Capacitated Network Design Problems: Hardness, Approximation Algorithms, and Connections to Group Steiner Tree

MohammadTaghi Hajiaghayi, Rohit Khandekar, Guy Kortsarz, and Zeev Nutov

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. We design combinatorial approximation algorithms for the Capacitated Steiner Network (Cap-SN) problem and the Capacitated Multicommodity Flow (Cap-MCF) problem. These two problems entail satisfying connectivity requirements when edges have costs and hard capacities. In Cap-SN, the flow has to be supported separately for each commodity while in Cap-MCF, the flow has to be sent simultaneously for all commodities. We show that the Group Steiner problem on trees ([12]) is a special case of both problems. This implies the first polylogarithmic lower bound for these problems by [17]. We then give various approximations to special cases of the problems. We generalize the well known Source location problem (see for example [19]), to a natural problem called the Connected Rent or Buy Source Location problem. We show that this problem is a simplification of Cap-SN and Cap-MCF and a generalization of Group Steiner on general graphs. We use Group Steiner Tree techniques, and more sophisticated techniques, to derive \( \log^{3+\epsilon} n \) approximation for the Connected Rent or Buy Source Location problem which is close to the best approximation known for Group Steiner on general graphs. Another special case we study is as follows. Given a bipartite graph \( G = (A \cup B, E) \) and an integer \( k \geq 0 \), find \( A' \subseteq A \) and \( B' \subseteq B \) of minimum total size \( |A'| + |B'| \) such that there exist \( k \) edge-disjoint paths in \( G \) from vertices in \( A' \) to vertices in \( B' \). This problem is a special case of the Steiner Network problem with vertex costs [20]. In [20] Nutov asked the open question if the Steiner network problem with vertex costs admits an \( o(k) \) ratio. We give an \( o(k) \) approximation for this special case, which could be a step toward resolving the open question of Nutov. We provide an \( O(\sqrt{k \log k}) \) approximation ratio for the problem. We also show that we can compute a solution of optimum value, while being able to route \( O(k/\text{polylog } n) \) flow, where \( n \) is

* Part of this work was done at DIMACS. We thank DIMACS for their hospitality.
** 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.
the number of vertices in $G$. The final special case of Cap-SN and Cap-MCF that we study is called the Unbalanced-P2P problem. Besides its practical applications to shift design problems [8], it generalizes many problems such as $k$-Steiner tree, Steiner Forest, and Point-to-Point Connection. We give a combinatorial logarithmic approximation algorithm for this problem.

1 Introduction

In this paper we assume all numbers are integral and bounded by a polynomial in $n$, the number of vertices in the graph.

We study the following two fundamental capacitated network problems.

\begin{itemize}
  \item \textbf{Capacitated Steiner Network (Cap-SN)}
  \item \textbf{Capacitated Multi Commodity Flow (Cap-MCF)}
\end{itemize}

The second problems we consider is:

\begin{itemize}
  \item \textbf{Group Steiner on trees (Group Steiner Tree)}
\end{itemize}

The case when the requirements have one source and one sink is called the Fixed Cost Flow problem in [11], and it is a special case of both Cap-SN and Cap-MCF. Even for the Fixed cost flow problem, the best known approximation ratio is $\Omega(|E|)$.

We show hardness for these problems by a reduction from Group Steiner on trees.

**Theorem 1.** [17] Unless $NP \subseteq ZTIME(n^{\log n})$ for some constant $c$, for every constant $\epsilon > 0$, Group Steiner Tree admits no $O(\log^{2-\epsilon} k)$ approximation, where $n$ and $k$ denote the number of vertices and groups respectively.
We prove an \( \Omega(\log^{2-\epsilon} n) \) hardness for both Cap-SN and Cap-MCF for any universal constant \( \epsilon > 0 \) by giving an approximation ratio preserving reduction from Group Steiner Tree problem to these problems. Halperin and Krauthgamer [17] prove an \( \Omega(\log^{2-\epsilon} k) \) lower bound on the approximation factor for Group Steiner Tree where \( k \) is the number of groups. However, the size of their construction is \( O(k) \). In particular, the number of vertices they have in the tree is less than \( 2k \). We present the lower bound in terms of the number \( n \) of vertices in their tree, which is between \( k \) and \( 2k \). Since the lower bound is polylogarithmic, \( k \) and \( n \) are essentially the same for our purposes.

