2008－AL－117 (8)
2008／3／7

最小費用枝架設問題に対する近似解法

Ehab Morsy, 永持 仁
京都大学大学院情報学研究科数理工学専攻

概要

枝重み $w : E \to R^+$ を持つグラフ $G = (V, E)$, シンク $s \in V$, 枝容量 $\lambda > 0$, ソース集合 $S \subseteq V$, 要求量 $0 \leq q(v) \leq \lambda$ $(v \in S)$ が与えられたとき，最小費用枝架設問題とは，各ソース $v \in S$ からシンク s への経路 P_v を G のパスとして選定し，すべての経路 P_v, $v \in S$ を確保するために必要な枝の架設費用を最小にする問題である．ここで，1本の枝 e を通過することのできる経路の要求量の和の上限が枝容量を λ を超えることはできないが，必要であれば枝 e の複製を何本でも架設することができる（ただし，要求量 $q(v)$ を複数の枝の複製に分離して流すことは許されない）．枝 e を1本架設する費用は $w(e)$ として与えられており，経路集合 $\mathcal{P} = \{P_v \mid v \in S\}$ の架設コストは枝 $e \in E$ を h 本架設する場合，$cost(\mathcal{P}) = \sum_{e \in E} h(e)w(e)$. として表される．本論文では，最小費用枝架設問題に対する $(15/8 + \rho_{ST})$-近似アルゴリズムを与える．ただし，$\rho_S T$ はスタイナー木問題に対する近似比である．

Approximation to the Minimum Cost Edge Installation Problem

Ehab Morsy, Hiroshi Nagamochi
Department of Applied Mathematics and Physics,
Graduate School of Informatics, Kyoto University

abstract

We consider the *minimum cost edge installation problem* (MCEI) in a graph $G = (V, E)$ with edge weight $w(e) \geq 0$, $e \in E$. We are given a vertex $s \in V$ designated as a sink, an edge capacity $\lambda > 0$, and a source set $S \subseteq V$ with demand $0 \leq q(v) \leq \lambda$, $v \in S$. For any edge $e \in E$, we are allowed to install an integer number $h(e)$ of copies of e. The MCEI asks to send demand $q(v)$ from each source $v \in S$ along a single path P_v to the sink s. A set of such paths can pass through a single copy of an edge in G as long as the total demand along the paths does not exceed the edge capacity λ (splitting the demand $q(v)$ of a source into two or more copied of an edge e is not allowed). The objective is to find a set $\mathcal{P} = \{P_v \mid v \in S\}$ of paths of G that minimizes the installing cost $cost(\mathcal{P}) = \sum_{e \in E} h(e)w(e)$. In this paper, we propose a $(15/8 + \rho_{ST})$-approximation algorithm to the MCEI, where ρ_{ST} is any approximation ratio achievable for the *Steiner tree problem*.

1 Introduction

We study a problem of finding routings from a set of sources to a single sink in a network with an edge installing cost. This problem is a fundamental and economically significant one that arises in hierarchical design of telecommunication networks [2] and transportation networks [8, 9]. In telecommunication networks this corresponds to installing transmission facilities such as fiber-optic cables, which represent the edges of the network. In other applications, optical cables may be replaced by pipes, trucks, and so on.

Consider an edge-weighted undirected graph G, where $V(G)$ and $E(G)$ denote the vertex set and edge set of G, respectively. We are given a set $S \subseteq V(G)$ of vertices specified as sources and a vertex $s \in V(G)$ specified as a sink. Each source $v \in S$ has a nonnegative demand $q(v)$, all of which must be routed to s through a single path. We are also given a finite set of different cable types, where each cable type is specified by its capacity and its cost per unit weight. The costs of cables obey economies of scale, i.e., the cost per unit capacity per unit weight of a high capacity cable is significantly less than that of a low capacity cable. The *single-sink buy-at-bulk problem* (SSBB) (also known as the *single-sink edge installation problem* [3]) asks to construct a network of cables in the graph by installing an integer number of each cable type between adjacent vertices in G so that given demands at the sources can be routed simultaneously to s. The goal is to minimize the costs of installed cables.

The problem of buy-at-bulk network design was first introduced by Salman et al. [8]. They showed that the problem is NP-hard by showing a reduction from the *Steiner tree problem*. The Steiner tree problem is a classical NP-hard optimization problem, and the current best approximation ratio for the Steiner tree problem is a bit less than 1.55 [7]. Moreover, they showed that the problem remains NP-hard even when only one cable type is available. The approximation ratio for the SSBB problem was gradually reduced from $O(\log^2 n)$ [1] to 24.92 [2] by a series of papers, where n is the number of vertices of the underlying graph.

In this paper, we study a special case of the SSBB that arises from transportation networks [9]. A multinational corporation wishes to enter a new geographic area, characterized by demand at each city. It has identified the location of its manufacturing facility. Suppose the shipping of the good will be carried out by some transport company. This transport company has only one truck type, with a fixed capacity. For each truck, the transport company charges at a fixed rate per mile, and offers no discount in the case where the truck is not utilized to full capacity. The problem facing the corporation is to decide a shipping plan of the finished good to each city, so that the total demand at each city is met and the total cost is minimized.

