Author/s:
Andersen, PJ;Ras, CJ

Title:
Minimum bottleneck spanning trees with degree bounds

Date:
2016-12-01

Citation:
Andersen, P. J. & Ras, C. J. (2016). Minimum bottleneck spanning trees with degree bounds. Networks, 68 (4), pp.302-314. https://doi.org/10.1002/net.21710.

Persistent Link:
https://hdl.handle.net/11343/291785

Minimum Bottleneck Spanning Trees with Degree Bounds

Patrick J. Andersen* and Charl J. Ras

School of Mathematics and Statistics, The University of Melbourne, Australia

September 2, 2016

Abstract

Given a graph G with edge lengths, the *minimum bottleneck spanning tree* (MBST) problem is to find a spanning tree where the length of the longest edge in tree is minimum. It is a well-known fact that every minimum spanning tree (MST) is a minimum bottleneck spanning tree. In this paper, we introduce the δ-MBST problem, which is the problem of finding an MBST such that every vertex in the tree has degree at most δ. We show that optimal solutions to the similarly defined δ-MST problem are not necessarily optimal solutions to the δ-MBST, and we establish that the δ-MBST problem is NP-complete for any $\delta \geq 2$. We show that when edge lengths of the graph are Euclidean distances between points in the plane, the problem is NP-hard for $\delta = 2$ and 3, and tractable for $\delta \geq 5$. We give a dual approximation scheme for the general graph version of the problem which is the best possible with respect to feasibility. For the Euclidean version, we give a $\sqrt{3}$-factor approximation algorithm for the 4-MBST. We also give a 2-factor algorithm for the Euclidean 3-MBST and a 3-factor approximation algorithm for the general Euclidean δ-MBST, both of which can be generalised to metric spaces.

Keywords: Minimum spanning trees, bottleneck objective, approximation algorithms, discrete geometry, bounded degree, combinatorial optimisation

*Email: pat.j.andersen@gmail.com; Phone: +61-413579238; Corresponding author

1 Introduction

The topic of spanning tree problems has been of much interest in combinatorial optimisation and network design due to its vast number of applications in areas such as telecommunication and computer networking, transportation, plumbing, and electrical circuit design [11]. These problems traditionally arise in scenarios where it is required that some set of terminals is joined together in an optimal way so that each terminal is part of a connected network. In the classic *minimum spanning tree* (MST) problem, we wish to connect the terminals together using edges such that the sum of the lengths of the edges is minimum. Polynomial time algorithms exist that can solve this problem exactly (see Kruskal [16], Prim [20]). In the related *minimum bottleneck spanning tree* (MBST) problem, we wish to connect the terminals with edges such that the length of a longest edge in the network is minimum. Bottleneck objectives for network optimisation problems are often studied by researchers due to their applications in areas such as VLSI layout, communication network design, and location problems [21]. For example, in wireless sensor networks, the power required for signal transmission increases as the distance between sensors increases, and since these sensors are often battery powered, the lifetime of the network is determined by the length of the longest edge [2]. Polynomial time algorithms specifically for the MBST problem were first introduced by Camerini [3] in which it was noted that algorithms for the MST problem can also be used for the bottleneck version. The classic *travelling salesman problem* (TSP) also has a bottleneck version in which the objective is to find a Hamiltonian cycle whose longest edge is of minimum length. Parker and Rardin [19] give a 2-factor approximation algorithm for this problem when restricted to edge lengths that satisfy the triangle inequality.

Often in real-world network problems, additional constraints are required for spanning tree problems. One such constraint is the degree bound constraint, which states that the number of edges incident on any terminal must be at most some constant δ, where δ is the *degree bound*. When this constraint is added to the MST problem (we refer to this as the *δ-MST problem*), it becomes an NP-complete problem for any degree bound $\delta \geq 2$ [9]. In fact, it was shown by Papadimitriou and Vazirani [18] that even when the terminals lie in the plane and the length of an edge is the Euclidean distance between its endpoints (we refer to this variant as the *δ-EMST problem*), the problem is NP-hard when $\delta = 2$ or 3, but tractable for $\delta \geq 5$. The complexity of the 4-EMST problem remained an open problem for over 20 years until it was shown by Francke and Hoffman [8] to also be NP-hard.

Approximation algorithms for the δ-MST and its variants have been well studied in the literature, and so we will only mention some key results. For the general δ-MST problem, simply finding a feasible solution (i.e., a spanning tree that satisfies the degree bound δ) is NP-complete [9] and so there is no polynomial time approximation algorithm unless P=NP. Hence, most of the algorithms for approximating the δ-MST problem in the literature are what are sometimes called *dual approximation algorithms*, which involve finding near feasible spanning trees of minimum cost. Singh and Lau [22] give a polynomial time algorithm that either determines that there is no δ-MST of a graph G, or it finds a spanning tree T of G, where no vertex in T has degree greater than $\delta + 1$, and where the sum of the weights of the edges of T is no greater than the sum of the weights of the edges of a δ-MST for G (if one exists). For the δ-EMST problem, polynomial time constant factor approximation algorithms are known, where the factor is given as the maximum ratio between a spanning tree found by the algorithm and the optimal MST [15].

For $\delta = 2$, the problem is equivalent to the Euclidean version of the *travelling salesman path problem (TSPP)*, which can be solved using existing Euclidean TSPP approximation algorithms. One such example is the 3/2-factor approximation algorithm of Hoogeveen [12], which is a modified Christofides' TSP algorithm [5]. For $\delta = 3$, Khuller, Raghavachari and Young [15] give a polynomial time approximation algorithm whose ratio is at most 1.5, a result which was later improved upon by Chan [4], where he describes an algorithm for the 3-EMST problem whose ratio is at most 1.402. In the same paper, Chan also gives an approximation algorithm for the 4-EMST problem, which was later proven by Jothi and Raghavachari [13] to have an approximation ratio of at most 1.1381. The authors of [7] provide an approximation algorithm based upon network flows that has an approximation ratio of at most $2-(\delta-2)/(d_{\max}(T)-2)$ for trees in metric spaces, where $d_{\max}(T)$ is the largest degree of a vertex in a given minimum spanning tree T for the problem instance and can be assumed to be at most 5 for the Euclidean metric. Arora and Chang [1] also give a quasi-polynomial time approximation scheme (QPTAS) for the δ-EMST problem.

Our results: We introduce the δ-MBST problem, which is the problem of finding a spanning tree whose longest edge is of minimum length and where the degree of each vertex in the tree is at most δ, and the related δ-EMBST problem which, like the δ-EMST problem, uses points in the plane and the Euclidean distances between points. We show that the δ-MBST problem is NP-complete

for any $\delta \geq 2$ when formulated as a general graph problem and we prove that the δ-EMBST problem is NP-complete for $\delta = 2$ or 3, and that neither of these δ-EMBST problems admits a PTAS unless P=NP. We also give a scheme that allows the use of existing dual approximation algorithms for the δ-MST to solve the δ-MBST problem, and we give constant factor approximation algorithms for the δ-EMBST problem based upon existing constant factor approximation algorithms for δ-EMST problems. The computational complexity of the 4-EMBST problem remains an open problem.

2 Graph Formulation

Let $G = (V, E)$ be a connected, undirected graph where $|V| = n$. Let $w : E \to \mathbb{R}$ be a weight function on the edges of G. For simplicity, we can assume that w is strictly positive since for the problems studied here, we can always add a constant value to w to obtain an equivalent problem with a strictly positive weight function. For an edge $e \in E$, we will often refer to $w(e)$ as the *length* of e. Let $T = (V, E' \subseteq E)$ be a spanning tree of G, i.e., T is a connected subgraph of G that contains every vertex in V, but contains no cycles. In the traditional minimum spanning tree problem, we wish to find a T such that $\sum_{e \in E'} w(e)$ is minimum. To further illustrate the problem, we give an integer linear programming formulation of the problem, adapted from [10]. For a set of vertices $S \subseteq V$, let $E(S)$ denote the set of edges in E that have both endpoints in S, i.e., $E(S) = \{(e_1, e_2) \in E : e_1, e_2 \in S\}$. For each $e \in E$, let x_e be a binary decision variable that is unit valued when the edge e is in the spanning tree, and zero otherwise. The integer program for the minimum spanning tree problem is

$$\text{minimise} \sum_{e \in E} w(e) x_e \tag{1}$$

$$\text{s.t.} \sum_{e \in E} x_e = |V| - 1 \tag{2}$$

$$\sum_{e \in E(S)} x_e \leq |S| - 1 \quad \forall S \subset V \tag{3}$$

$$x_e \in \{0, 1\} \qquad \forall e \in E \tag{4}$$

In the *minimum bottleneck spanning tree problem*, we also wish to find a spanning tree T that is minimum with respect to an objective function; however our objective function is different. Let e^* be an edge in T with the largest weight, i.e., $w(e^*) \geq w(e) \quad \forall e \in E'$. We say that e^* is a *bottleneck edge* of T and that

$w(e^*)$ is the *bottleneck length* or *bottleneck value* of T. In the MBST, we wish to find a T such that the bottleneck length is minimum. By modifying the objective function (1), we can obtain a formulation of this problem as

$$\text{minimise } \max_{e \in E}\{w(e)x_e\} \tag{5}$$

$$\text{s.t. } \sum_{e \in E} x_e = |V| - 1 \tag{6}$$

$$\sum_{e \in E(S)} x_e \leq |S| - 1 \quad \forall S \subset V \tag{7}$$

$$x_e \in \{0, 1\} \qquad \forall e \in E \tag{8}$$

This formulation is not linear; however we can obtain a linear formulation by adding a new decision variable z to represent the bottleneck value. Our formulation then becomes the mixed integer linear program

$$\text{minimise } z \tag{9}$$

$$\text{s.t. } z \geq w(e)x_e \qquad \forall e \in E \tag{10}$$

$$\sum_{e \in E} x_e = |V| - 1 \tag{11}$$

$$\sum_{e \in E(S)} x_e \leq |S| - 1 \quad \forall S \subset V \tag{12}$$

$$x_e \in \{0, 1\} \qquad \forall e \in E \tag{13}$$

$$z \in \mathbb{R} \tag{14}$$

As long as G is connected, both the formulations (1) - (4) and (9) - (14) will have feasible solutions, and an optimal solution for (9) - (14) can be directly obtained from the optimal solution for (1) - (4). This is due to the connection between these problems and the problem of finding bases of weighted matroids. Let $M(G)$ denote the graphic matroid whose independent sets are the subsets of E whose edge induced subgraphs are forests. Since the edge sets of the spanning trees of a connected graph G are the bases of $M(G)$, the problem of finding an MST is equivalent to finding a minimum weight basis of $M(G)$. Due to the properties of matroids, we can use a greedy algorithm to find a minimum weight basis for $M(G)$. Such an algorithm always minimises the largest element and hence every MST of G is an MBST of G, although the converse is not necessarily true.

