Geometric Minimum Diameter Minimum Cost Spanning Tree Problem[⋆]

Dae Young Seo[1], D.T. Lee[2,⋆⋆], and Tien-Ching Lin[2]

[1] Department of Computer Engineering, Korea Polytechnic University, Korea
seody@kpu.ac.kr

[2] Institute of Information Science, Academia Sinica, Taipei, Taiwan
{dtlee,kero}@iis.sinica.edu.tw

[⋆⋆]Dept. of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan

Abstract. In this paper we consider bi-criteria geometric optimization problems, in particular, the minimum diameter minimum cost spanning tree problem and the minimum radius minimum cost spanning tree problem for a set of points in the plane. The former problem is to construct a minimum diameter spanning tree among all possible minimum cost spanning trees, while the latter is to construct a minimum radius spanning tree among all possible minimum cost spanning trees. The graph-theoretic minimum diameter minimum cost spanning tree (MDMCST) problem and the minimum radius minimum cost spanning tree (MRMCST) problem have been shown to be NP-hard. We will show that the geometric version of these two problems, GMDMCST problem and GMRMCST problem are also NP-hard. We also give two heuristic algorithms, one MCST-based and the other MDST-based for the GMDMCST problem and present some experimental results.

1 Introduction

Given a connected, undirected and weighted graph $G = (V, E)$ where each edge is associated with a nonnegative real number, referred to as the *cost*, a spanning tree of G whose total cost is minimum among all possible spanning trees is called a *minimum cost spanning tree* (MCST) of G or simply *minimum spanning tree* (MST). There are two algorithms commonly used, Prim's algorithm [8] and Kruskal's algorithm [7] for constructing a minimum spanning tree. Note that the MST may not be unique if two or more edges have the same cost. Given a tree T in which each edge has a cost representing the length of the edge, the *eccentricity* of a vertex v in T is defined to be the total length of the path from v to the farthest vertex u_v in T, denoted by $ecc(v)$. The longest path from vertex v to u_v in T whose $ecc(v)$ is the maximum among all v in T is referred to as the *diameter* of T, and the total cost of the diameter of T is denoted

[⋆] Research supported in part by the National Science Council under the Grants No. NSC-94-2213-E-001-004, NSC-95-2221-E-001-016-MY3, and NSC 94-2752-E-002-005-PAE, and by the Taiwan Information Security Center (TWISC), National Science Council under the Grant No. NSC94-3114-P-001-001-Y.

$D(T)$. Note that there exist at least two extreme vertices in T such that their eccentricity is the maximum. The vertex v that attains the minimum $ecc(v)$ for all v in T is called the *center* of T, and the minimum $ecc(v)$ is referred to as the *radius* of T, denoted $R(T)$. If the costs of edges of T are positive, then either we have a unique center vertex or at most two vertices for which the eccentricity is the minimum.

A spanning tree of a graph $G(V, E)$ that minimizes its diameter is called the *minimum diameter spanning tree*. The geometric version of this problem (GMDST) is defined as follows. Given a set P of n points in the plane in which the cost of an edge connecting any two points is the Euclidean distance between them, find a spanning tree of P such that its diameter is the minimum among all possible spanning trees. Given a graph $G(V, E)$ and a source vertex or *center* $v \in V$, the spanning tree that minimizes $ecc(v)$ is called the *minimum radius spanning tree*. The geometric version of this problem (GMRST) is defined similarly. In [5], Ho, et al. described an algorithm for finding the GMDST of a set of n points in $O(n^3)$ time. They proved that there always exists a GMDST such that it either has a *center point*, called *monopole*, and the rest of points are directly connected to the center point, or has a *center edge* with two points, called *dipole*, and the rest of the points are connected to one of the endpoints of the center edge. They further showed that the above results can be extended to any graph in which the edge costs satisfy the triangle inequality. In [3], Hassin and Tamir showed that the graph-theoretic version of the minimum diameter spanning tree (MDST) problem, where the edge costs do not necessarily satisfy the triangle inequality, is reducible to the *absolute 1-center problem* introduced by Hakimi [2]. The *absolute 1-center problem* can be solved in $O(mn + n^2 logn)$ time [6], where m and n denote respectively the numbers of edges and of vertices of G. Chan [10] improved the bound of GMDST problem in d-dimensional space R^d. He described a semi-online model that computes GMDST of an n-point set $P \subset R^d$ within $\tilde{O}(n^{3-\frac{1}{(d+1)(d/2+1)}})$ time by maintaining a dynamic data structure. (The $\tilde{O}$ notation hides factors that are $o(n^\epsilon)$ for any fixed $\epsilon > 0$.) In other words, the GMDST of an n-point set P in the plane can be found within $\tilde{O}(n^{17/6})$ time. The time bound of the GMDST problem is still very close to the cubic time.

