Approximating Spanning Tree with Weighted Inner Nodes

Jian Li[⋆] , Haitao Wang, Rudolf Fleischer, Qi Ge, Hong Zhu

Shanghai Key Laboratory of Intelligent Information Processing
Department of Computer Science and Engineering
Fudan University
Shanghai, China, 200433
Email: {lijian83, wanghaitao, rudolf, qge, hzhu}@fudan.edu.cn

Abstract. We consider a problem coming from practical applications: finding a minimum spanning tree with both edge weights and inner node (non-leaf node) weights. This problem is NP-complete even in the metric space. We present two polynomial time algorithms which achieve approximation factors of $2.35 \ln n$ and $2(H_n - 1)$, respectively, where n is the number of nodes in the graph and H_n is the n-th Harmonic number. This nearly matches the lower bound: no polynomial-time approximation algorithm can achieve an approximation factor of $(1-\epsilon)H_n$, for any $\epsilon > 0$. For metric case where the edge weights are symmetric and satisfy the triangle inequality, it also proves to be an NP-hard problem and we give a 3.52 approximation algorithm and an improved factor 3.106 one and also show that an approximation factor of 1.463 is impossible unless $NP \subseteq DTIME[n^{O(loglogn)}]$. We also give an approximation algorithm with factor $\Delta - 1$, where Δ is the maximum degree of the graph.

Keywords: Minimum spanning tree, approximation algorithm, NP-hard

1 Introduction

1.1 Problem Statement

Minimum spanning trees have been widely studied and many efficient polynomial algorithms have been designed to compute them optimally. But when we consider the variant where nodes have weights and the weights of inner nodes of the spanning tree also contribute to the total weight of a tree, then the problem becomes NP-complete. In the paper, we call the non-leaf nodes of the spanning tree inner nodes and call the variant minimum spanning tree *minimum spanning tree with weighted inner nodes (*MSTI*)* problem. In the MSTI problem, we are given a connected undirected graph $G(V, E)$ with n nodes and m edges and a positive weight function w on the edges and nodes of G, i.e., $w : E \cup V \to \mathbb{R}^+$.

If T is a spanning tree of G, then let I_T denote the set of inner nodes of T, also called the *backbone* of T. The *sti-weight* (*s*panning *t*ree with *i*nner node weight) of T is defined as the sum of all edge weights plus all node weights of the backbone, i.e., $sti(T) = \sum_{e \in E(T)} w(e) + \sum_{v \in I_T} w(v)$. The goal is to find a minimum sti-weight spanning tree $T_{msti}(G)$ of G whose overall weight we denote by $msti(G)$ (i.e.$msti(G) = T_{msti}(G)$).

Many practical problems can naturally be modeled as an MSTI problem. For example, consider the problem of connecting a number of computers by some network. Then the

[⋆] Corresponding author. Email: lijian83@fudan.edu.cn Address: Department of Computer Science, Fudan University, 220 Han Dan Road. Shanghai, China, 200433 cost is determined by the cost of all cables plus the cost of additional routing hardware needed at intermediate nodes of the network.

If we set all edge weights to zero and all node weights to one, we obtain as a special case the *connected dominating set problem* (CDS). Since CDS is NP-complete [2], MSTI is NP-complete. Furthermore, MSTI has no $(1-\epsilon)H_n$ approximation, for any $\epsilon > 0$, unless $NP \subseteq DTIME(n^{O(\log\log n)})$, as this lower bound holds already for CDS [3].

1.2 Our Results

First we show that the MSTI problem is also NP-hard even in the metric space. We then give some approximation algorithms for MSTI. We assume throughout the paper that $n \geq 3$. In the first algorithm we combine an approximation algorithm for the uncapacitated facility location problem with an approximation algorithm for the node weighted Steiner tree problem to get an algorithm with an approximation factor of $2.35 \cdot \ln(n)$. If restricted to metric MSTI, the above algorithm gives a 3.52-approximation. By using a slightly different algorithm for facility location which considers a tradeoff between facility opening costs and connection costs we obtain a 3.106-approximation algorithm. The lower bound of 1.463 is also proved for the approximation factor.

We present a second algorithm with an improved approximation factor of $2(H_n - 1)$. This algorithm follows a greedy approach. In each iteration, we choose a star so as to minimize the ratio of the weight of the star to its size. A similar approach, based on spiders, was used by Klein and Ravi [5] and Guha and Khuller [4] to approximate the node weighted Steiner tree problem.

When the maximum degree Δ of the graph is small, we give an algorithm with an approximation factor of $\Delta - 1$. In the paper, we often use the following inequality.

Fact 1 *Let $a_1, \ldots, a_k, b_1, \ldots, b_k$ be positive real numbers. Then,*

$$\min\{\frac{a_1}{b_1}, \ldots, \frac{a_k}{b_k}\} \leq \frac{\sum_{i=1}^{k} a_i}{\sum_{i=1}^{k} b_i}$$

□

This paper is organized as follows. In Section 2, we prove MSTI is NP-hard in the metric space. In Section 3 we present a $(2.35 \cdot \ln n)$-approximation algorithm in general space and the 3.52-approximation algorithm in the metric space. In Section 4, we give an improved 3.106-approximation algorithm in the metric space. In Section 5 we present the $2(H_n - 1)$-approximation algorithm for general space. In Section 6 we present a $(\Delta - 1)$-approximation algorithm. We end the paper with some remarks in Section 7.

