Balanced Partition of Minimum Spanning Trees

Mattias Andersson[1], Joachim Gudmundsson[2⋆], Christos Levcopoulos[1], and Giri Narasimhan[3]

[1] Department of Computer Science, Lund University, Box 118, 221 00 Lund, Sweden. christos@cs.lth.se, dat97mae@ludat.lth.se

[2] Department of Computer Science, Utrecht University, PO Box 80.089, 3508 TB Utrecht, the Netherlands. joachim@cs.uu.nl

[3] School of Computer Science, Florida International University, Miami, FL 33199, USA. giri@fiu.edu.

Abstract. To better handle situations where additional resources are available to carry out a task, many problems from the manufacturing industry involve "optimally" dividing a task into k smaller tasks. We consider the problem of partitioning a given set $\mathcal{S}$ of n points (in the plane) into k subsets, $\mathcal{S}_1, \ldots, \mathcal{S}_k$, such that $\max_{1 \leqslant i \leqslant k} |MST(\mathcal{S}_i)|$ is minimized. A variant of this problem arises in the shipbuilding industry [2].

1 Introduction

In one interesting application from the shipbuilding industry, the task is to use a robot to cut out a set of prespecified regions from a sheet of metal while minimizing the completion time. In another application, a salesperson needs to meet some potential buyers. Each buyer specifies a region (i.e., a *neighborhood*) within which the meeting needs to be held. A natural optimization problem is to find a salesperson tour of shortest length that visits all of the buyers' neighborhoods and finally returns to his initial departure point. Both these problems are related to the problem known in the literature as the *Traveling Salesperson problem with Neighborhoods* (TSPN) and which has been extensively studied [4, 5, 7–10]. The problem (TSPN) asks for the shortest tour that visits each of the neighborhoods. The problem was recently shown to be APX-hard [8].

Interesting generalizations of the TSPN problem arise when additional resources ($k > 1$ robots in the sheet cutting problem, or $k > 1$ salespersons in the second application above) are available. The k-TSPN problem is a generalization of the problem where we are given k salespersons and the aim is to minimize the completion time, i.e., minimize the distance traveled by the salespersons making the longest journey.

The need for partitioning the input set such that the optimal substructures are balanced gives rise to many interesting theoretical problems. In this paper we consider the problem of partitioning the input so that the sizes of the minimum

⋆ Supported by the Swedish Foundation for International Cooperation in Research and Higher Education

P.M.A. Sloot et al. (Eds.): ICCS 2002, LNCS 2331, pp. 26–35, 2002.

spanning trees of the subsets are balanced. Also, we restrict our inputs to sets of points instead of regions. More formally, the *Balanced Partition Minimum Spanning Tree problem* (k-BPMST) is stated as follows:

Problem 1. Given a set of n points $\mathcal{S}$ in the plane, partition $\mathcal{S}$ into k sets $\mathcal{S}_1, \ldots, \mathcal{S}_k$ such that the weight of the largest minimum spanning tree,

$$W = \max_{1 \leqslant i \leqslant k} (|M(\mathcal{S}_i)|)$$

is minimized. Here $M(\mathcal{S}_i)$ is the minimum spanning tree of the subset $\mathcal{S}_i$ and $|M(\mathcal{S}_i)|$ is the weight of the minimum spanning tree of $\mathcal{S}_i$.

The paper is organized as follows. In section 2, we show that the problem is NP-hard. In section 3, we present an approximation algorithm with approximation factor $4/3 + \varepsilon$ for the case $k = 2$, and with an approximation factor $(2 + \varepsilon)$ for the case $k \geqslant 3$. The algorithm runs in time $O(n \log n)$.

2 NP hardness

In this section we show that the k-BPMST problem is NP-hard. In order to do this we need to state the recognition version of the k-BPMST problem:

Problem 2. Given a set of n points $\mathcal{S}$ in the plane, and a real number $\mathcal{L}$, does there exist a partition of $\mathcal{S}$ into k sets $\mathcal{S}_1, \ldots, \mathcal{S}_k$ such that the weight of the largest minimum spanning tree,

$$W = \max_{1 \leqslant i \leqslant k} (|M(\mathcal{S}_i)|) \leq \mathcal{L}?$$

In a computational model in which we can handle square roots in polynomial time, such as the real-RAM model (which will be used for simplicity), this formulation of the problem is sufficient in order to show that the k-BPMST problem is NP-hard. Note, however, that it may be inadequate in more realistic models, such as the Turing model, where efficient handling of square roots may not be possible. The computation of roots is necessary to determine the length of edges between points, which, in turn, is needed in order to calculate the weight of a minimum spanning tree. So in a realistic computational model the hardest part may not be to partition the points optimally, but instead to calculate precisely the length of the MST's. Thus, in these more realistic computational models we would like to restrict the problem to instances where the lengths of MST's are easy to compute. For example, this can be done by modifying the instances created in the reduction below, by adding some points so that the MST's considered only contain vertical and horizontal edges.