In such a transportation network, we have a single cable type with a fixed capacity $\lambda > 0$ for all edges, and we are interested in constructing a set $\mathcal{P}$ of paths each of which connects one of given sources to a single sink s. The cost of installing a copy of an edge e is represented by the weight of e. A subset of paths of $\mathcal{P}$ can pass through a single copy of an edge e as long as the total demand of these paths does not exceed the edge capacity λ; any integer number of separated copies of e are allowed to be installed. However, the demand of each source is not allowed to be split at any vertex or over two or more copies of the same edge. The cost of a set $\mathcal{P}$ of paths is defined by the minimum cost of installing copies of edges such that the demand of each source can be routed to the sink under the edge capacity constraint, i.e.,

$$cost(\mathcal{P}) = \sum_{e \in E(G)} h(e)w(e),$$

where $h(e)$ is the minimum number of copies of e required for routing the set of all demands along e, simultaneously. The goal is to find a set $\mathcal{P}$ of paths that minimizes $cost(\mathcal{P})$. We call this problem, the *minimum cost edge installation problem* (MCEI). Notice that, in order to get a feasible solution to the MCEI, such edge capacity λ should be as much as the maximum demand in the network. The MCEI can be formally defined as follows, where R^+ denotes the set of nonnegative reals.

Minimum Cost Edge Installation Problem (MCEI):

Input: A connected graph G, an edge weight function $w : E(G) \to R^+$, a sink $s \in V(G)$, a set $S \subseteq V(G)$ of sources, an edge capacity $\lambda > 0$, and a demand function $q : S \to R^+$ such that $q(v) \leq \lambda$, $v \in S$.

Feasible solution: A set $\mathcal{P} = \{P_v \mid v \in S, \{s, v\} \subseteq V(P_v)\}$ of paths of G.

Goal: Find a feasible solution $\mathcal{P}$ that minimizes $cost(\mathcal{P})$.

The MCEI is closely related to the *capacitated network design problem* (CND), which can be stated as follows. We are given an undi-

rected graph G such that each edge $e \in E(G)$ is weighted by nonnegative real $w(e)$, a subset $S \subseteq V(G)$ of sources, and a vertex $s \in V(G)$ designated as a sink. Each source $v \in S$ has a nonnegative demand $q(v)$, all of which must be routed to s through a single path. A cable with fixed capacity λ is available for installing on the edges of the graph, where installing i copies of the cable on edge e costs $iw(e)$ and provides $i\lambda$ capacity. The CND asks to find a minimum cost installation of cables that provides sufficient capacity to route all of the demand simultaneously to s. The problem requires choosing a path from each source to the sink and finding the number of cables to be installed on each edge such that all the demand is routed without exceeding edge capacities. Demands of different sources may share the capacity on the installed cables and the capacity installed on an edge has to be at least as much as the total demand routed through this edge. For this problem, Mansour and Peleg [6] gave an $O(\log n)$-approximation algorithm for a graph with n vertices. Salman et al. [8] designed a 7-approximation algorithm for the CND based on a construction from [5]. Afterwards Hassin et al. [4] gave a $(2 + \rho_{\mathrm{ST}})$-approximation algorithm, where ρ_{ST} is any approximation ratio achievable for the Steiner tree problem. By using of a slight intricate version of this algorithm, they improved the approximation ratio to $(1 + \rho_{\mathrm{ST}})$ when every source has unit demand. When all non-sink vertices are sources, the approximation ratio of Hassin et al. [4] becomes 3 for general demands and 2 for unit demands, since the Steiner tree problem in this case is a minimum spanning tree problem.

Note that, a solution to each of the MCEI and the CND can be characterized by specifying for each source v, the path P_v along which the demand $q(v)$ of v will be sent to the sink. The cables installed on each edge of the network are induced by these paths. In particular, for each edge e, a feasible solution to the MCEI assigns an integer number of separated cable copies required for routing all demands in $\{q(v) \mid e \in E(P_v)\}$, simultaneously. On the other hand, a feasible solution to the CND assigns on e at least $\lceil \sum_{v:e\in E(P_v)} q(v)/\lambda \rceil$ copies of the cable. That is, on contrary to the MCEI, the CND allows the demand from a source to be split among different copies of the same edge. Note that, the algorithm of Hassin et al. [4] to the CND takes the advantage (over the MCEI) of this assumption only for routing demands larger than λ to the sink. Hence, their algorithms can be used to obtain approximate solutions to the MCEI with approximation ratios $1 + \rho_{\mathrm{ST}}$ and $2 + \rho_{\mathrm{ST}}$ for the unit and general demand networks, respectively. In this paper, we proved that there is a $(15/8 + \rho_{\mathrm{ST}})$-approximation algorithm to the MCEI with general demands. Our result is based on a new and elaborated method for partitioning the source set of a given tree. When $S = V(G)$, the approximation ratio becomes 2.875.

The rest of this paper is organized as follows. Section 2 introduces terminologies on graphs and two lower bounds on the optimal value of the MCEI. Section 3 describes some results on tree partitions. Section 4 gives a framework of our approximation algorithm for the MCEI, analyzing its approximation ratio. Section 5 makes some concluding remarks.