In this paper we consider the geometric versions of these two related bi-criteria problems, the MDMCST problem and the MRMCST problem. The MDMCST problem is to construct a MDST among all possible MCSTs and the MRMCST problem is to construct for a given center a MRST among all possible MCSTs. The graph-theoretic version of both problems have been shown to be NP-hard [4,5]. We will show that the geometric versions of both minimum diameter minimum cost spanning (GMDMCST) problem and minimum radius minimum cost spanning (GMRMCST) problem are NP-hard. We then give two heuristic algorithms, called MCST-based algorithm and MDST-based algorithm for the GMDMCST problem and present some experimental results.

The rest of the paper is organized as follows. In Section 2 we show that the GMDMCST and GMRMCST problems are NP-hard. In Section 3 we give MCST-based and MDST-based heuristic algorithms for the GMDMCST problem. In Section 4 we give the implementation results of these two kinds of heuristic algorithm on the *Algorithm Benchmark System* (ABS) [11] and conclude with an analysis and some remarks.

2 NP-Hardness of the GMDMCST and GMRMCST

Given a connected undirected graph $G(V, E)$ with nonnegative edge costs, we consider the set of all minimum cost spanning trees of G. In the following we will simply refer to minimum cost spanning tree as minimum spanning tree, when the cost measure is understood. In [9], Seo defined the *minimum spanning tree intersection graph* (MSTIG) which is the intersection of all the MST's, and the *minimum spanning tree union graph* (MSTUG) which is the union of all the MST's. The edges in MSTIG are called the *essential edges* and those in MSTUG but not in MSTIG are called the *optional edges.*

Given a set P of n points in the plane in which the cost of an edge connecting any two points is the Euclidean distance between them, Seo [9] gave an algorithm that runs in $O(n^2 \log n)$ time for finding the MSTIG(G) and MSTUG(G) and coloring all edges in G according the following coloring rule. The essential edges are colored blue, the optional edges are colored green and the edges not in the MSTUG are colored red. Thus all the edges of a graph G will be partitioned into three classes, each assigned a distinct color. The MSTUG will be a connected graph, and the MSTIG will be a forest unless the MST is unique. The subtrees in the forest of MSTIG that consist solely of essential edges are called *blue trees.*

In this section, we show that the decision version of the optimization GMRMCST problem, the so-called the *geometric bounded radius bounded cost spanning tree* (GBRBCST) problem, is NP-complete. We first show that the PARTITION problem, a well-known NP-complete problem [1], is polynomially reducible to the GBRBCST problem, and then show that the GBRBCST problem is polynomially reducible to the GBDBCST problem, and finally show that the GBRBCST and GBDBCST problems are respectively polynomially reducible to the GMRMCST and GMDMCST problems.

The PARTITION problem is defined as follows. Given a finite set W and a size $s(w) \in \mathbf{Z}^+$ for each $w \in W$, decide if there exists a subset $W' \subseteq W$ such that $\sum_{w \in W'} s(w) = \sum_{w \in W-W'} s(w)$?

The GBRBCST is defined as follows. Given a point set P in the plane, a center s and two positive values R and C, decide if there exists a Euclidean spanning tree T such that the distance, $ecc(s)$, from s to the farthest site along the tree is bounded above by R and that the total cost of the tree is bounded above by C?

