steiner on Trees, G is a tree rooted at a node r, every group is a subset of the leaves (a leaf may belong to many groups), and H should be a subtree of T rooted at r that contains at least one leaf from every group. Halperin and Krauthgamer [9] prove that unless $\mathrm{NP} \subseteq \mathrm{ZTIME}\left(n^{\log^c n}\right)$ for some constant c, for every constant $\epsilon > 0$, Group Steiner on Trees admits no $O(\log^{2-\epsilon} n)$ approximation. In Sections 2 we give an approximation ratio preserving reduction from Group Steiner on Trees to the Fixed Cost k-Flow problem, thus obtaining the following result.

Theorem 1. Fixed Cost k-Flow *admit no* $O\left(\log^{2-\epsilon} n\right)$ *approximation for any constant* $\epsilon > 0$, *unless* $\mathrm{NP} \subseteq \mathrm{ZTIME}\left(n^{\log^c n}\right)$ *for some constant* c. *Consequently, this gives the first non constant hardness result for both* Capacitated Steiner Network, Capacitated Multicommodity Flow *and Capacitated Buy at Bulk.*

In [18] question is posed: does Node-Weighted k-Flow admits an $o(k)$ ratio? In Section 3 we resolve this question for the special case of Node-Minimum Bibartite k-Flow.

Theorem 2. Node-Minimum Bibartite k-Flow *admits a* $O(\sqrt{k \log k})$ *approximation algorithm. For unit capacities, the problem admits a* bicriteria *approximation algorithm that finds* $S \subseteq A \cup B$ *with* $|S| \leq |OPT|$ *such that* $\lambda_G(S \cap A, S \cap B) = \Omega(k/\text{polylog } n)$*. In addition, for unit capacities the problem is equivalent to the minimization version of the dense k-subgraph problem, hence admits an* $O(n^{0.172})$ *ratio [9].*

In Section 4, we prove the following theorem.

Theorem 3. Generalized-P2P *with* $b(V) := \sum_{v \in V} b_v = 0$ *admits a 2-approximation algorithm. Furthermore, if* $b(V)$ *is polynomially bounded in* $n = |V|$*, then* Generalized-P2P *admits an exact algorithm on instances when the input graph is a tree, and an approximation algorithm with ratio* $O(\log \min\{n', 2 + b(V)\})$ *on general graphs, where* $n' = |V^+ \cup V^-|$*.*

As was mentioned, Generalized-P2P generalizes the k-Steiner Tree and the Steiner Forest problems [10]. Our algorithm gives a *single* algorithm for both problems, but with logarithmic ratio. It would be very interesting to find a constant ratio approximation algorithm for Generalized-P2P, as it would give a unifying constant ratio algorithm for both k-Steiner Tree and Steiner Forest.

Finally, in Section 5, we prove the following theorem.

Theorem 4. Connected Rent or Buy Multicommodity Flow *admits an* $O(\log^{3+\epsilon} n)$*-approximation scheme.*

1.3 Previous work

Andrews et al. [1] present a polylogarithmic approximation ratio for Capacitated Multicommodity Flow problem under the assumption of *soft capacities*.

The Capacitated Steiner Network problem is a fundamental problems in combinatorial optimization. Even the Fixed-Cost Flow problem (the case of a single source and single sink) includes several fundamental problems. The directed Fixed-Cost Flow was shown to be Label-Cover hard by Even et al. [10] in 2002, which implies the same lower bound for directed Capacitated Steiner Network. The same hardness result was rediscovered independently by Chakrabarty et al. [6].

Goemans et al. [14] are the first who consider approximation algorithms for Capacitated Steiner Network with multiple pairs. However they mainly consider "soft capacities", where multiple copies of an edge are allowed. Carr et al. [5] observed that the natural cut-based LP-relaxation has an unbounded integrality gap even for the unicast case. Motivated by this observation they strengthened the basic cut-based LP by adding so-called Knapsack-Cover inequalities. Using these inequalities, they obtained constant factor approximation algorithms for some special graph topologies. However, in the general case, the integrality gap of the basic cut-based LP enhanced by Knapsack-Cover inequalities is $\Theta(n^2)$. Recently, In [6] various special cases of Capacitated Steiner Network are considered. For soft capacities, they give an $O(\log k)$ upper bound where k is the number of pairs with positive requirement. They algo give $O(\log n)$ approximation ratio for the case when requirements r_{ij} are equal for all $i, j \in V$. From results in our paper and [6], one can argue that the hard capacity case of Capacitated Steiner Network is **provably** harder to approximate than the soft capacity case. In [6], an $\Omega(\log \log n)$ hardness result for the case of soft capacities is presented. They gave no hardness result for the hard capacity case namely, the Capacitated Steiner Network problem. Approximation ratios or hardness results for the soft capacity case do not extend to Capacitated Steiner Network.

Garg, Konjevod, and Ravi [12] present an $O(\log N \cdot \log k)$-approximation algorithm for Group Steiner on trees where k is the number of groups, and N is the maximum size of a group. A combinatorial $O(\log^{2+\epsilon} n)$ ratio algorithm, is given for the problem in [8] and a primal dual algorithm is given in [20]. Krauthgamer [9] give a lower bound of $\Omega(\log^{2-\epsilon} n)$ for any fixed ϵ, unless NP has a quasi-polynomial-time Las Vegas algorithm.

