Report

Low-cost fault-tolerant spanning graphs for point sets in the Euclidean plane

Author(s):
Nardelli, Enrico; Stege, Ulrike; Widmayer, Peter

Publication Date:
1997-03

Permanent Link:

https://doi.org/10.3929/ethz-a-006651988 →

Low-Cost Fault-Tolerant Spanning Graphs for Point Sets in the Euclidean Plane*

Enrico Nardelli † Ulrike Stege ‡ ‡ Peter Widmayer §

March 27, 1997

Abstract

The concept of the minimum spanning tree (MST) plays an important role in topological network design, because it models a cheapest connected network. In a tree, however, the failure of a vertex can disconnect the network. In order to tolerate such a failure, we generalize the MST to the concept of a cheapest biconnected network. For a set of points in the Euclidean plane, we show that it is NP-hard to find a cheapest biconnected spanning graph, where edge costs are the Euclidean distances of the respective points. We propose a different type of subgraph, based on forbidding (due to failure) the use of a vertex. A minimum spanning multi-tree is a spanning graph that contains for each possible forbidden vertex a spanning tree that is minimum among the spanning trees that do not use the forbidden vertex. We propose a worst-case time optimal algorithm for computing a minimum spanning multi-tree for a planar Euclidean point set. A minimum spanning multi-tree is cheap, even though it embeds a linear number of MSTs: Its cost is more than the MST cost only by a constant factor. Furthermore, we propose a linear time algorithm for computing a cheap vertex failure tolerant graph, given the Delaunay triangulation. This graph bounds the cost of the minimum spanning multi-tree from above.

1 Introduction

In the past few years, networks with several desirable properties have been studied. Based on the classical minimum spanning tree (MST) concept for a graph with edge costs, a *most vital edge* of a MST has been defined as an edge whose removal from the underlying graph increases the cost of a MST by the largest amount. Efficient algorithms for finding a most vital edge have been proposed [17, 18]. Along a different track, spanning graphs for point sets in Euclidean space have been studied with the goal of preserving all Euclidean distances up to a constant factor in the shortest paths of the graphs – the *Euclidean spanners* [3, 4, 5, 6, 11, 12]. Recently, it has been shown that Euclidean spanners with additional desirable properties can be found efficiently, such as low total cost, bounded degree, and low diameter [4].

In this paper, we aim at a different combination of network properties, motivated by the possibility of a vertex failure in a computer network. In a simply connected network, such a failure can disconnect the network into several parts. We study the question of how we can save a network from being disconnected by a failing vertex, while at the same time keeping the total

*We gratefully acknowledge the support of the Swiss Nationalfonds.

†Univ. of L'Aquila, Via Vetoio, Coppito, I-67010 Aquila. E-Mail: nardelli@univaq.it.

‡corresponding author

§Departement Informatik, ETH Zentrum, CH-8092 Zürich. E-Mail: {stege,widmayer}@inf.ethz.ch.

cost of the network low. At one end of the spectrum of problems in this context, we generalize the concept of the minimum spanning tree of a graph towards a higher degree of connectivity: instead of searching for a cheapest simply connected spanning graph (i.e. a MST), we search for a cheapest biconnected spanning graph. The underlying graph is defined as the complete graph induced by a set of points in the plane and the Euclidean metric; that is, the cost of an edge is the Euclidean distance of its incident points. We show that even in this highly structured case, the computation of a cheapest biconnected spanning subgraph is NP-hard. This is in sharp contrast with the complexity of the computation of a MST for a given point set: via the Voronoi diagram and its dual, the Delaunay triangulation, the Euclidean MST of a set of n planar points can be computed in time $O(n \log n)$ [21].

We then turn to a restricted type of MST, where the restriction forbids the use of a particular vertex. The idea behind this concept has a similar focus as the investigations to the most vital edge: how can we still have a cheap connected network if a vertex fails? We are interested in a graph that contains a MST of the remaining vertices and edges, no matter which vertex is forbidden. Since this graph is the union of all possible restricted MSTs, we call it a *vertex fault-tolerant minimum spanning multi-tree* (VMT). A vertex failure has the least possible consequences in a VMT: we are guaranteed to still have a working MST available that avoids the failed vertex. We show that we even do not need to pay a lot for this property: the total cost of a VMT is less than four times the cost of a MST. As a side effect, it may therefore even serve as an approximation (although not the best possible) for a cheapest biconnected graph. Furthermore, we propose an algorithm to compute a VMT for a given set of n points in the Euclidean plane in time $O(n \log n)$.

Our algorithm first computes an appropriate generalization of a Delaunay triangulation of the point set, and then finds the result as a subgraph of the generalized Delaunay graph. In this generalization, the planarity of the Delaunay triangulation is lost, and also the resulting minimum spanning multi-tree need not be planar. Note that in case the output must represent adjacency explicitly (e.g. it is an adjacency list representation of a graph), our algorithm is asymptotically worst-case time optimal, since sorting can be reduced to VMT computation in linear time (just as it can be reduced to MST computation.).

In our problem description so far, the failure of a vertex was our only concern. Concerning the case of edge failures, our concepts and results carry over. It is interesting to note that the problem of finding all the replacement edges when edges of the minimum spanning tree are deleted was considered for the case of general graphs by Chin and Houck [9]. They gave an $O(n^2)$ worst case time algorithm, which was later improved by Tarjan [23] (pp. 712-713) to $O(m\alpha(m, n))$ worst case time and space, where n denotes the number of vertices and m the number of edges of the given graph. Hence we do not discuss an algorithm to compute an *edge fault-tolerant minimum spanning multi-tree* (EMT).

Work related to our problem has been done in the field of dynamic algorithms for Euclidean MST maintenance under insertion or deletion [1, 13, 14]. But since the objective there is to compute the new MST resulting from each update, even the currently best solution for dynamic maintenance of a Euclidean MST [13] only gives a $O(n \log^2 n)$ time bound when directly applied to our problem.

The paper is organized as follows. Section 2 recalls a few central definitions. Section 3 defines the problem of computing a cheapest biconnected spanning graph precisely and shows its NP-hardness. Section 4 introduces and discusses minimum spanning graphs for edge and vertex failures, respectively. Furthermore, it discusses a few useful properties of an *edge fault-tolerant minimum spanning multi-tree* EMT and a VMT and the relation between the two. Section 4.4 derives an upper bound on the total cost of EMT and VMT, showing that this

cost is only a constant times that of a MST. In Section 5, we propose an algorithm for the efficient computation of VMT and analyze its performance. Section 6 proposes a linear time algorithm for building a vertex failure tolerant graph with cost also at most $4{\cdot}\mathrm{MST}(V)$, given the Delaunay triangulation. Section 7 concludes the paper.

2 Definitions

We refer to standard graph terminology [7]. Let $G = (V, E)$ be a weighted, undirected graph, and let $c(e) \in \mathbb{R}^+$ be the *cost* (weight) *of edge* $e \in E$. To avoid the discussion of singularities, let us assume throughout the paper that all edge costs are pairwise different and every graph has at least three vertices. The *cost of a graph* $G = (V, E)$ is the sum of the costs of its edges, denoted as $c(G)$. When V is a set of points in $\mathbb{R}^2$ and the cost of each edge is the Euclidean distance between both incident points, we call G a *Euclidean graph*. In particular, *the complete Euclidean graph* $C(V)$ induced by point set V is the graph $C(V) = (V, E)$ with $E = V \times V$ and $c(x, y) = d_2(x, y)$ for $x, y \in V$, where d_2 denotes the Euclidean distance. Our assumption on different edge costs implies the following general position assumption for the point set: no two distances between points are the same.