Theorem 1. *The GBRBCST problem is NP-complete.*

Proof. GBRBCST is obviously in NP. For simplicity we assume that an instance of PARTITION is a multiset of positive integers (with repetition permitted) sorted in nondecreasing order, i.e., $w_1, w_2, \ldots, w_n$ where $w_i \leq w_{i+1}$ for $i = 1, ..., n-1$. We shall construct an instance of GBRBCST based on the input instance of PARTITION. The construction (see Fig. 1) is as follows. Let $ISqr^+(k)$

and $ISqr^-(k)$ be imaginary squares corresponding to w_k such that each side is of length $2w_k$. The top right (TR) corner of $ISqr^+(i)$ and the bottom left(BL) corner of $ISqr^+(i+1)$ are coincident, while BL($ISqr^-(i)$) and TR($ISqr^-(i+1)$) are coincident. Suppose that BL($ISqr^+(1)$) and TR($ISqr^-(1)$) are located at the origin s. For $k > 1$, BL($ISqr^+(k)$) = TR($ISqr^+(k-1)$) has coordinates $(2\sum_{i=1}^{k-1} w_i, 2\sum_{i=1}^{k-1} w_i)$, while TR($ISqr^-(k)$) = BL($ISqr^-(k-1)$) has coordinates $(-2\sum_{i=1}^{k-1} w_i, -2\sum_{i=1}^{k-1} w_i)$. The bottom and left sides of each $ISqr^+(k)$ or each $ISqr^-(k)$ are parallel to $x-$ and $y-$coordinate axes, respectively. Figure 2 shows a more detailed positioning of the points in a square of dimension $w \times w$. There are three types of points in our construction.

1. For each pair of imaginary squares $ISqr^+(k)$ and $ISqr^-(k)$, there is one basic TYPE1 point p at $(2\sum_{i=1}^{k} w_i, -2\sum_{i=1}^{k} w_i)$ shown in Fig. 2 on a *matching path* connecting $ISqr^+(k)$ and $ISqr^-(k)$, where the matching path consists of an upward vertical line from p to meet the right side of $ISqr^+(k)$ and a leftward horizontal line from p to meet the bottom side of $ISqr^-(k)$. Extraneous TYPE1 points on this matching path are added so that they equally divide the vertical and horizontal line segments $\overline{h,p}$ and $\overline{h',p}$ in Fig. 2, into $\lfloor\frac{(4\sum_{i=1}^{k} w_i)-w_k}{0.5w_k}\rfloor + 1$ edges, respectively. The divided edges will be of length strictly less than $0.5w_k$. Since w_i's are sorted in nondecreasing order, the number of TYPE1 extra points for each k is bounded by $O(n)$. Hence there will be $O(n^2)$ TYPE1 points.
2. Now consider TYPE2 points. For each ISqr, there will be 10 basic points (e.g. a,b,c,d,e,f,g,h,i and j for $ISqr^+(k)$ in Fig. 2) and 10 extra dividing points (e.g. in the middle of 5 edges (c,d),(d,e),(e,f),(g,j) and (i,j), 3 points equally dividing (a,b), and 2 points equally dividing (b,c), as shown). These dividing points are introduced to ensure that the distance of any divided segment in the corresponding ISqr is strictly less than $0.5w_k$. Similarly we have 20 points for $ISqr^-(k)$. Note that the distances between points c and h, between points h and i, and between points f and g in $ISqr^+(k)$ (or correspondingly those between points c' and h', between points h' and i', and between points f' and g' in $ISqr^-(k)$) are $0.5w_k$. Figure 2 shows the x and y-coordinates of basic TYPE2 points in $ISqr^+(k)$ and $ISqr^-(k)$; the subscript k is omitted from the figure. Note that points f and g are shifted so that the distance between points f and i is strictly greater than $0.5w_k$. Points f' and g' are shifted similarly.
3. Finally there are TYPE3 points, t_1, t_2, t_3, t_4 and other auxiliary points. Points t_1 and t_2 have x- and y- coordinates $((2+4.5\sqrt{2})\sum_{i=1}^{n} w_i, (2+4.5\sqrt{2})\sum_{i=1}^{n} w_i)$ and $(-(2+4.5\sqrt{2})\sum_{i=1}^{n} w_i, -(2+4.5\sqrt{2})\sum_{i=1}^{n} w_i)$, respectively. The distances between points j_n and t_1, and between points j'_n and t_2 are equal to $9\sum_{i=1}^{n} w_i$. Note that this distance is longer than the length of any matching path. The length of the matching path (h_k, p_k, h'_k) for $ISqr(k)$ is $8\sum_{i=1}^{k} w_i - 2w_k$. The shortest distance from s to t_1 (and to t_2) in a minimum spanning tree is $13\sum_{i=1}^{n} w_i$. Points t_3 and t_4 form an isosceles triangle with the apex s. The line segments $\overline{s,t_3}$ and $\overline{s,t_4}$ are of length $13.5\sum_{i=1}^{n} w_i$.