Remark: Recemtly Deeparnab Chakrabarty and C. Seshadhri improved our hardness for Fixed Cost k-Flow in a paper to appear in Approx 2013.

2 Hardness of Fixed Cost k-Flow (Theorem 1)

Given an instance $(G = (V, E), \{c_e \geq 0 \mid e \in E\}, r, \{S_1, \ldots, S_k\})$ of Group Steiner on Trees, we construct an instance of Fixed Cost k-Flow as follows (see Figure 1 for an illustration). For a positive integer k, let $[k] = \{1, \ldots, k\}$. Construct a graph $G_+ = (V_+, E_+)$ from G by adding some new vertices and edges as follows. Let $V_+ = V \cup \{s\} \cup \{g_i \mid i \in [k]\}$ and $E_+ = E \cup F$ where $F = \{\{s, v\} \mid v \in \cup_{i \in [k]} S_i\} \cup \{\{v, g_i\} \mid v \in S_i, i \in [k]\} \cup \{\{g_i, r\} \mid i \in [k]\}$. Each edge $e \in E$ is assigned cost c_e and capacity $u_e = \infty$. Each edge $e = \{s, v\}$ for $v \in \cup_i S_i$ is assigned cost $c_e = 0$ and capacity $u_e = |\{i \mid v \in S_i, i \in [k]\}|$, i.e., number of groups v belongs to. Each edge $e = \{v, g_i\}$ for $v \in S_i, i \in [k]$ is assigned cost $c_e = 0$ and capacity $u_e = 1$. Each edge $e = \{g_i, r\}$ for $i \in [k]$ is assigned cost $c_e = 0$ and capacity $u_e = |S_i| - 1$, i.e., one less than the number of vertices in group S_i. Finally we set sink as $t = r$ and demand as $d = \sum_{i \in [k]} |S_i| = \sum_{v \in V} |\{i \mid v \in S_i, i \in [k]\}|$.

Now we show the following one-to-one correspondence between the feasible solutions of the original Group Steiner on Trees instance and that of the created Fixed Cost k-Flow instance.

Lemma 1. *There exists a solution for the* Group Steiner on Trees *instance of cost at most C if, and only if, there exists a solution for* Fixed Cost k-Flow *instance of cost at most C. Furthermore, the solution to* Group Steiner on Trees *can be computed in polynomial time from that to the Fixed cost flow instance, and vice versa.*

Proof. Let subtree $T = (V_T, E_T)$ be a solution of cost C to the Group Steiner on Trees instance. Let $H = E_T \cup F$ be a subgraph of G_+. Since all edges in F have cost 0, the cost of H is also C. We now argue that H forms a feasible

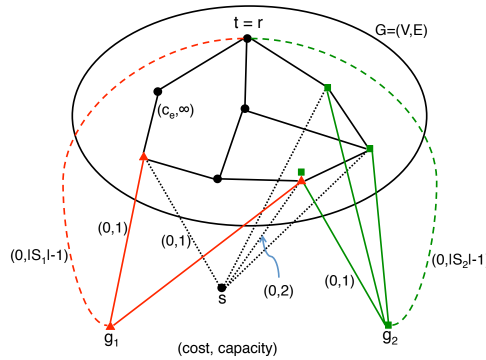

Fig. 1. The instance of Fixed Cost k-Flow created in the reduction from Group Steiner on Trees. The labels on the edges denote (cost, capacity). Not all labels are shown in the figure.

solution to Fixed cost flow, i.e., a flow of d units can be routed from s to t in H. We start by routing flow of $u_{\{s,v\}} = |\{i \mid v \in S_i, i \in [k]\}|$ units from s path from it to r in the tree T. This flow can be supported since received flow to each g_i for which $v \in S_i$ along the most $|S_i| - 1$ units of flow from all the vertices $v \in S_i$. This is because at most $|S_i| - 1$ vertices in S_i do not belong to T, which along edge $\{g_i, r\}$ of capacity $|S_i| - 1$. Thus indeed H forms a feasible solution to the Fixed cost flow instance.

Now let H be a solution of cost C to the Fixed cost flow instance. Since all edges in F have zero cost, we can assume that $F \subseteq H$, without loss of generality. It is enough to prove that $i \in [k]$. Suppose this is not true for some group S_j for $j \in [k]$. We extract an s-t-cut in graph H with capacity strictly less than d contradicting the existence of flow of value d from s to t in H. Let $U \subseteq V$ denote the set of vertices connected to some vertex in S_j in $H \cap E$ and let $U = \{s, g_j\} \cup U$. Note that $s \in U$ while from our assumption $t \notin U$. We now prove the following claim.

Claim. The total capacity of edges in H that leave U is strictly less than d.