For a given graph $G = (V, E)$ and a vertex $v \in V$, $\deg_G(v)$ denotes the *degree of* v *in* G, i.e., the number of incident edges of v in G. For simplicity, $G - e$ (resp. $G + e$) for a graph $G = (V, E)$ and an edge $e \in V \times V$ is an abbreviation for $(V, E - \{e\})$ (resp. $(V, E \cup \{e\})$); similarly, we use $G - v$ for $G = (V, E)$ and a vertex $v \in V$ as an abbreviation for $(V - \{v\}, E \cap ((V - \{v\}) \times (V - \{v\})))$.

We are interested in low cost spanning, connected subgraphs of $C(V)$ that satisfy certain constraints. Without constraints, a *minimum spanning tree of a graph* $G = (V, E)$, denoted as $\mathrm{MST}(G)$, is a lowest cost spanning, connected subgraph of G. For a given point set in the Euclidean plane, we write $\mathrm{MST}(V)$ for $\mathrm{MST}(C(V))$. The constraints of interest refer to the connectedness of the subgraph. A *cut vertex* (resp. *bridge*) in a graph G is a vertex (resp. an edge) whose removal increases the number of connected components of G. A connected graph G is *two-vertex* (resp. *two-edge*) *connected*, if G contains no cut vertex (resp. bridge).

Observation 2.1 *Let $|V| \geq 3$. Then any two-vertex connected graph $G = (V, E)$ is two-edge connected, but a two-edge connected graph need not be two-vertex connected.*

3 Minimum biconnected spanning graph computation

We are interested in the computation of a two-vertex (resp. two-edge) connected graph that spans a set V of points in the Euclidean plane and has minimum cost, denoted by $\mathrm{M2VG}(V)$ (resp. $\mathrm{M2EG}(V)$). This is a natural generalization of the minimum spanning tree problem, when simple connectedness is replaced by biconnectedness. We will now show that we need not distinguish the two types of biconnectedness:

Lemma 3.1 *Consider a set V of points in the Euclidean plane. Let $G = (V, E)$ be a M2EG(V). If $v \in V$ is a cut vertex of G then the three following conditions are satisfied:*

1. *v and all its adjacent vertices are collinear;*
2. *there exist two pairs v_{11}, v_{12} and v_{21}, v_{22} of the adjacent vertices of v such that v_{i1} and v_{i2} are connected only by paths in G which contain v (for $i = 1, 2$);*

3. $\deg_G(v) = 4$.

Proof. Let v be a cut vertex of G.

1. Pick any two neighbors $v_1, v_2 \in V$ of v. Assume v_1, v_2, and v are not collinear, (v_1, v), $(v_2, v) \in E$. If v_1 and v_2 are in the same connected component C_1 of $G - v$, then there exists a vertex $v_3 \in V$, which is adjacent to v and lies in a different connected component $C_2 \neq C_1$ of $G - v$, since v is cut vertex. Hence v, v_3 and at least one vertex of v_1 and v_2 are not collinear. Therefore w.l.o.g. let v_1 and v_2 be in two different connected components of $G - v$. Let $G' := G - (v_1, v) - (v_2, v) + (v_1, v_2)$. G' is two-edge connected, since G contains no bridges and therefore there is a path from v_1 to v_2 in G' containing v. Because of the triangle inequality $c(G') < c(G)$. This is a contradiction to the minimum cost requirement for G.

2. v is cut vertex in G. Hence $G - v$ consists of at least two connected components C_1, C_2, $C_1 \cap C_2 = \emptyset$. Since G is two-edge connected and v is cut vertex in G, each path from any vertex $x \in C_1$ to any vertex $y \in C_2$ contains v. Since G is two-edge connected there are at least two edge-disjoint paths between x and y, and therefore there are at least two pairs v_{11}, v_{12} and v_{21}, v_{22}, $(v_{11}, v_{21} \in C_1$, $v_{12}, v_{22} \in C_2)$ of adjacent vertices of v containing v in any connecting path.

3. If $\deg_G(v) < 4$, G contains a bridge, since v is cut vertex, a contradiction. Therefore assume $\deg_G(v) > 4$. We know from 1. that all adjacent vertices of v are collinear. Consider both halflines through v and all its neighbors; their union is one straight line. Since v is a cut vertex and G is minimum, v has neighbors on both halflines. Order the k adjacent vertices of v on one of the halflines according to their distance to v, with v_1 being closest to v, then v_2 and so on. Adding $(v_1, v_2), \ldots (v_{k-1}, v_k)$ to E and deleting $(v, v_2), \ldots, (v, v_{k-1})$ from E results in a cheaper graph than G, because of the triangle inequality. Contradiction.

$\diamond$

From Observation 2.1 we get that any M2EG(V) not containing a cut vertex is also a M2VG(V). Otherwise, the next lemma tells us how to transform any M2EG(V) into a M2VG(V) of the same cost.

Lemma 3.2 *Consider a set V of points in the Euclidean plane. Let $G = (V, E)$ be a M2EG(V). If G is not two-vertex connected, we can transform G into a M2VG(V) with the same cost.*

Proof. Let G be two-edge connected but not two-vertex connected. As long as the number k of cut vertices in G is at least one, transform G into a spanning two-edge connected graph G' containing less than k cut vertices such that $c(G) = c(G')$, as follows. Pick any cut vertex v in G; v satisfies the conditions in Lemma 3.1. Let $v_1, v_2 \in V$ be in different connected components of $G - v$ and $(v_1, v), (v_2, v) \in E$. Let C_1 be the connected component of $G - v$ containing v_1, C_2 the connected component of $G - v$ containing v_2. Furthermore, let $(v_3, v), (v_4, v) \in E$, $v_3 \in C_1$, $v_4 \in C_2$. Because G is two-edge connected, there is a path p from v_3 to v_1 in G such that p is edge disjoint from the path v_3, v, v_1. Analogously, there exists a path p' in G from v_4 to v_2 in G, with p' being edge disjoint from the path v_4, v, v_2. Let $G' = G - (v_1, v) - (v_2, v) + (v_1, v_2)$. There is a path not containing v in G' from any vertex v_0 in C_1 via v_1 and edge (v_1, v_2) to any

vertex v_0' in C_2. Hence G' is two-edge connected and the number of cut vertices is strictly less than k. Since $c((v_1, v_2)) = c((v_1, v)) + c((v_2, v))$, we have $c(G) = c(G')$. ◇

Let us state the problem of computing a M2EG(V) for the special case of a point set V on an integer grid in a decision version:

Problem: M2EG
Input: A set V of points in the plane with integer coordinates and an integer bound $c > 0$.
Question: Is there a two-edge connected Euclidean graph spanning V with cost at most c?

Theorem 3.3 *The problem M2EG is NP-complete.*

Proof. The problem is in NP, because we can guess a spanning graph and verify in polynomial time that it is two-edge connected and costs at most c. It is NP-hard by a reduction from Exact Cover by 3-Sets [19], in close analogy to the proof of the NP-hardness of the travelling salesman problem in [16] (the travelling salesman tour in that proof happens to be a M2EG). ◇

Note that from the construction in the proof it follows that for a weighted graph instead of a point set, the corresponding problem is also NP-complete.

4 Minimum spanning graphs for vertex failures and edge failures

4.1 Vertex fault-tolerant minimum spanning multi-trees

Let us now turn our attention to a different concept of a cheap spanning graph with stronger connectivity than a MST. Intuitively, we require that the removal of a vertex leaves us with a spanning graph that contains a MST of the new set of vertices.