2 NP-hard Proof of the Metric MSTI

Theorem 2. *The* MSTI *problem is NP-hard even in the metric space, where the edge weights are symmetric and satisfy the triangle inequality.*

Proof. We prove it by reduction from *Maximum Leaf Spanning Tree (MLST)* problem which is NP-hard[2]. Recall that the `MLST` problem is to find a spanning tree which has the maximum number of leaves in a given graph. The reduction algorithm is as follows:

For any given graph $G(V, E)$, we transform it to a new graph $G'(V', E')$ just by connecting u and v, if (u, v) is not in E. Now G' is a complete graph. For each node $v \in V'$, set $w(v) = \beta$. Here $\beta = \frac{1}{n^2}$ and n is the number of nodes in G. For each edge $(u, v) \in E'$, if $(u, v) \in E$, set $w(u, v) = 1$; otherwise, $w(u, v) = 2$. It is not difficult to see that the edge weights are symmetric and satisfy triangle inequality. Obviously, the reduction algorithm is polynomial.

Now we claim that there exits a spanning tree in G which has at least k leaves if and only if there exists a spanning tree of G' whose overall sti-weight is at most $(n-k)\beta+n-1$. We prove it as follows.

Suppose there is a spanning tree T of G which has $t \geq k$ leaves. Then in G', since the weight of each edge in T is 1, the weight of each inner node in T is β and the number of inner nodes of T is $n - t$, the sti-weight of T in G' is $(n - t)\beta + n - 1$ which is at most $(n - k)\beta + n - 1$.

If there is a spanning tree T' in G' whose weight is at most $(n - k)\beta + n - 1$, then we know the weight of T' is less than n because $(n - k)\beta = (n - k)\frac{1}{n^2} < 1$. Since there are $n - 1$ edges in T', it is easy to see that each edge in T' has weight 1. This implies T' is also a spanning tree in G. Because weight of the backbone is at most $(n - k)\beta$, we know that the number of the inner nodes in T' is at most $n - k$ which means the number of leaves is at least k.

$\square$

3 A Simple Algorithm

We first solve an uncapacitated facility location problem where all the nodes of G are clients. Each node is a facility candidate and the cost of opening a facility at node v is $w(v)$. Let $F \subset V$ be the set of facilities chosen to open in the solution of the facility location. Then we set the weight of the nodes in F to zero and subdivide each edge of G into two edges by introducing a new node, whose weight is the weight of the subdivided edge, in the middle of the subdivided edge. Then we solve the node weighted Steiner tree problem, where terminals are the nodes in F. Finally we combine the solution of the facility location problem (i.e., the set of edges used to connect nodes to their closest facility) with the obtained Steiner tree to get a solution to the `MSTI` problem, breaking cycles (if there exits) by deleting edges arbitrarily on the cycles. For the metric case, after finding the solution of facility location, since all nodes in F have direct connections, we just find a normal minimum spanning tree in the subgraph SG_F which is induced by F. The following is the two algorithms:

Algorithm 1(G)

```
Solve the uncapacitated facility location problem in $G$ by the approximation algorithm in [7]. Let $SOL_{ufl}$
        denote the solution. Let $F$ be the set of facilities opened in $SOL_{ufl}$.

for each $v \in F$
    $w(v) = 0$.
```

```
for each edge $e(u,v) \in G$
    delete $e$
    add a node $a_{uv}$
    add two edges $e_1(u, a_{uv})$, $e_2(a_{uv}, v)$
    $w(a_{uv}) = w(e)$
    $w(e_1) = w(e_2) = 0$ //Now every edge's weight is 0.

Considering the nodes in $F$ as terminals, find a node weighted Steiner tree by the approximation algorithm in
    [4]. Let $SOL_{steiner}$ denote the steiner tree.

Combining $SOL_{ufl}$ with $SOL_{steiner}$ to get a graph $G^*$, if there are cycles in $G^*$, break cycles by arbitrarily
    deleting edges on the cycles to get a spanning tree $T$ of $G$.
return $T$
```

Algorithm 2(G) (in metric space)

```
Solve the uncapacitated facility location problem in $G$ by the approximation algorithm in [6]. Let $SOL_{ufl}$
    denote the solution. Let $F$ be the set of facilities opened in $SOL_{ufl}$.

Using Prim's␣algorithm␣in␣[1]␣to␣find␣a␣minimum␣spanning␣tree␣$T_F$ in the subgraph $SG_F$ induced by $F$.

Combine $SOL_{ufl}$ with $T_F$ to get a spanning tree $T$ of $G$.

return $T$
```

Theorem 3.

(a) The Algorithm 1 is a factor $(2.35 \ln n)$*-approximation algorithm for* MSTI.

(b) The Algorithm 2 is a factor 3.52*-approximation algorithm for the metric* MSTI .

Proof.

(a) Since each node of G is either a facility or a client connected to a facility, and the Steiner tree connects all facilities, combining both solutions produces a spanning network of G, which we can turn into a spanning tree by deleting edges on cycles.

Let OPT_F denote the minimum cost of the facility location problem, and OPT_S the minimum weight of the node weighted Steiner tree problem with terminals F. It is easy to see that $T_{msti}(G)$ could also be a feasible solution (maybe not optimal) for solving the facility location problem, so we have $OPT_F \leq msti(G)$. The facility location problem can be $\ln n$-approximated [7].

We also have $OPT_S \leq msti(G)$. Here we give a simple idea of the proof. We assume T is the optimal spanning tree of $msti(G)$, I_T is the set of inner nodes of T, and F is the terminals set of the steiner tree problem. We delete the leaves which are in T but not in F to obtain T'. Then T' is a steiner tree of F. We assume $steiner(T')$ is the sum of the weight of the steiner tree. $steiner(T') = \sum_{e \in E(T')} w(e) + \sum_{v \in V(T')-F} w(v)$. Remember that $msti(G) = \sum_{e \in E(T)} w(e) + \sum_{v \in I_T} w(v)$. Because $E(T') \subseteq E(T)$ and $V(T') - F \subseteq I_T$, we can get $steiner(T') \leq msti(G)$. Then we obtain the inequality $OPT_S \leq steiner(T') \leq msti(G)$. The best approximation algorithm for the node weighted steiner tree problem achieves an approximation factor of $1.35 \ln n$ [4].