Proof. It is easy to note that all the edges in H that leave U are (1) $\{g_j, r\}$ with capacity $|S_j| - 1$, (2) $\{v, g_i\}$ with capacity 1, for all $i \neq j$ and $v \in S_i \cap U$, and (3) $\{s, v\}$ with capacity $|\{i \mid v \in S_i, i \in [k]\}|$ for all $v \in V \setminus U$. Thus the total capacity of these edges is

$$
\begin{aligned}
&|S_j| - 1 + \sum_{i \neq j} \sum_{v \in S_i \cap U} 1 + \sum_{v \in V \setminus U} |\{i \mid v \in S_i, i \in [k]\}| \\
&= |S_j| - 1 + \sum_{v \in U} |\{i \mid v \in S_i, i \in [k], i \neq j\}| + \sum_{v \in V \setminus U} |\{i \mid v \in S_i, i \in [k]\}| \\
&= -1 + \sum_{v \in U} |\{i \mid v \in S_i, i \in [k]\}| + \sum_{v \in V \setminus U} |\{i \mid v \in S_i, i \in [k]\}| \\
&= -1 + \sum_{v \in V} |\{i \mid v \in S_i, i \in [k]\}| \\
&= d - 1.
\end{aligned}
$$

This finishes the proof of the claim.

The above claim implies that $H \cap E$ indeed contains a path from some vertex in S_i to r for each $i \in [k]$, establishing that it is a feasible solution to Group Steiner on Trees. From the reduction, it is also clear that the solution to Group Steiner on Trees can be computed in polynomial time from that to the Fixed Cost k-Flow instance, and vice versa. This completes the proof of Lemma 1.

Theorem 1 now follows from Lemma 1 and the hardness result for Group Steiner on Trees given in [9].

This hardness gives the first non constant hardness for Capacitated Steiner Network and Capacitated Multicommodity Flow. For those two problems the best hardness known before this paper was constant. Also, it implies the same lower bound on Capacitated Buy at Bulk.

Remark: We later show that the minimization version of the Dense k-subgraph is a special case of Fixed Cost k-Flow. While this does not provide any formal hardness, it is widely believed that the minimization version of Dense k-subgraph admits only polynomial ratios, which would imply that Fixed Cost k-Flow admits only polynomial ratio.

3 Node-Minimum Bibartite k-Flow (Theorem 2)

3.1 An $O(\sqrt{k \log k})$ approximation algorithm

Let S be any inclusion minimal solution to Node-Minimum Bibartite k-Flow. The above lemma implies that $|S| \leq 2k$, and it is also easy to see that in the case of unit capacities (recall that G is a simple graph), $|OPT| \geq |S| \geq 2\sqrt{k}$. This immediately implies ratio $\sqrt{k}$ for unit capacities.

Let us now consider the case of arbitrary capacities. We have two claims here and the main one works even if the nodes have arbitrary costs $c(u)$. Thus we describe the algorithm for the weighted case.

Algorithm Greedy

1. $S \leftarrow \emptyset$.
2. While $f(S) < k$ do:
   - Find $a \in A$ and $b \in B$ such that $(f(S \cup \{a, b\}) - f(S))/(c(a) + c(b))$ is maximum.
   - $S \leftarrow S \cup \{a, b\}$.
3. Return S.

In [7] the following is proven

Theorem 5. *Say that we iteratively find a partial solution with density $\rho \cdot opt/k$ for some function ρ, the resulting solution is an $O(\log n) \cdot \rho$ approximation. If $\rho = n^{\epsilon}$ then the approximation ratio is $O(\rho)$.*

We now show that we can find a solution with density *opt* times the optimal density.

The *best pair* is the pairs s, t so that the flow $f(s,t)$ s can deliver t over $c(s) + c(t)$ is maximum.

Note that we can decompose any flow into paths. This implies that the flow can be associated with a collection of pairs of vertices $a \in A$ and $b \in B$ so that the optimum sends $f^*(a, b)$ flow from a to b so that $\sum_{a,b} f^*(a, b) = k$.

Claim. The density of the best pair is $O(opt)$ times the optimal density.

Proof. Consider

$$\frac{sum_{x\in OPT,y\in OPT}(c(x)+c(y))}{\sum_{x\in OPT,y\in OPT} f^*(x,y)}.$$

Let $deg(a)$ and $deg(b)$ be the the number of neighbors of a,b among vertices chosen by the optimum. Note that $opt \geq deg(a), deg(b)$ because all the $deg(a)$ vertices chosen by opt have cost at least 1 (because of the costs are integral). Thus

$$\sum_{x\in OPT,y\in OPT}\left(\frac{c(x)+c(y)}{f^*(x,y)}\right) \leq \sum_{x\in OPT\cap A} deg(x)c(x)/k + \sum_{b\in OPT\cap B} deg(y)\cdot c(y)/k \leq \frac{opt^2}{k}.$$

Since $f(x,y) \geq f^*(x,y)$ we get

$$sum_{x\in OPT,y\in OPT}\frac{c(x)+c(y)}{f(x,y)} \leq \sum_{x\in OPT\cap A} deg(x)c(x)/k + \sum_{b\in OPT\cap B} deg(y)\cdot c(y)/k \leq \frac{opt^2}{k}.$$

By averaging, the best pairs has has density $opt \cdot opt/k$.

We can also use a simple algorithm that works well if opt is large. Unfortunately, this part of the algorithm works only for uniform costs. The algorithm chooses arbitrarily k pairs one after the other and increases the flow by 1 per pair. This solution has cost $2k$. In case $opt \leq \sqrt{k/\log k}$ since the density in the greedy algorithm is opt times the optimum, from the density claim (see [7]) we get an $O(\sqrt{k \log k})$ ratio. In the other case $opt \geq \sqrt{2k/\log k}$ the ratio is $O(\sqrt{k \log k})$ as well because of the $2k$ cost solution.