2 Preliminaries

This section introduces some notations and definitions. Let G be a simple undirected graph. We denote by $V(G)$ and $E(G)$ the sets of vertices and edges in G, respectively. An edge-weighted graph is a pair (G, w) of a graph G and a nonnegative weight function $w : E(G) \to R^+$. The length of a shortest path between two vertices u and v in (G, w) is denoted by $d_{(G,w)}(u, v)$. Given a vertex weight function $q : V(G) \to R^+$ in G, we denote by $q(Z)$ the sum $\sum_{v\in Z} q(v)$ of weights of all vertices in a subset $Z \subseteq V(G)$.

Let T be a tree. A *subtree* of T is a connected subgraph of T. For a subset $X \subseteq V(T)$ of vertices, let $T\langle X\rangle$ denote the minimal subtree of T that contains X (note that $T\langle X\rangle$ is uniquely determined). Now let T be a rooted tree. We denote by $L(T)$ the set of leaves in T. For a vertex v in T, let $Ch(v)$ and $D(v)$ denote the sets of children and descendants of v, respectively, where $D(v)$ includes v. A *subtree* T_v *rooted* at a vertex v is the subtree induced

by $D(v)$, i.e., $T_v = T\langle D(v)\rangle$. For an edge $e = (u, v)$ in a rooted tree T, where $u \in Ch(v)$, the subtree induced by $\{v\} \cup D(u)$ is denoted by T_e, and is called a *branch* of T_v. For a rooted tree T_v, the *depth* of a vertex u in T_v is the length (the number of edges) of the path from v to u.

For a set Z, a set $\{Z_1, Z_2, ..., Z_\ell\}$ of pairwise disjoint subsets of Z is called a *partition* of Z if $\cup_{i=1}^{\ell} Z_i = Z$.

The rest of this section presents two lower bounds on the optimal value to the MCEI. The first lower bound has been proved and used to derive approximation algorithms to the CND in [4].

Lemma 1 *For an instance $I = (G = (V, E), w, S, q, s, \lambda)$ of the MCEI, let $opt(I)$ be the weight of an optimal solution to I, and T^* be the minimum weight of a tree that spans $S \cup \{s\}$ in G. Then*

$$\max\left\{w(T^*), \frac{1}{\lambda}\sum_{t\in S} q(t)d_{(G,w)}(s,t)\right\} \leq opt(I),$$

where $w(T^*)$ is the sum of weights of edges in T^*. □

The second lower bound is derived from an observation on the distance from sources $t \in S$ with $q(t) > \lambda/2$ to sink s.

Lemma 2 *For an instance $I = (G = (V, E), w, S, q, s, \lambda)$ of the MCEI with $q(t) \in [0, \lambda]$, $t \in S$, let $opt(I)$ be the weight of an optimal solution to I. Then*

$$\sum_{t\in S'} d_{(G,w)}(s,t) \leq opt(I),$$

where $S' = \{t \in S \mid q(t) > \lambda/2\}$.

Proof. The proof is followed directly by noting that for any two sources $u, v \in S'$, the paths P_v and P_u of the optimal solution cannot share the capacity of a single copy of any edge $e \in E$. □

Given an instance $I = (G = (V, E), w, S, q, s, \lambda)$ of the MCEI, our algorithm firstly produces a tree T of G that spans all vertices in $S \cup \{s\}$, finds a partition $\mathcal{S}$ of S, and assigns a vertex $t_Z \in Z$ for each subset $Z \in \mathcal{S}$ such that when all demands in each subset $Z \in \mathcal{S}$ are routed to t_Z simultaneously, the total flow on each edge of T is at most λ, where we call such a vertex t_Z the *hub vertex* of Z. Afterward, for each $Z \in \mathcal{S}$, we install a copy of each edge in a shortest path $SP(s, t_Z)$ between s and t_Z in G, and construct path P_t, $t \in Z$, by adding $SP(s, t_Z)$ to the path between t and t_Z in T. The running time of this algorithm is dominated by the approximation algorithm for the Steiner tree problem to compute tree T.

3 Tree partition

The purpose of this section is to describe how to construct a tree partition in a tree that spans a source set, that is, how to find a partition of the source set of the tree. Such a tree partition will be the basis of our approximation algorithm to the MCEI in the next section. We first present some results for special cases of tree partitioning.

3.1 Tree partition in special trees

In this subsection, we prepare several lemmas on tree partition problem for a tree with special structure. We first introduce a subgraph which plays a key role in our algorithm.

Definition 1 *For a vertex v in a rooted tree, a source set $Z_v \subseteq V(T_v) - \{v\}$, a demand function $q : Z_v \to R^+$, and a positive number λ, a binary rooted tree T_v is said to be a* balance-tree *if $q(Z_v) > \lambda$ holds and the total demand in each of its branches is less than $(4/7)\lambda$.*

We are given a binary rooted tree T_x with a source set $Z_x = L(T_x)$, an edge capacity $\lambda > 0$, a demand function $q : Z_x \to R^+$ such that $q(t) \leq \lambda/2$ for all $t \in Z_x$, and a vertex weight function $d : Z_x \to R^+$. Moreover, for each $u \in Ch(x)$, if $q(V(T_u) \cap Z_x) \geq (4/7)\lambda$, then T_u contains a balance-tree and satisfies $q(V(T_u) \cap Z_x) < (8/7)\lambda$. We partition Z_x into subsets, and choose a hub vertex from each subset such that, when demands of each subset are routed to its hub vertex simultaneously, the total flow on each edge of T_x is bounded from above by λ. For brevity, we use $(T_x, Z_x, q, d, \lambda)$ to refer to this tree throughout this subsection.