Using the above two approximation algorithms, we combine their solutions and break cycles (if there exits) by arbitrarily deleting edges on the cycles, and then we can obtain a $(\ln n + 1.35 \ln n)$-factor approximation solution for MSTI.

(b) The factor of approximation algorithm in [6] for the facility location problem in the metric space is 1.52. And then we claim that the weight of the spanning tree T_F found in the second step of Algorithm 2 is less than $2msti(G)$. Here we also give the simple idea about the 2-approximation. Assume T is the optimal solution of the metric `MSTI` problem in G, T_F is the minimum spanning tree of the subgraph SG_F which is induced by the nodes in F, and T_G is the minimum spanning tree of G. For each edge $e(u,v) \in T_G$, we add a secondary edge $e'(u,v)$ to T_G and finally get a new graph T'_G. Now we can easily get a Euler tour P in T'_G since each node has even degree. Then we connect directly each node in F according to the order in which they appear firstly in P to get a new tree (in fact, a path) T'_F which is a spanning tree of SG_F. Due to the triangle inequality, it is not difficult to get $\sum_{e\in T'_F} w(e) \le \sum_{e\in P} w(e) = 2\sum_{e\in T_G} w(e)$. Since T_G is the minimum spanning tree in G, we have $\sum_{e\in T_G} w(e) \le \sum_{e\in T} w(e)$. Since T_F is the minimum spanning tree of SG_F, we have $\sum_{e\in T_F} w(e) \le \sum_{e\in T'_F} w(e)$. Finally we can obtain $\sum_{e\in T_F} w(e) \le \sum_{e\in T'_F} w(e) \le 2\sum_{e\in T_G} w(e) \le 2\sum_{e\in T} w(e) \le 2msti(G)$. Then we just combine the above two results and obtain a factor (1.52+2) approximation solution of the metric `MSTI` problem.

□

4 Metric MSTI

4.1 The Improved Approximation Algorithm

In this section, we give a better approximation algorithm for the metric `MSTI` problem. Our improved metric `MSTI` algorithm runs in three stages.

First, we scale up the facility costs by a constant factor of $\delta > 1$ (which will be fixed later) and then run the greedy facility location algorithm of [9] (often called *JMS algorithm*) to find a solution of the scaled facility location problem.

In the second stage, we scale down the facility costs back to their original values. If at any point during this process a facility could be opened without increasing the total cost, then we open that facility and connect each client to its closest open facility. In other words, if the opening cost of the facility is equal to or less than the total cost that clients can save by switching their service facilities to that facility, then we open the facility and connect those clients to it. Let F denote the set of facilities chosen to open after the second stage, and let C denote the total cost of the facilities opening and the connection.

Finally, we connect the facilities in F by finding a minimum spanning tree T_F in the subgraph which is induced by the nodes in F. Let C_{msti} denote the cost of the final solution.

For an optimal solution of `MSTI`, let C_I^* denote the weight of all inner nodes and C_T^* the weight of all edges. In [9], a general lemma shows the relation between the solution of JMS algorithm and any other solution of the facility location problem. For any fixed parameter $\gamma_f \ge 1$, a second parameter γ_c can be computed as the supremum of the maximum solutions of an infinite family of optimization problems. It was shown in [6] that $\gamma_c \le 1.78$ when $\gamma_f = 1.11$.

Lemma 1. *[9] Let $\gamma_f \ge 1$ and $\gamma_c := sup_k\{z_k\}$, where z_k is the solution of the following optimization algorithm:*

$\boldsymbol{maximize}\ z_k = \frac{\sum_{i=1}^{k} \alpha_i - \gamma_f f}{\sum_{i=1}^{k} d_i}$

subject to:

$\forall 1 \le i < k : \alpha_i \le \alpha_{i+1}$

$\forall 1 \le j < i < k : r_{j,i} \ge r_{j,i+1}$

$\forall 1 \le j < i \le k : \alpha_i \le r_{j,i} + d_i + d_j$

$\forall 1 \le i \le k : \sum_{j=1}^{i-1} max(r_{j,i} - d_j, 0) + \sum_{j=i}^{k} max(\alpha_i - d_j, 0) \le f$

$\forall 1 \le j \le i \le k : \alpha_j, d_j, f, r_{j,i} \ge 0$

then for every instance $\mathcal{I}$ of the facility location problem and for every solution of $\mathcal{I}$ with facility opening cost C_{open} and connection cost C_{conn}, the cost of the solution found by the JMS algorithm is at most $\gamma_f C_{open} + \gamma_c C_{conn}$. □

It was also proved in [6] that after the second stage of our algorithm the cost C can be bounded by $C \le max\{\gamma_f + 1 - \frac{1}{\delta}, \gamma_f + \ln \delta\}C^*_{open} + (1 + \frac{\gamma_c - 1}{\delta})C^*_{conn}$, where C^*_{open} and C^*_{conn} are, respectively, the facility opening cost and connection cost of an optimal solution of the unscaled facility location problem. It is not difficult to prove that $\gamma_f + 1 - \frac{1}{\delta} < \gamma_f + \ln \delta$ when $\delta > 1$. So we have $C \le (\gamma_f + \ln \delta)C^*_{open} + (1 + \frac{\gamma_c - 1}{\delta})C^*_{conn}$. Actually, by carefully analyzing the proof of the above inequality in [6], we can easily find that in the right side of the inequality the facilities opening cost and connection cost could be not only the optimal solution but any other feasible solution as well. The following lemma shows the idea.