The case of unit capacities: In the case of unit capacities, the optimum uses at least k edges that carry flow because any cut must contain at least k saturated edges. In addition, given any subgraph that has at least k edges, we can send a flow of k directly through these edges. This implies that the uniform cost problem is equivalent to the minimization version of the dense k-Subgraph problem. In this problem we are given a graph G and a bound Q, and the goal is to select a minimum size U so that the number of edges with both endpoints in U is at least Q. In particular, it shows that the minimization version of the Dense k-subgraph problem is a special case of Fixed Cost k-Flow. While this does not give any lower bound, it is widely believed that the minimization version of the Dense k-subgraph problem admits only polynomial ratios.

The best ratio known for the minimization version of the Dense k-subgraph problem is $O(n^{0.172})$ and thus the uniform capacities case admits the same ratio.

3.2 A bicriteria approximation algorithm

We first reduce the problem to the tree instances using the theorem of Harrelson, Hildrum, and Rao [16]. A tree decomposition $\mathcal{T}$ of a groundset V is a sequence $\Pi_0, \ldots, \Pi_d$ of partitions of V, where $\Pi_0 = \{V\}$, $\Pi_d = \{\{v\} : v \in V\}$ and each Π_i is a refinement of Π_{i-1}. Such tree decomposition can be represented by a rooted tree, which we also denote by $\mathcal{T}$. The root of $\mathcal{T}$ is $\{V\}$. The nodes in layer i are the sets in Π_i and the leaves correspond to the sets in Π_d, i.e., the nodes in V. The edges of the tree go between the consecutive layers and are given by set inclusion. If $G = (V, E)$ is a graph with edge capacities u_e, then the weight of an edge (S, T) of $\mathcal{T}$ is $w(S, T) = u(\delta_G(S))$.

Now consider an instance of the multi-commodity flow demands $M = \{d_{ij} \geq 0 \mid i, j \in V\}$ between pairs of vertices. Let $c_G(M)$ (resp., $c_{\mathcal{T}}(M)$) denote the minimum maximum ?? edge-congestion under which M can be routed in G (resp., $\mathcal{T}$). Harrelson et al. [16] proved the following theorem.

Theorem 6 (Harrelson et al. [16]). *In time polynomial in $n = |V|$, one can compute a tree decomposition $\mathcal{T}$ with depth $d = O(\log n)$ such that for any multi-commodity flow instance M, we have*

- $c_{\mathcal{T}}(M) \leq c_G(M)$, *and*
- *given a routing of M with maximum edge-congestion $c_{\mathcal{T}}(M)$ in $\mathcal{T}$, we can compute in polynomial time, a routing of M with maximum edge-congestion $O(\log^2 n \log\log n) \cdot c_{\mathcal{T}}(M)$ in G.*

We use the above theorem to compute a tree decomposition $\mathcal{T}$ for the given bipartite graph $G = (A, B, E)$. It is easy to see that the optimum solution of the Node-Minimum Bibartite k-Flow instance in G induces a solution A^*, B^* in the tree $\mathcal{T}$ with same value and so that we can route at least k units of flow between A^* and B^* in $\mathcal{T}$. We next give an exact algorithm to find sets $A' \subseteq A, B' \subseteq B$ in $\mathcal{T}$ with minimum $|A'| + |B'|$ so that we can route k units of flow between them. The optimum solution A', B' in $\mathcal{T}$, in turn, induces a solution A', B' in G of value at most that of the optimum such that we can route $\Omega(k/\log^2 n \log\log n)$ flow.

The algorithm on the tree instances uses dynamic programming. For each vertex $u \in \mathcal{T}$ and values $0 \leq F, F^+, F^- \leq k$, we use $S^+(u, F, F^+)$ (resp., $S^-(u, F, F^-)$) to denote the minimum value $|A'| + |B'|$ such that there exists subsets $A' \subseteq A$ and $B' \subseteq B$ in the subtree $\mathcal{T}_u$ of $\mathcal{T}$ hanging below u so that we can route a flow of F units between A' and B' and send out (resp., bring in) a flow of F^+ (resp., F^-) units from vertices in A' (resp., from u) to u (resp., to vertices in B'), using only the edges in $\mathcal{T}_u$ and without violating the capacity of any edge. It is easy to compute S^+ and S^- values for leaf vertices $u \in \mathcal{T}$. Furthermore, given a non-leaf vertex $v \in \mathcal{T}$ and its children $u_1, \ldots, u_p$, it is easy to compute S^+ and S^- values for v from the corresponding values for its children. Finally, we read off the value of $S^+(r, k, 0)$ (or, equivalently $S^-(r, k, 0)$) (where r is the root of $\mathcal{T}$) to compute the optimum solution.

4 Generalized-P2P (Theorem 3)

4.1 An exact algorithm for Generalized-P2P on trees

Here we show how to solve Generalized-P2P optimally on instances when the input graph is a tree T, using dynamic programming. Root T at some node s. By adding zero-charge nodes and zero-cost edges to T if necessary, we can assume that T is a binary tree. If a node v has one child, then we add an additional child to v of charge 0, connected by an edge of cost 0. If v has at least 3 children, we add a binary tree rooted at v whose leaves are the children of v. In this binary tree, each leaf u is connected to to its parent by an edge of cost c_{uv}; non-leaf edges have cost 0 and non-leaf nodes distinct from v have charge 0. It is easy to see that the problem essentially remains unchanged by this modification.