In the Source location problem (see [19]), we are given a graph \( G(V, E) \) with capacities \( u_e \) for every edge \( e \), cost \( c_v \) for every vertex \( v \), and a single sink \( t \). The goal is to buy minimum cost subset \( V' \) of vertices (that serve as sources) so that for every \( v \not\in V' \), \( v \) can deliver \( d_v \) flow units to \( t \), namely, every cut containing \( v \cup V' \) on the one side and \( t \) on the other side has capacity at least \( d_v \). The Connected Rent or Buy Source Location problem generalizes Source location and is related to the Connected (non metric) Facility Location problem and the Single source Rent or Buy problem (see [16]). It is also a simplification of our two main problems, Cap-SN and Cap-MCF by adding a constraint on top of these two problems, and a generalization of The Group Steiner problem on general graphs. Let \( M \) be some (possibly large) integer. We have two options with regards to every edge \( e \). First, we can buy \( e \). In this case we pay \( M \cdot c_e \) cost once and we are free to deliver as much flow as we wish on \( e \), namely bought edges are assigned infinite capacities. Naturally buying an edge is more expensive. 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 algorithm can handle this more general case as well. The second possibility is to rent edges. An edge that is rented induced \( c_e \) cost per unit of flow that goes via \( e \) (renting means that you have to pay per every use of \( e \) separately). Thus the cost incurred by rented edges is \( \sum e \) is rented \( f(e) \cdot c_e \). The connectivity requirement constraint is that the edges bought need to induce a tree \( T' \) containing \( t \). This tree serves as backbone tree. This constraint is motivated by Connected Facility Location that requires the facilities to be connected. Its also motivated by the Connected dominating set problem and similar problems. In Connected Rent or Buy Source Location it is required that after we set the capacities of edges in \( T' \) to \( \infty \), every vertex \( v \not\in V(T') \) should be able to send \( d_v \) flow to \( t \). The goal is to minimize \( c(T) = M \cdot \sum e \) bought \( c_e + \sum e \) rented \( f(e) \cdot c_e \) with \( f(e) \) the flow on \( e \). The Capacitated Rent or Buy source location problem is like Connected Rent or Buy Source Location problem, except that we do not impose that the edges bought will be a tree. Then even with buy edges only, the problem is equivalent to the Cap-SN problem with a single source, and if there is a single source and a single sink the problem is equivalent to Cap-MCF problem with one source and one sink. Thus Connected Rent or Buy Source Location is therefore a simplification of our two main problems (albeit, we do not know if its a special case of these problems). Our problem is not to be confused with uncapacitated Rooted Rooted Rent or Buy problem [16] in which the capacities are \( \infty \) which makes the problem much much simpler.
**Connected Rent or Buy Source Location** (Connected Rent or Buy Source Location)

*Instance:* An undirected graph $G = (V, E)$ 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 $d_v$ for every $v \in V$.

*Objective:* Buy a subtree $T'$ of $G$ containing $t$, so that after the edges of $T'$ are given infinite capacity, $v \notin T'$ can deliver $d_v$ flow units to $t$. Minimize $c(T) = M \cdot \sum_{e \in T'} c_e + \sum_{e \notin T'} f(e)c_e$ with $f(e)$ the flow over $e$.