More precisely, for a set V of points in the Euclidean plane, a *vertex fault-tolerant minimum spanning multi-tree* VMT(V) $= (V, E)$ is a connected minimum cost graph with the property that for each $v \in V$, MST($V - v$) $\subseteq$ VMT(V).

Lemma 4.1 *For any two-vertex connected graph $G = (V, E)$ and any vertex $v \in V$, ($MST(G) - v$) $\subseteq$ $MST(G - v)$.*

Proof. Let $v \in V$, MST(G) $= (V, E_G)$, MST($G - v$) $= (V - v, E_{G-v})$. Assume there is an edge f in MST(G) $- v$, $f \notin E_{G-v}$. (Note that f is an edge in MST(G).) Then MST($G - v$) $+ f$ has a cycle K containing f, and f is longest in K. Therefore $K \subseteq G - v \subseteq G$. That is, there is the cycle K in G and f is longest edge in K. Therefore f cannot be an edge of MST(G), a contradiction. ◇

Corollary 4.2 *For each point set V in the Euclidean plane, $MST(V) \subseteq VMT(V)$. Furthermore, $VMT(V) = MST(V)$ if $|V| < 3$, and $VMT(V) = \bigcup_{v \in V} MST(V - v)$ if $|V| \geq 3$, and $VMT(V)$ is two-vertex connected.*

Proof. Lemma 4.1 implies $\mathrm{MST}(V) \subseteq \mathrm{VMT}(V)$, and $\mathrm{VMT}(V) = \mathrm{MST}(V)$ if $|V| < 3$, and $\mathrm{VMT}(V) = \bigcup_{v \in V} \mathrm{MST}(V - v)$. We show $\mathrm{VMT}(V)$ is two-vertex connected. If $|V| < 3$, $\mathrm{VMT}(V) = \mathrm{MST}(V)$. Since $\mathrm{MST}(V)$ has no cut vertex for $|V| = 1$ or $|V| = 2$, $\mathrm{VMT}(V)$ is two-vertex connected. Let $|V| \geq 3$. Since $\mathrm{MST}(V - v) \subseteq \mathrm{VMT}(V)$ for all $v \in V$, $\mathrm{VMT}(V) - v$ is connected for all $v \in V$ and therefore $\mathrm{VMT}(V)$ contains no cut vertex. ◇

This suggests a method for the construction of $\mathrm{VMT}(V)$: we start with $\mathrm{MST}(V) = (V, E_M)$. For each vertex $v \in V$, by removing v and all its incident edges we break the $\mathrm{MST}(V)$ into $\deg(v)$ connected components (in this paragraph for simplicity $\deg(v)$ denotes $deg_{MST(V)}(v)$); note that $\deg(v) < 6$. Let $a_1(v), \ldots, a_{\deg(v)-1}(v)$ be the $\deg(v) - 1$ edges that connect all $\deg(v)$ components into a single component at the minimum cost. That is, $\mathrm{MST}(V - v) = \mathrm{MST}(V) - v + a_1(v) + \ldots + a_{\deg(v)-1}(v)$. Clearly, the set of edges of $\mathrm{VMT}(V)$ is $E_M \cup \bigcup_{v \in V} \bigcup_{i=1}^{\deg(v)-1} a_i(v)$.

4.2 Edge fault-tolerant minimum spanning multi-trees

In analogy to vertex failures, we define a graph that survives an edge failure and has minimum cost among all such graphs. Intuitively, we require that the removal of an edge leaves us with a spanning graph that contains a MST of the new set of edges. This concept is related to the notion of the most vital edge (see [17, 18]) and to the notion of survivability of a network in presence of an edge failure [10, 22].

For a set V of points in the Euclidean plane, $|V| \geq 3$, an *edge fault-tolerant minimum spanning multi-tree* $\mathrm{EMT}(V) = (V, E)$ is a connected minimum cost graph with the property that for each $x, y \in V$, $\mathrm{MST}(C(V) - (x, y)) \subseteq \mathrm{EMT}(V)$. Due to our non-degeneracy assumption with respect to edge costs, the MST as well as the EMT are unique for a set V of points.

Let us clarify the relation between $\mathrm{MST}(V)$ and $\mathrm{EMT}(V)$.

Lemma 4.3 *For any two-edge connected graph $G = (V, E)$ and any edge $e \in E$, $(MST(G) - e) \subseteq MST(G - e)$.*

Proof. Let $e \in E$, $\mathrm{MST}(G) = (V, E_G)$, $\mathrm{MST}(G - e) = (V, E_{G-e})$. Assume there is an edge f in $\mathrm{MST}(G) - e$, $e \notin E_{G-e}$. (Note that f is an edge in $\mathrm{MST}(G)$.) Then $\mathrm{MST}(G - e) + f$ has a cycle K containing f, and f is longest in K. Therefore $K \subseteq G - e \subseteq G$. That is, there is the cycle K in G and f is longest edge in K. Therefore f cannot be an edge of $\mathrm{MST}(G)$, a contradiction. ◇

Corollary 4.4 *For each point set V in the Euclidean plane, $|V| \geq 3$, $MST(V) \subseteq EMT(V)$. Furthermore $EMT(V) = \bigcup_{e \in V \times V} MST(C(V) - e)$ and $EMT(V)$ is two-edge connected.* ◇

Proof. $\mathrm{EMT}(V)$ is two-edge connected, since $\mathrm{MST}(C(V) - (x, y)) \subseteq \mathrm{EMT}(V)$ for each $x, y \in V$ and therefore $\mathrm{EMT}(V) - (x, y)$ is connected for all $x, y \in V$, and therefore $\mathrm{EMT}(V)$ contains no bridge. ◇

This suggests a method for the construction of EMT(V): we start with MST(V) = (V, E_M). For each edge $e \in E_M$, by removing e we break the MST(V) into two connected components, say trees T_1 and T_2. Let $a(e)$, the addition for the removal of e, be the second cheapest edge connecting T_1 with T_2 (the cheapest is e). Clearly, the set of edges of EMT(V) is $E_M \cup \bigcup_{e \in E_M} \{a(e)\}$. An algorithm implementing this method in $O(m\alpha(m, n))$ worst case time and space has been given by Tarjan [23] (pp. 712-713). Here m is the number of edges, n the number of vertices and α is a functional inverse of Ackermann's function.

4.3 Useful properties of minimum spanning multi-trees

In order to compute VMT efficiently, let us have a look at some useful properties of EMT and VMT and their relation.

Lemma 4.5 *Consider a set V of points in the Euclidean plane, $|V| \geq 3$. Then EMT(V) $\subseteq$ VMT(V).*

Proof. Let (x, y) be an edge in MST(V). Consider the cut (X, Y) induced by (x, y) in MST(V), where X and Y are disjoint subsets of V, $X \cup Y = V$, X and Y are the node sets of the connected components of MST(V) $- (x, y)$. Let (u, v) be the cheapest edge with $u \in X$ and $v \in Y$, $(u, v) \neq (x, y)$. Then $(u, v) \in$ EMT(V). From Corollary 4.4 we know that every edge in EMT(V) is such a second cheapest edge across a cut induced by a MST edge. W.l.o.g let $x \neq u$ and $x \neq v$. Assume (u, v) is not in MST($V - x$). Then MST($V - x$) contains an edge (u', v'), $u' \in X, v' \in Y$, $c((u, v)) < c((u', v'))$. Contradiction. $\diamond$

Figure 1 illustrates for a given point set V in the Euclidean plane the relations between MST(V), EMT(V), and VMT(V).