Lemma 2. *For any feasible solution of the original (unscaled) facility location problem with facility opening cost C_{open} and connection cost C_{conn}, $C \le (\gamma_f + \ln \delta)C_{open} + (1 + \frac{\gamma_c - 1}{\delta})C_{conn}$.* □

Theorem 4. *Our algorithm is a factor* 3.106 *approximation algorithm for the metric* `MSTI` *problem.*

Proof. We have proved in theorem 3 that $cost(T_F) \le 2C^*_T$. Remember that C^*_I and C^*_T respectively denote the weight of all inner nodes and the weight of all edges in the optimal solution of the metric MSTI problem. According to Lemma 2, we have the inequality,

$$\begin{aligned} C_{msti} &\le C + cost(T_F) \le (\gamma_f + \ln \delta)C^*_I + (1 + \frac{\gamma_c - 1}{\delta})C^*_T + 2C^*_T \\ &\le \max(\gamma_f + \ln \delta, 3 + \frac{\gamma_c - 1}{\delta})msti(G) \ . \end{aligned}$$

Choosing $\gamma_f = 1.11$ and $\gamma_c = 1.78$, we can minimize the approximation factor by choosing $\delta = 7.36$. We then get an approximation factor of 3.106. □

Mahdian [8] have numerically calculated many pairs (γ_f, γ_c) (by solving the first 100 optimization problems), which should give a good approximation of the trade-off between facility cost approximation ratio and connection cost approximation ratio. We try all these pairs and find the pair $(1.11, 1.78)$ is the best and fortunately, this pair has been proved mathematically in [6].

4.2 A Lower Bound for Approximating Metric MSTI

In this section, we will slightly modify the lower bound proof for the non-approximability of the metric uncapacitated facility location problem [10] to get a lower bound for the non-approximability of metric `MSTI`. In particular, we establish a relation between the set cover problem (`SC`) and metric `MSTI` and show that if metric `MSTI` can be approximated within a factor of 1.463, then we can have a polynomial time approximation algorithm for `SC` with approximation factor $c \cdot ln(n)$ for some $c < 1$, which implies $NP \subseteq DTIME[n^{O(loglogn)}]$ [11].

An instance of `SC` is given by a set of elements $X = \{x_1, x_2, \ldots, x_n\}$ and a collection of subsets $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$, where each $S_i \subseteq X$. The goal is to choose a minimum number of subsets in $\mathcal{S}$ to cover all elements in X.

We construct an instance of `MSTI` as follows. We introduce a node for each $x_i \in X$ and each $S_j \in \mathcal{S}$, and add an additional node r with its weight $w(r) = 0$. With a slightly abuse of notation, we also use x_i and S_j to denote the corresponding nodes in the constructed instance of `MSTI`. If $x_i \in S_j$, then the two nodes are connected by an edge of weight 1 (i.e. $w(e(x_i, S_j)) = 1$). All S_j are connected to r by an edge of weight 2. All the other edge weights are defined by the shortest path in the graph defined so far. Obviously, the edge weights satisfy the triangle inequality. We assign a very large weight to all nodes x_i (so that they will not be chosen as inner nodes in an optimal `MSTI` tree). The weights of the S_j will be specified later.

The basic idea is to use the solution of `MSTI` to obtain a partial solution of the set cover problem in one iteration. If we can find a good approximation of `MSTI`, say with factor 1.463, then we can approximate the set cover problem within a factor of $c \cdot \ln n$ for some $c < 1$.

Because all nodes of x_i have a large weight, any `MSTI` approximation algorithm with a reasonable approximation factor will find a tree T in which all these x_i nodes are leaves. If T contains an edge $e = (S_i, S_j)$, then we delete e. Then there are two connected components left and S_i and S_j are in different connected components. Here we assume S_i is in the same connected components as r and thus S_j is in the other components. We then add the edge $e'(r, S_j)$ to $T - e$ to obtain a new spanning tree T' (i.e. $T' = T - e + e'$). Since $w(e) \geq 2$ and $w(e') = 2$, we know $msti(T) \geq msti(T')$. The inner nodes S_j are the subsets we will choose for the set cover problem, and the x_i connected to them are the elements covered by those subsets.

Now we formally give the algorithm to solve `SC` by using an approximation algorithm for `MSTI` as a subroutine. Suppose k is the optimal solution of an `SC` instance and γ a constant we will specify later. Since we do not know k, we run the following algorithm for $k = 1, 2, 3, \ldots, |\mathcal{S}|$.

```
Set Cover($X$,$\mathcal{S}$)
While $X \neq \emptyset$ do
        Create an MSTI instance corresponding to $(X, \mathcal{S})$, where all the nodes of $S_j$ are assigned the same weight $f = \gamma \frac{|X|}{k}$;
        Compute an MSTI-approximation $T$ by the approximation algorithm;
        Let $X'$ be the set of all $x_i$ that are connected by an edge of weight 1 in $T$ and $\mathcal{S}'$ the set of inner nodes $S_j$;
        Set $X = X - X'$ and $\mathcal{S} = \mathcal{S} - \mathcal{S}'$;
    end while
```

The following lemma was proved in [10].

Lemma 3. *[10] Suppose we have a set cover instance $(X, \mathcal{S})$ where the minimum set cover size is k. If there is a polynomial time algorithm that can pick βk sets (for any constant β) covering $c'|X|$ elements, where $c' > c_\beta = (1 - \frac{1}{e^\beta})$, then $NP \subseteq DTIME[n^{O(loglogn)}]$.* □

Theorem 5. *If there is a polynomial time approximation algorithm with an approximation factor smaller than 1.463 for metric* `MSTI`*, then $NP \subseteq DTIME[n^{O(loglogn)}]$.*

Proof. Consider iteration j of the above algorithm with sets X_j and $\mathcal{S}_j$. Let $n_j = |X_j|$, $m_j = |S_j|$, and f_j be the weight of S_j. Since there is a solution of size k of the set cover problem, there is a solution of `MSTI` of cost $kf_j + 2m_j + n_j$. If there is an α-approximation for `MSTI`, then we obtain a solution of cost at most $\alpha(kf_j + 2m_j + n_j)$.

