Approximation Algorithms for Generalized MST and TSP in Grid Clusters

Binay Bhattacharya[1], Ante Ćustić[1], Akbar Rafiey[1](✉), Arash Rafiey[1,2], and Vladyslav Sokol[1]

[1] Simon Fraser University, Burnaby, Canada
{binay,acustic,arafiey,arashr,vsokol}@sfu.ca
[2] Indiana State University, Terre Haute, IN, USA
arash.rafiey@indstate.edu

Abstract. We consider a special case of the generalized minimum spanning tree problem (GMST) and the generalized travelling salesman problem (GTSP) where we are given a set of points inside the integer grid (in Euclidean plane) where each grid cell is 1×1. In the MST version of the problem, the goal is to find a minimum tree that contains exactly one point from each non-empty grid cell (cluster). Similarly, in the TSP version of the problem, the goal is to find a minimum weight cycle containing one point from each non-empty grid cell. We give a $(1 + 4\sqrt{2} + \epsilon)$ and $(1.5 + 8\sqrt{2} + \epsilon)$-approximation algorithm for these two problems in the described setting, respectively.

Our motivation is based on the problem posed in [6] for a constant approximation algorithm. The authors designed a PTAS for the more special case of the GMST where non-empty cells are connected end dense enough. However, their algorithm heavily relies on this connectivity restriction and is unpractical. Our results develop the topic further.

Keywords: Generalized minimum spanning tree · Generalized travelling salesman · Grid clusters · Approximation algorithm

1 Introduction

The *generalized minimum spanning tree problem* (GMST) is a generalization of the well known *minimum spanning tree problem* (MST). An instance of the GMST is given by an undirected graph $G = (V, E)$ where the vertex set is partitioned into k *clusters* V_i, $i = 1, \dots, k$, and a weight $w(e) \in \mathbb{R}^+$ is assigned to every edge $e \in E$. The goal is to find a tree with minimum weight containing one vertex from each cluster.

The GMST occurs in telecommunications network planning, where a network of node clusters need to be connected via a tree architecture using exactly one node per cluster [9]. More precisely, local subnetworks must be interconnected by a global network containing a gateway from each subnetwork. For this

Supported by NSERC Canada.

inter-networking, a point has to be chosen in each local network as a hub and the hub point must be connected via transmission links such as optical fiber, see [14]. Furthermore, the GMST has some applications in design of backbones in large communication networks, energy distribution, and agricultural irrigation [10].

The GMST was first introduced by Myung, Lee and Tcha in 1995 [14]. Although MST is polynomially solvable [7], it was shown in [14] that the GMST is strongly NP-hard and there is no constant factor approximation algorithm, unless P = NP. However, several heuristic algorithms have been suggested for the GMST, see [9,10,16,17]. Furthermore, Pop, Still and Kern [18] used an LP-relaxation to develop a $2\rho-$approximation algorithm for the GMST where the size of every cluster is bounded by ρ.

In [6], Feremans, Grigoriev and Sitters consider the *geometric generalized minimum spanning tree problem in grid clusters*, GGMST for short. In this special case of the GMST, a complete graph $G = (V, E)$ is given where the set of vertices V correspond to a set of points in the planar integer grid. Every non-empty 1×1 cell of the grid forms a cluster. The weight of the edge between two vertices is given by their Euclidean distance. Figure 1 depicts one instance of the GGMST.

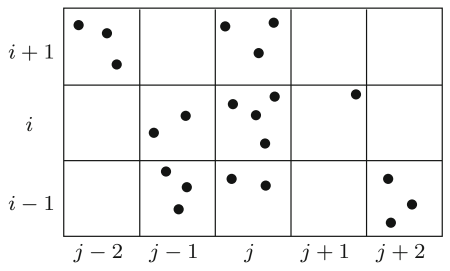

Fig. 1. An GGMST instance with $n = 21$ points and $N + 1 = 8$ non-empty cells, which are connected and fit into a 3×5 sub-grid

We say that two grid cells are connected if they share a side or a corner. Furthermore, we say that a set of grid cells is connected if they form one connected component. The authors in [6] show that the GGMST is strongly NP-hard, even if we restrict to instances in which non-empty grid cells are connected and each grid cell contains at most two points. Furthermore, they designed a dynamic programming algorithm that solves in $\mathcal{O}(l\rho^{6k}2^{34k^2}k^2)$ time the GGMST for which the set of non-empty grid cells is connected and fits into $k \times l$ sub-grid. (Note that the algorithm is polynomial if k is bounded.) Moreover, the authors used this algorithm to develop a polynomial time approximation scheme (PTAS) for the GGMST for which non-empty cells are connected and the number of non-empty cells is superlinear in k and l. The GGMST instances are often used to test heuristics for the GMST which, in light of the results in [6], is not adequate. The objective of this paper is to develop this topic further and to design a simple approximation algorithms for the GGMST and of its variants without restricting only to connected and dense instances.

Analogously as the GMST and the GGMST, the *generalized travelling salesman problem* (GTSP) and the *geometric generalized travelling salesman problem in grid clusters* (GGTSP) can be defined. The GTSP was introduced by Henry-Labordere [11] and is also known in the literature as *set TSP*, *group TSP* or *One-of-a-Set TSP*. This problem has many applications, including airplane routing, computer file sequencing, and postal delivery, see [2,12,13]. Elbassioni, Fishkin, Mustafa and Sitters [5] considered the GTSP in which non-empty clusters (i.e. regions) are disjoint α-fat objects with possibly varying size. In this setting they obtained a $(9.1\alpha + 1)$-approximation algorithm. They also give the first $\mathcal{O}(1)$-approximation algorithm for the problem with intersecting clusters (regions). Note that in the GGTSP, fatness of each cluster is 4 (each cluster is a square).