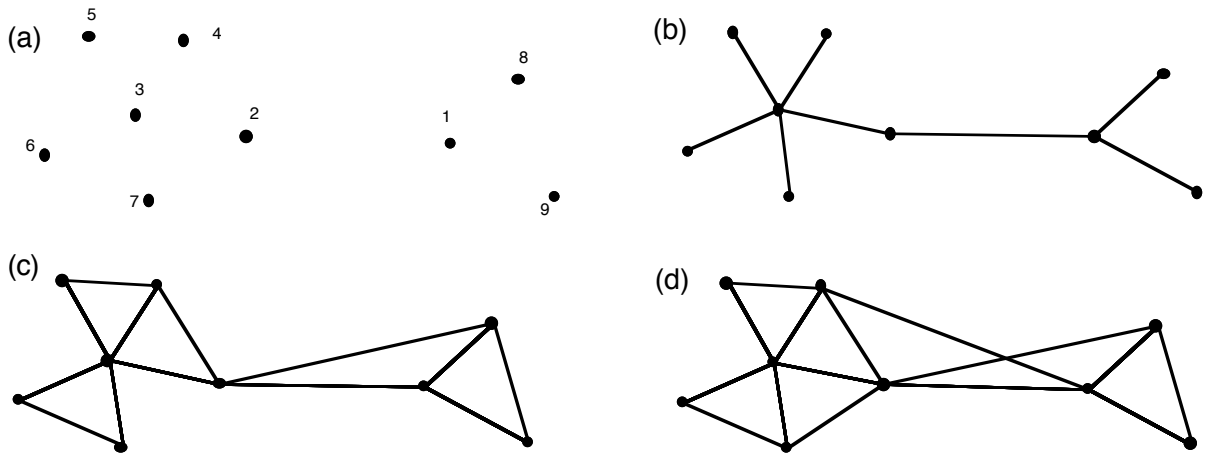

Figure 1: (a) a given planar point set V, (b) MST(V), (c) EMT(V), and (d) VMT(V).

Lemma 4.6 *For each set V of n points in the Euclidean plane, the number of edges in VMT(V) is at most $2n - 3$.*

Proof. In the VMT-construction suggested in Section 4, we add to MST(V) at most $\sum_{v \in V} (\deg_{MST(V)}(v) - 1)$ edges. Since $\sum_{v \in V} (\deg_{MST(V)}(v) - 1) = 2(n-1) - n$, the claim is proved. ◇

With Lemma 4.5 we immediately get:

Corollary 4.7 *For each set V of n points in the Euclidean plane, the number of edges in EMT(V) is at most $2n - 3$.* ◇

With the number of edges being bounded so strongly, it is natural to ask whether the multi-trees are planar; planarity might help in the efficient computation. Unfortunately, we get:

Lemma 4.8 *EMT(V) for a set V of points in the Euclidean plane is not necessarily planar.*

Proof. Figure 5 in the Appendix shows an example for V for which EMT(V) is not planar. ◇

With Lemma 4.5 we get

Corollary 4.9 *VMT(V) for a set V of points in the Euclidean plane is not necessarily planar.*◇

4.4 A bound on the cost of multi-trees

Since both minimum spanning multi-trees contain MST(V), we aim at bounding their costs in terms of the cost of the MST(V).

Theorem 4.10 *For a set V of points in the Euclidean plane, $c(VMT(V)) < 4 \cdot c(MST(V))$.*

Proof. We prove the existence of a graph $G = (V, E)$ with the following properties:

1. $c(G) < 4 \cdot c(\text{MST}(V))$;
2. $\text{MST}(V) \subseteq G$
3. G tolerates the failure of a vertex; more precisely, for each $v \in V$, G contains exactly $\deg_{MST(V)}(v) - 1$ edges that do not belong to MST(V) and connect MST(V) $- v$ into one connected component.

Since VMT(V) has minimum cost among all spanning graphs for V that contain MST(V) and tolerate the failure of a vertex, the three properties together imply the theorem. Graph $G = (V, E)$ is constructed as follows. Pick an arbitrary vertex $r \in V$ and consider MST(V) as a tree T rooted at r. Order the children of each vertex according to their geometric location, in clockwise order around the vertex. For each vertex v that possesses a parent $p(v)$ and children $q_1(v), \ldots, q_{\deg(v)-1}(v)$, let edge-set $E(v) = \{(q_i(v), q_{i+1}(v)) | 1 \leq i < \deg(v) - 1\} \cup \{(q_{\deg(v)-1}(v), p(v))\}$. (In this paragraph for simplicity $\deg(v)$ denotes $\deg_{MST(V)}(v)$.) For the root r let $E(r) = \{(q_i(r), q_{i+1}(r)) | 1 \leq i < \deg(r)\}$. Now, let $E = \bigcup_{v \in V} E(v) \cup E_M$. It is clear that G has properties 2 and 3.

To show that G has property 1, we charge the cost for each edge in some $E(v)$ to an edge of the MST(V) in such a way that each MST edge is charged for at most three times its own cost. The cost of a new edge $(q_i(v), q_{i+1}(v))$ between two adjacent children of the same vertex v in T is no more than the sum of the costs of edges $(v, q_i(v))$ and $(v, q_{i+1}(v))$. In this way, a MST edge is charged at most twice its own cost, once from each of the at most two adjacent siblings. In addition, we charge the cost for a new edge from $q_{\deg(v)-1}(v)$ to its grandparent $p(v)$ to edges $(q_{\deg(v)-1}(v), v)$ and $(v, p(v))$, adding a third time its own cost to both edges. Since the edges in T incident to the leaves of T are charged at most twice, we have cost strictly less than $3 \cdot c(\text{MST}(v))$ for new egdes and cost $c(\text{MST}(v))$ for the MST. $\diamond$

With Lemma 4.5 we immediately get:

Corollary 4.11 *For a set V of points in the Euclidean plane, $c(EMT(V)) < 4 \cdot c(MST(V))$.* $\diamond$

5 Efficient algorithms for vertex fault-tolerant multi-trees

It is by now standard textbook knowledge that the computation of a MST(V) for a set V of n points in the Euclidean plane takes only $O(n \log n)$ time in the worst case [21], even though the underlying graph $C(V)$ has $\Theta(n^2)$ edges. This is possible simply because $C(V)$ is never computed explicitly; instead, the fact that MST(V) is a subgraph of the Delaunay triangulation $DT(V)$ for V is used to first compute DT(V) and then apply a MST algorithm to DT(V). DT(V) can be computed (directly or via the Voronoi diagram) in $O(n \log n)$ time; it has $O(n)$ edges and is planar (by definition). A special MST algorithm for planar graphs even runs in $O(n)$ time [21].

For the computation of EMT(V) and VMT(V), we also want to avoid the explicit computation of $C(V)$ for efficiency reasons. We therefore search, similar in spirit to the role of DT(V) for MST(V), for a graph D(V) that has few edges, can be computed efficiently, and contains VMT(V) (and therefore also EMT(V), by Lemma 4.5). Let us define D(V) as an appropriate generalization of DT(V), taking vertex failures into account. More precisely, let D(V) = (V, E) be the connected minimum cost graph with the property that for each $v \in V$, DT($V - v$) $\subseteq$ D(V). Since MST(V) $\subseteq$ DT(V) and therefore for all $v \in V$, MST($V - v$) $\subseteq$ DT($V - v$) we have VMT(V) $\subseteq$ D(V). Let us call D(V) the *fault-tolerant Delaunay graph* for V (see Figure 2).

This definition suggests a construction method for D(V). We start with DT(V). For each vertex $v \in V$, we remove v from DT(V) and update the Delaunay triangulation with the algorithm of Aggarwal et al., given in [2].