Approximating the Connected Rent or Buy Source Location problem requires combining many tools and using some new ideas. We use [9] to transform the graph to a (random) tree. We then use [4] that gives a polylogarithmic approximation for the submodular cover problem with tree costs. Further, we define submodular problems in which leaves in the bough tree $T'$ serve as sources, while vertices in $V \setminus T'$ that are sources (want to send $d_v$ flow to $t$) serve as sinks. The idea is that if $d_v$ flow units could be delivered from the leaves of $T'$ to $v$, since the graph is undirected there is a similar flow from $v$ to the leaves of $T'$. Since bought edges have infinite capacity, $d_v$ flow units can be delivered from $v$ to $t$. On top of that we need a two stage greedy solution to derive the approximation. One to assure that the flow is maximum and one to reduce the rent cost.

We also study the following problem, which was suggested to us by Deeparnab Chakrabarty in a personal communication. This is a special case of Steiner network with vertex costs problem [20]. Nutov [20] posed the question if Steiner network with vertex costs has an $o(k)$ ratio. We prove such a result but for this special case.

**$k$-Bipartite Flow**

*Instance:* A bipartite graph $G = (A \cup B, E)$ with bipartition $A$ and $B$ and an integer $k > 0$.

*Objective:* Find subsets $A' \subseteq A$ and $B' \subseteq B$ of minimum total size $|A'| + |B'|$ such that $G$ has $k$ edge-disjoint paths from vertices in $A'$ to vertices in $B'$.

It is our hope that these methods, may assist in solving the general open problem of Nutov.

The next problem we study is called the Unbalanced Point to Point Connection, or Unbalanced-P2P. In [8], it is shown that Unbalanced-P2P is a special case of the Fixed cost flow problem.

**Unbalanced Point to Point Connection** (Unbalanced-P2P)

*Instance:* An undirected graph $G = (V, E)$ with edge-costs $\{c_e \mid e \in E\}$ and integer charges $\{b_v : v \in V\}$ (which can be negative).

*Objective:* Find a minimum-cost subgraph $H$ of $G$ containing all vertices, such that for every connected component $H'$ $b(H') := \sum_{v \in H'} b_v \geq 0$.

It is easy to see that the problem has a feasible solution if, and only if, $G$ is a feasible solution, i.e., every connected component $C$ of $G$ satisfies $b(C) \geq 0$, and that any inclusion-minimal solution is a forest. Given an instance of Unbalanced-P2P, let $V^+ = \{v \in V \mid b_v > 0\}$ and let $V^- = \{v \in V \mid b_v < 0\}$.
1.1 Our results

In Sections 3 and 4, we prove the following theorems respectively.

**Theorem 2.** The Cap-SN and Cap-MCF problems admit no better than $\Omega(\log^{2-\epsilon} n)$ approximation for any constant $\epsilon > 0$, unless $NP \subseteq ZTIME(n^{\log^c n})$ for some constant $c$.

**Theorem 3.** There exists a polynomial-time combinatorial $O(\log^{3+\epsilon} n)$-approximation algorithm for Connected Rent or Buy Source Location for any $\epsilon > 0$.

The best known result for Group Steiner on graphs (which is a special case of Connected Rent or Buy Source Location) is $O(\log^3 n)$ [12], hence our algorithm is almost optimal with respect to the knowledge today.

**Theorem 4.** The $k$-Bipartite Flow problem admits an $O(\sqrt{k \log k})$ combinatorial approximation algorithm.

**Theorem 5.** The $k$-Bipartite Flow problem admits a bicriteria polylogarithmic approximation in the following sense. Given a $k$-Bipartite Flow problem instance, we can find subsets $A' \subseteq A$ and $B' \subseteq B$ with $|A'| + |B'| \leq \text{value of the optimum solution}$ such that $G$ can support $\Omega(k/\text{polylog } n)$ flow from vertices in $A'$ to those in $B'$.

The proofs of Theorems 4 and 5 are given in an appendix. In Section 5, we prove the following theorem.