As a special case of the GTSP we can look at each geometric region as an infinite set of points. This problem, called the *TSP with neighbourhood*, was introduced by Arkin and Hassin [1]. In the same paper they present constant factor approximation algorithm for two cases in which the regions are translates of disjoint convex polygons, and for disjoint unit disks. For the general problem Mata and Mitchell [15] and later on Gudmundsson and Levcopoulos [8], gave an $\mathcal{O}(\log n)$-approximation algorithm. For intersecting unit disks an $\mathcal{O}(1)$-approximation algorithm is given in [4]. Safra and Schwartz [19] show that it is NP-hard to approximate the TSP with neighbourhood within $(2 - \epsilon)$. In this context, it is natural to consider the GTSP in which points are sitting inside geometric objects such as the integer grid.

Notation. We will usually refer to vertices as points. Throughout this paper, the number of points ($|V|$) will be denoted by n. Furthermore, N denotes the number of edges in every feasible solution (tree) of the GGMST, i.e. N is the number of non-empty cells minus 1. The edge between two points u and v will be denoted by $e_{u,v}$. We naturally extend the notation for the weight to sets of edges and graphs, i.g. the weight of a tree T is denoted by $w(T) = \sum_{e \in T} w(e)$, where $e \in T$ means that e is an edge of T. We assume that every point is in just one cell, i.e. points on the cell borders are assigned to only one neighbouring cell by any rule. An optimal solution of the GGMST will be denoted by T_{opt} throughout this paper.

Our results and organization of the paper. The main result of this paper is a $(1 + 4\sqrt{2} + \epsilon)$-approximation algorithm for the GGMST. We do not assume any restrictions on connectivity, density or cardinality of non-empty cells. The algorithm is presented and analyzed in Sect. 2. A lower bound for the weight of an optimal solution in terms of N is used to prove the approximation quality of the algorithm. Section 3 is devoted to proving this lower bound. Lastly, in Sect. 4 we use our GGMST algorithm to develop an approximation algorithms for the GGTSP.

2 The GGMST Approximation Algorithm

In this section we present a $(1+4\sqrt{2}+\epsilon)$-approximation algorithm (Algorithm 3) for the GGMST. Main part of the algorithm is Algorithm 1 which we describe next.

Algorithm 1. $\left(1+4\sqrt{2}+\frac{2\sqrt{2}}{w(T_{opt})}\right)$-approximation alg. for the GGMST

```
1  $T \leftarrow$ solution of the MST problem on non-empty cells (where the distance
2      between a pair of cells is the length of the shortest edge between them);
3  $G \leftarrow$ the graph consisting of the set of edges (and points) that correspond to the
4      edges in $T$;
5  for all cells $C$ that contain more than one point from $G$ do
6      $C_G \leftarrow$ the set of points from $G$ that are in $C$;
7      $p \leftarrow$ point from $C$ that is a median for $C_G$;
8      Replace $C_G$ by $p$, i.e. reconnect to $p$ all edges of $G$ that enter $C$;
9  end
10 return $G$;
```

Algorithm 1 is divided into two parts; in the first part we solve an MST instance defined as follows: non-empty cells play the role of vertices, and the weight of the edge between two cells C_1, C_2 is the smallest weight edge e_{p_1,p_2} where $p_1 \in C_1$ and $p_2 \in C_2$. Let T be an optimal tree of such MST instance, and let graph G be the set of edges (with its endpoints) of the original GGMST instance that correspond to the edges of T. Note that G has N edges and spans all non-empty cells but it can have multiple points in some cells. In the second part of the Algorithm 1 (i.e. the **for** loop), we modify G to obtain the GGMST feasibility, by iteratively replacing multiple cell points by a single point p. We choose point p to be the one that has the minimum sum of distances to other points of G that are in the corresponding cell.

Next we present an upper bound for solutions obtained by Algorithm 1 in terms of the number of edges N.

Theorem 1. *Algorithm 1 produces a feasible solution T_A of the GGMST such that $w(T_A) \leq w(T_{opt}) + \sqrt{2}N - \sqrt{2}$, where N is the number of edges of T_A.*

Proof. Denote by G_0 the non-feasible graph obtained in the first part of the algorithm, i.e. the first version of graph G. Then the weight of the solution T_A obtained by the algorithm is equal to $w(G_0) + ext$, where ext is the amount by which we increase (extend) the weight of G_0 in the second part of the algorithm. Note that $w(G_0) \leq w(T_{opt})$, as G_0 is an optimal solution of the problem for which T_{opt} is a feasible solution (find a minimum weight set of edges that spans all non-empty cells, with all GGMST edges being allowed). In the rest of the proof we will bound the value of ext.

In every run of the **for** loop we replace the set of points C_G with p. In doing so, every edge $e_{q,c}$, $c \in C_G$ from G, is replaced by $e_{q,p}$. From the triangle inequality we get that $w(e_{q,p}) - w(e_{q,c}) \leq w(e_{c,p})$. Hence, the increase (extension) of the weight of G in every run of the **for** loop is less or equal than $\sum_{c \in C_G} w(e_{c,p})$. Instead of bounding such absolute values, we will bound its average per edge adjacent to the corresponding cell. More precisely, we will calculate an average extension per *half-edge* assigned to the corresponding cell. Namely, every edge will be extended at most two times, once on each endpoint, so we can look at each extension as an extension of a half-edge. Furthermore, note that edges that contain leafs will be extended only on one side. We will use this fact to assign half-edges that contain leafs to other cells to lower their average half-edge extension. To every cell C, we will assign $|C_G| - 2$ leaf half-edges. Intuitively, we can do this because every node v of a tree *generates* $\deg(v) - 2$ leafs. Formally, it follows from the following well know equality:

$$|V_1| = 2 + \sum_{i \geq 2} |V_i|(i-2), \tag{1}$$

where $V_i = \{v \in V\colon \deg(v) = i\}$, and V is the set of vertices of a graph.