The slopes of $\overline{s,t_3}$ and $\overline{s,t_4}$ are -4 and -1/4, respectively. The auxiliary points on $\overline{s,t_3}$ are those with y-coordinates $(2\sum_{i=1}^{k} w_i)$ and $(2\sum_{i=1}^{k} w_i - w_k)$ for $1 \leq k \leq n$. On the other hand, the auxiliary points on $\overline{s,t_4}$ are those with x-coordinates $(-2\sum_{i=1}^{k} w_i)$ and $(-2\sum_{i=1}^{k} w_i + w_k)$. The slopes of $\overline{s,t_3}$ and $\overline{s,t_4}$ are selected so that the closest point from any auxiliary point must be s, t_3, t_4, or another auxiliary point on the same slope.

Let C be equal to the cost of the minimum spanning tree and R be equal to

$$13\sum_{i=1}^{n} w_i + \frac{1}{2}S = 13.5\sum_{i=1}^{n} w_i.$$

We now show that the instance of PARTITION has a solution if and only if the constructed instance of the GBRBCST problem has answer YES.

Let the solution be denoted by T^*. Note that all the edges shown as solid lines in Fig. 1 must belong to T^*, i.e., they are essential edges, and that those shown as dashed lines may or may not belong to T^*, i.e., they are optional edges. Let L_1 and L_2 denote the path length from s to t_1 and from s to t_2 in T^*, respectively. An imaginary square is said to be *straight*, *crooked*, or *bad* if it satisfies the following constraints. If both (c_ℓ, h_ℓ) and (h_ℓ, i_ℓ) of $\mathit{ISqr}^+(\ell)$ (or correspondingly (c'_ℓ, h'_ℓ) and (h'_ℓ, i'_ℓ) of $\mathit{ISqr}^-(\ell)$) are in T^*, then $\mathit{ISqr}^+(\ell)$ (or correspondingly $\mathit{ISqr}^-(\ell)$) is said to be *straight*. If only (f_ℓ, g_ℓ) of $\mathit{ISqr}^+(\ell)$ (or correspondingly (f'_ℓ, g'_ℓ) of $\mathit{ISqr}^-(\ell)$) is in T^*, then $\mathit{ISqr}^+(\ell)$ (resp. $\mathit{ISqr}^-(\ell)$) is said to be *crooked*. An imaginary square which is neither straight nor crooked is said to be *bad*. By Lemma 5, we can obtain from T^* a solution to the PARTITION problem by assigning w_ℓ to W' if $\mathit{ISqr}^+(\ell)$ is crooked and assigning w_ℓ to $W - W'$ if $\mathit{ISqr}^+(\ell)$ is straight, for $\ell = 1, 2, \ldots, n$. This completes the proof that the GBRBCST problem is NP-complete.

According to the construction of an instance of GBRBCST above, we have the following lemmas. Due to the page limit, some proofs are omited.

Lemma 1. $L_1 + L_2 \leq 27\sum_{i=1}^{n} w_i$.

Lemma 2. *Each pair of imaginary squares $\mathit{ISqr}^+(\ell)$ and $\mathit{ISqr}^-(\ell)$ contributes*
$8w_\ell$ to $L_1 + L_2$ if both of them are straight,
$9w_\ell$ to $L_1 + L_2$ if one of them is straight and the other one is crooked,
$10w_\ell$ to $L_1 + L_2$ if both of them are crooked, and
$\geq 14w_\ell$ $L_1 + L_2$ if at least one of them is bad.