For updating the Delaunay triangulation, if one vertex v is deleted, we only have to retriangulate the polygon in which v lies. For each vertex v with $\deg_{DT(V)}(v) \neq 2$, the updated Delaunay triangulation contains at most $\deg_{DT(V)}(v) - 2$ new edges. If $\deg_{DT(V)}(v) = 2$, that is v is element of the convex hull of V, the new convex hull simply drops v and both incident edges and does not add any new edge to the triangulation.

Because the number of edges of $DT(V)$ is at most $3n - 6$ due to planarity, we get a total of at most $3n - 6 + \sum_{v \in V} \max\{0, (\deg_{DT(V)}(v) - 2)\} = O(n)$ edges in D(V). Using the technique proposed in [2], all updates to DT(V) and the computation of new edges can be performed in time $O(n)$ overall in the worst case. We summarize the discussion in the following theorem:

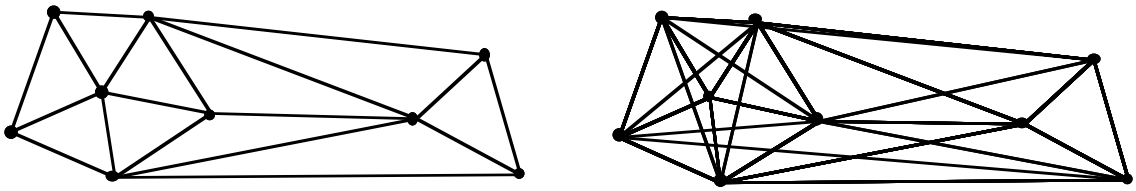

Figure 2: DT(V) (left) and D(V) of point set V in Figure 1.

Theorem 5.1 *Given DT(V) for a set V of n points in the Euclidean plane, a fault-tolerant Delaunay graph D(V) can be computed in time $O(n)$. D(V) contains VMT(V) and EMT(V).*

Proof. From the discussion above. ◇

Let us now turn to the computation of VMT(V) = (V, E), given D(V). We first compute MST(V) from D(V) in time $O(n \log n)$ for instance with Kruskal's algorithm [20] (applying more sophisticated MST algorithms does not pay off at this point, because the edges will anyway need to be sorted in increasing order of their cost in our algorithm). We keep adding edges to MST(V) as follows. We inspect the edges in D(V)−MST(V) in order of increasing cost. We maintain a collection $\mathcal{C}$ of subgraphs of D(V), called *bicomponents*, where each bicomponent $s = (V_s, E_s)$ is a subgraph of VMT(V) and represents either a single edge $e \in E$ (which is the case for each edge of the minimum spanning tree initially) or a two-vertex connected component. Furthermore at every particular stage of computing VMT(V), we have that $\bigcup_{s \in \mathcal{C}} V_s = V$, the bicomponents are pairwise different, and if s_1, s_2 are two bicomponents, then $s_1 \not\subseteq s_2$. Additionally, $E_{s_1} \cap E_{s_2} = \emptyset$ for two edge sets s_1, s_2 in the same stage of computing VMT(V).

For an inspected edge $e = (v, w)$, we distinguish two cases. If e connects two vertices in the same bicomponent, we discard e. Otherwise e connects two vertices in different bicomponents. In this case, we select e. We consider the path $p = (\{v, x_1, x_2, \ldots x_k, w\}, \{(v, x_1), (x_1, x_2), \ldots, (x_k, w)\})$ from v to w in MST(V). Path p and all bicomponents $s = (V_s, E_s)$ that have a common edge with p, build in the particular stage of computing VMT(V) a bicomponent. Therefore they are combined into one bicomponent. This computation ends with a single bicomponent $s_0 = (V_{S_0}, E_{S_0})$ where $V_{s_0} = V$ (cfr. the example, shown in Fig. 3 and 4).

In order to prove the correctness of this algorithm, we show that for a given point set V in the Euclidean plane, in every stage of the computation, i.e., after rejecting or choosing an edge e in D(V), for every bicomponent $s = (V_s, E_s)$, we have s =VMT(V_s).
First we consider a useful lemma:

Lemma 5.2 *For a set V of points in the Euclidean plane, let MST(V) = (V, E) be the minimum spanning tree of V, and let $G' = (V', E')$ with $V' \subseteq V$ be a connected graph for which $G' \subseteq$ MST(V). Then MST(V') = G'.*

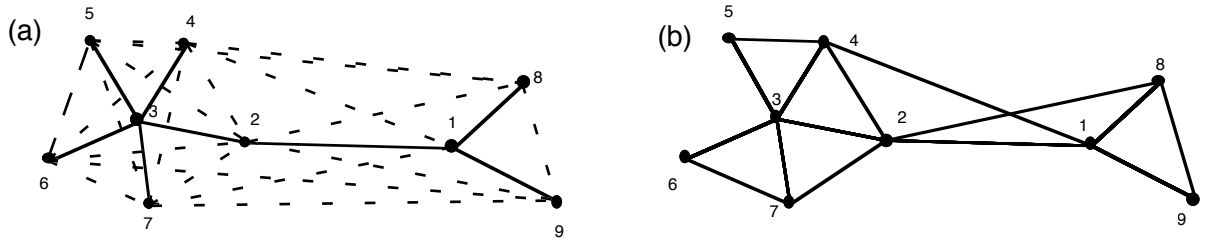

Figure 3: (a) D(V) of point set V in Figure 1. The straight edges are edges of MST(V). The edges in D(V) in increasing length are: $(4,5)$, $(6,7)$, $(2,4)$, $(2,7)$, $(8,9)$, $(5,6)$, $(4,7)$, $(2,5)$, $(2,6)$, $(4,6)$, $(2,8)$, $(1,4)$, $(1,7)$, $(4,8)$, $(7,9)$, $(5,8)$, $(6,9)$. (b) VMT(V) of point set V.

edge	choose	E_{s_i} of bicomponents s_i
		$\{(1,2)\}$, $\{(2,3)\}$, $\{(3,4)\}$, $\{(3,5)\}$, $\{(3,6)\}$, $\{(3,7)\}$, $\{(1,8)\}$, $\{(1,9)\}$
$(4,5)$	yes	$\{(1,2)\}$, $\{(2,3)\}$, $\{(3,4),(4,5),(3,5)\}$, $\{(3,6)\}$, $\{(3,7)\}$, $\{(1,8)\}$, $\{(1,9)\}$
$(6,7)$	yes	$\{(1,2)\}$, $\{(2,3)\}$, $\{(3,4),(4,5),(3,5)\}$, $\{(3,6),(3,7),(6,7)\}$, $\{(1,8)\}$, $\{(1,9)\}$
$(2,4)$	yes	$\{(1,2)\}$, $\{(2,3),(3,4),(4,5),(3,5),(2,4)\}$, $\{(3,6),(3,7),(6,7)\}$, $\{(1,8)\}$, $\{(1,9)\}$
$(2,7)$	yes	$\{(1,2)\}$, $\{(2,3),(3,4),(4,5),(3,5),(2,4),(2,7),(3,6),(3,7),(6,7)\}$, $\{(1,8)\}$, $\{(1,9)\}$
$(8,9)$	yes	$\{(1,8),(1,9),(8,9)\}$, $\{(2,3),(3,4),(4,5),(3,5),(2,4),(2,7),(3,6),(3,7),(6,7)\}$, $\{(1,2)\}$
$(5,6)$	no	$\{(1,8),(1,9),(8,9)\}$, $\{(2,3),(3,4),(4,5),(3,5),(2,4),(2,7),(3,6),(3,7),(6,7)\}$, $\{(1,2)\}$
$(4,7)$	no	$\{(1,8),(1,9),(8,9)\}$, $\{(2,3),(3,4),(4,5),(3,5),(2,4),(2,7),(3,6),(3,7),(6,7)\}$, $\{(1,2)\}$
$(2,5)$	no	$\{(1,8),(1,9),(8,9)\}$, $\{(2,3),(3,4),(4,5),(3,5),(2,4),(2,7),(3,6),(3,7),(6,7)\}$, $\{(1,2)\}$
$(2,6)$	no	$\{(1,8),(1,9),(8,9)\}$, $\{(2,3),(3,4),(4,5),(3,5),(2,4),(2,7),(3,6),(3,7),(6,7)\}$, $\{(1,2)\}$
$(4,6)$	no	$\{(1,8),(1,9),(8,9)\}$, $\{(2,3),(3,4),(4,5),(3,5),(2,4),(2,7),(3,6),(3,7),(6,7)\}$, $\{(1,2)\}$
$(2,8)$	yes	$\{(2,3),(3,4),(4,5),(3,5),(2,4),(2,7),(3,6),(3,7),(6,7)\}$, $\{(1,2),(1,8),(1,9),(8,9),(2,8)\}$,
$(1,4)$	yes	$\{(1,2),(1,8),(1,9),(8,9),(2,8),(2,3),(3,4),(4,5),(3,5),(2,4),(2,7),(3,6),(3,7),(6,7),(1,4)\}$