**Theorem 6.** There exists a polynomial-time combinatorial 2-approximation algorithm for the special case of Unbalanced-P2P with $b(V) := \sum_{v \in V} b_v = 0$, by the assumption that numbers are polynomial in $n$. Unbalanced-P2P admits an exact algorithm on tree instances (i.e., $G$ is a tree) and ratio $O(\log \min\{n', 2 + b(V)\})$ on general graphs, where $n' = |V^+ \cup V^-|$ is the number of vertices with non-zero charge.

An interesting challenge: The Unbalanced-P2P problem generalizes $k$-Steiner tree and Steiner Forest problems [8]. Our algorithm gives a single algorithm for both problems, but with logarithmic ratios. The challenge is to find a constant ratio for Unbalanced-P2P giving a single algorithm that can approximate within constant the $k$-Steiner tree and the Steiner forest problems. This seems to us to be a significant goal.

2 Previous work

Andrews et al. [1] present a polylogarithmic approximation ratio for Cap-MCF problem under the assumption of soft capacities.

The Cap-SN problem is a fundamental problem 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. [8] in 2002, which implies the same
lower bound for directed Cap-SN. The same hardness result was rediscovered independently by Chakrabarty et al. [6].

Goemans et al. [14] are the first who consider approximation algorithms for Cap-SN 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 by adding so-called Knapsack-Cover inequalities is $\Theta(n^2)$. Recently, In [6] various special cases of Cap-SN are considered. For soft capacities, they give an $O(\log k)$ upper bound where $k$ is the number of pairs with positive requirement. They also 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 Cap-SN 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 Cap-SN problem. Approximation ratios or hardness results for the soft capacity case do not extend to Cap-SN.

Garg, Konjevod, and Ravi [12] present an $O(\log N \cdot \log k)$-approximation algorithm for Group Steiner Tree on tree 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 [7] and a primal dual algorithm is given in [22]. Krauthgamer [17] give a lower bound of $\Omega(\log^{2-\epsilon} n)$ for any fixed $\epsilon$, unless NP has a quasi-polynomial-time Las Vegas algorithm.

3 Hardness of Cap-SN and Cap-MCF (Proof of Thm 2)

We prove the hardness result for the case of a single source and a single sink, called the Fix cost flow problem [11]. This in turn implies the same hardness for Cap-SN and Cap-MCF. 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 the Fixed cost flow problem. 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 $F = \{\{s, v\} \mid v \in \bigcup_{i \in [k]} S_i\} \cup \{\{v, g_i\} \mid v \in S_i, i \in [k]\} \cup \{\{g_i, r\} \mid i \in [k]\}$. Each edge $e \in E$ is assigned cost $c_e$ and capacity $u_e = \infty$. Each edge $e = \{s, v\}$ for $v \in \bigcup_i S_i$ 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 Tree and that of the created Fixed cost flow instance.
Lemma 1. There exists a solution for the Group Steiner Tree with cost at most $C$ if, and only if, there exists a solution for Fixed cost flow instance with cost at most $C$. Furthermore, the solution to Group Steiner Tree 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 Tree 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 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 $u(s,t) = |\{i \mid v \in S_i, i \in [k]\}|$ 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

$$|S_j| - 1 + \sum_{i \neq j} \sum_{v \in S_i \cap U} 1 + \sum_{v \in V \setminus U} |\{i \mid v \in S_i, i \in [k]\}|$$

$$= |S_j| - 1 + \sum_{v \in U} |\{i \mid v \in S_i, i \in [k], i \neq j\}| + \sum_{v \in V \setminus U} |\{i \mid v \in S_i, i \in [k]\}|$$
\[
= -1 + \sum_{v \in U} |\{i \mid v \in S_i, i \in [k]\}| + \sum_{v \in V \setminus U} |\{i \mid v \in S_i, i \in [k]\}|
\]

\[
= -1 + \sum_{v \in V} |\{i \mid v \in S_i, i \in [k]\}|
\]

\[
= d - 1.
\]

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 Tree. From the reduction, it is also clear that the solution to Group Steiner Tree can be computed in polynomial time from that to the Fixed cost flow instance, and vice versa. This completes the proof of Lemma 1.