Lemma 3. *For each pair of imaginary squares $\mathit{ISqr}^+(\ell)$ and $\mathit{ISqr}^-(\ell)$, at least two and at most four of the six optional edges (c_ℓ, h_ℓ), (h_ℓ, i_ℓ), (f_ℓ, g_ℓ), (c'_ℓ, h'_ℓ), (h'_ℓ, i'_ℓ) and (f'_ℓ, g'_ℓ) are in T^*.*

Lemma 4. *For each pair of imaginary squares $\mathit{ISqr}^+(\ell)$ and $\mathit{ISqr}^-(\ell)$, at most one of them is straight.*

Lemma 5. *For each pair of imaginary squares $\mathit{ISqr}^+(\ell)$ and $\mathit{ISqr}^-(\ell)$, exactly one of them is straight and the other one is crooked.*

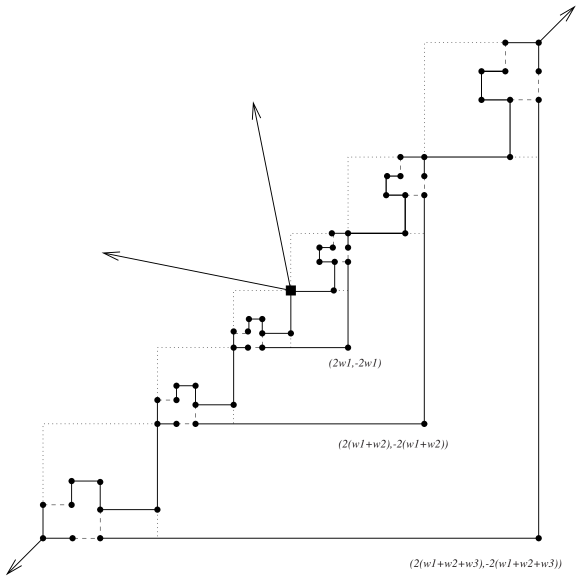


Fig. 1. MSTUG/MSTIG constructed from an instance of PARTITION

Lemma 6. *Given a graph G, suppose that the optional edges in MSTUG(G) are partitioned into l_{max} equivalence classes K_1, K_2, ..., $K_{l_{max}}$ such that the edges in the same class have an identical weight and the weight of an edge in K_i is strictly less than that in K_j if and only if $i < j$. Let $NO_i(T)$ denote the number of optional edges in K_i that are contained in the MST T. Then, for any pair of MST's T and T', $NO_i(T) = NO_i(T')$ for $1 \leq i \leq l_{max}$.*

Theorem 2. *The GBDBCST problem is NP-complete.*

Proof. Let $D = 2R$. Then the previous construction can be used to prove that the GBRBCST problem is polynomially reducible to the GBDBCST problem.

Theorem 3. *The GMRMCST and the GMDMCST problems are NP-hard.*

Proof. It is obvious that the GBRBCST problem and the GBDBCST problem are polynomially reducible to the GMRMCST problem and the GMDMCST problem, respectively.

3 Heuristic Algorithms for GMDMCST

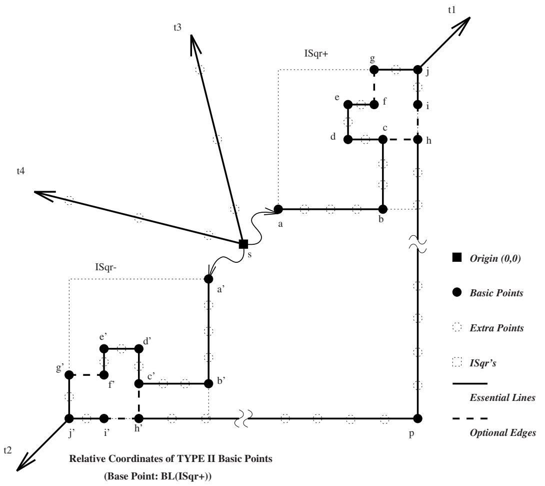
In this section, we assume that MSTIG(G) and MSTUG(G) have been computed and will be used as the input of our heuristic algorithms [9]. Let BT_i denote a blue-tree in MSTIG(G). We define the *pseudo-center*, denoted *pc*(BT_i), of BT_i to be the point on an edge or a vertex in BT_i such that the distances from the two extremes of the diameter $D(BT_i)$ to $pc(BT_i)$ are the same. It is trivial to see that $ecc(pc(BT_i)) = D(BT_i)/2$.