We give the following three lemmas without proofs due to space limitation.

Lemma 3 *Given a tree $(T_x, Z_x, q, d, \lambda)$ with $(8/7)\lambda \leq q(Z_x) < (12/7)\lambda$, there is a partition $\{X, Y\}$ of Z_x such that $q(Y) \geq (4/7)\lambda$, and when $q(X)$ and $q(Y)$ are routed to $t_X = \mathrm{argmin}\{d(t) \mid t \in Z_x\}$ and $t_Y = \mathrm{argmin}\{d(t) \mid t \in Y\}$, respectively, the total amount of these flow on each edge of T_x is at most λ.* □

Lemma 4 *Given a tree $(T_x, Z_x, q, d, \lambda)$ with $q(Z_x) \geq (12/7)\lambda$, there is a partition $\{A, B, C\}$ of Z_x and a subset $Z'_x \subseteq Z_x$ with $q(Z'_x) \geq (12/7)\lambda$ such that $q(A \cap Z'_x), q(B \cap Z'_x) > (3/7)\lambda$, $q(C \cap Z'_x) \geq (5/7)\lambda$, and when $q(A)$, $q(B)$, and $q(C)$ are routed to $t_A = \mathrm{argmin}\{d(t) \mid t \in Z'_x\}$, $t_B = \mathrm{argmin}\{d(t) \mid t \in Z'_x - A\}$, and $t_C = \mathrm{argmin}\{d(t) \mid t \in C \cap Z'_x\}$, respectively, the total amount of these flow on each edge of T_x is at most λ.* □

Lemma 5 *Given a tree $(T_x, Z_x, q, d, \lambda)$ with $Z_x \neq \emptyset$, there is a partition $\mathcal{Z}_1 \cup \mathcal{Z}_2$ of Z_x such that $q(Z) \geq (4/7)\lambda$ for each $Z \in \mathcal{Z}_1$, $q(Z) < (4/7)\lambda$ for each $Z \in \mathcal{Z}_2$, and when demands in each $Z \in \mathcal{Z}_1$ and $Z \in \mathcal{Z}_2$ are routed to $t_Z = \mathrm{argmin}\{d(t) \mid t \in Z\}$ and x, respectively, the total amount of these flow on each edge of T_x is at most λ.* □

3.2 Algorithm for tree partition

In this subsection, we present an algorithm that exploits the results in Lemmas 3-5 to compute a partition of the source set of a general tree given in the next theorem.

Theorem 1 *Given a tree T rooted at s, an edge capacity $\lambda > 0$, a source set $S \subseteq V(T)$, a demand function $q : S \to R^+$ such that $q(t) \leq \lambda/2$, $t \in S$, and a vertex weight function $d : S \to R^+$, there is a partition $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2 \cup \mathcal{S}_3 \cup \mathcal{S}_4$ of S, where $\mathcal{S}_3 = \cup_{1 \leq i \leq k}\{X_i, Y_i\}$ and $\mathcal{S}_4 = \cup_{1 \leq i \leq \ell}\{A_i, B_i, C_i\}$, and a set $\mathcal{H} = \{t_Z \in S \mid Z \in \mathcal{S}\}$ of hub vertices, that satisfy:*

(i) *For each subset $Z \in \mathcal{S}_1$, $q(Z) < (4/7)\lambda$ and $t_Z = s$.*

(ii) *For each subset $Z \in \mathcal{S}_2$, $q(Z) \geq (4/7)\lambda$ and $t_Z = \mathrm{argmin}\{d(t) \mid t \in Z\}$.*

(iii) *For $i = 1, 2, \ldots, k$, $q(Y_i) \geq (4/7)\lambda$, $q(X_i \cup Y_i) \geq (8/7)\lambda$, $t_{X_i} = \mathrm{argmin}\{d(t) \mid t \in X_i \cup Y_i\}$, and $t_{Y_i} = \mathrm{argmin}\{d(t) \mid t \in Y_i\}$.*

(iv) *For $i = 1, 2, \ldots, \ell$, $q(A_i \cap Z'_x), q(B_i \cap Z'_x) > (3/7)\lambda$, and $q(C_i \cap Z'_x) \geq (5/7)\lambda$, where $Z'_x \subseteq A_i \cup B_i \cup C_i$ with $q(Z'_x) \geq (12/7)\lambda$, and $t_{A_i} = \mathrm{argmin}\{d(t) \mid t \in Z'_x\}$, $t_{B_i} = \mathrm{argmin}\{d(t) \mid t \in Z'_x - A_i\}$, and $t_{C_i} = \mathrm{argmin}\{d(t) \mid t \in C_i \cap Z'_x\}$.*

(v) *When the total demand of each subset $Z \in \mathcal{S}$ is routed to t_Z simultaneously, the total amount of these flow on each edge of T is bounded from above by λ.*

Furthermore, such a partition $\mathcal{S}$ can be computed in polynomial time. □

To prove Theorem 1, we can assume without loss of generality that in a given tree T, (i) all sources are leaves, i.e., $S = L(T)$, by introducing a new edge of weight zero for each non-leaf source, and (ii) $|Ch(v)| = 2$ holds for every non-leaf $v \in V(T)$, i.e., T is a binary tree rooted at s, by splitting vertices of degree more than 3 with new edges of zero weights.