We wish to consider only the spanning trees whose vertices satisfy the degree bound $\delta \in \mathbb{N}$, i.e., for each vertex v in our desired spanning tree, the number of edges incident on v is no more than δ. We can alter our formulations to reflect this degree bound by adding the constraints

$$\sum_{e \in E(\{v\})} x_e \leq \delta \quad \forall v \in V \tag{15}$$

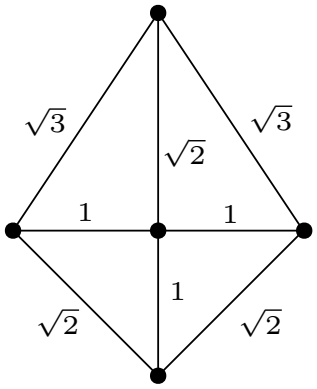
to the formulations (1) - (4) and (9) - (14). It should be noted that adding the constraints (15) to the formulations may result in an infeasible problem, as although every connected graph has spanning tree, a connected graph may not necessarily have a spanning tree that satisfies a specified degree bound. Also, once we add the condition that the degree of any vertex in the spanning tree does not exceed δ, then the edge sets of the spanning trees of G that satisfy this property can no longer be thought of as the bases of a graphic matroid derived from G. As a consequence, for a given δ, the set of optimal solutions for the δ-MST problem for a graph G may be disjoint from the set of optimal solutions for the δ-MBST problem for G. Fig. 1 shows an example in which this is the case, where $\delta = 3$ and the weight of each edge in the graph is given as the Euclidean distance between the edge's endpoints.

2.1 Complexity

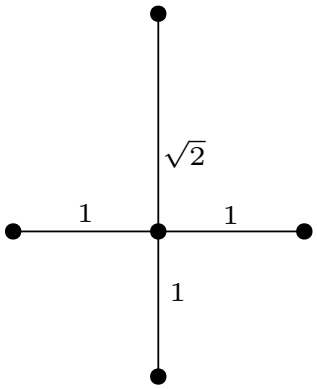
Garey and Johnson [9] state that given a graph $G = (V, E)$ and a positive integer $\delta \leq |V| - 1$, it is an NP-complete problem to determine whether or not there is a spanning tree of G with no vertex having degree greater than δ. Furthermore, this problem remains NP-complete for any fixed $\delta \geq 2$. This directly implies that the δ-MBST problem is NP-complete in general as it is an NP-complete problem to determine whether or not a specific instance of the δ-MBST problem has a feasible solution. Since the details of Garey and Johnson's reduction are omitted in [9], we provide our proof of the NP-completeness of our problem.

Theorem 1. *The δ-MBST problem is NP-complete for any fixed degree bound $\delta \geq 2$.*

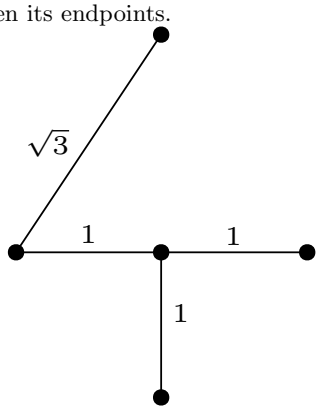
Proof. We use a similar approach to that outlined in [18] in order to justify the NP-completeness of the δ-MST problem, although we perform our reduction directly from the Hamiltonian path problem instead of the TSPP. Starting with a graph $G = (V, E)$, it is NP-complete to determine whether or not there is a Hamiltonian path between some pair of vertices in V. To transform this into an instance of the δ-MBST problem, for every vertex $v \in V$, we add $\delta - 2$ vertices



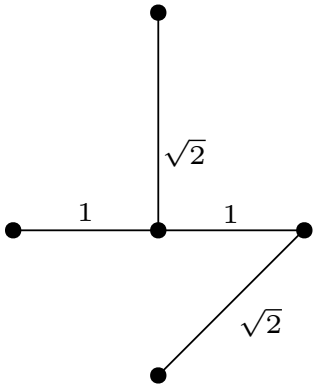
(a) The graph G, where edges are labelled by their respective weights, and where the weight of an edge is the Euclidean distance between its endpoints.



(b) The unique MST for G.



(c) A 3-MST for G, where the total edge weight is approximately 4.732.



(d) A 3-MBST for G, where the total edge weight is approximately 4.828.

Figure 1: An example of a graph G embedded in the Euclidean plane where a 3-MST is not a 3-MBST.

to V, which we refer to as the *additional neighbour vertices* of v, and we add an edge from v to each of its additional neighbour vertices. Hence, every vertex in V will have increased its degree by $\delta - 2$ and every additional neighbour vertex will have degree 1. We denote the resultant graph by $G^* = (V \cup V^*, E \cup E^*)$, where V^* and E^* are the added vertices and edges, and we assign a unit weight to every edge in $E \cup E^*$. We claim that there exists a δ-MBST of G^* of unit value if and only if G contains a Hamiltonian path. To see this, note that a spanning tree of G^* must use every edge in E^* since each of the additional neighbour vertices only has degree 1. This implies that for a δ-MBST to exist, each vertex

in V can only be incident to at most two edges from E in the δ-MBST since the degree bound is δ. Therefore, if a δ-MBST exists for G^*, each vertex in V must belong to a Hamiltonian path in G. Conversely, if G has a Hamiltonian path, then we can construct a δ-MBST for G^*, where the edge set of the tree is the union of the edges in the Hamiltonian path with E^*. Since each vertex in V is incident to at most $\delta - 2$ edges from E^* and at most two edges from the Hamiltonian path, it can be seen that every vertex in this tree will have degree at most δ. □

2.2 Dual Approximation Algorithms

Since it has been shown that it is an NP-complete problem to find a feasible spanning tree of a general graph, where the vertices of the tree satisfy a given degree bound $\delta \geq 2$, we cannot find any polynomial approximation algorithm for the δ-MBST problem in general graphs unless P = NP. Hence, we wish to find a dual approximation algorithm that will return a near feasible δ-MBST in polynomial time, where length of the longest edge in the returned tree is as close as possible to the length of the longest edge in an optimal δ-MBST. It turns out that by using existing dual approximation algorithms for the δ-MST problem, we can obtain the desired algorithms for the δ-MBST problem. This is based upon the ideas of the threshold algorithm of Edmonds and Fulkerson [6], which is an exact algorithm for bottleneck problems. Our dual approximation scheme can be thought of as a dual approximation version of the threshold algorithm, specifically used for the δ-MBST problem. To simplify our arguments, we will give the properties a dual approximation algorithm must satisfy in order for it to be considered a feasible algorithm for our dual approximation scheme.

For a graph $G = (V, E)$, we define an *edge supergraph* of G to be any graph $G' = (V, E')$, where $E \subseteq E'$. An *edge subgraph* is similarly defined with $E' \subseteq E$. We say that a dual approximation algorithm A is a *feasible* $(\delta + k)$*-algorithm*, where $k \in \mathbb{N}$, if it has polynomial time complexity, and if for any finite input graph G, A satisfies the following properties.

1. A terminates.

2. If G admits a feasible δ-MST, then A returns a feasible $(\delta + k)$-MST.

3. If A returns a feasible $(\delta + k)$-MST for G, then A will return a feasible $(\delta + k)$-MST when given an edge supergraph of G as input.

Suppose we have a feasible $(\delta+k)$-algorithm A as described above and an input graph $G=(V,E)$ that admits a feasible δ-MST. Let $w : E \to \mathbb{R}_+$ be the weight function for the edges of G. For a given positive integer l, let $G_{\leq l}$ denote the subgraph of G induced by the set of edges $\{e \in E : w(e) \leq l\}$. Let $A(H)$ denote the output returned by A when it is given the input graph H. We assume that $A(H)$ is either a $(\delta+k)$-MST or the empty set, where having $A(H)=\emptyset$ implies that H does not admit a δ-MST and the algorithm gave no output (note that if a graph H does not admit a δ-MST, then we do not guarantee that $A(H)=\emptyset$ as this would mean that we could use A to solve the NP-complete problem of determining the existence of a feasible $(\delta+k)$-MST). We describe our dual approximation scheme for the δ-MST problem as Algorithm 1.

Algorithm 1

Input: A feasible $(\delta+k)$-algorithm A and input graph $G=(V,E)$.

Let L be the sorted array of the lengths of the edges of E ordered by non-decreasing value, where $L[i]$ denotes the i-th element of L.
if $|L|=0$
  Return $\emptyset$.
while $|L|>1$
  Let $m := \lfloor \frac{|L|}{2} \rfloor$.
  Let $H := A(G_{\leq L[m]})$.
  if $H=\emptyset$
    $L := \{L[m+1], L[m+2], \ldots, L[|L|]\}$.
  else $L := \{L[1], L[2], \ldots, L[m]\}$.
Return $A(G_{\leq L[1]})$.

Our scheme performs a binary search on the unique edge lengths of G in order to find the smallest edge length l such that $A(G_{\leq l})$ is a feasible $(\delta+k)$-MST. We prove the correctness of this approach in the following theorem.

Theorem 2. *Let A be a feasible $(\delta+k)$-algorithm with complexity $P(n)$, where n is the number of vertices in the input and P is a polynomial function. The dual approximation scheme given by Algorithm 1, with algorithm A and graph G as input, is a feasible $(\delta+k)$-algorithm such that if the scheme returns a feasible $(\delta+k)$-MST of G, denoted T, then the bottleneck value of T is no worse than*

that of the optimal δ-MBST for G. Furthermore, the scheme with algorithm A has complexity $O(P(n)log(n))$.

Proof. Note that although the theorem statement gives an objective value guarantee for Algorithm 1, the algorithm A does not require such a guarantee, it only needs to be a feasible $(\delta + k)$-algorithm. As mentioned previously, Algorithm 1 performs a binary search on the lengths of edges of the input graph in order to find the first edge length l such that $A(G_{\leq l}) \neq \emptyset$ (if it exists). If we think of $A(G_{\leq l})$ as a function of l, then for our binary search to be a valid means of finding the smallest l such that $A(G_{\leq l}) \neq \emptyset$, this function must be "monotonic", i.e., for a given value l, the following must be true.