For $v \in V$ let T_v denote the subtree of T that consist of v and its descendants. For an integer $B \in [b(V^-), b(V^+)]$ let $T(v, B)$ be the minimum-cost of a subgraph H of T_v satisfying the following:

- The connected component in H containing v has total charge B.
- Every other connected component in H has non-negative total charge.

If there is no subgraph H satisfying the above conditions, then $T(v, B) = \infty$. The optimal solution value is $\min\{T(s, B) \mid B \geq 0\}$. The dynamic program computes quantities $T(v, B)$ for all $v \in T$ and integer $B \in [b(V^-), b(V^+)]$. Since each b_u is polynomially bounded, the number of such quantities is polynomial. We assume that the corresponding minimum-cost subgraph H is also stored in the dynamic program table.

The quantities $T(v, B)$ can be computed as follows. If v is a leaf, then computing $T(v, B)$ is trivial. For an internal node v, we compute $T(v, B)$ as follows. Let u_1 and u_2 be the two children of v. Depending on the set $F \subseteq \{vu_1, vu_2\}$ picked to the solution, we get four possibilities.

1. $F = \emptyset$. Then $T(v, B) = \min\{T(u_1, B_1) + T(u_2, B_2) \mid B_1, B_2 \geq 0\}$ if $B = b_v$, and $T(v, B) = \infty$ otherwise.
2. $F = \{vu_1\}$. Then $T(v, B) = \min\{c_{vu_1} + T(u_1, B_1) + T(u_2, B_2) \mid B_2 \geq 0\}$ if $B = b_v + B_1$, and $T(v, B) = \infty$ otherwise.
3. $F = \{vu_2\}$. Then $T(v, B) = \min\{c_{vu_2} + T(u_2, B_2) + T(u_1, B_1) \mid B_1 \geq 0\}$ if $B = b_v + B_2$, and $T(v, B) = \infty$ otherwise.
4. $F = \{vu_1, vu_2\}$. Then $T(v, B) = \min\{c_{vu_1} + T(u_1, B_1) + c_{vu_2} + T(u_2, B_2)\}$ if $B = b_v + B_1 + B_2$, and $T(v, B) = \infty$ otherwise.

Among these possibilities, we pick the minimum-cost solution corresponding to each value of the charge of the connected component containing v.

4.2 A 2-approximation algorithm for the case $b(V) = 0$

Our 2-approximation algorithm generalizes the algorithm of [14] which is the case $b_v \in \{-1, 0, 1\}$. We say an edge e *covers* a set S if e has exactly one endvertex in S; we say that an edge-set/graph covers a set family $\mathcal{F}$ if for every $S \in \mathcal{F}$ there is an edge in H covering S. Given a set-family $\mathcal{F}$ and an edge-set H the residual set-family $\mathcal{F}_H$ consists of the members of $\mathcal{F}$ not covered by H. Recall that a set-family $\mathcal{F}$ is *uncrossable* if for any $X, Y \in \mathcal{F}$ at least one of the following holds: $X \cap Y, X \cup Y \in \mathcal{F}$ or $X \setminus Y, Y \setminus X \in \mathcal{F}$. It is known and easy to see that if $\mathcal{F}$ is uncrossable, so is $\mathcal{F}_H$, for any edge-set H.

Goemans et al. [13] give a primal-dual 2-approximation algorithm for the problem of finding a minimum-cost edge-cover of an uncrossable set-family $\mathcal{F}$. A polynomial time implementation of this algorithm requires only that for any edge-set H, the minimal members of the residual set-family $\mathcal{F}_H$ can be computed in polynomial time (but $\mathcal{F}$ itself may not be given explicitly). Now the 2-approximation algorithm follows from the following lemma.

Lemma 2. *Given an instance of* Generalized-P2P *with* $b(V) = 0$, *let* $\mathcal{F} = \{S \subseteq V \mid b(S) \neq 0\}$. *Then the following holds.*

(i) *An edge-set* $H \subseteq E$ *is a feasible solution to* Generalized-P2P *if, and only if,* H *covers* $\mathcal{F}$.
(ii) *For any edge set* $H \subseteq E$, S *is an inclusion-minimal members of* $\mathcal{F}_H$ *if, and only if* S *is a connected component of the graph* (V, H) *and* $b(S) \neq 0$.
(iii) $\mathcal{F}$ *is uncrossable.*

Proof. Parts (i) and (ii) are straightforward, so we prove only part (iii). Let $X, Y \in \mathcal{F}$, so $b(X), b(Y) \neq \emptyset$. We will show that if $X \cap Y \notin \mathcal{F}$ or if $X \cup Y \notin \mathcal{F}$, then $X \setminus Y, Y \setminus X \in \mathcal{F}$. Suppose that $X \cap Y \notin \mathcal{F}$, so $b(X \cap Y) = 0$. Then $b(X \setminus Y) = b(X) - b(X \cap Y) = b(X) \neq 0$ and $b(Y \setminus X) = b(Y) - b(Y \cap X) = b(Y) \neq 0$; hence $X \setminus Y, Y \setminus X \in \mathcal{F}$. Suppose that $X \cup Y \notin \mathcal{F}$, so $b(X \cup Y) = 0$. Then $b(X \setminus Y) = b(X \cup Y) - b(Y) = -b(Y) \neq 0$ and $b(Y \setminus X) = b(X \cup Y) - b(X) = -b(X) \neq 0$; hence $X \setminus Y, Y \setminus X \in \mathcal{F}$. □