The proof is done (considering the real-RAM model) by a straight-forward polynomial reduction from the following recognition version of PARTITION.

Problem 3. Given integers $a = \{a_1 \leq \ldots \leq a_n\}$, the recognition version of the partition problem is: Does there exist a subset $P \subseteq I = \{1, 2, \ldots, n\}$ such that

$$\#P = \#I/P \quad \text{and} \sum_{j \in P} a_j = \sum_{j \in I/P} a_j$$

We will denote $\#P$ by h, $h = n/2$. This version of PARTITION is NP-hard [3].

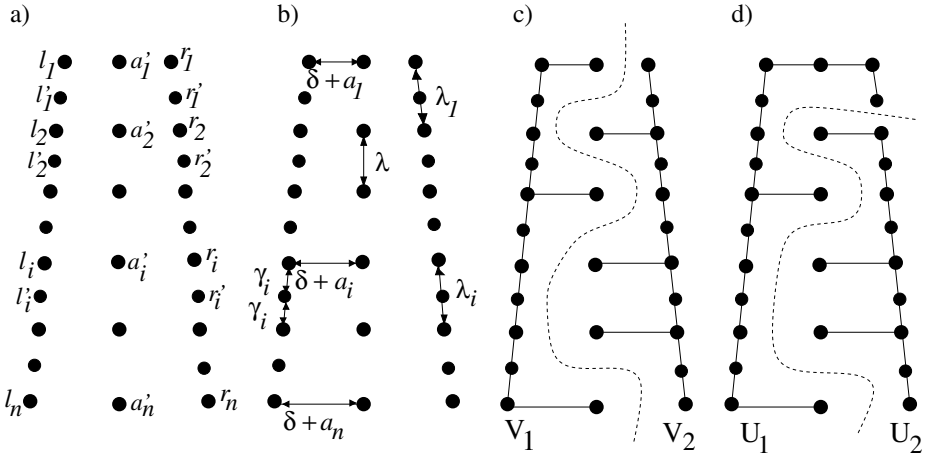

Fig. 1. The set of points $\mathcal{S}$ created for the reduction. In Figure (a) all notations for the points are given. Similarly, in Figure (b) the notations for the distances between points are given. Figure (c) illustrates a class 1 partition, and (d) illustrates a class 2 partition.

Lemma 1. *The k-*BPMST *problem is NP-hard.*

Proof. The reduction is done as follows. Given a PARTITION instance we create a 2-BPMST instance, in polynomial time, such that it is a yes-instance if, and only if, the PARTITION-instance is a yes-instance. Obviously PARTITION then polynomially reduces to 2-BPMST. Given that the PARTITION-instance contains n integers $a_1, \ldots, a_n$, we create the following 2-BPMST instance. A set of points $\mathcal{S}$, as shown in Figure 1a is created, with inter point distances as shown in Figure 1b. A closer description of these points and some additional definitions is given below:

- $a' = \{a'_1, \ldots, a'_n\}$, where $a'_i = (0, i\lambda)$,
- $l = \{l_1, \ldots, l_n\}$, where $l_i = (-\delta - a_i, i\lambda)$,
- $r = \{r_1, \ldots, r_n\}$, where $r_i = (\delta + a_i, i\lambda)$,
- $l' = \{l'_1, \ldots, l'_{n-1}\}$, where l'_i is the midpoint on the line between l_i and l_{i+1}, and

– $r' = \{r'_1, \ldots, r'_{n-1}\}$, where r'_i is the midpoint on the line between r_i and r_{i+1}

We also define the following set of points, $a^* = \{a'_{P[1]}, \ldots, a'_{P[h]}\}$. Further, let $\lambda = 11n(a_n + n)$ and let $\delta = 7n(a_n + n)$. Note that $\lambda_i^2 \leqslant \lambda^2 + a_n^2$ which implies that $\lambda_i \leqslant 12n(a_n + n)$, which means that $\gamma_i = \lambda_i/2 \leqslant 6n(a_n + n)$. Finally let (see definition 2)

$$\mathcal{L} = (\sum_{i \in I} a_i)/2 + n/2 \cdot \delta + \sum_{i=1}^{n-1} \lambda_i$$

Since the number of points in $\mathcal{S}$ is polynomial it is clear that this instance can be created in polynomial time. Next we consider the "if", and the "only if" parts separately.