Then for a cell C the average extension per assigned half-edges is bounded above by

$$\frac{\sum_{c \in C_G} w(e_{c,p})}{|C_G| + (|C_G| - 2)}. \tag{2}$$

Note that the maximum distance between two cell points is $\sqrt{2}$. Since points from C_G are candidates for p, it follows that $\sum_{c \in C_G} w(e_{c,p}) \leq \sqrt{2}(|C_G| - 1)$. Hence, (2) is bounded above by

$$\frac{\sqrt{2}(|C_G| - 1)}{2|C_G| - 2} = \frac{\sqrt{2}}{2}.$$

Hence, in average, every half-edge (except 2 leaf half-edges, see (1)) is extended by at most $\sqrt{2}/2$. Note that this average bound is a constant, i.e. does not depend on C. Now *ext* can be bounded by

$$ext \leq \frac{\sqrt{2}}{2}(2N - 2) = \sqrt{2}N - \sqrt{2}. \tag{3}$$

Finally, we can bound the solution T_A of the algorithm by

$$w(T_A) \leq w(G_0) + ext \leq w(T_{opt}) + \sqrt{2}N - \sqrt{2}. \qquad \square$$

The following theorem gives a lower bound for the optimal solution in terms of the number of edges N. Section 3 is dedicated to proving the theorem.

Theorem 2. *If T_{opt} is an optimal solution of the GGMST on $N+1$ non-empty cells, then $N \leq 4w(T_{opt}) + 3$.*

Now from Theorems 1 and 2 the following approximation bound for Algorithm 1 follows.

Corollary 1. *Algorithm 1 produces a feasible solution T_A of the GGMST such that $w(T_A) \leq (1+4\sqrt{2})w(T_{opt}) + 2\sqrt{2}$.*

Note that, due to the constant $2\sqrt{2}$, Corollary 1 does not gives us a constant approximation ratio for Algorithm 1. Namely, the approximation ratio that we get is equal to $1 + 4\sqrt{2} + \frac{2\sqrt{2}}{w(T_{opt})}$. Next we focus on improving Algorithm 1 so that $\frac{2\sqrt{2}}{w(T_{opt})}$ is replaced by arbitrary small $\epsilon > 0$. Note that the optimal solution weight does not necessarily increase with the increase of the number of points n, namely all points can be in the same cells. Hence we cannot use the standard approach. However, the following two facts will do the trick. First, note that the weight of the GGMST optimal solution increases as the number of non-empty cells increases. Second, given a spanning tree structure of non-empty cells T, we can in polynomial time find the minimum weight GGMST feasible solution T' with the same tree structure as T (i.e. there is an edge in T' between two cells if and only if these two cells are adjacent in T). Next we design one such dynamic programming algorithm (see Algorithm 2).

Given an GGMST instance, let T be a spanning tree of the complete graph where the set of vertices correspond to the set of non-empty cells. Denote by X_i the set of points inside cell C_i. We observe T as a rooted tree with C_r as its root. If C_i is a leaf of T then the weight $W(z)$ of each point z in set X_i is set to zero. If C_i is not a leaf then T has some children $C_{i_1}, \ldots, C_{i_k}$ and the weight for points inside sets $X_{i_1}, \ldots, X_{i_k}$ has already been computed. Then for each point p in cell C_i (set X_i) we compute:

$$W(p) = \sum_{j=1}^{k} \min_{q \in X_{i_j}} \{W(q) + w(e_{p,q})\}$$

Algorithm 2 computes $W(p)$ for all $p \in C_r$. Note that it is easy to adapt Algorithm 2 to store selected points at each step.

Now we have all ingredients to design a $(1 + 4\sqrt{2} + \epsilon)$-approximation algorithm, see Algorithm 3. Note that $1 + 4\sqrt{2}$ is approximately equal to 6.66.

Theorem 3. *For any $\epsilon > 0$, Algorithm 3 is a $(1 + 4\sqrt{2} + \epsilon)$-approximation algorithm for the GGMST.*

Proof. If $N \leq 15$ or $N \leq 10\sqrt{2}/\epsilon$, then we can enumerate all spanning trees on $N + 1$ non-empty cells, and apply Algorithm 2 on each of them. That will give us an optimal solution in polynomial time.

Assume $N > 15$ and $N > 10\sqrt{2}/\epsilon$. By Corollary 1 it follows that Algorithm 1 will produce a solution T_A such that

$$w(T_A) \leq \left(1 + 4\sqrt{2}\right) w(T_{opt}) + 2\sqrt{2}. \tag{4}$$

Algorithm 2. Optimal GGMST solution for a given spanning tree of cells

```
    Data: A spanning tree $T$ of non-empty cells
    Result: An optimal weight of the GGMST tree with the same structure as $T$
 1  Choose an arbitrary cell $C_r$ as the root of $T$;
 2  for each leaf $C_i$ of $T$ do
 3      for each $p \in X_i$ do
 4          $W(p) = 0$;
 5      end
 6  end
 7  $CurrentLevel$ = height of $T$;
 8  while $CurrentLevel \geq$ root level do
 9      for each node $C_i$ of $CurrentLevel$ do
10          Let $C_{i_1}, \ldots, C_{i_k}$ be children of $C_i$ in $T$;
11          for each $p \in X_i$ do
12              $W(p) = \sum_{j=1}^{k} \min_{q \in X_{i_j}} \{W(q) + w(e_{p,q})\}$;
13          end
14      end
15      $CurrentLevel = CurrentLevel - 1$;
16  end
17  return $\min_{p \in X_r} W(p)$;
```