Theorem 2 now follows from Lemma 1 and the hardness result for Group Steiner Tree given in [17].

### 4 Approximating the Connected Rent or Buy Source location problem

#### 4.1 Submodular cover problems

For a set \( U \), a set-function \( f : 2^U \rightarrow \mathbb{Z} \) is called submodular if

\[
f(A) + f(B) \geq f(A \cap B) + f(A \cup B).
\]

A set-function is called non-decreasing if \( A \subseteq B \) implies \( f(A) \leq f(B) \). Let \( U \) be a collection of items and \( f : 2^U \rightarrow \mathbb{Z}^+ \) be a non-negative and non-decreasing submodular function. We assume that \( f(U) \) is bounded by a polynomial in \( n = |U| \). Let \( c : U \rightarrow \mathbb{R}^+ \) 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 [21].

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

#### 4.2 The Submodular Cover with Tree Costs problem

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 \( \infty \). Because of the \( \infty \) 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] a log^{2+\epsilon} n ratio approximation is given for this problem. The solution is highly complex and uses the complex algorithm of [7] and changes it to an even more complex algorithm.

### 4.3 The Connected Rent or Buy Source Location problem

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

Let G = (V, E) denote the graph instance for the Connected Rent or Buy Source Location problem. Transform G with costs M \cdot c_e into a random tree T using [9]. 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 [9] 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 \infty, and assure that every v \not\in 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. 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'.

---

5 Being unaware of [4], we derived it independently in an earlier draft of this paper [15]
Claim. If $U'$ satisfies $f(U') = f(V)$, $v \not\in U'$ 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 $v \not\in U'$. 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 $v \not\in U'$. Since $G$ is undirected, we can reverse the direction of the flow and send $d_v$ flow units from every $v \not\in U'$ 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 $O(\log^{1+\epsilon} n \cdot \log(\sum_{v \in T'} d_v)) = O(\log^{2+\epsilon} n)$ in the approximation algorithm is due to the use of [4]. We further lose an $O(\log n)$ factor due to the use of [9]. 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 $\alpha$ be the rent cost in the optimum. As we assume all numbers are polynomial in $n$, we may try all different $\alpha$ 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 $\sum_{e \text{ rented by } W' \cup U'} f^2 - \sum_{e \text{ rented by } W' \cup U'} f(e) c_e < -\alpha$ 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 $O(\log^{3+\epsilon} n) \cdot \log(\sum_e u_e c_e)$, with an additional $\log n$ penalty by [9] deriving a $O(\log^{3+\epsilon} n)$ for Connected Rent or Buy Source Location.

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 $\alpha$. This implies a $\log^{2+\epsilon} n$ approximation for Connected Rent or Buy Source Location.
5 Approximating Unbalanced-P2P

In [8] it is shown that Unbalanced-P2P is a special case of Cap-SN and that Steiner forest and k-Steiner tree are special cases of the problem. Thus a constant approximation for Unbalanced-P2P will be a uniform constant approximation for both k-Steiner tree and for Steiner forest. A fascinating open problem.

5.1 An exact algorithm for Unbalanced-P2P on trees

See Appendix C.

5.2 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 Unbalanced-P2P with $b(V) = 0$, let $\mathcal{F} = \{S \subseteq V \mid b(S) \neq 0\}$. Then the following holds. An edge-set $H \subseteq E$ is a feasible solution to Unbalanced-P2P if, and only if, $H$ covers $\mathcal{F}$. 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 $b(S) \neq 0$. $\mathcal{F}$ is uncrossable.

**Proof.** Parts (i) and (ii) are straightforward, so we prove only part (iii). Let $X, Y \in \mathcal{F}$, so $b(X), b(Y) \neq 0$. 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 $X \setminus Y, Y \setminus X \in \mathcal{F}$.