Suppose this solution selects βk nodes in $\mathcal{S}$ as inner nodes and cn nodes in X as leaves within distance 1 of these inner nodes. By Lemma 3, $c \leq c_\beta$ unless $NP \subseteq DTIME[n^{O(loglogn)}]$. We know the overall cost of this solution is at least $\beta k f_j + cn_j + 2m_j + 3(n_j - cn_j)$, but at most $\alpha(kf_j + 2m_j + n_j)$.

Thus, we can obtain the following inequality,

$$\alpha \geq \frac{\beta k f_j + 2m_j + 3n_j - 2cn_j}{kf_j + 2m_j + n_j} \geq \frac{\beta\gamma n_j + 2m_j + 3n_j - 2c_\beta n_j}{\gamma n_j + 2m_j + n_j}$$

The last inequality holds because $c < c_\beta$ and $f_j = \gamma\frac{n_j}{k}$. The right hand side is minimized with $\beta = ln\frac{2}{\gamma}$, i.e.,

$$\alpha \geq \frac{n_j(1 + \gamma + \gamma ln\frac{2}{\gamma}) + 2m_j}{n_j(1+\gamma) + 2m_j} \approx \frac{n_j(1 + \gamma + \gamma ln\frac{2}{\gamma})}{n_j(1+\gamma)}$$

.

The last approximation holds when $m_j/n_j \leq \epsilon$ for some small $\epsilon > 0$. We can assume that this is true for every j because we can reduce the original set cover problem to an equivalent set cover problem where we reproduce many copies of each single element, say $|\mathcal{S}|\frac{1}{\epsilon}$ many. The containment relationship is reserved. This leaves the number of subsets unchanged but increases the number of elements such that $m_j/n_j \leq \epsilon$. Choosing $\gamma = 0.463$ we get $\alpha \geq 1.463$. □

5 A $2(H_n - 1)$-Approximation for the General MSTI

In the previous section, we used approximation algorithms for facility location and node weighted Steiner tree, both of which use the similar technique to classical set cover approximation algorithms. For the algorithm in this section, we also follow a greedy approach, in each iteration we select a minimum ratio partial solution. The similar ideas appear in[4, 5] for example. However, to find a minimum ratio partial solution is not always a trivial work like in the set cover approximation. The main difficulty here is how to define the structure of partial solution and how to find a minimum ratio partial solution.

Now, we describe our algorithm which runs in iterations. In the beginning, every node of G forms a singleton tree. In each iteration, we combine some trees together into a new larger tree. The algorithm terminates when only one tree is left which will be our `MSTI` approximation. Since at any time of our algorithm our partial solution forms a forest, we use F_i to denote this forest just before going into iteration i.

In each iteration we choose a *treestar* which consists of a tree T_0, k trees $T_1, \ldots, T_k$ and k edges $e_1, \ldots, e_k$, where $k \geq 1$ and e_i connects a node in T_0 to a node in T_i. We call T_0 the *center-tree* of the treestar, $T_1, \ldots, T_k$ the *leaf-trees*, and e_i the *leaf-edges*. We then combine the center-tree plus the leaf-trees and the leaf-edges into one larger tree.

We distinguish *paid nodes* from *unpaid nodes*. If the weight of the node has been counted, it is paid. Otherwise, it is unpaid. Initially, all nodes are unpaid. We will make sure that all the inner nodes of F_i are paid before iteration i and each node will be paid at most once. In each iteration, we pay the cost of the treestar we choose. Formally, the cost of treestar TS is composed of the cost of all the leaf-edges in TS plus all *unpaid* nodes that are leaves in original trees but become inner nodes in the new larger tree. The only exception is that if the center is an unpaid singleton tree (a node) and the newly produced larger tree consists only of two nodes, then the center node must be paid in the new larger tree although it is a leaf. We call the above two types of nodes which should be paid in this iteration *switch nodes* of the treestar TS. After this iteration, all switch nodes become paid. Fig. 1 shows a treestar.

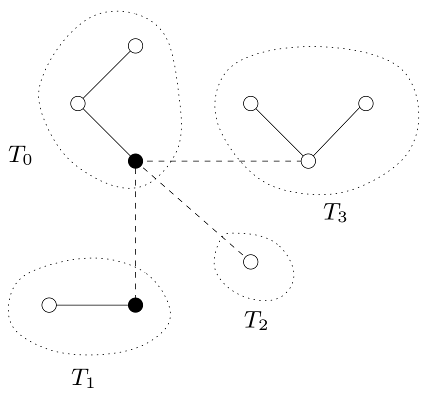

Fig. 1. A treestar with center-tree T_0 and leaf-trees T_1, T_2, T_3. The dashed lines are leaf-edges. The solid nodes are switch nodes. The cost of the treestar is the sum of the weights of the three leaf-edges and the two switch nodes.

To achieve a good approximation ratio we must in each step select the treestar carefully. Note that the *cost* of a treestar is the sum of the leaf-edge weights plus the weights of all switch nodes. As a consequence, the *sti*-weight of the tree is at most the sum of all the treestar costs involved in its iterative construction. Every node of the backbone is counted for once, namely in the iteration when it becomes an inner node. And every edge is counted in the iteration when it is added as a leaf-edge.

We define the *relative cost* of a treestar as the quotient of its cost and its number of tree components (i.e. one plus the number of leaf-trees). In each iteration we choose a treestar of minimum relative cost, a so-called *mrc-treestar*. We will first show that we can find such an mrc-treestar efficiently, and then analyze the approximation ratio of this construction.