Algorithm 3. $(1 + 4\sqrt{2} + \epsilon)$-approximation algorithm for the GGMST

```
1  if $N \leq 15$ or $N \leq 10\sqrt{2}/\epsilon$ then
2      Output minimum weight solution obtained by Algorithm 2 on all spanning
       trees of non-empty cells;
3  else
4      Run Algorithm 1;
5  end
```

From Theorem 2 and $N > 15$ it follows that $1 \leq 5w(T_{opt})/N$. Applying that on the rightmost element of inequality (4) we get

$$w(T_A) \leq \left(1 + 4\sqrt{2}\right) w(T_{opt}) + \frac{10\sqrt{2}}{N} w(T_{opt}),$$
$$\leq \left(1 + 4\sqrt{2} + \frac{10\sqrt{2}}{N}\right) w(T_{opt}).$$

Now from $N > 10\sqrt{2}/\epsilon$ it follows that

$$w(T_A) \leq \left(1 + 4\sqrt{2} + \epsilon\right) w(T_{opt}),$$

which proves the theorem. □

3 The Lower Bound Proof

This section is entirely devoted to proving Theorem 2 which gives us a lower bound on the weight of an optimal solution. The lower bound is expressed in terms of the number of edges N.

Throughout this section we identify 1×1 grid cell with its coordinates (i, j), where $i, j \in \mathbb{Z}$ is the row and the column of the cell inside the infinite integer grid. For example, in Fig. 1, cell $(i, j + 1)$ contains one point which is near its upper right corner.

We start by proving lower bounds for trees of small size.

Lemma 1. *The weight of any subtree of T_{opt} with four edges is at least* 1*.*

Proof. Consider a subtree T' of T_{opt} with four edges. Let H denote the set of the five cells that contain vertices of T'. Note that there will be two cells in H with coordinates (i, j) and (i', j') such that $|i - i'| \geq 2$ or $|j - j'| \geq 2$. Hence, Euclidean distance between a vertex from the cell (i, j) and a vertex from the cell (i', j') is a least 1. This implies $w(T') \geq 1$. See Fig. 2 for an example. $\square$

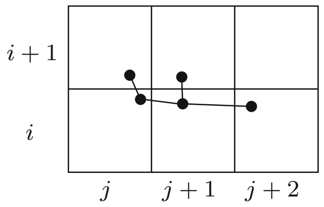

Fig. 2. An example of a tree T' with four edges

Lemma 2. *The weight of any subtree of T_{opt} with seven edges is at least $\frac{1}{3}(2\sqrt{6}+\sqrt{6-3\sqrt{3}})$ (which is greater than* 1.93*).*

Proof. Let T' be a subtree of T_{opt} with seven edges. If T' does not fit in any 3×3 sub-grid of the original grid, then there are two vertices u, v of T' which are from cells with coordinates (i, j) and (i', j') such that $|i - i'| \geq 3$ or $|j - j'| \geq 3$. In that case $w(e_{u,v}) \geq 2$ and therefore $w(T') \geq 2$.

Next we consider the case when T' fits into 3×3 grid. Since T' has eight vertices, at least three of them are in the corner cells of a 3×3 grid. Without loss of generality we assume that these three vertices are vertex v in cell (i, j), vertex u in cell $(i + 2, j)$ and vertex y in cell $(j + 2, i)$. Let P be a shortest path in T' from v to u and let Q be the shortest path in T' from v to y. Note that $w(e_{v,u}) \geq 1$ and $w(e_{v,y}) \geq 1$. If P and Q do not have a common vertex apart from v, then $w(T') \geq 2$. Thus we are left with the case when P and Q have a common vertex other than v, which we denote by x.

First we assume that P and Q do not go through the point in cell $(i+1, j+1)$. In this case, up to symmetry, one of the configurations depicted in Fig. 3(a,b)

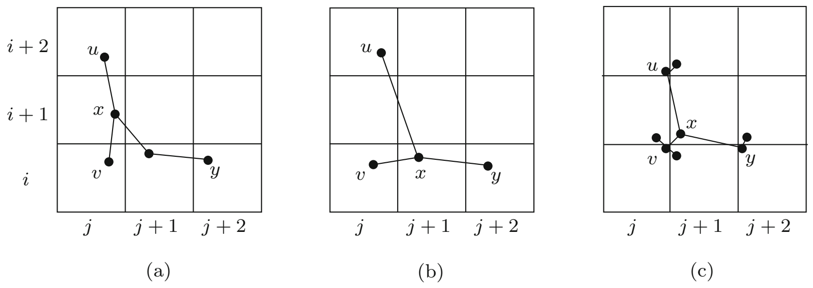

Fig. 3. Layouts of P and Q

occurs. However, it is clear that $w(e_{v,x}) + w(e_{x,y}) + w(e_{x,u}) \geq 2$ and hence $w(T') \geq 2$.

Lastly, we observe the case when vertex x is in cell $(i+1, j+1)$. Then $w(P \cup Q)$ is at least $w(e_{x,v}) + w(e_{x,u}) + w(e_{x,y})$, which is minimized when x is the Fermat point for the three corners of cell $(i+1, j+1)$ and T' has the structure depicted in Fig. 3(c). Therefore it can be computed that $w(T') \geq \frac{1}{3}(2\sqrt{6} + \sqrt{6 - 3\sqrt{3}}) > 1.93$. □

Lemma 3. *The weight of any subtree of T_{opt} with eight edges is at least* 2.