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

In Appendix C, we prove that the Unbalanced-P2P problem 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 [9]. There is a one-to-one correspondence between \( V' \) and the set \( L \) of leaves of \( T \). The resulting instance of Unbalanced-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 instance of Unbalanced-P2P to-one correspondence between \( V \) results of Bartal [3] and Fakcharoenphol, Rao and Talwar [9]. There is a one-to-one correspondence between \( V' \) and the set \( L \) of leaves of \( T \). The resulting instance of Unbalanced-P2P on \( T \) inherits the charges on the leaves of \( T \) from 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 given 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 \((V, E')\).

### 5.4 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 Unbalanced-P2P computes an edge set \( E' \subseteq E \) of cost \( \leq 4\tau^* \), where \( \tau^* \) 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 \( \tau \), which is an estimate for \( \tau^* \). Create an instance of Unbalanced-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 \( 4\tau \), and the number of edges incident to \( s \) is at most \( 4\tau/\tau = 4b(V) \). Using binary search, we find the minimum integer \( \tau \) for which the procedure returns an edge-set \( E'' \) of cost \( 4\tau \). 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 \( O(\log(2 + b(V)))\)-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 \( c(E') + c(F) = O(\tau^*) + O(\log n') \cdot \tau^* = O(\log(2 + b(V))) \cdot \tau^* \).
References


A Proof of Theorem 4: a greedy algorithm for k-Bipartite Flow

We analyze the most basic greedy algorithm for the problem. The algorithm maintains sets $A' \subseteq A, B' \subseteq B$ that cannot yet deliver $k$ units of flow. Then it examines all pairs
of vertices \( a \in A \setminus A' \) and \( b \in B \setminus B' \) and adds to \( A', B' \) the pair that increases the flow by the largest amount. In the algorithm, let \( \text{Flow}(A', B') \) be the flow from source set \( A' \) to sink set \( B' \).

**Algorithm Greedy** \((A, B, E, k)\):

1. \( A' \leftarrow \emptyset, B' \leftarrow \emptyset \)
2. While \( \text{Flow}(A', B') < k \), add to \( A' \cup B' \) the pair \((a', b') \in A \setminus A' \times (B \setminus B')\) for which \( \text{Flow}(A' + a, B' + b) - \text{Flow}(A', B') \) is maximum.
3. Return \( A', B' \)

### A.1 Analysis

Let \( U \) be a universe and \( f \) be a non-decreasing function \( f : 2^U \rightarrow R^+ \). Let \( c : 2^U \rightarrow R^+ \) be a subadditive function defined on subsets of \( U \). Let \( \text{Add}_f(S) = \min \{ f(T + S) - f(S), k - f(S) \} \). Let the density of \( T \) with respect to \( S \) be \( \text{Add}_f(S)/c(T) \). Let \( OPT \) be the minimum cost solution and let \( \text{opt} \) be its cost. The optimum density with respect to \( S \) is defined as \( \text{Add}_f(OPT(S))/\text{opt} \). For the proof of the following theorem see [10].

**Theorem 8.** Say that we greedily add sets \( T \) to \( S \) until \( f(S) = f(U) \) so that at every iteration \( \text{Add}_f(S)/c(T) \) is at least \( \text{Add}_f(OPT(S))/(\rho \cdot \text{opt}) \). Then the approximation ratio is at most \( \rho (\ln(f(U)) + 1) \).

Let \( D \subseteq A + B \) be the vertices that generate flow in the optimum. Thus the cost of the solution is \( \text{opt} = |D| \).

**Claim.** We can start with the flow on \( A' \cup B' \), use only pairs of vertices in \( D \) and increase the flow by at least \( \text{Add}_f(OPT(A' \cup B')) \).