We prove Theorem 1 by showing that the next algorithm actually delivers a desired partition $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2 \cup \mathcal{S}_3 \cup \mathcal{S}_4$. We first choose a vertex $v \notin Q \cup \{s\}$ with the maximum depth in the current tree such that the total demand of a source set Z_v of the tree rooted at v is at least $(4/7)\lambda$, where Q is initialized to be empty and is used to keep track of vertices v in the current tree such that T_v contains a balance-tree and satisfies $q(Z_v) < (8/7)\lambda$. Depending on the total demand of Z_v, we add Z_v to $\mathcal{S}_2$, add v to Q, or compute a partition of Z_v by using Lemma 3 or 4. In the latter case, we add the subsets of the obtained partition to one of $\mathcal{S}_3$ or $\mathcal{S}_4$. We then remove all sources in $\mathcal{S}_2 \cup \mathcal{S}_3 \cup \mathcal{S}_4$ from S and repeat these steps on the minimal subtree of T that spans s and the current source set until there is no such vertex v. Finally, we partition the remaining set of sources by using Lemma 5 and add the resulting partition to one of $\mathcal{S}_1$ or $\mathcal{S}_2$. A formal description of the algorithm is the following.

Algorithm TREEPARTITION

Input: A binary tree $\widehat{T}$ rooted at s, a capacity λ of each edge, a set $S = L(\widehat{T})$ of sources, a demand function $q : S \to R^+$ such that $q(t) \leq \lambda/2$, $t \in S$, and a vertex weight function $d : S \to R^+$.
Output: A pair $(\mathcal{S}, \mathcal{H})$ that satisfies the conditions in Theorem 1.
Initialize $T := \widehat{T}$; $Q := \mathcal{H} := \mathcal{S}_1 := \mathcal{S}_2 := \mathcal{S}_3 := \mathcal{S}_4 := \emptyset$.
1 **while** there exists a vertex $v \in V(T) - \{s\} - Q$ such that $q(V(T_v) \cap S) \geq (4/7)\lambda$ **do**
2 Choose such v with the maximum depth from s;
3 Let $Z_v := D_T(v) \cap S$; $T_v := T\langle Z_v \rangle$;
4 **begin** /* Distinguish four cases. */
5 **Case-1** $q(Z_v) \leq \lambda$: Let $\mathcal{S}_2 := \mathcal{S}_2 \cup \{Z_v\}$;
6 $t_{Z_v} = \text{argmin}\{d(t) \mid t \in Z_v\}$; $\mathcal{H} := \mathcal{H} \cup \{t_{Z_v}\}$;
7 **Case-2** $\lambda < q(Z_v) < (8/7)\lambda$: Let $Q := Q \cup \{v\}$;
8 **Case-3** $(8/7)\lambda \leq q(Z_v) < (12/7)\lambda$:
9 Apply Lemma 3 to $(T_v, Z_v, q, d, \lambda)$ to get a partition $\{X, Y\}$ of Z_v and vertices t_X and t_Y that satisfy the conditions in the lemma;
10 $\mathcal{S}_3 := \mathcal{S}_3 \cup \{X, Y\}$; $\mathcal{H} := \mathcal{H} \cup \{t_X, t_Y\}$
11 **Case-4** $(12/7)\lambda \leq q(Z_v) < (16/7)\lambda$:
12 Apply Lemma 4 to $(T_v, Z_v, q, d, \lambda)$ to get a partition $\{A, B, C\}$ of Z_v and vertices t_A, t_B, and t_C that satisfy the conditions in the lemma;
13 $\mathcal{S}_4 := \mathcal{S}_4 \cup \{A, B, C\}$ and $\mathcal{H} := \mathcal{H} \cup \{t_A, t_B, t_C\}$
14 **end**; /* Cases-1,2,3,4 */
15 Let $S := S - (\mathcal{S}_2 \cup \mathcal{S}_3 \cup \mathcal{S}_4)$; $T := T\langle S \cup \{s\}\rangle$
16 **endwhile**;
17 **if** $S \neq \emptyset$
18 Regard T as a tree T_s rooted at s and apply Lemma 5 to (T_s, S, q, d, λ) to get a partition $\mathcal{Z}_1 \cup \mathcal{Z}_2$ of S and a vertex t_Z for each $Z \in \mathcal{Z}_1 \cup \mathcal{Z}_2$ that satisfy the conditions in the lemma;
19 $\mathcal{S}_1 := \mathcal{Z}_2$; $\mathcal{S}_2 := \mathcal{S}_2 \cup \mathcal{Z}_1$; $\mathcal{H} := \mathcal{H} \cup \{t_Z \mid Z \in \mathcal{Z}_1 \cup \mathcal{Z}_2\}$
20 **endif**.