The crux of this problem lies in the selection of optional edges to be included in GMDMCST so that the resulting diameter of the MST is minimized. We shall adopt different strategies in our heuristic algorithms to select optional edges. The following properties and operations are needed for our heuristics. For each blue tree BT_i, we will maintain diameter $D(BT_i)$, pseudo-center $pc(BT_i)$, and the distance, denoted $\ell(v)$, from v to $pc(BT_i)$ for each vertex v in BT_i. When we select an optional edge $e = (u, v)$ concatenating two blue trees, BT_i and BT_j, we will create a new blue tree $BT_{i,j}$, which is the union of BT_i, BT_j and the optional edge e. $D(BT_{i,j}) = \max\{D(BT_i), D(BT_j), c(e) + \ell(u) + \ell(v) + ecc(pc(BT_i)) + ecc(pc(BT_j))\}$, where $c(e)$ denotes the cost of edge e.

We now start to develop our heuristic algorithms for GMDMCST. Both kinds of our heuristic algorithm for GMDMCST use the greedy method. The general idea is as follows: We greedily select an optional edge concatenating two blue trees so that the new blue tree has a minimum diameter until all the blue trees are connected to become a single MST. Note that we can easily modify our heuristic algorithm for GMDMCST to become a heuristic algorithm for GMRMCST by using a different greedy criterion which is to greedily select an optional edge concatenating two blue trees so that the new blue tree has a minimum radius.

3.1 The MST-Based Heuristic

In this subsection, we use the ideas of the Prim's and Kruskal's MST algorithm to develop our MST-based heuristic algorithm.



Relative Coordinates of TYPE II Basic Points

(Base Point: BL(ISqr+))

a: (0,0)	*b:(1.5w,0)*	*c:(1.5w,w)*	*d:(w,w)*	*e:(w,1.5w)*
f:(1.375w,1.5w)	*g:(1.375w,2w)*	*h:(2w,w)*	*i:(2w,1.5w)*	*j:(2w,2w)*

(Base Point: TR(ISqr-))

a':(0,0)	*b':(0,-1.5w)*	*c':(-w,-1.5w)*	*d':(-w,-w)*	*e':(-1.5w,-w)*
f':(-1.5w,-1.375w)	*g':(-2w,-1.375w)*	*h':(-w,-2w)*	*i'(-1.5w,-2w)*	*j':(-2w,-2w)*

Fig. 2. The detailed picture for ISqr's in Figure 1

3.1.1 Prim-Like Heuristic

We start with a randomly selected blue tree BT_s among all blue trees in MSTIG(G) and consider the optional edges that are incident to BT_s with the minimum cost. Let $e = (u, v)$ be one of the optional edges with the minimum cost that is incident to BT_s, where $u \in BT_s$ and $v \in BT_j$ for some $j \neq s$.

$$D(BT_s) = \max\{D(BT_s), D(BT_j), c(e) + \ell(u) + \ell(v) + (D(BT_s) + D(BT_j))/2\}.$$

We greedily select among all optional edges with the minimum cost that are incident to BT_s the optional edge e that gives the minimum diameter according to the formula above. After selecting this optional edge e, we obtain a new blue tree BT_s by concatenating these two blue trees.

3.1.2 Kruskal-Like Heuristic

Let $e = (u, v)$ be one of the optional edges with the minimum cost, where $u \in BT_i$ and $v \in BT_j$. We will concatenate BT_i and BT_j to be a new blue tree for which $\delta(e)$ is minimum, where $\delta(e) = D(BT_{i,j}) - (D(BT_i) + D(BT_j))$.