4.3 An $O(\log |V^+ \cup V^-|)$-approximation algorithm

We already proved that Generalized-P2P can be solved optimally on tree instances. We next reduce the general problem to the case when the input graph is a tree with a loss of $O(\log n')$ factor in the approximation ratio, where $n' = |V^+ \cup V^-|$. This is achieved as follows. Consider the shortest-path metric on $V' = V^+ \cup V^-$ w.r.t. the edge-costs c_e. We probabilistically embed this metric into a tree metric T, c' with $O(\log n')$ distortion using the results of Bartal [3] and Fakcharoenphol, Rao and Talwar [11]. There is a one-to-one correspondence between V' and the set L of leaves of T. The resulting instance of Generalized-P2P on T inherits the charges on the leaves of T from the original charges on vertices of V', while the charge of internal vertices of T is 0. We compute an optimal solution to the obtained tree instance, and return the corresponding subgraph H of G. Note that any feasible solution with cost C for the original instance induces a solution with cost $O(C \log n')$ for the new instance on tree T. Similarly any feasible solution with cost C for the new instance induces a solution with cost C for the original instance. Hence the approximation ratio is bounded by the distortion of the reduction, which is $O(\log n')$.

Now consider the augmentation version of the problem, when we are give an edge subset $E' \subseteq E$ of cost 0. Then we can contract every connected component F of (V, E') into a single vertex v_F with charge $b(v_F) = b(F)$. Thus the approximation ratio in this case is $O(\log n')$, where n' is the number of connected components with non-zero charge in the graph (V, E').

4.4 An $O(\log(2 + b(V)))$-approximation algorithm

The main novelty in this result is that the ratio becomes smaller as $b(V)$ becomes smaller. In general, $b(V)$ may be very small as compared to $|V^- \cup V^+|$.

Lemma 3. *There exists a polynomial time algorithm that given an instance of* Generalized-P2P *computes an edge set* $E' \subseteq E$ *of cost* $\leq 4\tau^*$*, where* τ^* *denotes the optimal solution value, such that the number* n' *of connected components with non-zero charge in the graph* (V, E') *is at most* $4b(V)$.

Proof. Fix a parameter τ, which is an estimate for τ^*. Create an instance of Generalized-P2P with total charge zero by adding a new vertex s with charge $-b(V)$ and connecting s to each vertex in V^+ by an edge of cost $\tau/b(V)$. Then apply the 2-approximation algorithm for the case $b(V) = 0$. The new instance admits a solution of cost at most $\tau^* + b(V) \cdot (\tau/b(V)) = \tau^* + \tau$, by taking an optimal solution to the original instance with edges that connect s to at most $b(V)$ vertices in V^+. Thus the procedure returns an edge-set of cost at most $2(\tau^* + \tau)$. Consequently, if $\tau \geq \tau^*$ then the procedure returns an edge-set of cost at most 4τ, and the number of edges incident to s is at most $4\tau/(\tau/b(V)) = 4b(V)$. Using binary search, we find the minimum integer τ for which the procedure returns an edge-set E'' of cost 4τ. Then $c(E'') \leq 4\tau \leq 4\tau^*$ and the number of edges in E'' incident to s is at most $4b(V)$. Let E' be obtained from E'' by removing the edges incident to s. Then $c(E') \leq c(E) \leq 4\tau^*$, and the number n' of connected components in (V, E') with non-zero-charge is at most the degree of s w.r.t. E'', hence at most $4b(V)$, as claimed. □

The entire algorithm has two steps. At step 1 we compute an edge set E' as in the above lemma. Step 2 applies the $O(\log n')$)-approximation algorithm from the previous section to compute an augmenting edge-set $F \subseteq E \setminus E'$ such that $E' \cup F$ is a feasible solution. The solution cost is bounded by $c(E') + c(F) = O(\tau^*) + O(\log n') \cdot \tau^* = O(\log(2 + b(V))) \cdot \tau^*$.

5 Connected Rent or Buy Multicommodity Flow (Theorem 4)

5.1 The Submodular Cover with Tree Costs problem

A set-function $f : 2^U \to \mathbb{Z}$ is *submodular* if $f(A) + f(B) \geq f(A \cap B) + f(A \cup B)$ for all $A, B \subseteq U$; F is called non-decreasing if $A \subseteq B$ implies $f(A) \leq f(B)$. Let $f : 2^U \to \mathbb{Z}^+$ be a non-negative and non-decreasing submodular set function on a groundset U. We assume that $f(U)$ is bounded by a polynomial in $n = |U|$. Let $c : U \to \Re^+$ be a cost function and denote the cost of a set $S \subseteq U$ by $c(S) = \sum_{i \in S} c(i)$. A *submodular cover problem* is to find a minimum-cost subset S so that $f(S) = f(U)$. The following is proved in [19].