Figure 4: Bicomponents in the computation of VMT(V) (cfr. Figure 3(b).)

Proof. Given $G' = (V', E') \subseteq \text{MST}(V)$, G' connected. Then G' is spanning tree of V'. Assume $\text{MST}(V') \neq G'$. Hence $c(\text{MST}(V')) < c((V', E'))$. Let G be MST(V), with the subgraph (V', E') replaced by MST(V'). G is spanning tree of V. Therefore $c(G) = c(\text{MST}(V)) - c((V', E')) + c(\text{MST}(V'))$ and, since $c(\text{MST}(V')) - c((V', E')) < 0$, $c(G) < c(\text{MST}(V))$. Contradiction. ◇

It is easy to see that at the beginning of the computation of VMT(V), every bicomponent $s_i = (V_{s_i}, E_{s_i})$ is a VMT, since $|E_{s_i}| = 1$ and therefore VMT(V_{s_i}) = MST(V_{s_i}). Now let us consider any stage of computing the VMT(V). All bicomponents are VMTs, and in the next step the algorithm considers e. Furthermore all edges $e' \notin$ MST, $c(e') < c(e)$, have already been chosen or discarded by the algorithm. Now, if we discard e, we do not change any bicomponent, and all bicomponents remain VMTs. If we choose edge $e = (v, w)$, v and w do not belong to the same bicomponent yet. The path $p = (V_p, E_p)$ from v to w in MST(V), (v, w), and all bicomponents having a common edge with p build the new bicomponent.

Let $s_i = (V_{s_i}, E_{s_i})$, $1 \leq i \leq bp \leq |E_p|$, be the bicomponents with $s_i \cap p \neq \emptyset$ and $s_i = \bigcup_{v \in V_{s_i}} \text{MST}(V_{s_i} - v)$, where bp is the number of bicomponents appearing on path p. For simplicity let $H = V_p \cup \bigcup_{E_{s_i} \cap E_p \neq \emptyset} V_{s_i}$ be the set of vertices in the new bicomponent and let $S = \bigcup_{E_{s_i} \cap E_p \neq \emptyset} s_i$ be the set of old bicomponents to be connected into one with path p and the chosen edge e. We show that $(p+e) \cup S =$VMT(H), i.e. $(p+e) \cup S = \bigcup_{v \in H} \text{MST}(H - v)$. Lemma 5.2 implies that $\text{MST}(H) = p \cup \bigcup_{s_i \in S} \text{MST}(V_{s_i})$. Assume $(p+e) \cup S \neq \bigcup_{v \in H} \text{MST}(H - v)$. Then there is an edge e' in $\bigcup_{v \in H} \text{MST}(H - v)$ and not in $(p+e) \cup S$ or there is an edge e' in $(p+e) \cup S$ and not in $\bigcup_{v \in H} \text{MST}(H - v)$. In the first case there exists a vertex $x \in V_p$ such that e' is the shortest connection of K_1 and K_2, who are two of the connected components of $\text{MST}(H) - x$. Therefore $c(e') < c(e)$ and e' is not chosen by the algorithm until now, a contradiction. In the second case $e' \in S + e$. Since the number of edges in $\bigcup_{v \in H} \text{MST}(H - v)$ is at least the number of edges in $(p + e) \cup S$, if $e' = e$ or $e' \neq e$ and e not in $\bigcup_{v \in H} \text{MST}(H - v)$, there is an edge e'' in $\bigcup_{v \in H} \text{MST}(H - v)$, e'' not in $(p+e) \cup S$. That is there exists a vertex $x \in V_p$ such that e'' is the shortest connection of K_1 and K_2, who are two of the connected components of $\text{MST}(H) - x$. Therefore $c(e'') < c(e)$ and e'' is not chosen by the algorithm until now, a contradiction. If $e' \neq e$ and e is in $\bigcup_{v \in H} \text{MST}(H - v)$, then $c(e') < c(e)$, and e' is chosen by the algorithm. That is there is a s_{i_0} $(1 \leq i_0 \leq |E_p|)$ with e' is edge in s_{i_0} and therefore e' is edge of $\bigcup_{v \in V_{s_{i_0}}} \text{MST}(V_{s_{i_0}} - v)$, a contradiction. This completes the proof of the correctness of the algorithm.$\diamond$

Let us consider how to implement this algorithm efficiently. Since for any two bicomponents $s_i = (V_{s_i}, E_{s_i})$, $s_j = (V_{s_j}, E_{s_j})$ in one step of computing VMT(V), $E_{s_i} \cap E_{s_j} = \emptyset$ for $i \neq j$, the sets of the bicomponents build a partition. For our purpose, it is sufficient to represent a bicomponent by the set of its edges. So to update the bicomponents efficiently, we put the edges of the bicomponents in a simple union find structure (cfr. [24]). MakeSet(e) creates a new bicomponent consisting of the single edge e. In general, the two vertices incident with edge e define a path in the rooted MST(V) that can be traversed by following MST(V) edges from both vertices towards the root of the MST(V) until the lowest common ancestor of the two vertices is met. To decide when this happens, we keep track for each bicomponent of the vertex closest to the root of the MST(V). We call this vertex the *root of the bicomponent* and use it as the indentifier of the bicomponent in the union-find structure. This identifier is created easily in a makeset operation by taking the vertex of e closer to the root of MST(V). Find(e) returns the identifier of the bicomponent containing edge e. Union(e, f) joins the bicomponents containing edges e and f.