If If P exists and we have a yes PARTITION-instance it is clear that the corresponding 2-BPMST instance is also a yes-instance. This follows when the partition $\mathcal{S}'_1 = a^* + l + l'$, $\mathcal{S}'_2 = \mathcal{S} - \mathcal{S}_1$ (a class 1 partition, as defined below) is considered. The general appearance of $M(\mathcal{S}'_1)$ and $M(\mathcal{S}'_2)$ (see Figure 1c) is determined as follows. The points $l + l'$ and the points $r + r'$ will be connected as illustrated in Figure 1c, which follows from the fact that $\gamma_i < \delta < \delta + a_1$. Next consider the remaining points a'. Any point a'_i will be connected to either l_i (in $M(\mathcal{S}'_1)$) or r_i (in $M(\mathcal{S}'_2)$), since r_i and l_i are the points located closest to a'_i (follows since $\lambda > \delta + a_n$). Thus,

$$|M(\mathcal{S}'_1)| = |M(\mathcal{S}'_2)| = (\sum_{i \in I} a_i)/2 + n/2 \cdot \delta + \sum_{i=1}^{n-1} \lambda_i$$

and we have that the created instance is a yes-instance.

Only if We have that P does not exist and we therefore want to show that the created 2-BPMST is a no-instance. For this two classes of partitions will be examined:

– All partitions $\mathcal{V}_1, \mathcal{V}_2$ such that $l + l' \subseteq \mathcal{V}_1$ and $r + r' \subseteq \mathcal{V}_2$
– All other partitions $\mathcal{U}_1, \mathcal{U}_2$ not belonging to class 1.

We start by examining the first class (illustrated by Figure 1c). Note that an optimal MST will contain the edges in $M(\mathcal{V}_1)$ and $M(\mathcal{V}_2)$ plus the edge between a'_1 and l_1 or r_1, hence $|M(\mathcal{S})| = |M(\mathcal{V}_1)| + |M(\mathcal{V}_2)| + \delta + a_1$. Note also that $|M(\mathcal{V}_1)| + |M(\mathcal{V}_2)| = 2 \cdot \mathcal{L}$. For all partitions $\mathcal{V}'_1 \subseteq \mathcal{V}_1, \mathcal{V}'_2 \subseteq \mathcal{V}_2$ such that each subset $\mathcal{V}'_1, \mathcal{V}'_2$ contains exactly $|a'|/2$ points from the set a' it is clear, since P does not exist, that $max\{|M(\mathcal{V}'_1)|, |M(\mathcal{V}'_2)|\} > \mathcal{L}$. This is true also for the partitions $\mathcal{V}^*_1 \subseteq \mathcal{V}_1, \mathcal{V}^*_2 \subseteq \mathcal{V}_2$ such that each subset does *not* contain exactly $|a'|/2$ points from the set a'. To see this consider any such partition and the corresponding subset $\mathcal{V}^*_i$ such that $|\mathcal{V}^*_i| = \max\{|\mathcal{V}^*_1|, |\mathcal{V}^*_2|\}$. We have that

$$|M(\mathcal{V}^*_i)| \geq \delta + n/2 \cdot \delta + \sum_{i=1}^{n-1} \lambda_i > (\sum_{i \in I} a_i) + n/2 \cdot \delta + \sum_{i=1}^{n-1} \lambda_i > \mathcal{L}$$

This implies that $\max\{|M(\mathcal{V}^*_1)|, |M(\mathcal{V}^*_2)|\} > \mathcal{L}$.

Next consider the class 2 partitions (illustrated by Figure 1d). There is always an edge of weight γ_i $(1 \leq i \leq n)$ connecting the two point sets of any such partition. This means that there can not exist a class 2 partition $\mathcal{U}_1, \mathcal{U}_2$ such that $\max\{|M(\mathcal{U}_1)|, |M(\mathcal{U}_2)|\} \leq \mathcal{L}$, because we could then build a tree with weight at most $2 \cdot \mathcal{L} + \gamma_i < |M(\mathcal{V}_1)| + |M(\mathcal{V}_2)| + \delta + a_1 = |M(\mathcal{S})|$, which is a contradiction. Thus, $\max\{|M(\mathcal{U}_1)|, |M(\mathcal{U}_2)|\} > \mathcal{L}$, which concludes this lemma.