Proof. Let T' be a subtree of T_{opt} with eight edges. If T' does not fit in any 3×3 sub-grid then by the same simple argument as in the proof of Lemma 2 we get $w(T') \geq 2$. If T' fits in a 3×3 grid, then there is one vertex of T' in any cell of such 3×3 grid. More specifically, there are vertices in cells (i, j), $(i+2, j)$, $(i, j+2)$ and $(i+2, j+2)$ from which easily follows that $w(T') > 2$. □

Lemma 4. *The weight of any subtree of T_{opt} with nine edges is at least* $1 + \sqrt{3}$.

Proof. Let T' be a subtree of T_{opt} with nine edges. If T' does not fit in any 4×4 sub-grid of the original grid, then there are two vertices u, v of T' which are in cells with coordinates (i, j) and (i', j') such that $|i - i'| \geq 4$ or $|j - j'| \geq 4$. In that case $w(e_{u,v}) \geq 3$ and therefore $w(T') \geq 3 > 1 + \sqrt{3}$.

Next we consider the case when the smallest rectangular sub-grid that contains T' is of the size 4×4, and let (i, j) be the bottom left corner cell of such 4×4 grid. In that case there are four (not necessarily distinct) vertices u, v, x, y of T' that for some $i \leq i', i'' \leq i+3$ and $j \leq j', j'' \leq j+3$ lie in cells $(i', j), (i, j'), (i'', j+3), (i+3, j'')$, respectively. Let P be the shortest path in T' from u to x and let Q be the shortest path in T' from v to y. Let us observe the union of paths P and Q. This union is a set of k edges we denote by e_ℓ, $\ell = 1, \ldots, k$. Let us denote by x_ℓ and y_ℓ the lengths of projections of e_ℓ on x-axis and y-axis, respectively. Then

$$w(P \cup Q) = \sum_{\ell=1}^{k} \sqrt{x_\ell^2 + y_\ell^2}. \tag{5}$$

Since distance between projections of u and x on x-axis is at least 2 and distance between projections of v and y on y-axis is at least 2, it follows that $\sum_{\ell=1}^{k} x_\ell \geq 2$ and $\sum_{\ell=1}^{k} y_\ell \geq 2$. Hence, (5) is minimized when $k = 1$ and $x_1 = y_1 = 2$ with minimal value being $2\sqrt{2}$. Therefore we get $w(T') \geq 2\sqrt{2} > 1 + \sqrt{3}$.

Lastly, we consider the case when T' fits into a rectangular sub-grid R of dimensions smaller than 4×4. Without loss of generality we can assume that R is of the size 4×3, and let (i, j) be the bottom left corner cell of R. Note that there are at least two vertices of T' that are in corner cells of R. Without loss of generality we assume that vertex v is in cell (i, j). Next we distinguish remaining cases with respect to the position of the second corner point which we denote by u.

Case 1. Vertex u is in cell $(i, j + 2)$. As there are ten vertices in T', one of them must be in cell $(i + 3, j')$ for some $j \leq j' \leq j + 3$. Denote such vertex by y. By calculating the Fermat point x it can be seen that weight of the Steiner tree containing u, v and y is at least $2 + \sqrt{3}/2$ which is greater than $1 + \sqrt{3}$, see Fig. 4(a).

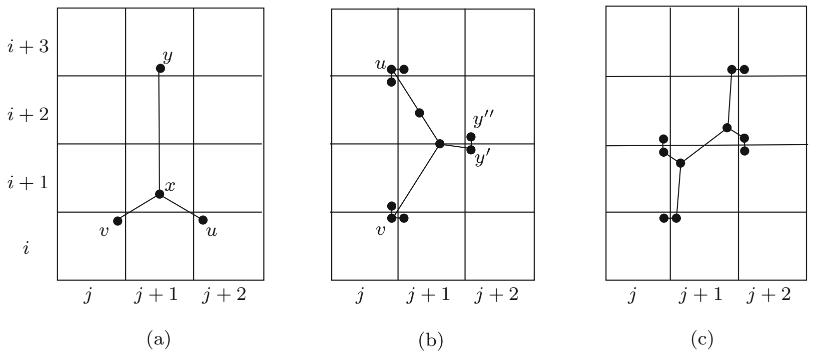

Fig. 4. T' configurations cases

Case 2. Vertex u is in cell $(i + 3, j)$. We can assume that there are no vertices of T' in cells $(i, j + 2)$ or $(i + 3, j + 2)$ as then **Case 1** applies. Then there must be vertices y', y'' in T' in cells $(i+1, j+2)$ and $(i+2, j+2)$. Hence, $w(T')$ must be at least as the weight of the Steiner tree that contains right upper corner of cell (i, j), right bottom corner of cell $(i + 3, j)$ and left bottom corner of cell $(i + 2, j + 2)$. By calculating the Fermat point, one can see that such Steiner tree has weight $1 + \sqrt{3}$, hence $w(T') \geq 1 + \sqrt{3}$. In Fig. 4(b) subtree T' has the configuration that mimics such Steiner tree.

Case 3. Vertex u is in cell $(i+3, j+2)$. We can assume that there are no vertices of T' in cells $(i, j + 2)$ or $(i + 3, j)$ as then **Case 1** or **Case 2** apply. In this case minimal weight T' mimics the Steiner tree that contains right upper corner of cell (i, j), left bottom corner of cell $(i + 3, j + 2)$, right bottom corner of cell $(i + 2, j)$ and left upper corner of the cell $(i + 1, j + 2)$, see Fig. 4(c). It is easy to calculate that the weight of such Steiner tree is $\sqrt{5+2\sqrt{3}}$ which is greater than $1+\sqrt{3}$. $\square$

Now we are ready to prove Theorem 2.