1. If $A(G_{\leq l})$ is a feasible $(\delta+k)$-MST, then $A(G_{\leq l^+})$ is a feasible $(\delta+k)$-MST for all $l^+ > l$.

2. If $A(G_{\leq l}) = \emptyset$, then $A(G_{\leq l^-}) = \emptyset$ for all $l^- < l$.

The first criterion holds by the definition of A since $G_{\leq l^+}$ is an edge supergraph of $G_{\leq l}$. Similarly, the second criterion holds since $G_{\leq l}$ is an edge supergraph of $G_{\leq l^-}$. Hence the binary search is valid and we get the worst case time complexity of the scheme as $O(P(n)\log(n))$. It is follows from the validity of the binary search that the scheme yields a feasible $(\delta + k)$-algorithm since A is a feasible $(\delta + k)$-algorithm.

Finally, if G permits a feasible δ-MBST and the bottleneck length in an optimal δ-MBST for G is l, then $G_{\leq l}$ permits a feasible δ-MBST and so $A(G_{\leq l})$ will return a feasible $(\delta + k)$-MST for $G_{\leq l}$. Note that $A(G_{\leq l})$ is a feasible $(\delta + k)$-MST for G also since $G_{\leq l}$ is an edge subgraph of G. However, since $G_{\leq l}$ does not contain any edge whose length is greater than l, $A(G_{\leq l})$ will also not contain any such edge, therefore $A(G_{\leq l})$ is a feasible $(\delta + k)$-MBST for G, where the longest edge in the spanning tree is no longer than l. Since the binary search of Algorithm 1 finds the smallest length l' such that $A(G_{\leq l'})$ is a feasible $(\delta + k)$-MST, it can be seen that the scheme will return a $(\delta + k)$-MBST of G where the bottleneck length of the tree is no greater than l. □

The polynomial time dual approximation algorithm described by Singh and Lau [22] is an example of one such algorithm A that satisfies the criteria of Theorem 2. The general idea of this algorithm is that it solves an LP-relaxation of an integer programming formulation of the δ-MST with decision variables representing the edges of the input graph, and then it iteratively refines the

problem by fixing a subset of the decision variables and removing degree bound constraints until an integer solution is recovered. For this algorithm, Singh and Lau have $k = 1$, which is clearly the smallest possible value of k if P $\neq$ NP. We note that if this process succeeds in finding a $(\delta+1)$-MST for an edge subgraph of a graph G, then it will succeed in finding a $(\delta + 1)$-MST for G since the feasible region of the the LP formulation of the problem for an edge subgraph of G is a subset of the feasible region of the LP formulation for G. Thus the algorithm can be seen to satisfy the criteria of Theorem 2 and hence we have shown the following statement.

Corollary 1. *There exists a polynomial time dual approximation algorithm to find a $(\delta + 1)$-MBST for an input graph G, provided G permits a feasible δ-MBST, where the longest edge in the output tree is no longer than the longest edge in an optimal δ-MBST for G.*

3 Geometric Formulation

In this section, we look at certain planar geometric versions of the MBST and δ-MBST problems. Given a set of points P in the plane and a distance function $d(p_1, p_2) \in \mathbb{R}_+$, where $p_1, p_2 \in P$, our goal is to find a MBST or δ-MBST of the complete graph with P as its vertices, where the weight of the edge (p_1, p_2) is simply the distance $d(p_1, p_2)$. Clearly this problem can be thought of as special case of the general graph version of the problem. For the remainder of this section, we will focus on the case where the points in P are points with integer coordinates in the Cartesian plane and where $d(p_1, p_2)$ is the Euclidean distance between p_1 and p_2, i.e., we will focus on the δ-EMBST problem. Some results will be generalised from Euclidean distance to the L_p metric which we define later, where Euclidean distance is the L_2 metric. Other results can be generalised to point sets in any metric space.

It can be shown using trigonometry that for an EMST, no vertex will have two incident edges that meet at an angle of less than 60°, since otherwise, we could swap one of the two incident edges with another edge to get a smaller valued spanning tree. This implies that no EMST has a vertex of degree greater than 6. In fact, it has been shown [17] that there always exists an EMST in the plane that has no vertex of degree greater than 5. Therefore, one can find a δ-EMST when $\delta \geq 5$ by finding an EMST, and since an MST is always a MBST, one can find a δ-EMBST when $\delta \geq 5$ by also finding an EMST. Hence the δ-EMBST problem is tractable when $\delta \geq 5$ and the only cases where a δ-EMST

may not be a solution for the δ-EMBST problem is when $\delta = 2, 3$ or 4. For the remainder of this section, we will assume $\delta \in \{2, 3, 4\}$.

3.1 Complexity

In this subsection, we will show that both the 2-EMBST and 3-EMBST problems are NP-complete. The decision version of the δ-EMBST problem can be easily seen to be in NP since given a spanning tree, we can verify in polynomial time whether or not the spanning tree is a feasible δ-EMBST, and we can determine in polynomial time whether or not its longest edge is smaller than some given bound. As is the case for Papadimitriou and Vazirani's NP-hardness proof for the 3-EMST problem, our proofs use a reduction from the Hamiltonian path problem in grid graphs with maximum degree 3.

We define a *grid graph* to be any finite connected vertex-induced subgraph of the infinite 2-dimensional grid embedded in the plane so that the vertices are at integer coordinates and vertices are adjacent if and only if they have unit distance from each other (see Fig. 2). We have the following theorem [18].

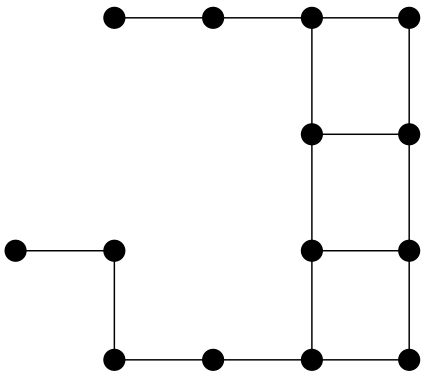


Figure 2: An example grid graph with a maximum degree of 3.

Theorem 3. *The Hamiltonian path problem for grid graphs with maximum degree 3 is NP-complete.*

We first use a simple reduction from the aforementioned Hamiltonian path problem to show that the problem of determining for some k, whether or not there is a spanning path on a given set of points in the plane whose longest edge has length at most k, is NP-complete.

Theorem 4. *The Euclidean bottleneck travelling salesman path problem (2-EMBST problem) is NP-complete.*

Proof. Suppose we are given a connected grid graph $G = (V, E)$ with maximum degree 3. Since G is a grid graph, its vertices are embedded in the plane and all its edges have unit length under the Euclidean metric. Let T be a 2-EMBST for V and let b be the length of its longest edge. The shortest distance between any pair of vertices in V is 1 which only occurs if the vertices are adjacent in G. If G has a Hamiltonian path, then this is an optimal solution to the 2-EMBST problem for V with $b = 1$. Conversely, if we obtained a T with $b = 1$, then this is only possible if T consists entirely of edges of E and is thus a Hamiltonian path of G. Hence G has a Hamiltonian path if and only if V has a 2-EMBST with a bottleneck value of at most 1. □

By using a similar approach to that of Papadimitriou and Vazirani [18], we obtain the following complexity result for the 3-EMBST problem.

Theorem 5. *The 3-EMBST problem is NP-complete.*

Proof. Suppose we are given a connected grid graph $G = (V, E)$ with maximum degree 3, where we will assume that $|V| \geq 2$. Since G is a grid graph, all its edges have unit length under the Euclidean metric. We perform a 2-colouring of G so that the vertex set V is partitioned into a set of black vertices B and white vertices W such that $E \subseteq B \times W$ (this is possible since grid graphs are bipartite). For each node $v \in V$, we add a single "pseudo node" u that is a small distance away from v at a direction leading towards a missing neighbour of v (such a direction is always possible since the degree of v is at most 3, see Fig. 3). If v is a black vertex, then we let u be a distance of ϵ from v and refer to it as a black pseudo node; otherwise we let u be a distance of δ from v and refer to it as a white pseudo node, where $0 < \delta < \epsilon < \frac{2-\sqrt{2}}{2}$. Thus, any two black pseudo nodes have distance of least $\sqrt{2}(1 - \epsilon) > 1$ from each other, any two white pseudo nodes have distance of least $\sqrt{2}(1 - \delta) > 1$ from each other, and each black pseudo node has a distance of at least $\sqrt{(\epsilon - \delta)^2 + 1} > 1$ from every white pseudo node.

Let U be the set of pseudo nodes. We now show that G has a Hamiltonian path if and only if there is a degree-3 spanning tree S on the set of points $V \cup U$ whose longest edge has a length of at most 1. If G has Hamiltonian path, then for each vertex $v \in V$, we can add the edge (v, u) to the path, where u is the pseudo node of v, to give a degree-3 spanning tree S whose longest edge has length 1. Conversely, suppose that we have a degree-3 spanning tree T on the

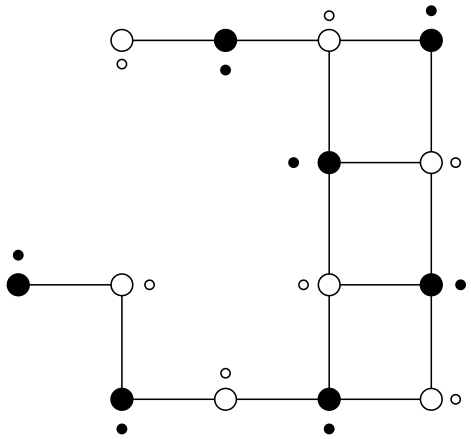


Figure 3: The grid graph from Fig. 2 with a 2-colouring and with the black and white pseudo nodes having been placed.

points $V \cup U$ whose longest edge has length $k = 1$. Note that the only edges whose lengths are strictly less than 1 are edges between the vertices in V and their respective pseudo nodes in U. Since the only edges of unit length are those between adjacent vertices in G, and since all other edges have lengths strictly greater than 1, we can conclude that T consists only of the edges from E and the edges (v, u) between the points $v \in V$ and their respective pseudo nodes $u \in U$. In this case, we can remove the edges (v, u) from T and we are left with a Hamiltonian path in G (see Fig. 4). Thus, there is a degree-3 spanning tree S on the set of points $V \cup U$ whose longest edge has length at most 1 if and only if G has a Hamiltonian path. This completes the proof. □

As a consequence of Theorem 5, we have the following corollary.

Corollary 2. *The 3-EMBST problem cannot be approximated in polynomial time within a constant factor strictly less than* $\frac{5\sqrt{2}}{7} \approx 1.0102$ *unless* $P = NP$.