3 A $2 + \varepsilon$ approximation algorithm

In this section a $2 + \varepsilon$ approximation algorithm is presented. Note also that a straight-forward greedy algorithm, that partitions $M(\mathcal{S})$ into k sets by removing the $k - 1$ longest edges gives an approximation of k. The main idea of the $2 + \varepsilon$ approximation algorithm is to partition $\mathcal{S}$ into a constant number of small components, test all valid combinations of these components and give the best combination as output. As will be seen later, one will need an efficient partitioning algorithm, denoted ValidPartition or VP for short. A partition of a point set $\mathcal{S}$ into two subsets $\mathcal{S}_1$ and $\mathcal{S}_2$ is said to be valid if $\max(|M(\mathcal{S}_1)|, |M(\mathcal{S}_2)|) \leqslant 2/3 \cdot |M(\mathcal{S})|$. The following lemma is easily shown [1] using standard decomposition methods.

Lemma 2. *Given a set of points $\mathcal{S}$,* VP *divides $\mathcal{S}$ into two sets $\mathcal{S}_1$ and $\mathcal{S}_2$ such that (i)* $\max\{|M(\mathcal{S}_1)|, |M(\mathcal{S}_2)|\} \leqslant \frac{2}{3}M(\mathcal{S})$*, and (ii)* $|M(\mathcal{S}_1)| + |M(\mathcal{S}_2)| \leqslant |M(\mathcal{S})|$*. If* VP *is given a MST of $\mathcal{S}$ as input then it holds that the time needed for* VP *to compute a valid partition is $O(n)$.*

3.1 Repeated ValidPartition

ValidPartition will be used repeatedly in order to create the small components mentioned in the introduction of this section. Consider the following algorithm, given a MST of $\mathcal{S}$ and an integer m. First divide $M(\mathcal{S})$ into two components using VP. Next divide the largest of these two resulting components, once again using VP. Continue in this manner, always dividing the largest component created thus far, until m components have been created. Note that in each division the number of components increase by one. This algorithm will be denoted RepeatedValidPartition, or RVP for short. The following lemma expresses an important characteristic of RVP.

Lemma 3. *Given a minimum spanning tree of a set of points $\mathcal{S}$ and an integer m,* RVP *will partition $\mathcal{S}$ into m components $\mathcal{S}_1, \ldots, \mathcal{S}_m$ such that* $\max(|M(\mathcal{S}_1)|, \ldots, |M(\mathcal{S}_m)|) \leqslant \frac{2}{m}|M(\mathcal{S})|$.

Proof. Consider the following algorithm $\mathcal{A}$. Start with $M(\mathcal{S})$ and divide with VP until the weight of all components is less than or equal to $\frac{2}{m}|M(\mathcal{S})|$. The order in which the components are divided is arbitrary but when a component weighs less than or equal to $\frac{2}{m}|M(\mathcal{S})|$ it is not divided any further. If it now could be shown that the number of resulting components is at most m the lemma would follow.

This is seen when the dividing process of RVP is examined. Since RVP always divides the largest component created thus far, a component of weight at most $\frac{2}{m}|M(\mathcal{S})|$ would not be divided unless all other components also have weight at most $\frac{2}{m}|M(\mathcal{S})|$. Further, VP guarantees that the two components resulting from a division always have weights less than the divided component. Thus, when m components have been created by RVP these m components would also have weight less than or equal to $\frac{2}{m}|M(\mathcal{S})|$. Therefore, the aim is to show that algorithm $\mathcal{A}$, given $M(\mathcal{S})$, produces at most m components. The process can be represented as a tree. In this tree each node represents a component, with the root being $M(\mathcal{S})$. The children of a node represent the components created when that node is divided using VP. Note that the leaves of this tree represent the final components. Thus the aim is to show that the number of leaves do not exceed m. For this purpose we will divide the leaves into two categories. The first category is all leaves whose sibling is not a leaf. Assume that there are m_1 such leaves in the tree. The second category is all remaining leaves, that is, those who actually have a sibling leaf. Assume, correspondingly, that there are m_2 such leaves.