Proof of Theorem 1. We first prove by induction the correctness of algorithm TREEPARTITION. We first consider the vertex v chosen in the first iteration of the while-loop. By the choice of v, $q(V(T_u) \cap S) < (4/7)\lambda$ for all $u \in Ch(v)$. Hence $(4/7)\lambda \leq q(Z_v) < (8/7)\lambda$ holds, which implies that $q(Z_v) \leq \lambda$ or $\lambda < q(Z_v) < (8/7)\lambda$ can occur in the first iteration. If $q(Z_v) \leq \lambda$ holds, then Z_v is removed from S and added to $\mathcal{S}_2$. Otherwise $\lambda < q(Z_v) < (8/7)\lambda$ holds and hence T_v is a balance-tree. In the latter case, v is added to a set Q.

Assume that the algorithm works correctly after the execution of the jth iteration, and let T be the current tree. We show the correctness of the algorithm during the execution of the $(j+1)$th iteration. Note that, for any vertex v chosen by the algorithm, Z_v will be removed from the current S except for the case where $\lambda < q(Z_v) < (8/7)\lambda$. Now let v be a vertex selected in the $(j+1)$st iteration. Then we see that, for each $u \in Ch(v)$, either (i) $q(V(T_u) \cap S) < (4/7)\lambda$ holds (if u has not been chosen before by the algorithm) or (ii) $u \in Q$ holds and T_u contains a balance-tree and satisfies $q(V(T_u) \cap S) < (8/7)\lambda$ (otherwise). Therefore, one of $(4/7)\lambda \leq q(Z_v) \leq \lambda$, $\lambda < q(Z_v) < (8/7)\lambda$, $(8/7)\lambda \leq q(Z_v) < (12/7)\lambda$, and $(12/7)\lambda \leq q(Z_v) < (16/7)\lambda$ holds. Let B_v^1 and B_v^2 denote the two branches of T_v, and let Z_v^i denote the set of sources in B_v^i, $i = 1, 2$, where $q(Z_v^1) \geq q(Z_v^2)$. Now if $q(Z_v) \leq \lambda$ holds, then Z_v is removed from the current S after it is added to $\mathcal{S}_2$. If $\lambda < q(Z_v) < (8/7)\lambda$ holds, then T_v is a balance-tree (if $q(Z_v^1), q(Z_v^2) < (4/7)\lambda$) or B_v^1 (consequently T_v) contains a balance-tree (by $q(Z_v^1) \geq q(Z_v^2)$). In this case, v is added to a set Q. Finally, if $(8/7)\lambda \leq q(Z_v) < (12/7)\lambda$ (resp., $(12/7)\lambda \leq q(Z_v) < (16/7)\lambda$) holds then T_v satisfies conditions of Lemma 3 (resp., Lemma 4) in this case. In the latter two cases, Z_v is removed from the current S after elements of its partition are added to appropriate subsets of $\mathcal{S}$. Therefore, the algorithm works correctly during the execution of all iterations of the while-loop.

After the final iteration, there is no vertex $v \in V(T) - \{s\} - Q$ such that $q(V(T_v) \cap S) \geq (4/7)\lambda$ for the current tree T. If the current $S \neq \emptyset$, then for each $u \in Ch(s)$, either (i) $q(V(T_u) \cap S) < (4/7)\lambda$ holds (if u has not been chosen before by the algorithm) or (ii) $u \in Q$ holds and T_u contains a balance-tree and satisfies $q(V(T_u) \cap S) < (8/7)\lambda$ (otherwise). That is, the current tree T satisfies the conditions in Lemma 5 and a desired partition of the current S can be constructed.

Now we prove that the partition obtained from algorithm TREEPARTITION satisfies conditions (i)-(v) in Theorem 1. Conditions (i)-(iv) follow immediately from construction of $\mathcal{S}_1$, $\mathcal{S}_2$, $\mathcal{S}_3$, and $\mathcal{S}_4$. Now we show (v). Let v be the vertex chosen in line 2 of an arbitrary iteration of the algorithm, where the subtree T_v of the current tree T is being processed in this iteration. Now, if Case-2 holds, then we just add v to Q and then move to the next iteration (the current S and T remain unchanged in this iteration). Otherwise (Case-1, 3, or 4 holds), the algorithm partitions the set Z_v of all sources of

T_v into subsets and chooses a hub vertex from each of these subsets. We then remove Z_v from the current source set S, that is, none of the vertices of T_v will become a hub vertex in the subsequent iterations of the algorithm. Thus it is sufficient to show that, overall iterations of the algorithm, when the demand of each source in Z_v is routed to its hub vertex simultaneously, the total flow on each edge of T_v is bounded from above by λ. Hence (v) follows from the conditions of Lemmas 3, 4, and 5. This completes the correctness of TREEPARTITION and the proof of Theorem 1. □

4 Approximation Algorithm to MCEI

This section describes a framework of our approximation algorithm for the MCEI and then analyzes its approximation ratio. The algorithm relies on the results on tree partition we provided in Section 3.