3.2 The MDST-Based Heuristic

3.2.1 k-Center Heuristic

In [5], Ho et al. proved that there exists a GMDST of a set of n points which is monopolar or dipolar. The idea of k-center heuristic is to extend monopolar and dipolar spanning trees to k-polar. Let b be the total number of blue trees in MSTIG(G). If each combination contains k distinct blue trees in MSTIG(G), it has total $C(b, k)$ distinsct combinations. For each combination π of k blue trees $BT_{\pi(1)}, BT_{\pi(2)}, \ldots, BT_{\pi(k)}$, we will partition the set of remaining blue trees into k subsets $S_1, S_2, \ldots, S_k$ by some greedy criteria, connect each blue tree in S_i to $BT_{\pi(i)}$ for each $1 \leq i \leq k$ and then connect $BT_{\pi(1)}, BT_{\pi(2)}, \ldots, BT_{\pi(k)}$ by Kruskal-like heuristic to obtain a minimum spanning tree BT_π. The k-center heuristic will consider all combinations of k blue trees as possible k-poles, and select one such that its resulting minimum spanning tree has the minimum diameter.

Note that we can apply the Prim-like heuristic algorithm to implement the 1-center heuristic algorithm. We can implement the 1-center heuristic by calling the Prim-like heuristic b times, each starts with blue tree BT_i, where $1 \leq i \leq b$, and then select the MST with the minimum diameter. The k-center heuristic can also be viewed as a combination of Prim-like heuristic and Kruskal-like heuristic. For each combination π of k blue trees $BT_{\pi(1)}, BT_{\pi(2)}, \ldots, BT_{\pi(k)}$, we first connect all blue trees in S_i to $BT_{\pi(i)}$ by Prim-like heuristic and then connect blue trees $BT_{\pi(1)}, BT_{\pi(2)}, \ldots, BT_{\pi(k)}$ by Kruskal-like heuristic.

4 Implementation

The MST-based heuristic algorithm and the MDST-based heuristic algorithm have been implemented on the ABS. Tables 1 and 2 summarize the experimental

Table 1. The experimental results for MST-based heuristic algorithms

				MST_PRIM		*MST_KRUSCAL*	
n	m	m_{opt}	m_{BT}	avg. d.	avg. t.	avg. d.	avg. t.
50	50	3.05	2.50	1265.028	0.0002	1263.837	0.0002
100	50	10.06	5.93	1437.457	0.0002	1431.074	0.0002
150	50	21.77	11.60	1509.909	0.0006	1496.996	0.0002
200	50	33.78	17.41	1584.962	0.0008	1555.767	0.0004
250	50	50.17	25.09	1646.061	0.0007	1596.836	0.0003
300	50	65.91	32.60	1639.306	0.0006	1608.231	0.0004
350	50	82.21	40.70	1693.195	0.0011	1631.054	0.0007
400	50	102.79	51.16	1655.356	0.0005	1598.080	0.0011

Table 2. The experimental results for MDST-based heuristic algorithms

				MDST_1C		*MDST_2C*		*MDST_3C*		*MDST_4C*		*MDST_5C*	
n	m	m_{opt}	m_{BT}	avg. d.	avg. t.	avg. d.	avg. t.	avg. d.	avg. t.	avg. d.	avg. t.	avg. d.	avg. t.
50	50	3.05	2.50	1262.6	0.0001	1262.0	0.0001	1262.4	0.0001	1262.4	0.0001	1263.3	0.0002
100	50	10.06	5.93	1424.4	0.0006	1423.7	0.0011	1424.3	0.0011	1426.4	0.0015	1428.6	0.0018
150	50	21.77	11.60	1463.8	0.0009	1459.3	0.0044	1457.2	0.0135	1456.2	0.0381	1454.8	0.0847
200	50	33.78	17.41	1527.0	0.0016	1520.9	0.0142	1507.8	0.0785	1505.2	0.3383	1504.6	1.1756
250	50	50.17	25.09	1560.5	0.0048	1537.1	0.0446	1520.9	0.3563	1509.9	2.1918	1504.7	10.7995
300	50	65.91	32.60	1530.0	0.0077	1513.1	0.1066	1487.7	1.1158	1474.2	9.0257	1466.9	59.0691
350	50	82.21	40.70	1553.0	0.0127	1512.0	0.2253	1486.4	2.9002	1467.4	28.6890	1454.2	230.5542
400	50	102.79	51.16	1534.0	0.0217	1495.7	0.4741	1464.2	7.5041	1444.8	91.5174	1425.9	516.9337