We start by examining the first category. Consider any leaf l_i of this category. Denote its corresponding sibling s_i and denote by p_i the parent of l_i and s_i. Further to each l_i we attach a weight $w(l_i)$ which is defined as $w(l_i) = |M(p_i)| - |M(s_i)|$. Since s_i is not a leaf it holds that $|M(s_i)| > \frac{2}{m}|M(\mathcal{S})|$, and since VP is used we know that $|M(s_i)| \leqslant \frac{2}{3}|M(p_i)|$. Thus, $|M(p_i)| > \frac{3}{m}|M(\mathcal{S})|$ which implies that $w(l_i) \geqslant \frac{1}{3}|M(p_i)| > \frac{1}{m}|M(\mathcal{S})|$ and $\sum_{i=1}^{m_1} w(l_i) > m_1 \cdot \frac{1}{m}|M(\mathcal{S})|$.

Next the second category of leaves is examined. Denote any such leaf l'_i and its corresponding parent p'_i. Since there are m_2 leaves of this category and each leaf has a leaf sibling these leaves have in total $m_2/2$ parent nodes. Further, for each such corresponding parent component $M(p'_i)$ we have that $|M(p'_i)| > \frac{2}{m}|M(\mathcal{S})|$ (they are not leaves). Thus, $\sum_{i=1}^{m_2} |M(p'_i)| > \frac{m_2}{2} \cdot \frac{2}{m}|M(\mathcal{S})| = m_2 \cdot \frac{1}{m}|M(\mathcal{S})|$.

Next consider the total weight of the components examined so far. We have that $m_1 \cdot \frac{1}{m}|M(\mathcal{S})| + m_2 \cdot \frac{1}{m}|M(\mathcal{S})| < \sum_{i=1}^{m_1} w(l_i) + \sum_{i=1}^{m_2} |M(p'_i)| \leqslant |M(\mathcal{S})|$, which implies that $m_1 + m_2 \leqslant m$. Thus, the number of leaves do not exceed m. □

3.2 The approximation algorithm

Now we are ready to state the algorithm CA. As input we are given a set $\mathcal{S}$ of n points, an integer k and a positive real constant ε. The algorithm differs in two separate cases, $k = 2$ and $k \geqslant 3$. First $k = 2$ is examined, in which the following steps are performed:

step 1: Divide $M(\mathcal{S})$ into $\frac{4}{\varepsilon'}$ components, using RVP, where $\varepsilon' = \frac{\varepsilon}{4/3+\varepsilon}$. The reason for the value of ε' will become clear below. Let W denote the heaviest component created and let w denote its weight.

step 2: Combine all components created in step 1, in all possible ways, into two groups.

step 3: For each combination tested in step 2, compute the MST for each of its two created groups.
step 4: Output the best tested combination

Theorem 1. *For $k = 2$ the approximation algorithm* CA *produces a partition which is within a factor $\frac{4}{3} + \varepsilon$ of the optimal in time $O(n \log n)$.*

Proof. Let V_1 and V_2 be the partition obtained from CA. Assume that $\mathcal{S}_1$ and $\mathcal{S}_2$ is the optimal partition, and let e be the shortest edge connecting $\mathcal{S}_1$ with $\mathcal{S}_2$. According to Lemma 3 it follows that $w \leqslant 2/(4/\varepsilon')|M(\mathcal{S})| = \frac{\varepsilon'}{2}|M(\mathcal{S})|$. We will have two cases, $|e| > w$, and $|e| \leqslant w$, which are illustrated in Figure 2 (a) and Figure 2 (b), respectively. In the first case every component is a subset of either

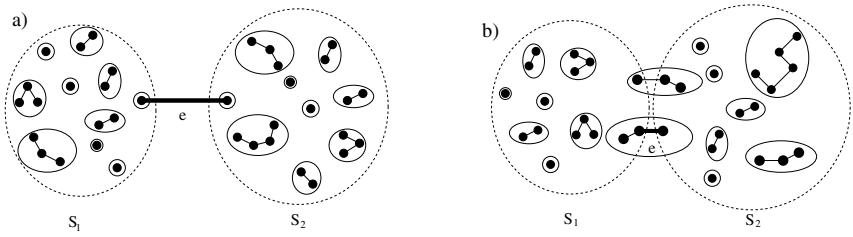

Fig. 2. The two cases for CA, $k = 2$. The edge e (marked) is the shortest edge connecting $\mathcal{S}_1$ with $\mathcal{S}_2$

$\mathcal{S}_1$ or $\mathcal{S}_2$. This follows since a component consisting of points from both $\mathcal{S}_1$ and $\mathcal{S}_2$ must include an edge with weight greater than w. Thus, no such component can exist among the components created in step 1. Further, this means that the partition $\mathcal{S}_1$ and $\mathcal{S}_2$ must have been tested in step 2 of CA and, hence, the optimal solution must have been found.