The basic idea of the algorithm is to first produce a tree T of minimum cost including all vertices in $S \cup \{s\}$. For each source $t \in S$ with $q(t) > \lambda/2$, we install a copy of each edge in a shortest path $SP(s,t)$ between s and t in (G,w), and let $P_t := SP(s,t)$. We then find a partition $\mathcal{S}$ of the remaining sources in S, and assign a hub vertex t_Z for each subset $Z \in \mathcal{S}$, such that when the total demand of each subset is routed to its hub vertex simultaneously, the amount of these flow on each edge of T is at most λ. Finally, for each set $Z \in \mathcal{S}$, we install a copy of each edge in a shortest path $SP(s,t_Z)$ between s and t_Z in (G,w), and construct a path P_t from the path between t and t_Z in T by adding $SP(s,t_Z)$ for all $t \in Z$.

Algorithm APPROXMCEI
Input: An instance $I = (G = (V,E), w, S, q, s, \lambda)$ of the MCEI.
Output: A solution $\mathcal{P}$ to I.

Step 1. Compute a Steiner tree T that spans $S \cup \{s\}$ in G.
Regard T as a tree rooted at s, and define $d : S \to R^+$ by setting

$$d(t) := d_{(G,w)}(s,t), \quad t \in S.$$

Step 2. Let $S' = \{t \in S \mid q(t) > \lambda/2\}$.
For each $t \in S'$, choose a shortest path $SP(s,t)$ between s and t in (G,w), join t to s by installing a copy of each edge in $SP(s,t)$, and let $P_t := SP(s,t)$.

Step 3. Apply Theorem 1 to $(T, S - S', q, s, d, \lambda)$ to obtain a partition

$$\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2 \cup \mathcal{S}_3 \cup \mathcal{S}_4$$

of $S - S'$, where $\mathcal{S}_3 = \cup_{1 \le i \le k}\{X_i, Y_i\}$ and $\mathcal{S}_4 = \cup_{1 \le i \le \ell}\{A_i, B_i, C_i\}$, and a set $\mathcal{H} = \{t_Z \in S \mid Z \in \mathcal{S}\}$ of hub vertices, that satisfy conditions (i)-(v) of the theorem.

Step 4. For each $t \in Z \in \mathcal{S}_1$, let P_t be the path between t and s in T.
For each $Z \in \mathcal{S} - \mathcal{S}_1$,
Choose a shortest path $SP(s,t_Z)$ between s and t_Z in (G,w) and join t_Z to s by installing a copy of each edge in $SP(s,t_Z)$.
For each $t \in Z$, let P_t be the path obtained from the path between t and t_Z in T by adding $SP(s,t_Z)$.

Step 5. Output $\mathcal{P} = \{P_t \mid t \in S\}$. □

Before analyzing the worst case performance of this algorithm, we show the following lemma. The proof is omitted due to space limitation.

Lemma 6 *Let $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2 \cup \mathcal{S}_3 \cup \mathcal{S}_4$ be a partition from a tree $\widehat{T}$ by algorithm* TREEPARTITION *and let $\mathcal{H} = \{t_Z \in S \mid Z \in \mathcal{S}\}$ be the associated set of hub vertices. Then, we have*

$$\sum_{t \in Z \in \mathcal{S}_2 \cup \mathcal{S}_3 \cup \mathcal{S}_4} q(t)d(t) \ge (4/7)\lambda \sum_{Z \in \mathcal{S}_2 \cup \mathcal{S}_3 \cup \mathcal{S}_4} d(t_Z).$$

□

We now turn to proving that the solution output from algorithm APPROXMCEI is within a factor of $(15/8 + \rho_{\mathrm{ST}})$ of the optimal solution.

Theorem 2 *For an instance $I = (G = (V,E), w, S, q, s, \lambda)$ of the* MCEI, *algorithm* APPROXMCEI *delivers a $(15/8 + \rho_{\mathrm{ST}})$-approximate solution $\mathcal{P}$, where ρ_{ST} is the performance ratio for approximating Steiner tree problem.*

Proof. Let $opt(I)$ denote the weight of an optimal solution. By the construction, the cost of $\mathcal{P}$ is bounded by

$$cost(\mathcal{P}) \le w(T) + \sum_{t \in S'} d(t) + \sum_{Z \in \mathcal{S}_2 \cup \mathcal{S}_3 \cup \mathcal{S}_4} d(t_Z).$$

For a minimum Steiner tree T^* that spans $S \cup \{s\}$, we have $w(T) \le \rho_{\mathrm{ST}} w(T^*)$ and $w(T^*) \le opt(I)$ by Lemma 1. Hence $w(T) \le \rho_{\mathrm{ST}} \cdot opt(I)$ holds. To prove the theorem, it suffices to show that

$$\sum_{t \in S'} d(t) + \sum_{Z \in \mathcal{S}_2 \cup \mathcal{S}_3 \cup \mathcal{S}_4} d(t_Z) \le (15/8) opt(I). \quad (1)$$

To prove this inequality, we distinguish two different cases. In the first case, $\sum_{t \in S'} q(t)d(t) \ge$