results. Columns 1 - 2 indicate the total number of input vertices (n) and the size of grid ($m \times m$) respectively. Columns 3 - 4 contain the average number of optional edges (m_{opt}) and the average number of the blue trees (m_{BT}) for the input instances. The other columns contain the average diameter (avg. d.) and average running times (avg. t.) of the Prim-like heuristic algorithm (MST_PRIM), the Kruskal-like heuristic algorithm (MST_KRUSCAL) and the k-center heuristic algorithms (1-center (MST_1C), 2-center (MST_2C) , 3-center (MST_3C), 4-center (MST_4C), 5-center (MST_5C)), respectively. We generate the test instances with input size from $50, 100, 150, \ldots, 400$ vertices on a 50×50 grid. For each input size we randomly generate 1000 test instances and then execute all heuristic algorithms for every test instance. In general, we see the MDST-based heuristics are better than the MST-based heuristics, but they spend more time. As far as the MST-based heuristics are concerned, the Kruskal-like heuristic is better than the Prim-like heuristic. We also see that the 2-center heuristic obtains the smallest diameter for smaller input size 50 and 100, and the 5-center heuristic yields the smallest diameter spanning tree when the input size is larger than or equal to 150. It seems that the number of centers in an optimal MDMCST increases as the input size (or the size of blue trees) increases.

5 Conclusion

In this paper, we have considered two related bi-criteria problems GMDMCST and GMRMCST and shown that they both are NP-hard. We have presented two kinds of heuristic algorithms for the GMDMCST problem, called MST-based algorithm and MDST-based algorithm. The MST-based heuristic algorithm includes Prim-like heuristic algorithm and Kruskal-like heuristic algorithm. We have also introduced a k-center heuristic algorithm, which is an MDST-based heuristic algorithm. The time complexity of the Prim-like heuristic algorithm and the Kruskal-like heuristic algorithm both are $\mathrm{O}(n^2 + m_{opt}m_{BT})$, where n is the number of vertices, m_{BT} is the number of blue trees and m_{opt} is the number of the optional edges. The time complexity of k-center heuristic algorithm is $O(n^k(n^2 + m_{opt}m_{BT} + m_{opt}\log m_{opt}))$.

So far, we have not been able to find any approximation algorithm for the GMDMCST. This is an interesting problem for further research. Our benchmark results give us a hint to find an approximation algorithm by approximating the number of centers in an optimal MDMCST.

References

1. Garey, M.R., Johnson, D.S.: Computers and Intractability: a guide to the theory of NP-completeness. Freeman, San Francisco (1979)
2. Hakimi, S.L.: Optimal locations of switching centers and medians of a graph. Operations Research 12, 405–459 (1964)
3. Hassin, R., Tamir, A.: On the minimum diameter spanning tree problem. Information Processing Letters 53, 109–111 (1995)
4. Ho, J.M.: Optimal Trees in Network Design, Ph.D Dissertation, Northwestern University (May 1989)
5. Ho, J.M., Lee, D.T., Chang, C.H., Wong, C.K.: Minimum Diameter Spanning Trees and Related Problems. SICOMP 20(5), 987–997 (1991)
6. Kariv, O., Hakimi, S.L.: An algorithmic approach to network location problems. I: The p-Centers. SIAM Journal on Applied Math. 37, 513–537 (1979)
7. Kruskal, J.B.: On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. Problem. Amer. Math. Soc. 7 (1956)
8. Prim, R.C.: Shortest Connection Networks and Some Generalizations. Bell System Tech. J. (1957)
9. Seo, D.Y.: On the Complexity of Bicriteria Spanning Tree Problems for a Set of Points in the Plane, Ph. D Dissertation, Northwestern University (1999)
10. Chan, T.M.: Semi-online maintenance of geometric optima and measures. SIAM Journal on Computing 32, 700–716 (2003)
11. Algorithm Benchmark System (ABS), `http://www.opencps.org`