Proof. Suppose we had an algorithm A that could approximate the problem within a constant factor c in polynomial time. This means that if we were given an instance of the problem whose optimal bottleneck length is L, then if we performed A on the same instance, it would yield a solution with a bottleneck value no worse than cL. If we applied A on set of points $V \cup U$ that we constructed in the proof of Theorem 5, with $\epsilon = 2/7$ and $\delta = 1/7$, then the algorithm would return a value $k \geq 1$. Note that the distance between any two black pseudo

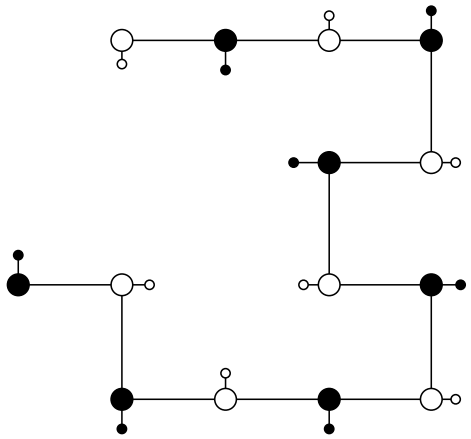


Figure 4: A 2-MBST for the vertices in Fig. 3. A Hamiltonian path results from removing the pseudo nodes.

nodes is at least

$$\sqrt{2}(1-\epsilon) = \sqrt{2}(1-\frac{2}{7}) = \frac{5\sqrt{2}}{7},$$

the distance between any white pseudo node and any black pseudo node is at least

$$\sqrt{(\epsilon-\delta)^2+1} = \sqrt{\frac{1}{49}+1} = \frac{\sqrt{50}}{7} = \frac{5\sqrt{2}}{7},$$

and the distance between a white pseudo node and an original black node is at least

$$\sqrt{1+\delta^2} = \sqrt{1+\frac{1}{49}} = \frac{5\sqrt{2}}{7}.$$

Hence, if $k < \frac{5\sqrt{2}}{7}$, then k must equal 1 since no pair of points in $V \cup U$ have a distance that lies in the open interval $(1, \frac{5\sqrt{2}}{7})$. Hence, if $c < \frac{5\sqrt{2}}{7}$, we could use our approximation algorithm to determine if G has a Hamiltonian path in polynomial time. Thus if $P \neq NP$, we must have $c \geq \frac{5\sqrt{2}}{7}$. □

The results of Theorem 5 and Corollary 2 can be generalised from the Euclidean metric to any L_p metric, where $p \in [1, \infty]$. For any two points $x, y \in \mathbb{R}^2$, let $d_p(x, y)$ denote the distance between x and y under the L_p metric. If $x = (x_1, x_2)$ and $y = (y_1, y_2)$, then

$$d_p(x, y) = (|x_1 - x_2|^p + |y_1 - y_2|^p)^{1/p}$$

for $p \in [1, \infty)$, and

$$d_\infty(x, y) = \max\{|x_1 - x_2|, |y_1 - y_2|\}.$$

The L_∞ metric (Chebyshev distance) in the two dimensional plane can be viewed as being equivalent to the L_1 metric through rotation and scaling, and so we will assume that the point sets have be transformed appropriately for the L_∞ case so that the results can follow from those of the L_1 case. Note that if $x_1 = x_2$ and $|y_1 - y_2| = 1$, then $d_p(x, y) = 1,\ \forall p \in [1, \infty]$ (and similarly for the case where $y_1 = y_2$ and $|x_1 - x_2| = 1$). Hence the distances between adjacent vertices in a grid graph are unit under any L_p metric with $p \in [1, \infty]$. From this observation, we can generalise Theorem 5 and Corollary 2.

Theorem 6. *For $p \in [1, \infty]$, the 3-MBST problem using the L_p metric is NP-complete. Furthermore, the problem does not admit a polynomial time approximation scheme (PTAS) unless $P = NP$.*

Proof. In our notation, we will assume $p \in [1, \infty)$ since the results for the case in which $p = \infty$ follow from the results for the case in which $p = 1$. If a problem admits a PTAS, it means that given any constant $c > 0$, there exists a polynomial time algorithm to solve the problem within a factor of $1 + c$. Given a grid graph $G = (V, E)$ whose maximum degree is 3, we add in the pseudo nodes $u \in U$ as in the proof of Theorem 5 such that the resultant set of vertices $V \cup U$ admits a 3-MBST whose bottleneck value is 1 if and only if G contains a Hamiltonian path. For the distances of our pseudo nodes, we now require $0 < \delta < \epsilon < 1 - 2^{-1/p}$. Thus, any two black pseudo nodes have distance of least $2^{\frac{1}{p}}(1-\epsilon) > 2^{\frac{1}{p}}(1-(1-2^{-1/p})) = 1$ from each other, any two white pseudo nodes have distance of least $2^{\frac{1}{p}}(1 - \delta) > 1$ from each other, and each black pseudo node has a distance of at least $[(\epsilon - \delta)^p + 1]^{\frac{1}{p}} > 1$ from every white pseudo node. As before, the only edges whose lengths are strictly less than 1 are edges between the vertices in V and their respective pseudo nodes in U, and the only edges of unit length are those between adjacent vertices in G. Thus we can use the same argument as in Theorem 5 to show the problem is NP-hard under the L_p metric with $p \in [1, \infty]$.

In order to show that there is no PTAS unless P=NP, we will show that given any L_p metric with $p \in [1, \infty)$, there exists some constant $C = C(p)$ such that the 3-MBST problem using the L_p metric cannot be approximated within a factor of C unless P=NP. Given a grid graph G, we use the construction in the reduction above. As $\delta \to 0$, the shortest distance between two white pseudo nodes approaches $2^{1/p}$ and the shortest distance between a white pseudo node

and a black pseudo node approaches $(\epsilon^p+1)^{\frac{1}{p}} < 2^{\frac{1}{p}}$. If we require that the shortest distance between two black pseudo nodes is equal to the shortest distance between a white pseudo node and a black pseudo, then we have

$$[(\epsilon-\delta)^p+1]^{\frac{1}{p}} = 2^{\frac{1}{p}}(1-\epsilon)$$

$$\Rightarrow (\epsilon-\delta)^p+1-2(1-\epsilon)^p = 0.$$

As $\delta \to 0$, the LHS of the equation becomes

$$\epsilon^p+1-2(1-\epsilon)^p,$$

where $0 < \epsilon < 1-2^{-1/p}$.
For $\epsilon = 0$,

$$\epsilon^p+1-2(1-\epsilon)^p = -1,$$

and for $\epsilon = 1-2^{-1/p}$,

$$\epsilon^p+1-2(1-\epsilon)^p = (1-2^{-1/p})^p > 0.$$

Thus the polynomial $\epsilon^p+1-2(1-\epsilon)^p$ has a root for $\epsilon \in (0, 1-2^{-1/p})$. We denote this root by ϵ^* and we let $\epsilon = \epsilon^*$ for our construction. Thus the shortest distance between two black pseudo nodes is $2^{\frac{1}{p}}(1-\epsilon^*)$ which, for a small enough choice of δ, is approximately equal to the shortest distance between a white pseudo node and a black pseudo node. We also have that the distance between a black pseudo node and an original white node is at least $(1+(\epsilon^*)^p)^{\frac{1}{p}} > 1$, and the distance between a white pseudo node and an original black node is at least $(1+\delta^p)^{\frac{1}{p}} > 1$. If we let $C = \min\{[(\epsilon^*-\delta)^p+1]^{\frac{1}{p}}, (1+\delta^p)^{\frac{1}{p}}\}$ (the smaller of the shortest distance between a black pseudo node and a white pseudo node and the shortest between a white pseudo node and an original black node), then if δ is chosen to be sufficiently small, there are no edges whose lengths are in the open interval $(1, C)$. Thus we cannot approximate the problem within a factor of C unless $P = NP$, and hence the 3-MBST problem does not admit a PTAS for any L_p metric with $p \in [1, \infty]$. □

By simply removing the pseudo nodes, we are able to establish an equivalent L_p norm generalisation and inapproximability result for the 2-EMST problem.

Theorem 7. *For $p \in [1, \infty]$, the 2-MBST problem using the L_p metric is NP-complete. Furthermore, the problem cannot be approximated in polynomial time within a constant factor strictly less than $2^{\frac{1}{p}}$, for $p \in [1, \infty)$, unless $P = NP$.*

Proof. The NP-hardness of the problem can be easily seen using the same argument as in Theorem 4 since the distances between two vertices of a grid graph

are still unit under the L_p metric. To see the inapproximability result, note that the shortest distance between two vertices that are not adjacent in the graph is $(1+1)^{\frac{1}{p}} = 2^{\frac{1}{p}}$. □

Francke and Hoffman [8] show the NP-hardness of the 4-EMST by using reduction from the vertex cover problem. In simple terms, they are able to construct an instance of the 4-EMST problem in such a way that it is possible to determine the size of an optimal vertex cover of a given graph from the optimal solution to their 4-EMST instance. This construction does not seem to generalise easily to the 4-EMBST however, and as such we leave the complexity of the 4-EMBST problem as an open problem

3.2 Euclidean Approximation Algorithms

In this subsection, we will describe polynomial time algorithms that give constant factor approximations for the δ-EMBST problem. Recall that the δ-EMBST can be solved exactly in polynomial time when $\delta \geq 5$. Thus, the only cases of interest are when $\delta = 2, 3$ or 4, and we describe and analyse constant factor approximation algorithms for these cases. For $\delta = 2$, the problem is equivalent the Euclidean bottleneck TSPP, and we describe a 3-factor approximation algorithm that is similar to Parker and Rardin's 2-factor algorithm for the bottleneck TSP [19]. Our 3-factor algorithm generalises to any metric space and any $\delta \geq 2$. For $\delta = 3$, we show that the existing algorithm of Khuller, Raghavachari and Young [15] for the 3-EMST problem yields a performance ratio of 2. For $\delta = 4$, we use a bottleneck version of Chan's constant factor approximation algorithm for the 4-EMST [4] to obtain a $\sqrt{3}$-factor algorithm.