5.1 Computing a Treestar of Minimum Relative Cost

We show how to find an mrc-treestar with fixed center T_0. Since T_0 may have many nodes, enumerate all the possibilities of the leaf-edge combinations will lead to an exponential time algorithm. The main observation is that we do not need to test all possible combinations.

Lemma 4. *There is an mrc-treestar with center T_0 such that all leaf-trees are connected to the same node of T_0.*

Proof. Assume there is a mrc-treestar TS where we have leaf-trees connected to nodes $v_1, \ldots, v_t$ of T_0, where $t \geq 2$. Let S_i be the set of leaf-trees connected to node v_i, for $i = 1, \ldots t$. Then we have

$$mrc(TS) = \frac{\sum_{i=1}^{t} cost(TS_i)}{1 + \sum_{i=1}^{t} |S_i|} = \frac{\sum_{i=1}^{t} cost(TS_i)}{\sum_{i=1}^{t}(\frac{1}{t} + |S_i|)},$$

where $mrc(TS)$ is the relative cost of TS and $cost(TS_i)$ is the cost of the treestar TS_i which consists of center-tree T_0 and the leaf-trees S_i connected to v_i. By Fact 1, there is a node v_i such that the treestar TS_i has relative cost

$$mrc(TS_i) = \frac{cost(TS_i)}{1 + |S_i|} < \frac{cost(TS_i)}{\frac{1}{t} + |S_i|} \leq mrc(TS).$$

□

For a fixed center-tree T_0 and node v_0 in T_0 we can compute the respective mrc-treestar as follows. Let $dis(v_0, T_i)$ denote the minimum cost of connecting the leaf-tree T_i to node v_0 in T_0 (but without the cost induced by v_0 if it becomes a switch node). Formally,$dis(v_0, T_i) = \min_{v \in T_i}(w(e(v_0, v)) + sn(v))$ where $sn(v) = w(v)$ if v becomes a switch node and $sn(v) = 0$ otherwise. If the mrc-treestar has k leaf-trees, it must use the k cheapest trees. So all we have to do is to compute $dis(v_0, T_i)$ for all other trees T_i. Assume there are l other trees T_i, we sort them by the value $dis(v_0, T_i)$ in non-decreasing order. Suppose they are $T_1, T_2, \ldots, T_l$, and then we find the value k that minimizes $rc(T_0, v_0) = \frac{sn(v_0) + \sum_{i=1}^{k} dis(v_0, T_i)}{k+1}$.

Then, the mrc-treestar TS in forest F is the tree with the relative cost $mrc(TS) = \min_{T \in F, v \in T} rc(T, v)$.

Theorem 6. *We can compute an mrc-treestar in time $O(m \log n)$.*

Proof. We must compute an mrc-treestar for any fixed center tree T and any node v in T. There may be up to n trees, but there are also only n nodes and fixing the node automatically fixes the tree which the fixed node belongs to. So we actually only need to compute n mrc-treestars with the fixed tree and the fixed node. The time to compute one such star is dominated by $O(n \log n)$, the time needed to sort the $dis(v_0, T_i)$ values. Since each T_i can only incur cost once for each adjacent edge, the total time is bounded by $O(m \log n)$. □

Here we give the outline of our algorithm. **Mrc-Treestar** is a subroutine which will be used in **Msti**

Mrc-Treestar (F) //F is the forest where we want to find the mrc-treestar

```
for each tree $T \in F$
   for each node $v \in T$
       if $paid(v) = 0$
          then $sn(v) = w(v)$
          else $sn(v) = 0$
       for each tree $T' \in F$ $(T' \neq T)$
           for each node $v' \in T'$ and if $(v, v') \in E(G)$
               if $paid(v') = 0$ and $deg_F(v') \geq 1$ //$deg_F(v')$ is the degree of node $v'$ in $F$
                  then $sn(v') = w(v)$
                  else $sn(v') = 0$
               $dis(v, v') = w(e(v, v')) + sn(v')$
           $dis(v, T') = \min_{v' \in T'} dis(v, v')$
       Sort all trees $T' \neq T$ by $dis(v, T')$ in nondecreasing order, suppose they are $T_1, T_2, \ldots, T_l$
       $rc(T, v) = \min_k(\frac{sn(v)+\sum_{i=1}^{k} dis(v,T_i)}{k+1})$
$mrc(TS) = \min_{T \in F, v \in T}(rc(T, v))$
update paid function according to $TS$
return $TS$
```

Msti (G)//G is the graph where we want to find a MSTI

```
$F_1 = \bigcup_{i=1}^{n}\{\{v_i\}\}$
$paid(v_i) = 0, 1 \leq i \leq n$
$T = \emptyset, i = 1$
while ($T$ is not a spanning tree of $G$)
   do $TS_i = Mrc - Treestar(F_i)$.
      Let $E(TS_i)$ be tree−edges of $TS_i$ and $T_1, T_2, .., T_k$ be tree components (center−tree and leaf−trees)
         involved in $TS_i$.
      $T = T \cup E(TS_i)$
      $F_{i+1} = F_i - \{T_1, T_2, .., T_k\} + \{T_1 \cup T_2 \cup \ldots \cup T_k\}$
      $i = i + 1$
end while
return $T$
```

5.2 Analysis of the Approximation Factor

We now prove that we eventually compute a $2(H_n - 1)$-approximation to the minimum *sti*-weight tree. Let F_i be the forest of n_i trees before iteration i. Let OPT_i denote the minimum cost of connecting *all* these trees (using a similar cost function as we used for the cost of a treestar, i.e. the cost of all edges and those nodes which are leaves in F_i but finally become inner nodes). In particular, F_1 is the set of nodes of G, $n_1 = n$, and $OPT_1 = OPT$, the minimum *sti*-weight for G. Let TS_i be the treestar computed in iteration i, with m_i tree components (the number of leaf-trees of TS_i plus one), and let

$cost(TS_i)$ be its cost. After t iterations (it is easy to see t must be smaller than n), the algorithm terminates.
We start with a lemma.