In the second case, $|e| \leqslant w$, there may exist components consisting of points from both $\mathcal{S}_1$ and $\mathcal{S}_2$, see Fig. 2. To determine an upper bound of the approximation factor we start by examining an upper bound of CA. The dividing process in step 1 of CA starts with $M(\mathcal{S})$ being divided into 2 components $M(\mathcal{S}'_1)$ and $M(S'_2)$, such that $\max(|M(\mathcal{S}'_1)|, |M(\mathcal{S}'_2)|) \leqslant \frac{2}{3}|M(\mathcal{S})|$. These two components are then divided into several smaller components. This immediately reveals an upper bound of $|CA| \leqslant \frac{2}{3}|M(S)|$. Next the lower bound is examined. We have:

$$|opt| \geqslant \frac{|M(\mathcal{S})| - |e|}{2} \geqslant \frac{|M(\mathcal{S})|}{2} - \frac{\varepsilon' \cdot M(\mathcal{S})}{2} \geqslant (1 - \varepsilon')\frac{M(\mathcal{S})}{2}.$$

Then, if the upper and lower bound are combined we get:

$$|CA|/|opt| \leqslant \frac{\frac{2}{3}|M(\mathcal{S})|}{(1 - \varepsilon')\frac{M(\mathcal{S})}{2}} \leqslant \frac{4/3}{1 - \varepsilon'} \leqslant 4/3 + \varepsilon.$$

In the third inequality we used the fact that $\varepsilon' \leqslant \frac{\varepsilon}{4/3+\varepsilon}$.

Next consider the complexity of CA. In step 1 $M(\mathcal{S})$ is divided into a constant number of components using VP. This takes $O(n)$ time. Then, in step 2, these components are combined in all possible ways. This takes $O(1)$ time since there are a constant number of components. For each tested combination there is a constant number of MST's to be computed in step 3. Further, since there are a constant number of combinations and $M(\mathcal{S})$ takes $O(n \log n)$ to compute, step 3 takes $O(n \log n)$ time. □

Next consider $k \geqslant 3$. In this case the following steps are performed:

step 1: Compute $M(\mathcal{S})$ and remove the $k-1$ heaviest edges $e_1, \ldots, e_{k-1}$ of $M(\mathcal{S})$, thus resulting in k separate trees $M(U'_1), \ldots, M(U'_k)$.

step 2: Divide each of the trees $M(U'_1), \ldots M(U'_k)$ into $\frac{k \cdot C}{\varepsilon'}$ components, using RVP. C is a positive constant and $\varepsilon' = \frac{\varepsilon}{2+\varepsilon}$. The reason for the value of ε' will become clear below. Denote the resulting components $M(U_1), \ldots, M(U_r)$, where $r = \frac{k \cdot C}{\varepsilon'} \cdot k$. Further set $w = \max\{|M(U_1)|, \ldots, |M(U_r)|\}$.

step 3: Combine $U_1, \ldots, U_r$ in all possible ways into $1, \ldots, k$ groups.

step 4: For each such combination do:

- Compute the MST for each of its corresponding groups.
- Divide each such MST in all possible ways, using RVP. That is, each MST is divided into $1, \ldots, i (i \leq k)$ components, such that the total number of components resulting from all the divided MST's equals k. Each such division defines a partition of $\mathcal{S}$ into k subsets.

step 5: Of all the tested partitions in step 4, output the best.

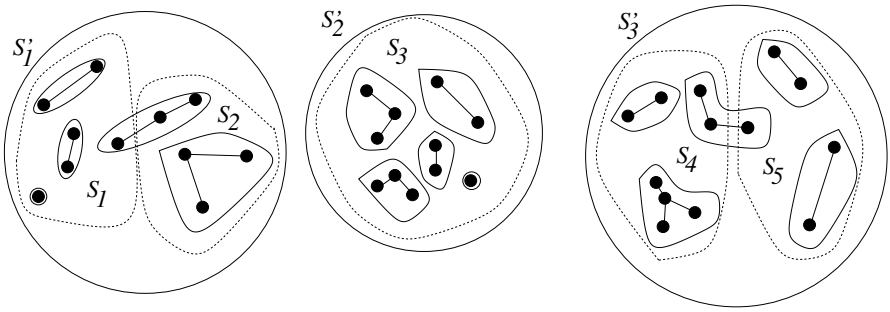

Fig. 3. $\mathcal{S}_1, \ldots, \mathcal{S}_k$ is an optimal partition of $\mathcal{S}$. All subsets that can be connected by edges of length at most w are merged, thus creating the new set $\mathcal{S}'_1, \ldots, \mathcal{S}'_{k'}$