Let P be an instance of the δ-EMBST problem, and let A be a polynomial time approximation algorithm for the δ-EMBST problem. Let $e_A^\delta(P)$ denote the longest edge in the δ-EMBST given by A when performed on P, and let $e_{\text{opt}}^\delta(P)$ denote the longest edge in an optimal δ-EMBST for P. We define the approximation ratio for the algorithm A as

$$\rho = \sup_P \frac{e_A^\delta(P)}{e_{\text{opt}}^\delta(P)}.$$

We also define $\hat{\rho}$ as the performance ratio of the approximation algorithm relative to the optimal EMST. That is, let $e_{\text{opt}}(P)$ be the longest edge in an optimal EMST for P. Then

$$\hat{\rho} = \sup_P \frac{e_A^\delta(P)}{e_{\text{opt}}(P)}.$$

For our analyses when $\delta = 3$ and 4, we will not explicitly find ρ; instead we find upper bounds for $\hat{\rho}$. Since the length of the longest edge in an MST is at most the length of the longest edge in a δ-MBST for the same set of points, we have $\rho \leq \hat{\rho}$, so an upper bound for $\hat{\rho}$ is also an upper bound for ρ.

3.2.1 A 3-factor approximation for the δ-MBST in metric spaces.

Our algorithm uses similar ideas to Parker and Rardin's 2-factor approximation for the bottleneck TSP which uses the square of a graph G, denoted G^2. The graph G^2 is a supergraph of G such that the edge (u, v) is in G^2 if and only if there is a path between u and v in G with two or fewer edges. Our algorithm uses the cube of G, denoted G^3, where the edge (u, v) is in G^3 if and only if there is a path between u and v in G with three or fewer edges. Both our algorithm and Parker and Rardin's can be generalised from points that lie in the Euclidean plane to points that lie in any metric space, whilst still maintaining the desired performance ratio. Hence, we let our point set P be a set of points in a metric space with distance function $d : P \times P \to \mathbb{R}_+$. The distances between points must obey the triangle inequality, i.e, for any three distinct points $x, y, z \in P$,

$$d(x, y) \leq d(x, z) + d(y, z).$$

Let $G = (V := P, E := P \times P)$ be the complete graph using the points P as vertices, where the length of an edge $(p_1, p_2) \in E$ is equal to the distance $d(p_1, p_2)$.
Our algorithm is summarised as follows.

1. Find an MBST for G, denoted T.

2. Compute T^3.

3. Find a Hamiltonian path in T^3. This is the desired approximation.

It has been shown that the cube of any connected graph has a Hamiltonian cycle. From Karaganis' inductive proof of this statement [14], it can be seen that a Hamiltonian cycle can be found in the cube of a connected graph in polynomial time by using a recursive approach over subgraphs, hence we can perform Step 3 in polynomial time by removing an edge from the Hamiltonian cycle. Since a Hamiltonian path is a feasible δ-MBST for all $\delta \geq 2$, it can easily be seen that the algorithm returns a feasible solution in polynomial time. All

that remains to be shown is that the longest edge in the returned solution is no more than 3 times the longest edge in an optimal δ-MBST for the input graph. This follows from the following theorem.

Theorem 8. *Let $G = (V := P, E := P \times P)$, where P is a set of points in a metric space, be such that the length of an edge $e \in E$ is given by the distance between its endpoints. Let T be an MBST for G, and let H be a Hamiltonian path in T^3. Then the longest edge in H is at most 3 times the longest edge in a δ-MBST for G, for any $\delta \geq 2$.*

Proof. For a graph G' let $w(G')$ denote the length of the longest edge in G'. Let (x, y) be an edge in T^3 that is not in T. This means that there is a path from x to y in T using either 2 or 3 edges. If the path in T uses two edges, say (x, z) and (z, y) for some $z \in P$, then the triangle inequality implies that

$$d(x, y) \leq d(x, z) + d(z, y) \leq 2 \max\{d(x, z), d(z, y)\},$$

hence the length of (x, y) is at most twice the length of an edge in T. If the path from x to y in T uses three edges, say (x, q), (q, z) and (z, y) for some $z, q \in P$, then the triangle inequality implies that

$$\begin{aligned} d(x, y) &\leq d(x, z) + d(z, y) \\ &\leq d(x, q) + d(q, z) + d(z, y) \\ &\leq 3 \max\{d(x, q), d(q, z), d(z, y)\}, \end{aligned}$$

hence the length of (x, y) is at most 3 times the length of an edge in T. Thus for all edges (x, y) in T^3 that are not already in T, the length of (x, y) is at most 3 times the length of an edge in T. Hence $w(T^3) \leq 3w(T)$. Since H is a subgraph of T^3, we have $w(H) \leq w(T^3)$. Let T_δ be an optimal δ-MBST for G for some $\delta \geq 2$. Since T_δ is a feasible solution to the MBST problem for G, and since T is an optimal solution to the MBST problem, we have $w(T_\delta) \geq w(T)$. After combining the inequalities, we have $w(H) \leq 3w(T_\delta)$ as required. □

3.2.2 A 2-factor approximation for the 3-MBST problem in metric spaces.

The algorithm presented here is an adaptation of that of Khuller, Raghavachari and Young [15]. Let P be a set of points in a metric space. The first stage of the algorithm is to find a rooted MST for P where the root can be chosen to be any vertex. The algorithm then replaces the edges of the local subtree consisting of a root and its children with a path starting from the root which contains all the child nodes. As a result, the root node and one child have degree one and all

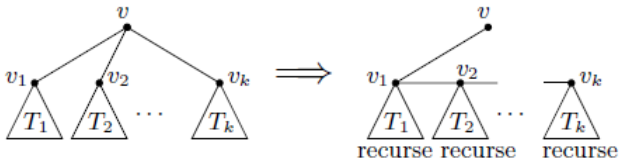


Figure 5: The illustration of the Khuller, Raghavachari and Young algorithm [4].

other children have degree two, with respect to the edges in the local subtree. The algorithm then iterates recursively over the subtrees rooted at each of the children, performing the same kind of edge replacements as for the previous root node. Therefore once the algorithm terminates, no vertex will have degree greater than three. See Figure 5 for an illustration.

We let T^* be the set containing the edges of the solution produced by the algorithm. The set T^* is initially empty and is added to during the recursive calls. For a rooted tree T, let T_v denote the subtree of T rooted at v, i.e., T_v is the subgraph of T induced by the vertex v and its descendants. The recursive part of the algorithm is presented as Algorithm 2, where we let the initial T_v be any rooted MST T for the point set P, and where the initial T^* is the empty set. The tree given by the edges of T^* after the last recursive call of the algorithm will be the desired approximation.

Algorithm 2

Input: A rooted tree T over a point set P with root v and a partially built solution T^*.

if v has at least one child
  Let the children of v be $v_1, \ldots, v_k$, where k is the number of children of v.
  Add the edge (v, v_1) to T^*
  Perform Algorithm 2 with $T := T_{v_1}$ as input.
  if $k \geq 2$
    for $i = 1, \ldots, k-1$
      Add the edge (v_i, v_{i+1}) to T^*.
      Perform Algorithm 2 with $T := T_{v_{i+1}}$ as input.

Clearly, Algorithm 2 terminates in linear time when given the MST as input. The performance ratio of the algorithm is a direct consequence of the triangle inequality.

Theorem 9. *For Algorithm 2,* $\hat{\rho} \leq 2$.

Proof. Let T be a rooted MST for P. Let T^* be the edges of the resultant tree when T is given as input to Algorithm 2. Suppose e is some edge in T^* that was not present in T. This means that e is an edge between two children in some subtree of T. Let $e = (v_i, v_j)$ and let v be the parent of v_i and v_j. The edges (v, v_i) and (v, v_j) were both present in T, and from the triangle inequality we have

$$\begin{aligned} d(v_i, v_j) &\leq d(v, v_i) + d(v, v_j) \\ &\leq 2 \max\{d(v, v_i), d(v, v_j)\}. \end{aligned}$$

Hence, the longest edge in T^* is no more than twice the length of the longest edge in T. □

3.2.3 A $\sqrt{3}$-factor approximation for the 4-EBMST problem.

The algorithm presented here is based on that of Chan [4]. Let P be a set of points in the Euclidean plane. The first stage of Chan's 4-EMST algorithm is to find a rooted MST for P where the root can be chosen to be any vertex. Note that since this MST is an EMST, we can assume that it is a 5-MST. The algorithm then works recursively from the root, performing local edge swaps for each subtree so that the desired degree bound is satisfied at each vertex (see Figure 6). At any given stage of the algorithm in which there is an edge swap, the algorithm swaps out an edge between the current root node and its child, and adds an edge between two of the children so that one child becomes the parent of the other child. Our algorithm works in a similar way.

For a rooted tree T, we let T_v denote the subtree of T rooted at the vertex v. If v has at least one child in the tree, we assume that at most one of the children of v is *adopted*. An adopted child is one that has been added as child of v due to a previous step of the algorithm. It is important to make this distinction for the analysis so that when we find the worst case ratio of the bottleneck lengths before and after a single edge swap, we can state this ratio in terms of the lengths of edges incident to vertices that are not adopted (note that these are the edges that were present in the original spanning tree T before the algorithm was applied). Also key to the algorithm is the existence of τ-mutable

pairs of children. For a root vertex v and a constant τ, a pair of children, v_1 and v_2, of v are a *τ-mutable pair* if either

1. One of v_1, v_2 is an adopted child of v and
$$d(v_1, v_2) \leq \max\{d(v, v_1), d(v, v_2)\},$$
2. Neither v_1 nor v_2 is an adopted child of v and
$$d(v_1, v_2) \leq \tau \max\{d(v, v_1), d(v, v_2)\}.$$

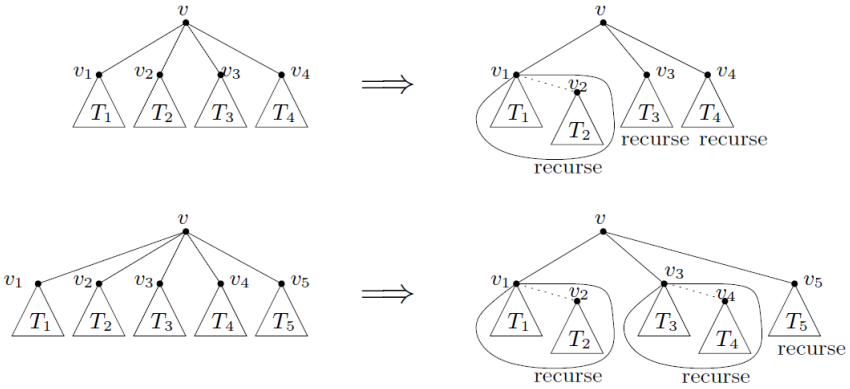


Figure 6: The illustration of Chan's algorithm [4].