Lemma 5. *Any tree of size at least 2 can be partitioned into vertex-disjoint stars of size at least 2, where a star is a tree which has at most one inner node.*

Proof. By induction on the size of the tree. A tree of size 2 is a star of size 2. If the tree has more than 2 nodes, let r be any vertex of degree at least 2. Let $T_1, T_2, T_3, \ldots$ be the subtrees rooted at r. If some of them have size one, we connect all of them to r to form a star of size at least 2; the other subtrees will have a star partition by the induction hypothesis. If all T_i have size at least 2, then $T_1 \cup r, T_2, T_3, \ldots$ have a star partition by the induction hypothesis. □

The next two lemmas show that we can bound the relative cost of the mrc-treestar found in any iteration by the relative cost of OPT.

Lemma 6. *For all $i \geq 1$, $\frac{cost(TS_i)}{m_i} \leq \frac{OPT_i}{n_i}$.*

Proof. By Lemma 5, the optimal connection of all trees in F_i is a tree (regarding the trees in F_i as nodes) which has a star partition. By Fact 1, there is one star (thus a treestar in F_i) in this partition whose relative cost is at most $\frac{OPT_i}{n_i}$. But we computed a treestar with minimum relative cost, so the claim follows.

Lemma 7. *For all $i \geq 1$, $OPT_{i+1} \leq OPT_i$.*

Proof. Suppose $F_i = \{T_1, T_2, \ldots T_{n_i}\}$ and we find the mrc-treestar $TS_i = \{T_1, T_2, \ldots T_k\}$, $2 \leq k \leq n_i$ and merge its components into a single tree. If $k = n_i$, the lemma holds since $OPT_{i+1} = 0$. Let G_i denote an optimal connection network of these trees in F_i. The cost of G_i is composed of the cost of all the edges of G_i and the cost of all nodes that are leaves in F_i but become inner nodes after adding G_i. We will construct a connection network G_{i+1} for F_{i+1} (G_{i+1} may not be the optimal solution for F_{i+1}). Consider all the edges in the cut $S = S(A; B)$ where $A = T_1 \cup T_2 \cup \ldots \cup T_k$ and $B = T_{k+1} \cup \ldots \cup T_{n_i}$ in G_i. we denote these cut edges (a_j, b_j) where $a_j \in A$ and $b_j \in B$. we can obtain a connection network G_{i+1} for F_{i+1} whose cost is not more than the cost of G_i by deleting some edges of S. Consider the subgraph H_i of $F_i \cup G_i$ induced by the nodes of B. Suppose H_i have several components, say $C_1, C_2, \ldots, C_l$. We let G_{i+1} consist of all edges of $H_i \cap G_i$ and l cut edges each of which connects one component to A. Since A is a tree in F_{i+1}, such a G_{i+1} can connect all trees in F_{i+1}. For a component C_i, arbitrary choosing a cut edge (a_x, b_x) that connects C_i to A may not work because of a bad situation where the cost of G_i does not include the cost of a_x while the cost of G_{i+1} does, rendering $G_{i+1} \geq G_i$. The bad situation happens only when a_x is a singleton tree in F_i and a leaf in tree $F_i \cup G_i$ (the cost of G_i does not contains the cost of a_x), but an unpaid leaf in F_{i+1} and an inner node in tree $F_{i+1} \cup G_{i+1}$ (so the cost of G_{i+1} contains the cost of a_x). We argue that there exists a cut edge adjacent to each component C_i such that the bad situation doesn't happen. Suppose for component C_1 there is no such edge, then l must be 1 since all choice a_xs are leaves in tree $F_i \cup G_i$; otherwise $F_i \cup G_i$ is not connected. So A is the union of all such

a_xs. However, they can't be all unpaid leaves in G_{i+1}, for at least the center must be paid. Therefore the contradiction follows.

□

In particular, $OPT_i \leq OPT_1 = OPT$ for all $i \geq 1$.

Theorem 7. *We can compute a $2(H_n - 1)$-approximation to the minimum sti-weight tree in time $O(nm \log n)$.*

Proof. First note that

$$\frac{m_i}{n_i} = \frac{m_i - 1}{n_i} + \frac{1}{n_i} \leq \sum_{k=n_i-m_i+2}^{n_i} \frac{1}{k} + \frac{1}{n_i} \leq 2 \cdot \sum_{k=n_i-m_i+2}^{n_i} \frac{1}{k}$$

In each iteration, we combine m_i trees into a single tree. Thus, $n_{i+1} = n_i - m_i + 1$. Because in the last iteration (the tth iteration) all of the leaf-trees will be combined into the final spanning tree which is our approximation solution, it is easy to know $n_t = m_t$. And there are more than one trees in F_t, which implies $n_t \geq 2$. Since $n_{t-1} > n_t$, we can get $n_{t-1} \geq 3$. Using Lemmas 6 and 7 we can can get the inequality:

$$\sum_{i=1}^{t} cost(TS_i) \leq \sum_{i=1}^{t-1} OPT_i \cdot \frac{m_i}{n_i} + \frac{m_t}{n_t} \cdot OPT_t \leq (2 \cdot \sum_{i=1}^{t-1} \sum_{k=n_i-m_i+2}^{n_i} \frac{1}{k} + 1) \cdot OPT$$

Since $n_1 = n$ and $n_{t-1} \geq 3$, we can obtain

$$\begin{aligned}\sum_{i=1}^{t-1} \sum_{k=n_i-m_i+2}^{n_i} \frac{1}{k} &= \frac{1}{n} + \frac{1}{n-1} + \cdots \frac{1}{n_{t-1}} \\ &\leq \frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{3} \\ &\leq H(n) - (\frac{1}{2} + 1)\end{aligned}$$

Then we can obtain the upper bound of our approximation solution:

$$\begin{aligned}\sum_{i=1}^{t} cost(TS_i) &\leq ((2(H(n) - \frac{3}{2}) + 1) \cdot OPT \\ &= 2(H(n) - 1) \cdot OPT\end{aligned}$$

The running time follows from theorem 6 and the fact that we have at most n iterations.