Proof (of Theorem 2). We will proof the theorem by induction on N. Recall that N is the number of edges in T_{opt}.

By Lemmas 1, 3 and 4, theorem holds for $N \leq 13$. Next we assume that theorem holds for all trees with number of edges strictly less than N.

We will perform the induction step as follows: through exhaustive case study we will show that there always exist a subtree T' of T_{opt} for which $w(T')$ is greater or equal to number of edges of T' divided by 4, and if we remove from T_{opt} the edges of T', it remains connected. In that case, by induction hypothesis the bound for T_{opt} holds.

We observe T_{opt} as a rooted tree, and given a vertex v of T_{opt}, we denote by T_v the maximal subtree of T_{opt} rooted at v.

Let u be a non-leaf vertex of T_{opt} with maximum number of edges in its path to the root.

Assumption 1: We may assume u has at most two children. Namely, in the case when u has four children u_1, u_2, u_3, u_4 let T' be a subtree of T_u induced by $\{u, u_1, u_2, u_3, u_4\}$. In the case when u has exactly three children u_1, u_2, u_3 set T' to be T_v where v is the parent of u. Note that in both cases T' has four edges. Let $T'' = T_{opt} \setminus E(T')$ where $E(T)$ denotes the set of edges of a tree T. Since T'' is a tree, by induction hypothesis it follows that $|E(T'')| = N-4 \leq 4w(T'')+3$. Furthermore, by Lemma 1 we have that $4 \leq 4w(T')$. Hence, $N \leq 4w(T'') + 4w(T') + 3 = 4w(T_{opt}) + 3$.

Assumption 2: If u has exactly two children u_1, u_2, we may assume that the parent of u (denoted by v) has degree strictly greater than two. Namely, if this is not the case, we set $T' = T_v \cup \{e_{v,w}\}$ where w is the parent of v, and we set $T'' = T_{opt} \setminus E(T_w)$. Since T' has four edges and T'' is a tree, by induction hypothesis for T'' and Lemma 1 we obtain the bound.

Case 1: Vertex u has exactly two children u_1, u_2. Then by **Assumption 2** v has at least two children. By the choice of u, the number of edges in any path from v to a leaf in T_v is at most 2. Let w' be another child of v. By **Assumption 1** w' has at most two children. Also note that we can assume that w' has at least one child. Otherwise the subtree T' induced by $\{w', v, u, u_1, u_2\}$ has four edges, hence by removing the edges of T' from T we can apply the induction hypothesis and obtain the bound.

Case 1.1: Vertex v has another child w''. In this case using the same arguments as above it can be shown that w'' must have exactly one or two children. Note that subtree T' induced by v, u, u_1, u_2 together with $T_{w'}, T_{w''}$ has at least seven edges and at most nine edges. Therefore, Lemmas 2, 3 or 4 can be applied for each of the cases. Furthermore, for the remaining subtree $T_{opt} \setminus E(T')$ the induction hypothesis can be applied to obtain the bound.

Case 1.2: Vertex v has only two children w', u. Let w be the parent of v. We can assume that w' has exactly one child, otherwise the subtree T' induced by the vertices of T_v and vertex w has exactly seven edges, hence we could use Lemma 2. If the degree of w is two, then let T' be the subtree induced by T_w together with the edge $e_{w,y}$, where y is the parent of w. T' has seven edges and therefore, the result follows. Now, we may assume that w has another child v'. Let $T_1 = T_v$ and observe that T_1 has 5 edges. Let $T_2 = T_{v'}$. By the same argument used for T_v, we conclude that T_2 has at most five edges. Let $T' = T_1 \cup T_2 \cup \{e_{w,v'}, e_{w,v}\}$. If T_2 has zero, one or two edges, then T' has at least seven and at most nine edges, and hence the bound follows. If T_2 has four edges then by induction hypothesis on $T_{opt} \setminus E(T_2)$ and by applying the Lemma 1 on T_2, we obtain the bound. It remains to consider the cases when T_2 has three or five edges. If T_2 has three edges, then we add edge $e_{w,v'}$ to T_2 and now the new tree has four edges, hence we can apply the same arguments as before. We are left only with the case when T_2 has five edges. In this case $w(T_2) \geq 1$, according to Lemma 1, and also $T_3 = T_1 \cup \{e_{w,v'}, e_{v,w}\}$ has seven edges. By Lemma 2, either $w(T_3)$ is at least 2, or it has the structure depicted in Fig. 3(c), and it is clear that every edge incident to the tree in Fig. 3(c) is grater than, say 0.5. Hence, in either case $w(T') \geq 3$. Since T' has twelve edges the bound is obtained by induction hypothesis on $T_{opt} \setminus E(T')$.

Case 2: Vertex u has exactly one child u_1.

Case 2.1 Vertex v has another child w'. In this case $T_{w'}$ has depth at most 1. If w' has more than one child, then from **Case 1** (w' instead of u) we are done. If w' has one child (denoted by w_1), then the subtree induced by $\{u_1, u, v, w', w_1\}$ has four edges and we are done.

We continue by assuming that w' has no child. If v has another child $w'' \notin \{u, w'\}$, then as we argued for w', we can assume that w'' has no child. However, in this case subtree induced by $\{u_1, u, v, w', w''\}$ has four edges and we are done. Therefore we can assume that v has exactly two children w' and u. Let w be the parent of v. Then the subtree induced by u_1, u, v, w', w has four edges and we are done.