For our algorithm, we will have $\tau = \sqrt{3}$ and all edge swaps will be centred around $\sqrt{3}$-mutable pairs of children. The existence of $\sqrt{3}$-mutable pairs of children will be shown by Theorem 11. Our algorithm is presented as Algorithm 3 which is located in the appendix. By providing an MST of the point set P as the initial input to Algorithm 3, we are able obtain a 4-EMST for P.

As in the case of Chan's algorithm, it is easy to see that Algorithm 3 runs in linear time and that it returns a feasible tree that satisfies the degree bound

(a) (b)

Figure 7: The notation for Theorem 11, originally given in [4].

of 4, assuming we can find the appropriate $\sqrt{3}$-mutable pairs in constant time. Before we prove this is possible, we will show that the correctness of Algorithm 3 implies that $\hat{\rho} \leq \sqrt{3}$. To prove that $\hat{\rho}$ is bounded above by $\sqrt{3}$, we need only prove that after any edge swap in the algorithm, the length of the new edge that is swapped in is at most $\sqrt{3}$ times the length of an edge that was present in the original MST before the algorithm. Clearly, this would imply that the length of any new edge swapped in by the algorithm is at most $\sqrt{3}$ times the bottleneck length of the original MST. We prove this constant upper bound for $\hat{\rho}$ in the following theorem.

Theorem 10. *For Algorithm 3, $\hat{\rho} \leq \sqrt{3}$.*

Proof. Let T be an EMST for the point set P. Since T is an EMST, we can assume that it is a 5-EMST. We make T a rooted tree by choosing an arbitrary vertex in P to be the root. Let T_A be the resultant tree after Algorithm 3 has been performed on T. Suppose that T_A is distinct from T, which means we have performed an edge swap at some stage of the algorithm. Consider one such edge swap. Let e_{old} be the edge that was swapped out, and let e_{new} be the edge that was swapped in to replace e_{old}. To show that $\hat{\rho} \leq \sqrt{3}$, we only need to show that for any such edge swap,

$$\frac{|e_{\text{new}}|}{|e_{\text{old}}|} \leq 1 \quad \text{if } e_{\text{old}} \text{ is incident on an adopted child, or}$$
$$\frac{|e_{\text{new}}|}{|e_{\text{old}}|} \leq \sqrt{3} \quad \text{otherwise.}$$

In other words, we need to show that we only swap out an existing edge in T if the length of the edge we swap in is no more than $\sqrt{3}$ times the length of the edge we swap out. We also need to show that we only swap out an edge that was swapped in at a previous stage of the algorithm if the new edge that replaces it is shorter or of equal length. However, since we only ever perform an edge swap with respect to a $\sqrt{3}$-mutable pair v_1, v_2 with root node v, and we always swap out the longer of the edges (v, v_1) and (v, v_2), these conditions are satisfied by the definition of τ-mutable pairs. □

All that remains is to prove the existence of the $\sqrt{3}$-mutable pairs required by the algorithm. In order to prove this, we will use a technical lemma presented in [4]. Let $|e|$ denote the length of the edge e, i.e., if $e = (e_1, e_2)$, then $|e| = d(e_1, e_2)$. The lemma is as follows.

Lemma 1. *If a triangle has sides x, y, z with $|x| \leq |y|$, and the angle opposite z is θ, then*

$$|z| \leq f(\theta)|x| + |y|, \quad \textit{where } f(\theta) := \max\{2\sin(\theta/2) - 1, 0\}.$$

We now prove the existence of the $\sqrt{3}$-mutable pairs.

Theorem 11. *Let P be a set of points in the plane with EMST T. Let v be the root of a tree T_v given as input to Algorithm 3, where T_v is a subtree of T after excluding any adopted children of v. If v has four children, then v has two children that form a $\sqrt{3}$-mutable pair. If v has five children, then v has two disjoint pairs of children, such that each pair is a $\sqrt{3}$-mutable pair.*

Proof. Suppose v_1 and v_2 are children of v. Let $x_1 = d(v, v_1), x_2 = d(v, v_2)$, and let θ be the angle between the edges (v, v_1) and (v, v_2). Assume that $x_1 \leq x_2$. If $\theta \leq 60°$, then by Lemma 1,

$$d(v_1, v_2) \leq (2\sin(\theta/2) - 1)x_1 + x_2 \leq x_2$$

and so v_1, v_2 are a $\sqrt{3}$-mutable pair, even if either v_1 or v_2 is an adopted child. If we assume that neither v_1 nor v_2 are adopted, then if $60° \leq \theta \leq 120°$, we have

$$\begin{aligned} d(v_1, v_2) &\leq 2\sin(\theta/2)x_2 \\ &\leq 2\sin(60°)x_2 \\ &= \sqrt{3}x_2 \end{aligned}$$

and so v_1, v_2 are a $\sqrt{3}$-mutable pair. Hence in order to find $\sqrt{3}$-mutable pairs of children, we need only check the angles between the edges incident to v. There are two cases to consider, depending on the number of children of v.

- **Case 1:** The root node v has four children. If v has an adopted child, then we denote it by v'; otherwise we choose any of the children to be denoted by v', and let v_1, v_2, v_3 be the other children of v. Let $\theta_1, \theta_2, \theta_3, \theta_4$ be the angles between children with respect to v as shown in Figure 7(a). If either θ_3 or θ_4 are less than or equal to 60°, then we have a $\sqrt{3}$-mutable pair, so assume that $\theta_3, \theta_4 > 60°$. However, this implies that one of θ_1, θ_2 is less than 120° and so we also have a $\sqrt{3}$-mutable pair in this case.

- **Case 2:** The root node v has five children. For this case, we need to show that there are two disjoint $\sqrt{3}$-mutable pairs. As in the previous case, if v has an adopted child, then we denote it by v'; otherwise we choose any of the children to be denoted by v', and let v_1, v_2, v_3, v_4 be the other children of v. Let $\theta_1, \ldots, \theta_5$ be the angles between children with respect to v as shown in Figure 7(b). If $\theta_5 \leq 60°$, then one of $\theta_1 + \theta_4 + \theta_5, \theta_2, \theta_3$ is less than or equal to 120° and so we have two disjoint $\sqrt{3}$-mutable pairs (and

similarly if $\theta_4 \leq 60°$). If both $\theta_4, \theta_5 \geq 60°$, then $\theta_1 + \theta_2 + \theta_3 \leq 240°$. Since (v, v_i), for $i = 1, \ldots, 4$, are all original edges of the EMST T, then we know that $\theta_1, \theta_2, \theta_3$ are all greater than or equal to $60°$. This implies that none of $\theta_1, \theta_2, \theta_3$ is strictly greater than $120°$ and so v_1, v_2 and v_3, v_4 are two disjoint sets of $\sqrt{3}$-mutable pairs.

□

To find the $\sqrt{3}$-mutable pairs at any stage of the algorithm, we only need to check the smallest angles between edges incident on the root node. Since the degrees of the vertices of the original EMST are bounded by 5, there are only a constant number of angles to check. Thus if the angles are known, we can find the $\sqrt{3}$-mutable pair(s) at any iteration of Algorithm 3 in constant time.

We also briefly investigated the use of Chan's algorithm for the 3-EMST problem described in [4]. It is similar to the previous algorithm, in that it is a recursive algorithm that uses local edge swaps. However, there are more rules that determine which edges are swapped, and the analysis of the algorithm is far more technical. When converting the rules of this algorithm to suit a bottleneck objective, it seems that Chan's analysis does not easily generalise. In particular, Chan uses a lemma to establish the existence of angles between edges of at least $72°$, which cannot be used for the bottleneck version. This is not unexpected due to the differences between the min-sum and bottleneck objectives. Consider a local edge swap, when the edge being added is longer than the edge being removed. Under the usual min-sum objective, the length of the new edge is added to objective, and the length of the edge being removed is subtracted from the objective, hence the length of the edge being removed is an important consideration for the swap. On the other hand, under the bottleneck objective, only the length of the new edge may affect the objective value of the tree. These differences have implications in the analysis of the edge swap algorithms and the performance ratios obtained. As such, it is not a straightforward exercise in this case to incorporate Chan's analysis into an algorithm for the bottleneck version of the problem, and we leave this task as future work.

4 Conclusion

In this paper, we have introduced the δ-MBST problem and have shown that it is possible to have optimal solutions to this problem which are not optimal solutions to the δ-MST problem for the same instance. We show that the prob-

lem is NP-complete for any $\delta \geq 2$, and we give a dual approximation scheme that can use an existing feasible $(\delta + k)$-algorithm for the δ-MST problem in order to obtain a feasible $(\delta + k)$-algorithm for the δ-MBST problem, where the bottleneck values of the solutions output by the scheme are no worse than those of optimal solutions for the same instances. We have also proven that the Euclidean version is NP-complete for $\delta = 2$ and 3, a result which generalises to arbitrary L_p metrics, and we show that these problems cannot be approximated within factors of $\sqrt{2}$ and $\sqrt{2}(\sqrt{3} - 1)$ respectively. Finally, we have adopted existing algorithms for the δ-EMST problem to give constant factor approximation algorithms for the geometric and Euclidean versions of the problem. For future work, we would like to establish the complexity of the 4-EMBST problem. We believe this problem to be NP-complete since that is case for the 4-EMST problem. However we believe a different reduction is required for this proof than what was used by Francke and Hoffman.

References

[1] S. Arora and K. Chang, Approximation schemes for degree-restricted MST and red-blue separation problems, Algorithmica 40 (2004), 189-210.

[2] M. Brazil, C. J. Ras, and D. J. Thomas, The bottleneck 2-connected k-Steiner network problem for $k \leq 2$, Discrete Applied Mathematics 160 (2012), 1028-1028.

[3] P. M. Camerini, The min-max spanning tree problem and some extensions, Information Processing Letters 7 (1978), 10-14.

[4] T. M. Chan, Euclidean bounded-degree spanning tree ratios, Proceedings of the Nineteenth Annual Symposium on Computational Geometry, ACM, 2003, pp. 11-19.

[5] N. Christofides, Worst-case analysis of a new heuristic for the travelling salesman problem, Tech. rep., DTIC Document (1976).

[6] J. Edmonds and D. R. Fulkerson, Bottleneck extrema, Journal of Combinatorial Theory 8 (1970), 299-306.

[7] S. P. Fekete, S.Khuller, M. Klemmstein, B. Raghavachari, and N. Young, A network-flow technique for finding low-weight bounded-degree spanning trees, Journal of Algorithms 24 (1997), 310-324.

[8] A. Francke and M. Hoffmann, The Euclidean degree-4 minimum spanning tree problem is NP-hard, Proceedings of the Twenty-Fifth Annual Symposium on Computational Geometry, ACM, 2009, pp. 179-188.