Theorem 2. *For $k \geqslant 3$ the approximation algorithm CA produces a partition which is within a factor of $2+\varepsilon$ of the optimal in time $O(n \log n)$*

Proof. The time complexity CA is the same as for the case $k = 2$. This follows as a constant number of components are created and a constant number of combinations and partitions are tested, hence the time complexity is $O(n \log n)$.

To prove the approximation factor we first give an upper bound on the weight of the solution produced by CA and then we provide a lower bound for an optimal solution. Combining the two results will conclude the theorem.

Consider an optimal partition of $\mathcal{S}$ into k subsets $\mathcal{S}_1, \ldots, \mathcal{S}_k$. Merge all subsets that can be connected by edges of length at most w. From this we obtain the sets $\mathcal{S}'_1, \ldots, \mathcal{S}'_{k'}$, where $k' \leqslant k$ (see Figure 3). Let m'_i denote the number of elements from $\mathcal{S}_1, \ldots, \mathcal{S}_k$ included in S'_i. The purpose of studying these new sets is that every component created in step 2 of CA belongs to exactly one element in $\mathcal{S}'_1, \ldots, \mathcal{S}'_{k'}$. A direct consequence of this is that a combination into k' groups equal to $\mathcal{S}'_1, \ldots, \mathcal{S}'_{k'}$ must have been tested in step 3.

Step 4 guarantees that $M(\mathcal{S}'_1), \ldots, M(\mathcal{S}'_{k'})$ will be calculated, and that these MST's will be divided in all possible ways. Thus, a partition will be made such that each $M(\mathcal{S}'_i)$ will be divided into exactly m'_i components. This partitions $\mathcal{S}$ into k subsets $\mathcal{V}_1, \ldots, \mathcal{V}_k$. Let $\mathcal{V}$ be a set in $\mathcal{V}_1, \ldots, \mathcal{V}_k$ such that $|M(\mathcal{V})| = \max_{1 \leqslant i \leqslant k}(|M(\mathcal{V}_i)|)$. We wish to restrict our attention to exactly one element of the set $\mathcal{S}'_1, \ldots, \mathcal{S}'_{k'}$. Thus, we note that $\mathcal{V}$ is a subset of exactly one element $\mathcal{S}'$ in $\mathcal{S}'_1, \ldots, \mathcal{S}'_{k'}$. Assume that $M(\mathcal{V})$ was created in step 4 when $M(\mathcal{S}')$ was divided into m' components using RVP. Thus, $M(\mathcal{V}) \leqslant \frac{2}{m'}|M(\mathcal{S}')|$, according to Lemma 3. Since the partition $\mathcal{V}_1, \ldots, \mathcal{V}_k$ will always be tested we have that $|CA| \leqslant |M(\mathcal{V})| \leqslant \frac{2}{m'}|M(\mathcal{S}')|$.

Next a lower bound of an optimal solution is examined. Let $|opt'|$ be the value of an optimal solution for $\mathcal{S}'$ partitioned into m' subsets. Note that $\mathcal{S}'$ consists of m' elements from $\mathcal{S}_1, \ldots, \mathcal{S}_k$. Assume w.l.o.g that $\mathcal{S}' = \mathcal{S}_1 + \ldots + \mathcal{S}_{m'}$. This means that $\mathcal{S}_1, \ldots, \mathcal{S}_{m'}$ is a possible partition of $\mathcal{S}'$ into m' subsets. Thus, $|opt| \geqslant \max_{1 \leqslant i \leqslant m'}(|M(S_i)|) = |opt'|$. Assume w.l.o.g. that $e'_1, \ldots, e'_{m'-1}$ are the edges in $M(\mathcal{S})$ connecting the components in $\mathcal{S}'$. We have:

$$|opt| \geqslant |opt'| \geqslant \frac{1}{m}(|M(S')| - \sum_{i=1}^{m'-1} |e'_i|) \geqslant \frac{1}{m}(|M(S')| - (m'-1)w) \qquad (1)$$