**Proof.** Let \( \text{old}_f \) be the old flow between \( A' \cup B' \). Find a path \( P \) in \( OPT \) that does not exist in \( \text{old}_f \). Such a path must exist as long as the flow delivered is less than \( k \). Also, observe that we may analyze different paths separately, since by definition these paths are edge disjoint. Now use \( P \) either to change the flow or to augment the flow by 1. Add a unit flow along \( P \). For edges in \( P \) of flow zero, the flow is changed to 1. For edges of flow 1 the flow remains 1. After adding flow along \( P \) the flow may not be feasible. If two flow units leave some \( v \), reduce to 0 the flow of the edge not in \( P \), and if two flow units enter some \( v \) reduce to 0 the flow of the edge not in \( P \). In addition some flow units may now become irrelevant as they are not delivering flow from \( A \) to \( B \) and are discarded. This operation is not equivalent to finding a path in the residual graph. In fact the flow may decrease due to this operation. Since the total flow between pairs in \( D \) is at least \( k \), at the end we can deliver an extra of \( \text{Add}_f(OPT(A' \cup B')) \) flow units, all between pairs in \( D \).

**Claim.** There exists a pair with density at least \( \text{Add}_f(OPT(A' \cup B'))/\text{opt}^2 \).

**Proof.** Let \( N \) be the set of pairs used by \( OPT \). Note that \( \sum_{(a,b) \in N} \text{Add}_{(a,b)}(A' \cup B') \geq \text{Add}_f(OPT(A' \cup B')) \). We have \(|N| \leq \text{opt}^2 \), thus there is a pair \( a', b' \) that delivers at least \( \text{Add}_f(OPT(A' \cup B'))/\text{opt}^2 \) flow between the two vertices. Its density is at least \( \text{Add}_f(OPT(A' \cup B'))/(2 \cdot \text{opt}^2) \) because the cost of every vertex is 1.

**Claim.** The greedy algorithm has \( O(\sqrt{k \log k}) \) approximation ratio.
Proof. By the density claim with \( \rho = 2opt \) we get a bound of \( 2opt \cdot (\ln(k) + 1) \cdot opt \) on the cost, and hence the approximation ratio of \( O(opt \cdot \sqrt{k}) \). We now consider two cases. If \( opt = |D| \leq \sqrt{k}/\log k \) then the \( O(\sqrt{k}/\log k) \) ratio follows. Otherwise, assume that \( opt > \sqrt{k}/\log k \). Note that the overall cost of our solution is at most \( 2k \), since in each iteration we increase the flow by at least \( 1 \) and increase the cost by \( 2 \). Thus we get an approximation ratio of \( O(k/\sqrt{k}/\log k) = O(\sqrt{k}/\log k) \).

B Proof of Theorem 5

In this section, we sketch the proof of Theorem 5. We first reduce the problem to the tree instances using the theorem of Harrelson, Hildrum, and Rao [18]. Let us state the result of Harrelson, Hildrum, and Rao [18] more formally. A tree decomposition \( \mathcal{T} \) of a graph \( G = (V, E) \) is described by a series of hierarchical partitions of the vertex set \( V \) of \( G \). The vertices of \( \mathcal{T} \) correspond to the subsets of \( V \). Consider a series of partitions \( \Pi_0, \ldots, \Pi_d \) where partition \( \Pi_{i+1} \) is a refinement of partition \( \Pi_i \). The partition \( \Pi_0 \) corresponds to a single set \( V \) while the partition \( \Pi_d \) corresponds to the sets of singletons \( \{v\} \) where \( v \in V \). These partitions give rise to a tree \( \mathcal{T} \) naturally. The root vertex of \( \mathcal{T} \) is \( V \) itself. The vertices in layer \( i \) are the sets in \( \Pi_i \) and the leaves correspond to the sets in \( \Pi_d \), i.e., the vertices in \( V \). The edges of the tree go between the consecutive layers and are given by set inclusion. The weights of the edges of the tree are given as follows. For a setpair \((S, T)\), where \( S \subseteq T, S \in \Pi_{i+1}, \text{ and } T \in \Pi_i \), the weight \( w(S, T) = \omega_C(d(S)) \) is defined to be the weight of the cut \((S, \overline{S})\) in the graph \( G \). Now consider an instance of the multi-commodity flow demands \( M = \{d_{ij} \geq 0 | i, j \in V\} \) between pairs of vertices. Let \( c_G(M) \) (resp., \( c_T(M) \)) denote the minimum maximum edge-congestion under which \( M \) can be routed in \( G \) (resp., \( \mathcal{T} \)). Harrelson et al. [18] proved the following theorem.