[9] M. R. Garey and D. S. Johnson, Computers and intractability: A guide to the theory of NP-completeness, W. H. Freeman, New York, NY, 1979.

[10] M. X. Goemans, Minimum bounded degree spanning trees, 47th Annual IEE Symposium on Foundations of Computer Science (FOCS'06), IEEE, 2006, pp. 273-282.

[11] R. L. Graham and P. Hell, On the history of the minimum spanning tree problem, Annals of the History of Computing 7 (1985), 43-57.

[12] J. Hoogeveen, Analysis of Christofides' heuristic: Some paths are more difficult than cycles, Operations Research Letters 10 (1991), 960-970.

[13] R. Jothi and B. Raghavachari, Degree-bounded minimum spanning trees, Discrete Applied Mathematics 157 (2009), 960-970.

[14] J.J. Karaganis, On the cube of a graph, Canad Math Bull 11 (1968), 295-296.

[15] S. Khuller, B. Raghavachari, and N. Young, Low-degree spanning trees of small weight, SIAM Journal on Computing 25 (1996), 355-368.

[16] J. B. Kruskal, On the shortest spanning subtree of a graph and the travelling salesman problem, Proceedings of the American Mathematical Society 7 (1956), 48-50.

[17] C. Monma and S. Suri, Transitions in geometric minimum spanning trees, Discrete & Computational Geometry 8 (1992), 265-293.

[18] C. H. Papadimitriou and U. V. Vazirani, On two geometric problems related to the travelling salesman problem, Journal of Algorithms 5 (1984), 231-246.

[19] R. G. Parker and R. L. Rardin, Guaranteed performance heuristics for the bottleneck travelling salesman problem, Operations Research Letters 2 (1984), 269-272.

[20] R. C. Prim, Shortest connection networks and some generalizations, Bell System Technical Journal 36 (1957), 1389-1401.

[21] M. Sarrafzadeh and C. Wong, Bottleneck Steiner trees in the plane, IEEE Transactions on Computers 41 (1992), 370-374.

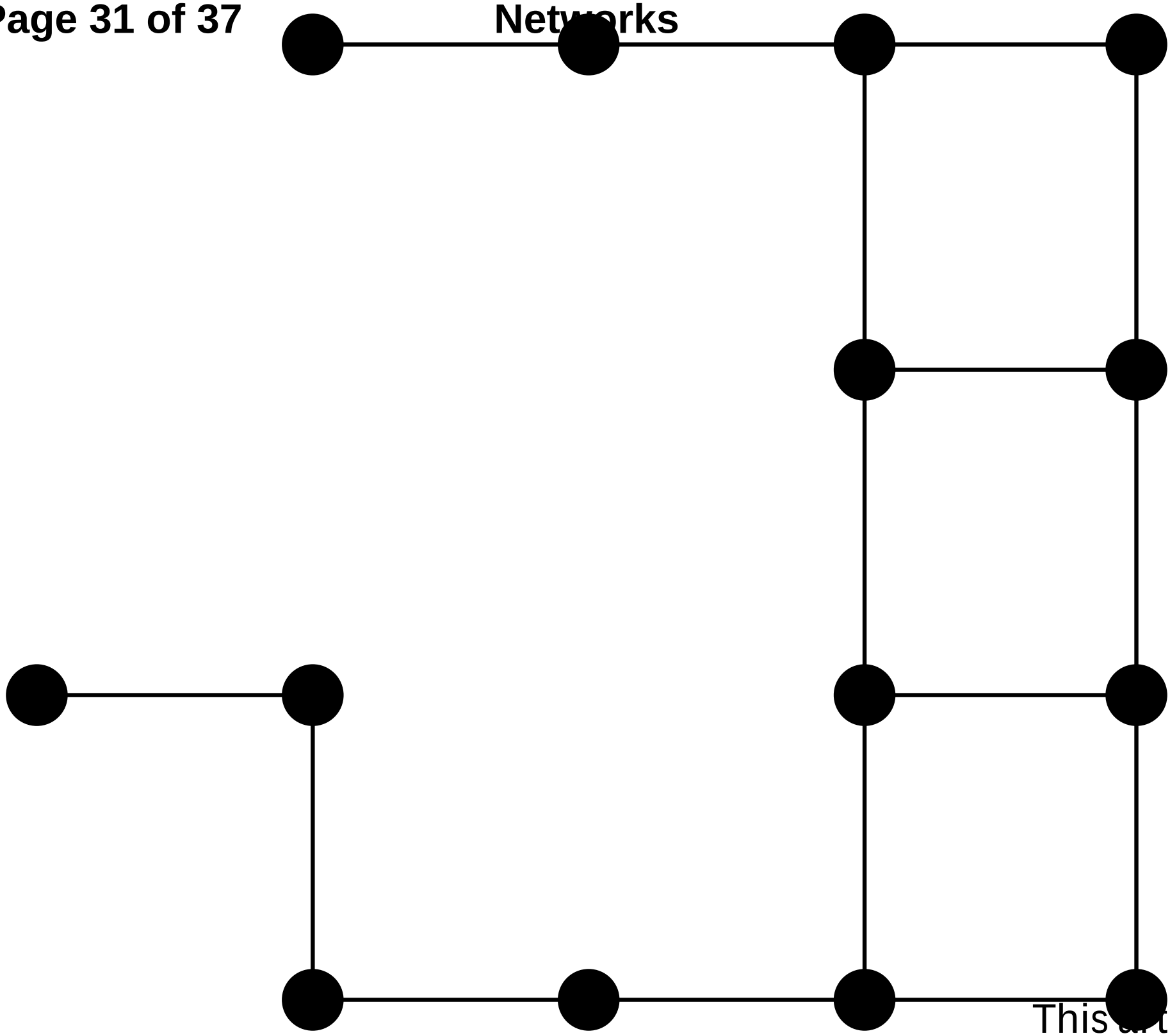
[22] M. Singh and L. C. Lau, Approximating minimum bounded degree spanning trees to within one of optimal, Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing, ACM, 2007, pp. 661-670.