Case 2.2: Vertex v has only child u. Let w be the parent of v. W can assume that v has a sibling node v', as otherwise we can remove the four edge subtree induced by $\{u_1, u, v, w, z\}$, where z is the parent of w. Furthermore, we can assume that v' has a child u', as otherwise we can remove the four edge subtree induced by $\{u_1, u, v, w, v'\}$.

Case 2.2.1: Vertex u' has no child but has a sibling u''. We can assume that no child of v' has a child, as we can observe such case as an instance of **Case 2.2.3**. Furthermore, we can assume that u' and u'' are only children of v'. Otherwise, in the case when v' has more than three children, there would exist a subtree of $T_{v'}$ with four edges that we could remove. Furthermore, in the case when v' has exactly three children, we can remove $T' = T_{v'} \cup \{e_{w,v'}\}$.

Hence we are left with the case when u'' is the only sibling of u'. In the case v and v' are only children of w, we can remove seven edge subtree $T' =$

$T_w \cup \{e_{w,z}\}$, where z denotes the parent of w. Lastly, we consider the case when there exist third child of w denoted by v''. From the assumptions and solved cases above, we can assume that $T_{v''}$ has at most two edges, hence subtree $T' = T_v \cup T_{v'} \cup T_{v''} \cup \{e_{w,v}, e_{w,v'}, e_{w,v''}\}$ has seven, eight or nine edges, therefore we can remove it.

Case 2.2.2: Vertex u' has no child nor sibling. In the case there exists a third child of w, from the assumptions and solved cases above if would follow that we can assume that it has only one child which has no child. In that case thee would exist a subtree of T_w with four edges that we can remove. Hence, we can assume that w has no other children besides v and v'. Then T_w is a path with five edges. If $w(T_w)$ is grater than 5/4, we can remove it and we are done. Otherwise it must be similar to the structure depicted in Fig. 5, i.e. with a path of approximate size 1 alongside a border of a cell, and with remaining vertices grouped at the endpoints of such path. Note that in that case, edge $e_{w,z}$ must be big enough so that $w(T_w \cup \{e_{w,z}\})$ is greater than 6/4. Hence we can remove $T_w \cup \{e_{w,z}\}$ and by induction hypothesis obtain the bound.

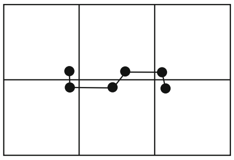

Fig. 5. A short path with five edges

Case 2.2.3: Vertex u' has a child u'_1. Note that from the assumption on maximality of depth of u, u'_1 has no children. As we solved **Case 2.1**, we can assume that u'_1 has no siblings. Furthermore, we can assume that there is no sibling of u' that has a child, as in that case there would exist subtree of $T_{v'}$ with four edges that we could remove. Now in the case that u' has more than one sibling, again, there would exist subtree of $T_{v'}$ with four edges that we could remove. In the case that u' has exactly one sibling, subtree $T' = T_{v'} \cup \{e_{w,v'}\}$ can be removed. We are left with the case when both T_v and $T_{v'}$ are paths with two edges. In the case there is a third child of w, denoted by v'', from the solved cases above if follows that we can assume that $T_{v''}$ is also a path with two edges. In that case there is a subtree of T_w with nine edges that can be removed. In the case there is no third child of w, the seven edges subtree $T' = T_w \cup \{e_{w,z}\}$ (with z being the parent of w), can be removed and the bound obtained. We considered all the cases, therefore proving the theorem. □

4 Approximation of the GGTSP

Our approximation algorithms for the GGMST can be used to obtain approximation algorithms for the geometric generalized travelling salesman problem on grid clusters (GGTSP) using standard methods.

Algorithm 4. $(2 + 8\sqrt{2} + 2\epsilon)$-approximation algorithm for the GGTSP

```
  Data: Instance $I$ of the GGTSP
  Result: Generalized travelling salesman tour
1 $T_A \leftarrow$ output of Algorithm 3 on $I$;
2 $G_E \leftarrow$ Eulerian graph obtained by doubling all edges in $T_A$;
3 $\mathcal{ET} \leftarrow$ an Euler tour of $G_E$;
4 $\mathcal{C} \leftarrow$ a GGTSP tour obtained by going along $\mathcal{ET}$ and skipping repeated vertices;
5 return $\mathcal{C}$;
```

We start with the approach of shortcutting a double MST, presented in Algorithm 4 and analyzed next.

By removing one edge from a GGTSP tour, one obtains a GGMST tree, hence $w(T_A)$ is less than $(1 + 4\sqrt{2} + \epsilon)OPT$, where OPT is the weight of an optimal solution of the GGTSP. Therefore, $w(G_E)$ is less than $2(1+4\sqrt{2}+\epsilon)OPT$. Due to triangle inequality, shorcutting the Euler tour in line 4 of the algorithm does not increase the weight. Hence, Algorithm 4 is a $(2 + 8\sqrt{2} + 2\epsilon)$-approximation algorithm for the GGTSP. Note that $2 + 8\sqrt{2}$ is approximately equal to 13.31.

Next we use the approach from the famous Christofides $\frac{3}{2}$-approximation algorithm for the metric TSP, see [3]. This approach will give us 0.5 decrease of the approximation ratio. We give a sketch of the algorithm and the analysis, and leave details to the reader.