Then we can describe the algorithm for VMT computation as follows:

procedure VMT(V: set of points in the Euclidean plane): set of edges;
Compute D(V);
Compute MST(D(V)) with Kruskal's algorithm; (* Computes the rooted MST with the algorithm of Kruskal, and a priority queue Q which contains the edges of D(V) without the edges

of MST(V) in sorted order. Let r be the root of the rooted MST. *)
for all vertices $v \in V$ **do** compute v.level (the level of v in the rooted MST);
    (* Here r.level := 1 and son(v).level := v.level +1. *)
for all edges $(v, w) \in$ MST **do** MakeSet((v, w));
while the number of bicomponents > 1 **do**
    (v, w) := Min(Q); DeleteMin(Q);
    if not SameBicomponent(v, w) **then** (* v and w join the bicomponents on the path p from v to w in MST(V) into a new one. *)
        finished := FALSE; (* finished is set TRUE, if all bicomponents in path p are joined into one bicomponent. *)
        $(v_0, w_0) := (v, w)$; (* Don't loose the original. *)
        MakeSet((v_0, w_0));
        while not finished **do** (* All bicomponents in path p are walked along in descending order of the levels. *)
            E_v := FALSE; E_w := FALSE;
            if v.level $\geq$ w.level **then**
                v' := Find(v, father(v));
                E_v := TRUE;
            if v.level $\leq$ w.level **then**
                w' := Find(w, father(w));
                E_w := TRUE;
            if not SameBicomponent(v', w') **then**
                if E_v **then** Union((v, father(v)), (v_0, w_0)); $v := v'$;
                if E_w **then** Union((w, father(w)), (v_0, w_0)); $w := w'$;
            else
                Union((v, father(v)), (v_0, w_0)); Union((w, father(w)), (v_0, w_0));
                if v'.level = w'.level **then** Union((v', father(v')), (v_0, w_0));
                finished := TRUE;
end VMT.

The procedure SameBicomponent is given by:

procedure SameBicomponent(v, w: node): boolean; (* Returns TRUE, if there are edges e and f incident to v resp. w with e and f are in the same bicomponent, otherwise FALSE. *)
    return Find((v, father(v))) = Find((w, father(w))) ;
end SameBicomponent.

Theorem 5.3 *For a set V of n points in the Euclidean plane, VMT(V) can be computed in time $O(n \log n)$.*

Proof. If the algorithm chooses an edge, the number of bicomponents decreases per one pass through the while-statement by at least one as follows: Choosing an edge $e = (v, w)$ means creating a new set e. This increases the number of bicomponents momentarily by 1. Then we join e and all bicomponents in the path p in MST(V) connecting v with w. Path p consists of at least two edges, and at least two edges in p are in different bicomponents (otherwise we do not choose e). The creation of the new bicomponent hence decreases the number of bicomponents by at least two. In summary after the pass through the loop, the number of bicomponents is decreased by at least one.

How often do we use the operations Find, Union, and SameBicomponent? At least n times, if we only add one edge to receive a VMT. In worst case we pass the outer while loop $(n-2)$ times and reduce the number of bicomponents by one in each pass through the loop. Then every pass causes two Find, two Union operations, and one SameBicomponent operation. In summary, we use any of the three operations at most $2n-2$ times.

MakeSet can be implemented in time $O(1)$, Union and Find in time $O(\log n)$, and since the degree of a vertex in the Euclidean MST is bounded by a constant, also SameBicomponent needs time $O(\log n)$. ◇

We also arrive at a $O(n \log n)$ time algorithm if we apply Eppstein's offline algorithm [13] of the dynamic minimum spanning tree problem to D(V). Although we are in the Euclidean case, this is possible, since we only remove one vertex in the MST at once. Eppstein's algorithm, however, solves a more general problem and, as a consequence, is more complex and therefore harder to implement.

6 A fault-tolerant graph with weaker conditions than a VMT

In this section, we relax the conditions of a VMT with the goal of arriving at a faster algorithm. We search for a vertex failure tolerant graph with cost no more than a constant times the cost of the MST. We give a linear time algorithm building a cheap two-vertex connected graph, C2VG(V), for a Euclidean point set V in the plane, $|V| \geq 3$, given DT(V). The cost of G is $c(G) \leq k \cdot c(MST(V))$, $k = 4$. This is the algorithm:

1. Compute MST(V) from DT(V) with the algorithm by Cheriton and Tarjan [8].
2. Build a new graph $G'(V) = (V, E')$ with edge set $E' := \{(v,x) | (v,w)$ and (w,x) are edges in MST(V)$\}$.
3. Compute the minimum spanning forest MSF(G') with the algorithm by Cheriton and Tarjan [8].
4. C2VG(V) = MST(V)$\cup$ MSF($G'(V)$).

The next Lemma shows that C2VG(V) has the desired properties.

Lemma 6.1 *1. $G'(V)$ has $O(n)$ edges.*

2. C2VG(V) is two-vertex connected.

3. $c(C2VG(V)) \leq 4 \cdot c(MST(V))$

4. Given DT(V), the algorithm runs in linear time.

Proof. Consider the following construction with a two-coloring of MST(V), using colors red and blue.

1. Let $e \in E'$, $e = (v,w)$. Then either v and w are red or both are blue, and there are $O(|V|)$ vertices of each color. All edges with red vertices are in one connected component in G', all edges with blue vertices are in a second connected component. W.l.o.g. we consider the graph with edges with red vertices. To every blue vertex v, we conceptually attach

all edges incident to the adjacent red vertices of v. The degree of a vertex in a Euclidean minimum spanning tree is bounded by 5 (given that all edge lengths are different), and therefore at most 10 edges belong to any blue vertex. That is, $G'(V)$ has $O(|V|)$ edges.

2. By construction, $\mathrm{MST}(V) \cap \mathrm{MSF}(G'(V)) = (V, \emptyset)$. Consider any vertex $v_0 \in V$. Let $v_1, \ldots v_k$ be the vertices adjacent to v_0 in $\mathrm{MST}(V)$; they all have the same color. All vertices of the same color are a connected component of $\mathrm{MSF}(G'(V))$. Since the color of v_0 differs from the color of each v_i, $1 \leq i \leq k$, in $\mathrm{C2VG}(V) - v_0$ vertices $v_1, \ldots, v_k$ are connected within $\mathrm{MSF}(G'(V))$. Hence, $\mathrm{C2VG}(V) - v_0$ is connected.

3. This follows directly from the proof of the cost bound of $\mathrm{VMT}(V)$.

4. The algorithm proposed by Cheriton and Tarjan [8] to build a MST of a graph $G = (V, E)$ runs in time $O(|E|)$. That is, steps 1 (the number of edges of the Delaunay triangulation is linear in the number of vertices) and 3 (cfr. Lemma 6.1, property 1) of our algorithm take $O(|V|)$ time. It is easy to see that also step 4 takes no more than linear time. Consider step 2. Every vertex in MST has degree at most 5. That is, since G' contains at most 20 edges incident to a vertex, we can find every edge in constant time.

$\diamond$

7 Conclusion

In this paper, we have taken a step towards the definition and efficient computation of networks for points in the plane that tolerate the failure of a vertex and do so at low cost. We have used the rich structure of the underlying Euclidean space throughout. A number of more general settings of the problem therefore suggest themselves: how does the problem change for other interesting metrics, such as obstacles in the plane, or on a terrain? It seems also interesting to look into the problem with more than one failure. A straight extension of our approach makes things technically very complicated and does not promise to adapt well to higher numbers of failures. Also, other measures of quality than the total cost should be investigated. In particular, we are interested in keeping the diameter of the network low after failures of edges and vertices. Finally, it would of course be desirable to integrate fault-tolerance with many of the properties that have been considered into one spanning graph structure, similar in spirit to what Arya et al. [4] have accomplished for Euclidean spanners. We have, however, no clue as to how this can be achieved.

References

[1] Agarwal, Eppstein, and Matoušek. Dynamic half-space reporting, geometric optimization and minimum spanning trees. 33rd IEEE Symposium on Foundations of Computer Science (FOCS'92), Pittsburgh, Pe., 80–87, 1992.