To obtain a useful bound we need an upper bound on w. Consider the situation after step 1 has been performed. We have $\max_{1 \leqslant i \leqslant k}(|M(U'_i)|) \leqslant |M(\mathcal{S})| - \sum_{i=1}^{k-1} |e_i|$. Since each U'_i is divided into $\frac{k \cdot C}{\varepsilon'}$ components we have that the resulting components, and therefore also w, have weight at most $2/(\frac{k \cdot C}{\varepsilon'}) \cdot (|M(\mathcal{S})| - \sum_{i=1}^{k-1} |e_i|)$, according to Lemma 3. Using the above bound gives us:

$$\frac{w}{|opt|} \leqslant \frac{2/(\frac{k \cdot C}{\varepsilon'}) \cdot (|M(\mathcal{S})| - \sum_{i=1}^{k-1} |e_i|)}{\frac{1}{k}(|M(\mathcal{S})| - \sum_{i=1}^{k-1} |e_i|)} \leqslant \frac{2 \cdot \varepsilon'}{C} \Rightarrow w \leqslant \frac{2 \cdot \varepsilon'}{C}|opt| \qquad (2)$$

Setting $C \geqslant 2$ and combining 1 and 2 gives us:

$$|opt| \geqslant \frac{1}{m}\left(|M(S')| - (m'-1)\frac{2 \cdot \varepsilon'}{C}|opt|\right) \geqslant (1 - \varepsilon')\frac{|M(\mathcal{S}')|}{m'}.$$

Combining the two bounds together with the fact that $\varepsilon' \leqslant \varepsilon/(2+\varepsilon)$ concludes the theorem.

$$|CA|/|opt| \leqslant \frac{\frac{2}{m'}|M(\mathcal{S}')|}{(1-\varepsilon')\frac{|M(\mathcal{S}')|}{m'}} \leqslant \frac{2}{1-\varepsilon'} \leqslant 2+\varepsilon.$$

□

4 Conclusion

In this paper it was first showed that the k-BPMST problem is NP-hard. After this had been determined the continued approach was to find an approximation algorithm for the problem. The algorithm is based on partitioning the point set into a constant number of smaller components and then trying all possible combinations of these small components. This approach revealed a $4/3+\varepsilon$ approximation in the case $k=2$, and an $2+\varepsilon$ approximation in the case $k \geqslant 3$. The time complexity of the algorithm is $O(n \log n)$.

References

1. M. Andersson. Balanced Partition of Minimum Spanning Trees, LUNDFD6/NFCS-5215/1–30/2001, Master thesis, Department of Computer Science, Lund University, 2001.
2. B. Shaleooi. Algoritmer för plåtskärning (*Eng. transl.* Algorithms for cutting sheets of metal), LUNDFD6/NFCS-5189/1–44/2001, Master thesis, Department of Computer Science, Lund University, 2001.
3. M. R. Garey and D. S. Johnson. Computers and Intractability: A guide to the theory of NP-completeness, W. H. Freeman and Company, San Francisco, 1979.
4. E. M. Arkin and R. Hassin. Approximation algorithms for the geometric covering salesman problem. Discrete Applied Mathematics, 55:197–218, 1994.
5. A. Dumitrescu and J. S. B. Mitchell. Approximation algorithms for TSP with neighborhoods in the plane. In *Proc. 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2001.
6. M. R. Garey, R. L. Graham and D. S. Johnson. Some NP-complete geometric problems. In *Proc. 8th Annual ACM Symposium on Theory of Computing*, 1976.
7. J. Gudmundsson and C. Levcopoulos. A fast approximation algorithm for TSP with neighborhoods. Nordic Journal of Computing, 6:469-488, 1999.
8. J. Gudmundsson and C. Levcopoulos. Hardness Result for TSP with Neighborhoods, Technical report, LU-CS-TR:2000-216, Department of Computer Science, Lund University, Sweden, 2000.
9. C. Mata and J. S. B. Mitchell. Approximation algorithms for geometric tour and network design problems. In *Proc. 11th Annual ACM Symposium on Computational Geometry*, pages 360–369, 1995.
10. J. S. B. Mitchell. Guillotine Subdivisions Approximate Polygonal Subdivisions: A Simple Polynomial-Time Approximation Scheme for Geometric TSP, k-MST, and Related Problems. SIAM Journal on Computing, 28(4):1298–1309, 1999.