□

6 A ($\varDelta - 1$)-Approximation

Let $\varDelta$ be the maximal degree of the graph. For small $\varDelta$ there is a simple and fast $(\varDelta-1)$-approximation algorithm.
We try to transfer the node weights to the edges and then find a normal minimum spanning tree in the new graph with only edge weights. For each edge $e(u,v)$, let the new weight be $w'(e) = w(e) + w(u) + w(v)$. Then we compute a minimum spanning tree T for edge weights w' [1].

Theorem 8. *T is a $\varDelta - 1$-approximation to the optimal solution $T_{msti}(G)$.*

Proof. For the cost $w'(T)$ and $w'(T_{msti}(G))$ of the two trees with respect to w' we have

$$w'(T) = \sum_{e\in E(T)} w'(e) = \sum_{e\in E(T)} w(e) + \sum_{v\in V(T)} deg_T(v)\cdot w(v)$$

and

$$w'(T_{msti}(G)) = \sum_{e\in E(T_{msti}(G))} w'(e) = \sum_{e\in E(T_{msti}(G))} w(e) + \sum_{v\in V(T_{msti}(G))} deg_{msti}(v)\cdot w(v)\ ,$$

where $deg_T(v)$ and $deg_{msti}(v)$ denote the degree of v in T and $T_{msti}(G)$, respectively. Since $w'(T) \le w'(T_{msti}(G))$, we can get the inequality:

$$\sum_{e\in E(T)} w(e) + \sum_{v\in I_T}(deg_T(v)-1)\cdot w(v) \le \sum_{e\in E(T_{msti}(G))} w(e) + \sum_{v\in I_{T_{msti}(G)}} (deg_{msti}(v)-1)\cdot w(v)$$

Thus, if $\varDelta \ge 2$, we get for the *sti*-weight of T in G

$$\begin{aligned} sti(T) &= \sum_{e\in E(T)} w(e) + \sum_{v\in I_T} w(v) \le \sum_{e\in E(T)} w(e) + \sum_{v\in I_T}(deg_T(v)-1)\cdot w(v) \\ &\le \sum_{e\in E(T_{msti}(G))} w(e) + \sum_{v\in I_{T_{msti}(G)}} (deg_{msti}(v)-1)\cdot w(v) \\ &\le (\varDelta-1)\cdot\left(\sum_{e\in E(T_{msti}(G))} w(e) + \sum_{v\in I_{T_{msti}(G)}} w(v)\right) = (\varDelta-1)\cdot msti(G)\ . \end{aligned}$$

□

7 Conclusion

We have presented the first approximation algorithms for the NP-complete problem minimum spanning tree with weighted inner nodes. Reducing the problem to an uncapacitated facility location and a node weighted Steiner tree problem gave us a $(2.35\ln n)$-approximation algorithm. In the metric space, the algorithm can get a factor 3.52 approximation solution. Using a slightly different algorithm for facility location, we obtain a 3.106-approximation algorithm. The lower bound of 1.463 is also proved for the approximation factor. Then we improved the approximation factor to $2(H_n - 1)$ in general case by using a greedy algorithm that repeatedly chooses minimum ratio stars and contracts them into single nodes. The computation of a minimum ratio star can be done efficiently. We believe that the factor $2(H_n - 1)$ is a tight bound of our approximation algorithm.

8 Acknowledgements

We thank Prof M. Mahdian for providing the data about the tradeoff between γ_f and γ_c and helpful suggestions.

References

1. T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* The MIT Press, Cambridge, MA, and London, England, 5. edition, 1990.
2. M. R. Garey and D. S. Johnson. *Computers and Intractability — A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, New York, 1979.
3. S. Guha and S. Khuller. Approximation algorithms for connected dominating sets. *Algorithmica*, 20:347–387, 1998.
4. S. Guha and S. Khuller. Improved methods for approximating node weighted Steiner trees and connected dominating sets. *Information and Computation*, 150:57–74, 1999.
5. P. Klein and R. Ravi. A nearly best-possible approximation algorithm for node-weighted Steiner trees. *Journal of Algorithms*, 19(1):104–115, 1995.
6. M. Mahdian, Y. Ye, and J. Zhang. Improved approximation algorithms for metric facility location problems In *Proceedings of the 5th International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX), LNCS 2462*, pages 229-242, 2002.
7. N. Young. k-medians, facility location, and the Chernoff-Wald bound. In *Proceedings of the 11th ACM-SIAM Symposium on Discrete Algorithms (SODA'00)*, pages 86–95, 2000.
8. M. Mahdian. Private communication, 2005.
9. K. Jain, M. Mahdian, and A. Saberi. A new greedy approach for facility location problems. In *Proceedings of the 34th ACM Symposium on the Theory of Computation (STOC'02)*, pages 731–740, 2002.
10. S. Guha and S. Khuller. Greedy strikes back: Improved facility location algorithms. *Journal of Algorithms*, 31:228–248, 1999.
11. U. Feige. A threshold of $\ln n$ for approximating set-cover. *Journal of the ACM*, 45(4):634–652, 1998.