Theorem 7. *The submodular cover problem admits a* $\max_{u \in U}\{\ln(f(\{u\})) + 1\}$*-approximation algorithm.*

In [4], the Submodular Cover with Tree Costs problem is studied. The difference between this problem and submodular cover is in the objective function. In the Submodular Cover problem the objective function is $c(U')$. In the submodular cover with tree costs problem, U are leaves in a tree T' rooted at a root r. After a feasible set U', so that $f(U') = f(U)$ is found, U' induces a unique subtree $T_{U'}$ with all paths leading from the root r to U'. The cost of U' is defined as the cost of $T_{U'}$. Submodular cover is just the special case that the tree is a star. Note that the edges of T' are *bought* so after buying them their capacity is set to ∞. Because of the ∞ capacity of the edges bought, it is enough for $v \notin T'$ to send d_v flow units to the leaves of T'. These leaves can forward the d_v flow units to t, over T', due to the infinite capacities.

In [4] [5] a $\log^{2+\epsilon} n$ ratio approximation is given for this problem. The solution is highly complex and uses the complex algorithm of [8] and changes it to an even more complex algorithm.

5.2 The Algorithm

We provide an $O(\log^{3+\epsilon} n)$ approximation for the Connected Rent or Buy Multicommodity Flow problem.

Le $G = (V, E)$ denote the graph instance for the Connected Rent or Buy Multicommodity Flow problem. Transform G with costs $M \cdot c_e$ into a random tree T using [11]. This incurs a loss of $O(\log n)$ in the approximation ratio in the cost of buying edges. Our goal is to buy a subtree T' of T containing t. Note that T' will induce a tree $G(T')$ containing t in G. This breaks G into two sets T' and $V \setminus T'$.

Luckily, in the [11] construction, the vertices V of G are leaves in T. Therefore we can set V as the universe. We meet the requirement of Submodular Cover with Tree Costs that the universe, namely V, is a collection of leaves in some tree.

Our algorithm works in two greedy phases:

The first greedy phase: In the first phase we define a submodular cover problem, which immediately implies a Submodular Cover with Tree Costs instance. The goal of this first stage is to buy some tree $T' \subseteq T$ containing t, transform the capacities of T' to ∞, and assure that every $v \notin T'$ will be able to send d_v flow units to t.

Remark: This first stage may lead to a very large rent cost. We use a second greedy phase to deal with that.

Add a new vertex w that will later serve as a *sink* in some flow functions we define. For every v, add an edge (v, w) of cost 0 and capacity d_v. Vertices in $V \setminus T'$ will serve as sinks in later computation, even though they are sources (namely want to send d_v flow to t). This slightly unusual situation will be "fixed" later.

We now define the following submodular cover function. Let $U' \subseteq V$ and let $f(U')$ be the minimum between $\sum_{v \in V} d_v$ and the flow that U' can deliver to w, when U' are sources. Note that for every vertex v in U', since v is a source, the edge (v, w) can be used to deliver d_v flow units to w. However, since vertices in $V \setminus U'$ are not sources delivering d_v flow units to via the edge (v, w) is not a simple matter. This function is submodular [2]. Note that $f(V)$ is $\sum_{v \in V} d_v$ because every vertex is connected directly to w with a capacity d_v. Since U' are leaves in T, this step automatically defines a Submodular Cover with Tree Costs problem. When a tree T' is chosen, if t does not belong to the tree, we add a shortest cost path from t to the tree. Note that as the optimum value *opt* is polynomial in n, we can guess the optimum value by going over all possibilities.

[5] Being unaware of [4], we derived it independently in an earlier draft of this paper [15]

We may discard vertices in G whose distance from t is more than *opt*. Thus the stage of adding t into T' adds at most *opt* cost. Recall that any U' defines a tree $T_{U'}$ induced by the leaves U'.

Claim. If U' satisfies $f(U') = f(V)$, $v \notin U'$ can deliver d_v flow units to t.

Proof. As $f(U') = f(V)$, U', by definition, U' can send $\sum_v d_v$ flow to w hence d_v flow to every $v \notin U'$. Note that for vertices in U' the direct edges to w is used to gets d_v flow units for $v \in U'$.

Now consider $v \notin U'$. Since G is undirected, we can reverse the direction of the flow and send d_v flow units from every $v \notin U'$ to the (leaves) vertices of U'. As edges of T' have infinite capacity, this flow can be forwarded to t via T'.

The ratio derived is as follows according to [4]. A factor of $O(\log^{1+\epsilon} n \cdot \log(\sum_{v \notin T'} d_v) = O(\log^{2+\epsilon} n)$ in the approximation algorithm is due to the use of [4]. We further lose an $O(\log n)$ factor due to the use of [11]. Eventually, the implies ratio is $\log^{3+\epsilon} n$ for this first phase.

The second greedy phase: reducing the flow cost.

Say that at stage 1 U' are the leaves of T'. At this time the rent cost may be very high. The only way to reduce it is to add more vertices of $W = V \setminus U'$ to T'. This gives more bought edges, a things that decrease the rent cost. Thus W is the universe. Let α be the rent cost in the optimum. As we assume all numbers are polynomial in n, we may try all different α and guess the optimum one.