$\sum_{t\in S-S'} q(t)d(t)$. By Lemma 1, this implies that

$$\begin{aligned} opt(I) &\geq (1/\lambda)\sum_{t\in S} q(t)d(t) \\ &= (1/\lambda)(\sum_{t\in S'} q(t)d(t) + \sum_{t\in S-S'} q(t)d(t)) \\ &\geq (2/\lambda)\sum_{t\in S-S'} q(t)d(t) \\ &\geq (2/\lambda)\sum_{t\in Z\in\mathcal{S}_2\cup\mathcal{S}_3\cup\mathcal{S}_4} q(t)d(t) \\ &\geq (8/7)\sum_{Z\in\mathcal{S}_2\cup\mathcal{S}_3\cup\mathcal{S}_4} d(t_Z), \qquad (2) \end{aligned}$$

where the last inequality follows from Lemma 6. Inequality (2) and Lemma 2 prove (1) in this case.

In the second case, $\sum_{t\in S'} q(t)d(t) < \sum_{t\in S-S'} q(t)d(t)$. Then it is easy to see that there exist two real numbers $0 \leq \alpha, \beta \leq 1$ such that $\alpha+\beta = 1$, $\alpha < \beta$, $(1/\lambda)\sum_{t\in S'} q(t)d(t) \leq \alpha opt(I)$, and $(1/\lambda)\sum_{t\in S-S'} q(t)d(t) \leq \beta opt(I)$. Since $q(t) > \lambda/2$ for all $t \in S'$, we have

$$(1/2)\sum_{t\in S'} d(t) < (1/\lambda)\sum_{t\in S'} q(t)d(t) \leq \alpha opt(I). \qquad (3)$$

On the other hand, Lemma 6 implies that

$$\begin{aligned} &(4/7)\sum_{Z\in\mathcal{S}_2\cup\mathcal{S}_3\cup\mathcal{S}_4} d(t_Z) \\ &\leq (1/\lambda)\sum_{t\in Z\in\mathcal{S}_2\cup\mathcal{S}_3\cup\mathcal{S}_4} q(t)d(t) \leq \beta opt(I). \qquad (4) \end{aligned}$$

By multiplying (3) and (4) by 2 and 7/4, respectively, and adding the obtained inequalities, we have

$$\begin{aligned} \sum_{t\in S'} d(t) + \sum_{Z\in\mathcal{S}_2\cup\mathcal{S}_3\cup\mathcal{S}_4} d(t_Z) &\leq (2\alpha + (7/4)\beta)opt(I) \\ &< (15/8)opt(I), \end{aligned}$$

by the assumptions on α and β. □

5 Concluding remarks

In this paper, we have studied the minimum cost edge installation problem (MCEI), a problem of finding a routing from a set of sources to a single sink in a network with an edge installing cost. The MCEI is closely related to the capacitated network design problem (CND). In particular, a solution to each of the MCEI and the CND can be characterized by a set of paths, each of which sends the demand of a source to the sink and the set of these paths induces the numbers of cables installed on each edge of the network. The CND allows the demand from a source to be split among different copies of the same edge, while the MCEI does not allow such splitting. We have designed a $(15/8 + \rho_{ST})$-approximation algorithm for the MCEI, improving the approximation ratio of the algorithm of Hassin et al. [4] designed for the CND, where ρ_{ST} is any approximation ratio achievable for the Steiner tree problem. Our improvement is based on an elaborate tree partition and a new lower bound on the optimal value which relies on the constraint that no demand is allowed to be split among different copies of the same edge.

References

[1] B. Awerbuch, Y. Azar, Buy-at-bulk network design, In IEEE Symposium on Foundations of Computer Science (FOCS), (1997) 542-547.

[2] F. Grandoni, G. F. Italiano, Improved approximation for single-sink buy-at-bulk, In Proceedings of International Symposium on Algorithms and Computation (ISAAC), (2006) 111-120.

[3] S. Guha, A. Meyerson, K. Munagala, A constant factor approximation for the single sink edge installation problem, In ACM symposium on the Theory of Computing (STOC), (2001) 383-388.

[4] R. Hassin, R. Ravi, F. S. Salman, Approximation algorithms for a capacitated network design problem, Algorithmica, 38 (2004) 417-431.

[5] S. Khuller, B. Raghavachari, N. N. Young, Balancing minimum spanning and shortest path trees, Algorithmica, 14 (1993) 305-322.

[6] Y. Mansour, D. Peleg, An approximation algorithm for minimum-cost network design, Tech. Report Cs94-22, The Weizman Institute of Science, Rehovot, (1994); also presented at the DIMACS Workshop on Robust Communication Network, (1998).

[7] G. Robins, A. Z. Zelikovsky, Improved Steiner tree approximation in graphs, In Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2000), (2000) 770-779.

[8] F. S. Salman, J. Cheriyan, R. Ravi, S. Subramanian, Approximating the single-sink link-installation problem in network design, SIAM J. Optim., 11 (2000) 595-610.

[9] H.-Z. Zheng, D.-H. Chu, D.-C. Zhan, Effective algorithm CFL based on Steiner tree and UFL problem, IJCSNS 6(9A), (2006) 24-27.