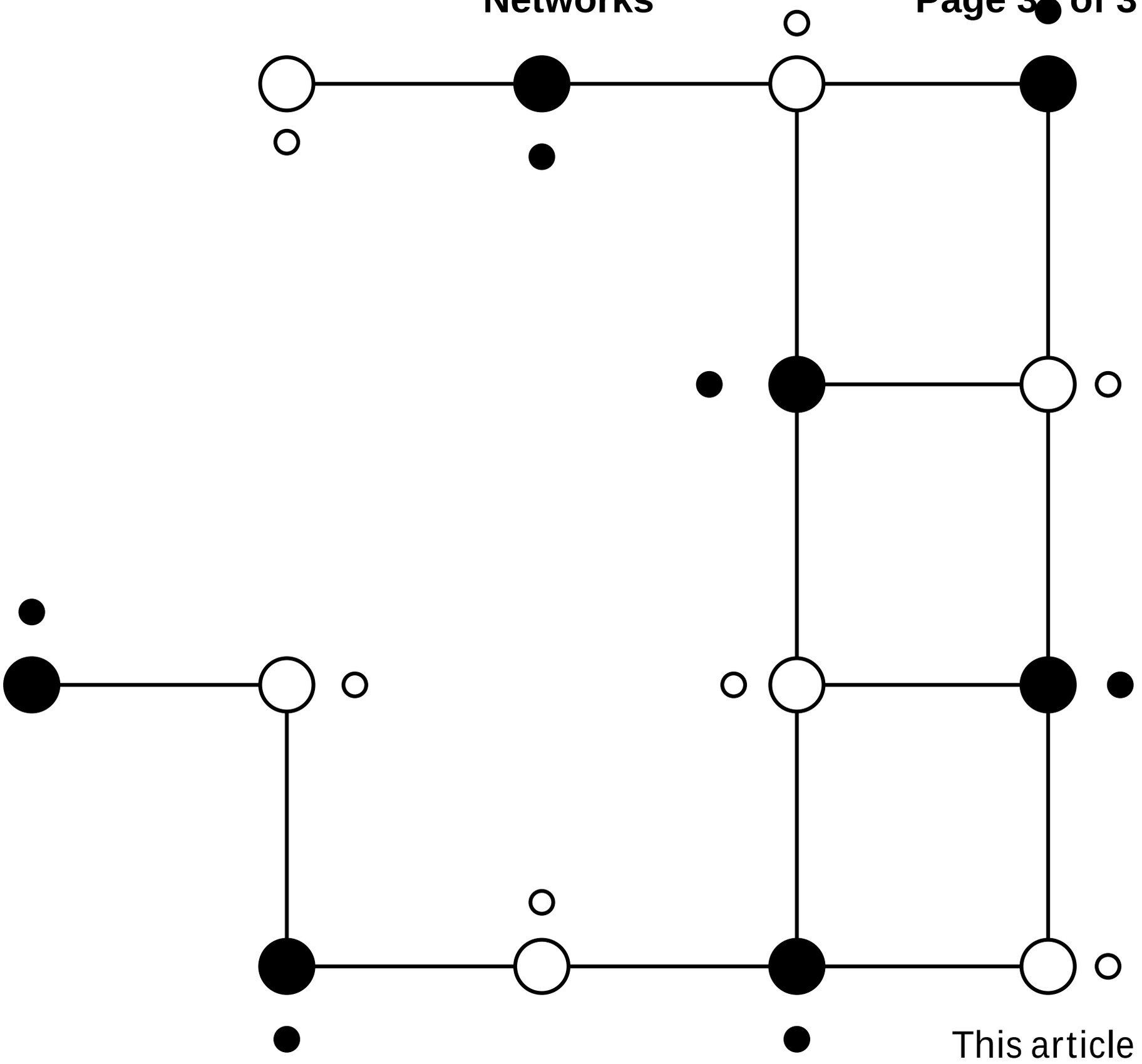
Appendix

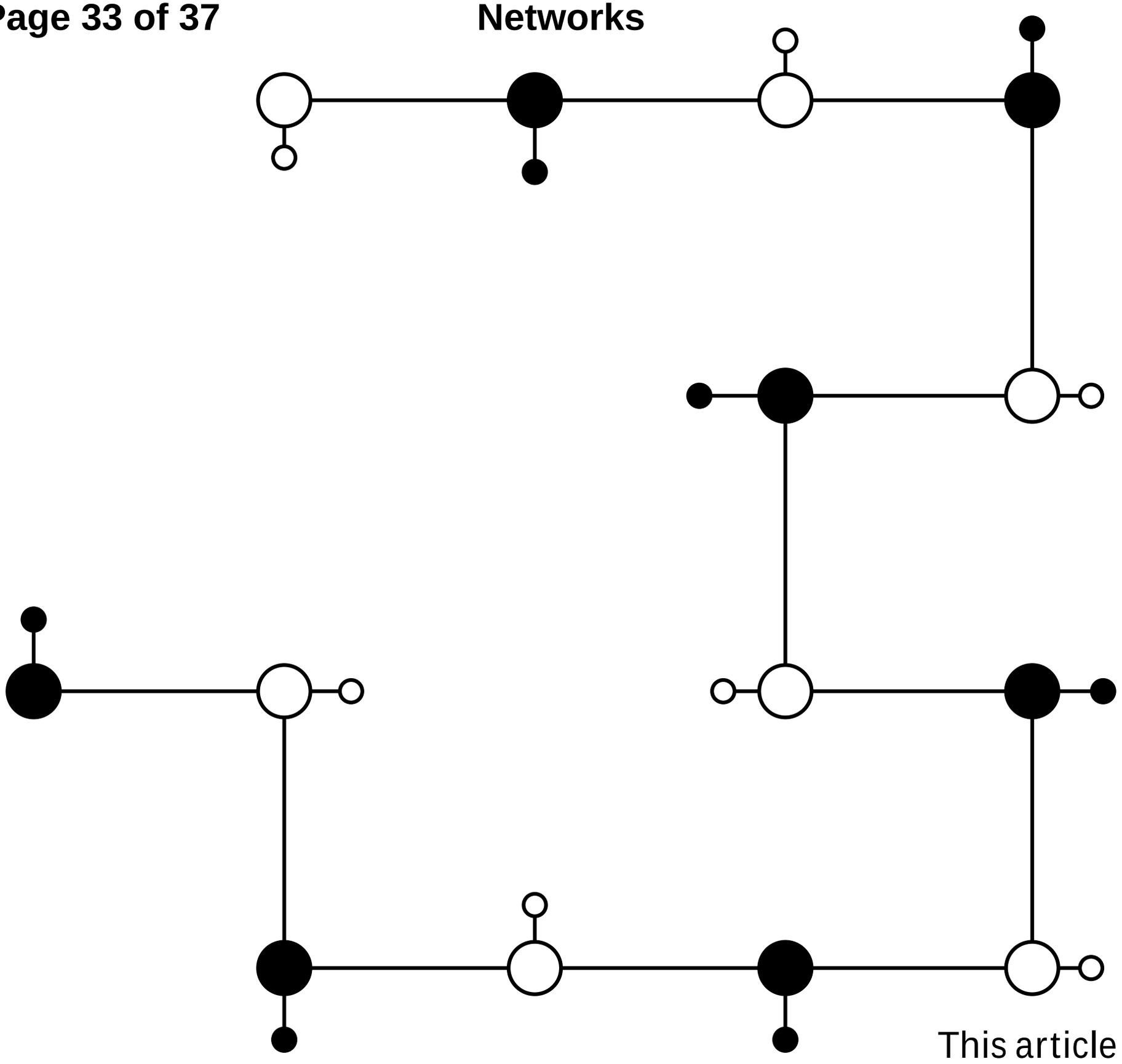
Algorithm 3

Input: A rooted tree T over a point set P with root v; v may have a single adopted child.
Output: A spanning tree for P in which the degree of each vertex is at most 4.

if v has at least one child
  if v has three children or fewer
    Perform the algorithm on each T_c, where c is a child of v.
  else if v has four children
    Let v_1,v_2,v_3,v_4 be the children of v, where v_1,v_2 are a $\sqrt{3}$-mutable pair.
    if $d(v, v_1) \geq d(v, v_2)$
      Remove the edge (v, v_1) and add the edge (v_1, v_2) to T.
      Perform the algorithm on T_{v_2},T_{v_3} and T_{v_4}.
    else
      Remove the edge (v, v_2) and add the edge (v_1, v_2) to T.
      Perform the algorithm on T_{v_1},T_{v_3} and T_{v_4}.
  else if v has five children
    Let v_1,v_2,v_3,v_4,v_5 be the children of v, where v_1,v_2 and v_3,v_4 are two $\sqrt{3}$-mutable pairs.
    if $d(v, v_1) \geq d(v, v_2)$
      Remove the edge (v, v_1) and add the edge (v_1, v_2) to T.
      if $d(v, v_3) \geq d(v, v_4)$
        Remove the edge (v, v_3) and add the edge (v_3, v_4) to T.
        Perform the algorithm on T_{v_2},T_{v_4} and T_{v_5}.
      else
        Remove the edge (v, v_4) and add the edge (v_3, v_4) to T.
        Perform the algorithm on T_{v_2},T_{v_3} and T_{v_5}.
    else
      Remove the edge (v, v_2) and add the edge (v_1, v_2) to T.
      if $d(v, v_3) \geq d(v, v_4)$
        Remove the edge (v, v_3) and add the edge (v_3, v_4) to T.
        Perform the algorithm on T_{v_1},T_{v_4} and T_{v_5}.
      else
        Remove the edge (v, v_4) and add the edge (v_3, v_4) to T.
        Perform the algorithm on T_{v_1},T_{v_3} and T_{v_5}.
return T.







This article is p

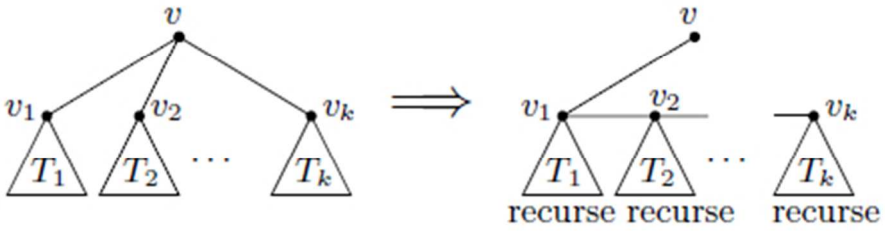
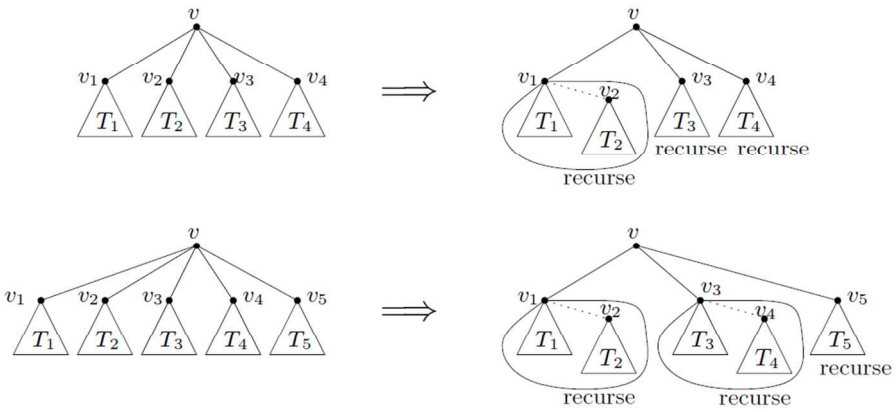
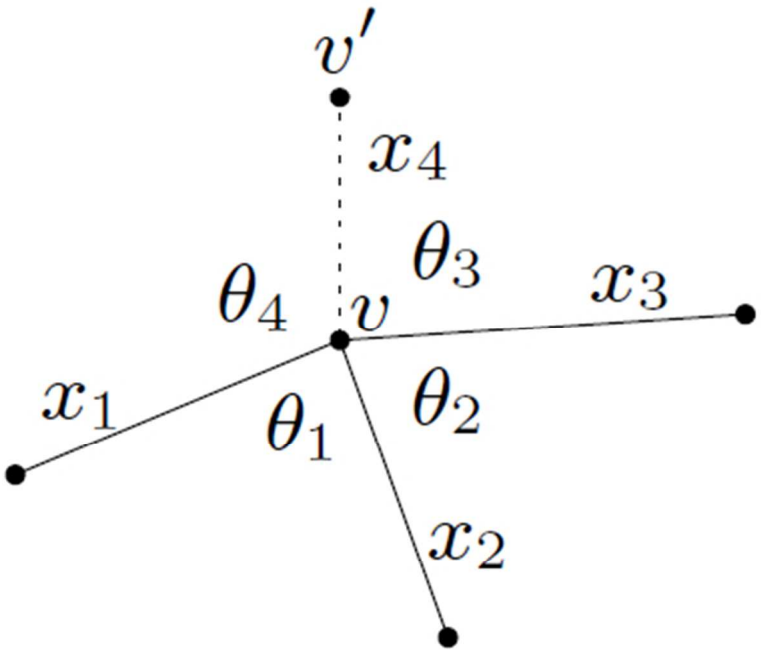


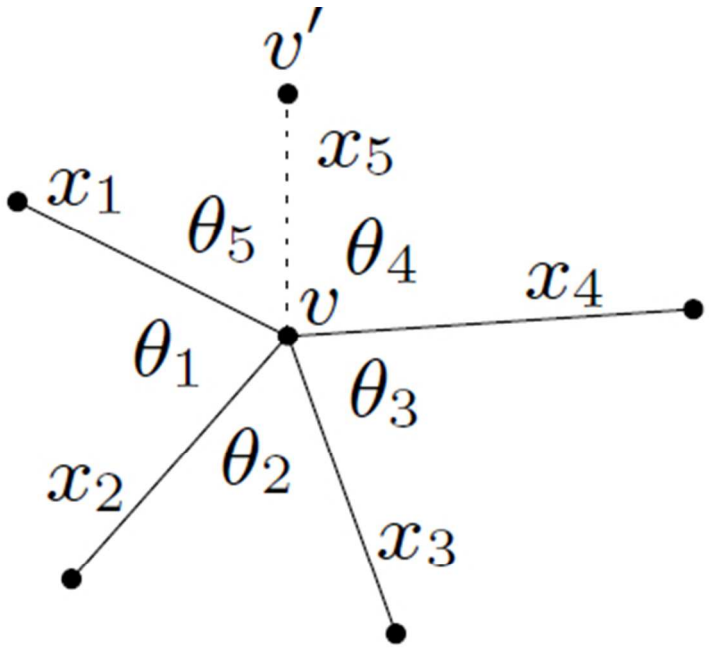
Figure 5: The illustration of the Khuller, Raghavachari and Young algorithm [4].



The illustration of Chan's algorithm [4].



The notation for Theorem 11, originally given in 4.



The notation for Theorem 11, originally given in 4.