We define a submodular cover instance on W and later a Submodular Cover with Tree Costs instance on W. For a set $W' \subseteq W$ consider adding these W' to the tree, and computing a min-cost max-flow solution with $U' \cup W'$ as sources and the vertex w defined above as the sink. Note that the cost of the flow is by definition *exactly* the rent cost of the edges not in T'. Define

$$f_2(W') = \min\{ \sum_{e \text{ rented by } W' \cup U'} -f(e) \cdot c(e), -\alpha \}.$$

To compute the first part of the maximization above we use the standard polynomial solution for the min cost max flow problem, with $U' \cup W'$ as sources. The function $f_2(W')$ is submodular [2].

Claim. If $f_2(W') = f(V)$ then the flow cost of W' $\sum_{e \text{ rented by } W' \cup U'} f(e) c_e \leq \alpha$ namely the flow cost of W' is at most the optimum flow cost.

Proof. Note that $f(V) = -\alpha$ because the flow cost of the direct edges (v, w) is 0, and $-\alpha < 0$. Say that $f(W') = f(V)$ and thus $f(W') = -\alpha$, This must imply that $\sum_{e \text{ rented by } W' \cup U'} f(e) c_e \leq \alpha$ for otherwise $\sum_{e \text{ rented by } W' \cup U'} -f(e) c_e < -\alpha$ contradicting the assumption that $f_2(W') = -\alpha$. Thus if $f(W' \cup U') = f(V)$ our rent cost is no larger than the one of the optimum.

Since edges we buy form a tree containing t, it does not matter *how do we join* W' to the tree. The least cost way to join W' to the tree is thus the best. This immediately defines a submodular cover problem with tree costs. The ratio implied by [4] for the cost is at most $O(\log^{2+\epsilon} n) \cdot \log(\sum_e u_e \cdot c_e)$. with an additional $\log n$ penalty by [11] deriving a $O(\log^{3+\epsilon} n)$ for Connected Rent or Buy Multicommodity Flow.

We have two stages in which we get $O(\log^{3+\epsilon} n)$ approximation with respect to the cost. Further, we reduced the rent cost to at most the optimum value α. This implies a $\log^{3+\epsilon} n$ approximation for Connected Rent or Buy Multicommodity Flow.

References

1. M. Andrews, S. Antonakopoulos, and L. Zhang. Minimum-cost network design with (dis)economies of scale. In *FOCS*, pages 585–592, 2010.
2. J. Bar-Ilan, G. Kortsarz, and D. Peleg. How to allocate network centers. *J. Algorithms*, 15(3):385–415, 1993.
3. Y. Bartal. On approximating arbitrary metrices by tree metrics. In *STOC*, pages 161–168, 1998.
4. G. Calinescu and A. Zelikovsky. The polymatroid steiner problems. In *ISAAC*, pages 234–245, 2004.
5. R. Carr, L. Fleischer, V. Leung, and C. Phillips. Strengthening integrality gaps for capacitated network design and covering problems. In *SODA*, pages 106–115, 2000.
6. D. Chakrabarty, C. Chekuri, S. Khanna, and N. Korula. Approximability of capacitated network design. In *IPCO*, 2010.
7. M. Charikar, C. Chekuri, T. Cheung, Z. Dai, A. Goel, S. Guha, and M. Li. Approximation algorithms for directed Steiner problems. *J. Algorithms*, 33:73–91, 1999.
8. C. Chekuri, G. Even, and G. Kortsarz. A greedy approximation algorithm for the group Steiner problem. *Discrete Appl. Math.*, 154(1):15–34, 2006.
9. E. Chlamtac, M. Dinitz, and R. Krauthgamer. Everywhere-sparse spanners via dense subgraphs. In *FOCS*, pages 758–767, 2012.
10. G. Even, G. Kortsarz, and W. Slany. On network design problems: fixed cost flows and the covering steiner problem. *ACM Trans. Algorithms*, 1:74–101, 2005.
11. J. Fakcharoenphol, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *J. Comput. System Sci.*, 69(3):485–497, 2004.
12. N. Garg, G. Konjevod, and R. Ravi. A polylogarithmic approximation algorithm for the group Steiner tree problem. *J. Algorithms*, 37(1):66–84, 2000.
13. M. X. Goemans, A. V. Goldberg, S. A. Plotkin, D. B. Shmoys, va Tardos, and D. P. Williamson. Improved approximation algorithms for network design problems. In *SODA*, pages 223–232, 1994.
14. M. X. Goemans and D. Williamson. A general approximation technique for constrained forest problems. *SIAM J. Comput.*, 24(2):296–317, 1995.
15. M. Hajiaghayi, R. Khandekar, G. Kortsarz, and Z. Nutov. Combinatorial algorithms for capacitated network design. *CoRR*, abs/1108.1176, 2011.
16. C. Harrelson, K. Hildrum, and S. Rao. A polynomial-time tree decomposition to minimize congestion. In *SPAA*, pages 34–43, 2003.
17. G. Kortsarz and Z. Nutov. A note on two source location problems. *J. Discrete Algorithms*, 6(3):520–525, 2008.
18. Z. Nutov. Approximating steiner networks with node-weights. *SIAM J. Comput.*, 39(7):3001–3022, 2010.
19. L. A. Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2:385–393, 1982.
20. L. Zosin and S. Khuller. On directed steiner trees. In *SODA'02*, pages 59–63, 2002.