Theorem 9 (Harrelson et al. [18]). 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

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

\[ T \]

\textbf{C An exact algorithm for Unbalanced-P2P on trees}

We now focus on the case when the charges \( b_v \) are polynomially bounded, but the total charge \( b(V) \) may not be zero. We show how to solve the problem on trees optimally, using dynamic programming.

Root the tree \( T \) at some vertex \( s \). By adding zero-cost edges to \( T \) if necessary, we can assume that \( T \) is a binary tree without loss of generality. In particular, if a vertex \( v \) has \( p \) children, we add a binary tree with \( p \) leaves at \( v \) and connect \( p \) leaves one-to-one to the \( p \) leaves. We give a cost of zero to each of the tree edges. It is easy to see that the instance essentially remains unchanged by this modification. For a vertex \( v \in T \), let \( T_v \) denote the subtree hanging below \( v \). The dynamic program computes quantities \( T(v, B) \) for all vertices \( v \in T \) and integer \( B \) in the range \( \left[ \sum_{u, b_u < 0} b_u, \sum_{u, b_u > 0} b_u \right] \).

Since each \( b_v \) is polynomially bounded, the number of such quantities is polynomial. The quantity \( T(v, B) \) is defined as the minimum-cost of a subgraph \( H \) of \( T_v \) satisfying the following:

- the connected component in \( H \) containing \( v \) has the total charge \( B \), and
- every other connected component in \( H \) has non-negative total charge.

If there is no subgraph \( H \) satisfying the above conditions, we define \( T(v, B) \) as \(-\infty\). We assume that the minimum-cost subgraph \( H \) is also stored in the dynamic program table.

The quantities \( T(v, B) \) can be computed as follows. For leaf vertices \( v \), it is trivial to compute \( T(v, B) \) and the corresponding optimum subgraphs. For an internal vertex \( v \), we compute \( T(v, B) \) as follows. Let \( u_1 \) and \( u_2 \) be the two children of \( v \). Depending on whether we pick edges \((v, u_1)\) or \((v, u_2)\) in the solution, we get four possibilities.

1. If we pick none of these edges in the solution, we get a solution of cost \( \min \{ T(u_1, B_1) + T(u_2, B_2) \mid B_1, B_2 \geq 0 \} \) corresponding to charge of the connected component containing \( v \) of \( b_v \).
2. If we pick edge \((v, u_1)\) but do not pick edge \((v, u_2)\) in the solution, we get a solution of cost \( \min \{ c(v, u_1) + T(u_1, B_1) + T(u_2, B_2) \mid B_2 \geq 0 \} \) corresponding to charge of the connected component containing \( v \) of \( b_v + B_1 \).
3. If we pick edge \((v, u_2)\) but do not pick edge \((v, u_1)\) in the solution, we get a solution of cost \( \min \{ c(v, u_2) + T(u_2, B_2) + T(u_1, B_1) \mid B_1 \geq 0 \} \) corresponding to charge of the connected component containing \( v \) of \( b_v + B_2 \).
4. If we pick both the edges \((v, u_1)\) and \((v, u_2)\) in the solution, we get a solution of cost \( \min \{ c(v, u_1) + T(u_1, B_1) + c(v, u_2) + T(u_2, B_2) \} \) corresponding to charge of the connected component containing \( v \) of \( b_v + B_1 + B_2 \).

We consider all these possibilities and pick the minimum-cost solution corresponding to each value of the charge of the connected component containing \( v \).

Finally, we output the solution corresponding to \( \min \{ T(s, B) \mid B \geq 0 \} \). It is easy to see that the above dynamic programming based algorithm computes the optimum solution our problem.