We start by running Algorithm 1 on the GGTSP instance. Let T_G be the resulting tree. Note that $w(T_G)$ is less or equal than $(1 + 4\sqrt{2})OPT + 2\sqrt{2}$, where OPT is the weight of an optimal solution of the GGTSP. Let S be a set of non-empty cells that contain a vertex of T_G with an odd degree. Note that $|S|$ is even. Let M be a minimum perfect matching among cells in S, where the distance between two cells $C_1, C_2 \in S$ is the smallest distance between two points p_1, p_2 among all $p_1 \in C_1$, $p_2 \in C_2$. It is not hard to show that $w(M) \leq \frac{1}{2}OPT$. Let M_G be the set of edges e_{t_1,t_2} for which t_1, t_2 are vertices of T_G and there exist an edge $e_{p_1,p_2} \in M$ such that p_1 and t_1 are in the same cell and p_2 and t_2 are in the same cell. Note that $w(M_G) \leq \frac{1}{2}OPT + N\sqrt{2}$, and hence by Theorem 2 we get that $w(M_G) \leq \frac{1}{2}OPT + 4\sqrt{2}OPT + 3\sqrt{2}$. By merging M_G and T_G we obtain an Eulerian graph, and by shortcutting one of its Euler tours we obtain a GGTSP tour with weight at most $(\frac{3}{2} + 8\sqrt{2})OPT + 5\sqrt{2}$. By similar approach as in Algorithm 3 and Theorem 3, we can get rid of $5\sqrt{2}$ error, and obtain a $(\frac{3}{2} + 8\sqrt{2} + \epsilon)$-approximation algorithm for every $\epsilon > 0$.

5 Conclusions

We presented a simple $(1+4\sqrt{2}+\epsilon)$-approximation algorithm for the geometric generalized minimum spanning tree problem on grid clusters (GGMST) and $(1.5+8\sqrt{2}+\epsilon)$-approximation algorithm for the geometric generalized travelling salesman problem on grid clusters (GGTSP).

To obtain guarantied approximation ratios for our algorithms, we used the following lower bound on the optimal solution: Every tree with N edges that contains at most one point from any 1×1 grid cell is of size at least $\frac{N-3}{4}$. Obtaining a tight lower bound in terms of the number of edges would decrees guaranteed approximation ratios of our (and other similar) algorithms. Moreover, it would be an interesting result on its own.

Acknowledgment. We would like to thank Geoffrey Exoo for many usefull discussions.

References

1. Arkin, E.M., Hassin, R.: Approximation algorithms for the geometric covering salesman problem. Discrete Appl. Math. **55**(3), 197–218 (1994)
2. Bovet, J.: The selective traveling salesman problem. Papers Presented at the EURO VI Conference, Vienna (1983)
3. Christofides, N.: Worst-case analysis of a new heuristic for the travelling salesman problem. In: Traub, J.F. (ed.) Symposium on New Directions and Recent Results in Algorithms and Complexity, p. 441. Academic Press, Orlando, Florida (1976)
4. Dumitrescu, A., Mitchell, J.S.B.: Approximation algorithms for TSP with neighborhoods in the plane. In: Symposium on Discrete Algorithms, pp. 38–46 (2001)
5. Elbassioni, K.M., Fishkin, A.V., Mustafa, N.H., Sitters, R.A.: Approximation algorithms for Euclidean group TSP. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 1115–1126. Springer, Heidelberg (2005)
6. Feremans, C., Grigoriev, A., Sitters, R.: The geometric generalized minimum spanning tree problem with grid clustering. 4OR **4**, 319–329 (2006)
7. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H Freeman, New York (1979)
8. Gudmundsson, J., Levcopoulos, C.: Hardness result for TSP with neighborhoods. Technical report LU-CS-TR:2000–216, Department of Computer Science, Lund Unversity, Sweden (2000)
9. Golden, B.L., Raghavan, S., Stanojevic, D.: Heuristic search for the generalized minimum spanning tree problem. INFORMS J. Comput. **17**(3), 290–304 (2005)
10. Jiang, H., Chen, Y.: An efficient algorithm for generalized minimum spanning tree problem. In: Proceedings of Genetic and Evolutionary Computation Conference, pp. 217–224 (2010)
11. Henry-Labordere, A.L.: The record balancing problem: a dynamic programming solution of a generalized traveling salesman problem. RIBO **B–2**, 736–743 (1969)
12. Laporte, G.: The traveling salesman problem: an overview of exact and approximate algorithms. Eur. J. Oper. Res. **59**, 231–247 (1992)
13. Laporte, G., Asef-Vaziri, A., Srikandarajah, C.: Some applications of the generalized traveling salesman problem. J. Oper. Res. Soc. **47**, 1461–1467 (1996)
14. Myung, Y.-S., Lee, C.-H., Tcha, D.-W.: On the generalized minimum spanning tree problem. Networks **26**(4), 231–241 (1995)

15. Mata, C.S., Mitchell, J.S.B.: Approximation algorithms for geometric tour and network design problems. In: Proceedings of the 11th Annual Symposium on Computational Geometry, pp. 360–369. ACM (1995)
16. Oncan, T., Corseau, J.F., Laporte, G.: A tabu search heuristic for the generalized minimum spanning tree problem. Eur. J. Oper. Res. **191**, 306–319 (2008)
17. Pop, P.C.: The generalized minimum spanning tree problem. Ph.D. Thesis, University of Twente (2002)
18. Pop, P.C., Still, G., Kern, W.: An approximation algorithm for the generalized minimum spanning tree problem with bounded cluster size. In: Proceedings of the First ACiD Workshop. Texts Algorithms vol. 4, pp. 115–121 (2005)
19. Safra, S., Schwartz, O.: On the complexity of approximating TSP with neighborhoods and related problems. In: Di Battista, G., Zwick, U. (eds.) ESA 2003. LNCS, vol. 2832, pp. 446–458. Springer, Heidelberg (2003)