[2] Aggarwal, Guibas, Saxe, and Shor. A linear time algorithm for computing the Voronoi diagram of a convex polygon, Discrete Comput. Geom., 4 (6), 1989, 39–45.

[3] Althöfer, Das, Dobkin. Joseph and Soares, On sparse spanners of weighted graphs, Discrete Comput. Geom. 9, 1993, 81–100.

[4] Arya, Das, Mount, Salowe and Smid. Euclidean spanners: short, thin, and lanky, STOC, 1995.

[5] Arya, Mount, and Smid. Randomized and deterministic algorithms for geometric spanners of small diameter, 35th Annu. IEEE Sympos. Found. Comp. Sci., 1994, 703–712.

[6] Arya and Smid. Effizient construction of a bounded degree spanner with low weight, 2nd Annu. European Sympos. Algorithms (ESA), Lecture Notes in Computer Science, 855, 1994, 48–59.

[7] Berge, Graphs and Hypergrahs. North-Holland, Amsterdam, 1973.

[8] Cheriton and Tarjan. Finding minimum spanning trees. SIAM J. Comput., Vol.5, No.4, December 1976, 724–742.

[9] Chin and Houck. Algorithms for updating minimal spanning trees. Journal of Computer and System Sciences, 16:333-344, 1978.

[10] Christofides and Whitlock. Network synthesis with connectivity constraints – a survey, Operational Research'81, North-Holland, Amsterdam, 1981, 705–723.

[11] Das and Heffernan. Constructing degree-3 spanners with other sparseness properties, Lecture Notes in Computer Science, 762, 992, 11–20.

[12] Das, Heffernan, and Narasimhan. Optimally Sparse Spanners in 3-dimensional Euclidean Space, 9th Annu. ACM Sympos. Comput. Geom., 1993, 53–62.

[13] Eppstein. Offline algorithms for dynamic minimum spanning tree problems. Journal of Algorithms, 17(2):237-250, 1994.

[14] Eppstein. Dynamic Euclidean minimum spanning trees and extrema of binary functions. Discrete and Computational Geometry, 13:237–250, 1995.

[15] Fredman and Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms, J. Assoc. Comput. Mach. 34, 1987, 596–615.

[16] Garey, Graham, and Johnson. Some NP-complete Geometric Problems, 8th Ann. ACM Symposium on Theory of Computing, 1976, 10–22.

[17] Hsu, Jan, Lee, Hung, and Chern. Finding the most vital edge with respect to minimum spanning tree in weighted graphs, Information Processing Letters, 39, 1991, 277–281.

[18] Iwano and Katoh. Efficient algorithms for finding the most vital edge of a minimum spanning tree, Information Processing Letters, 48, 1993, 211–213.

[19] Karp. Reducibility among combinatorial problems, Complexity of Computer Computations, 1972, 85–104.

[20] Kruskal. On the shortest spanning subtree of a graph and the travelling salesman problem, Proc. AMS 7, 1956, 48–50.

[21] Preparata and Shamos. Computational Geometry, Springer-Verlag, New York, 1985.

[22] Steiglitz, Weiner, and Kleitman. The design of minimum cost survivable networks, IEEE Transaction of Circuit Theory, 16, 1969, 455–460.

[23] Tarjan. Applications of path compression on balanced trees. Journal of the ACM, 26(4):690–715, October 1979.

[24] Tarjan and Leeuwen. Worst-Case analysis of set union algorithms. J. Assoc. Comput. Mach. 31, 1984, 245–281.

Appendix

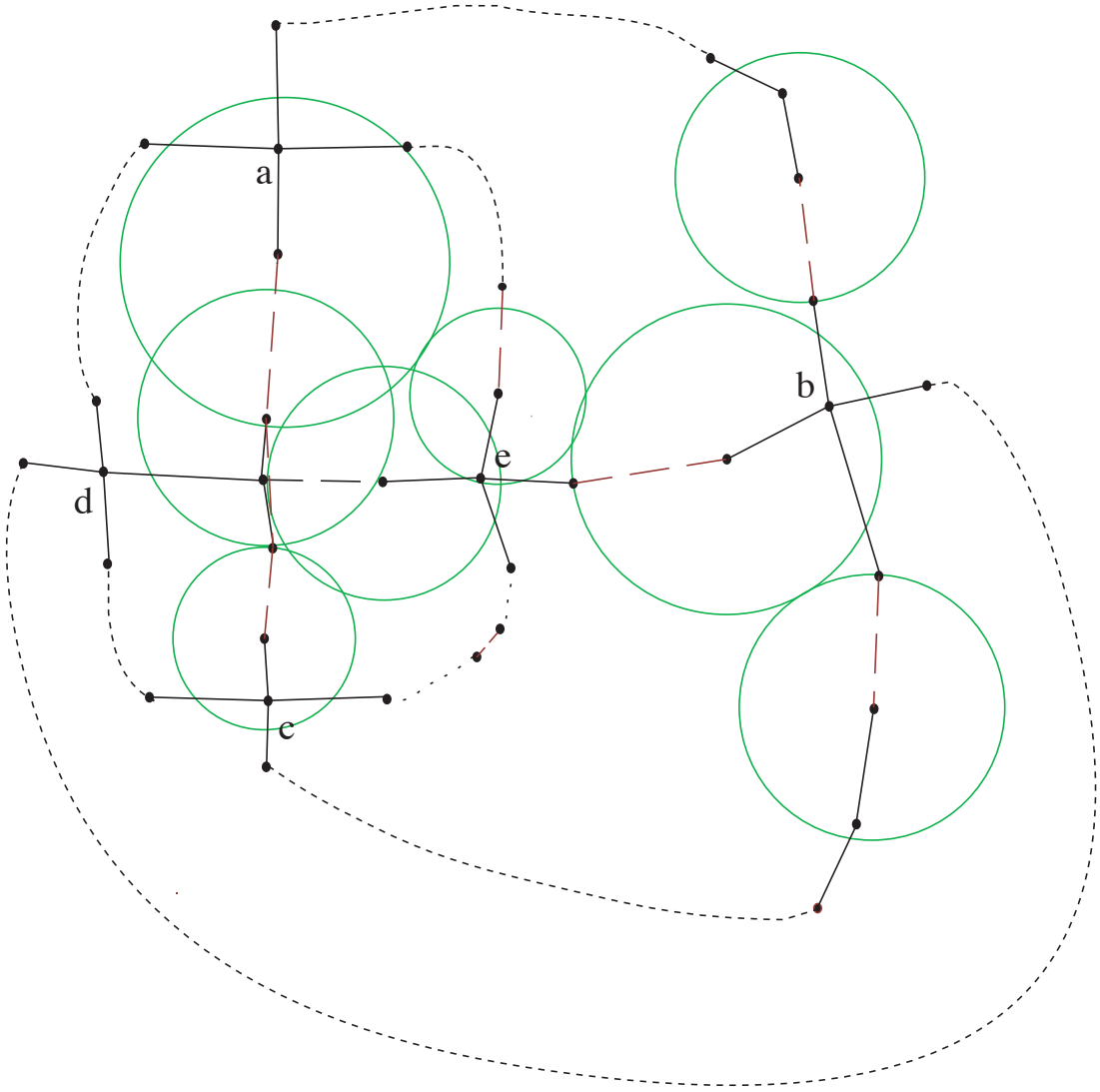

Figure 5: A subgraph sketch of a non planar EMT: the subgraph is isomorphic to K_5. The MST is shown with dotted lines and solid edges. The dashed edges are edges in EMT\MST: deleting a solid edge that is fully contained in one of the cyles forces the addition of a dashed one. It is important that in the cycles there are no other points than the ones shown.