How path-like is a tree?
compatible transformations from tree to path and vice versa

van Garderen, M.

Award date:
2013

Technische Universiteit
Eindhoven
University of Technology

Department of Mathematics and Computer Science
Den Dolech 2, 5612 AZ Eindhoven
P.O. Box 513, 5600 MB Eindhoven
The Netherlands
www.tue.nl

Supervisors
Prof. dr. Bettina Speckmann (TU/e)
Wouter Meulemans, MSc. (TU/e)

Section
Algorithms and Visualization

Date
October 14, 2013

How Path-like is a Tree?

Compatible transformations from tree to path and vice versa

Master's Thesis

Mereke van Garderen

Abstract

An important topic in information visualisation is set visualisation, displaying how individual elements are grouped together in sets. We focus on set visualisation in a geometric setting, where we want to show set relationships on elements with fixed geometric positions. Where classic set visualisation methods focus on the use of convex shapes to enclose the elements of a set, we want to use a graph to connect them. Existing methods can produce visualisations in which set membership is represented by a path or a tree on the elements. To find the best in-between visualisation, we search for a visually attractive, compatible, stepwise transformation from a tree to a path, such that the tree becomes more path-like in every step. We define a pathliness measure, to determine how path-like a given tree is, based on the diameter of the tree and the furthest distance from each vertex. We show that a single-step transformation is not always possible, since not every tree has a compatible path. Furthermore, we discuss the possibilities and limitations of an existing approach for tree-to-path transformations based on constrained Delaunay triangulations. We provide our own edge-flip based solution to transform a tree into path in at most $n - 3$ steps for n points in convex position. For points in general position, we show that edge flips do not always suffice to transform a tree into a path while improving the pathliness in every step. Whether or not a Euclidian minimum spanning tree can be transformed into a path by using only edge flips is still an open problem.

Acknowledgements

I would like to thank my advisor Bettina Speckmann and my tutor Wouter Meulemans for their guidance and support throughout the project, their valuable remarks and feedback on this thesis, and countless hours of fruitful discussion on the topic. In addition, I would like to thank Nikhil Bansal for reviewing this thesis and serving on the examination committee. Furthermore, I would like to thank the algorithms group at the university of Konstanz for their interesting questions and remarks after a presentation on the topic. I would also like to thank my office-mates Quirijn Bouts and Sander Alewijnse for supporting as well as distracting me whenever necessary, and Sander de Putter for some inspirational walks around the building.

Contents

Abstract **i**

Acknowledgements **iii**

Table of Contents **v**

1 Introduction **1**
1.1 Motivation . . . 1
1.2 Problem description . . . 2
1.3 Related work . . . 2
1.4 Thesis overview . . . 5

2 Pathliness Measure **7**
2.1 How path-like is a tree . . . 7
2.2 Unsuitable measures . . . 8
2.2.1 Degree . . . 8
2.2.2 Geometric length . . . 9
2.3 Furthest-distance vectors . . . 10
2.3.1 Definition . . . 10
2.3.2 Experimental results . . . 11

3 Compatible Path **15**
3.1 Tóth-tree . . . 15
3.2 Counterexample . . . 16

4 Constrained Delaunay Triangulations **19**
4.1 Method . . . 19
4.2 Lowerbound . . . 20

5 Edge Flips **23**
5.1 Convex position . . . 23
5.2 General position . . . 24
5.2.1 Visibility order . . . 24
5.2.2 Increasing the diametrical path . . . 25
5.2.3 Increasing pathliness . . . 29

6 Conclusion **31**

Chapter 1

Introduction

In this introductory chapter, we first explain our motivation for studying compatible transformations from tree to path and give a more detailed description of the problem. Furthermore, we review some of the literature in the areas related to this topic and give an overview of the contents of the remainder of this thesis.

1.1 Motivation

To get a good overview of a large data set, it is important to understand the relations between individual elements and how they can be grouped together in sets. Visualising these sets and the relations between them can be a useful aid in comprehending the structure of the data. A currently very popular area of research for which this is relevant is social network science. Social scientists can group people in a social network according to the communities they belong to, such as schools and sports clubs, and study the interactions between these groups. Set visualisation can also be useful in linguistics, to identify related words, or in archaelogy, to study relations between different sites, or even for regular citizens who quickly want to see which restaurants in their town serve Italian food rather than Indian.

Because of its many and diverse applications, set visualisation is an important topic in information visualisation. For most people, Venn diagrams and the more general Euler diagrams are the most familiar set representations. In an Euler diagram, each set is enclosed by a circle, and circles overlap when an element belongs to multiple sets. With this simple and intuitive approach, one can see in an instant the differences and commonalities between the represented sets. However, this works well only with a small number of elements. When applied to larger data sets, Venn diagrams and Euler diagrams can get very cluttered and hard to interpret. If the elements have fixed positions, for example in geographic data, it might not even be possible to draw a traditional Euler diagram. If simple circles or ellipses no longer suffice, it is a natural next step to use other shapes to enclose or connect the sets. This is where trees and paths come into play: we can connect the elements of each set using a tree structure. Since we can define many different spanning trees on the same set of elements, the challenge then becomes to find the nicest one. Would it be best to connect the elements with a single polyline, a path? Or would a more complex branching structure, a tree, give a nicer result? To answer this question, it would be helpful if we could transform one representation into the other within a reasonable number of steps, and take a look at the intermediate steps. Finding such a transformation is the problem studied in this thesis.

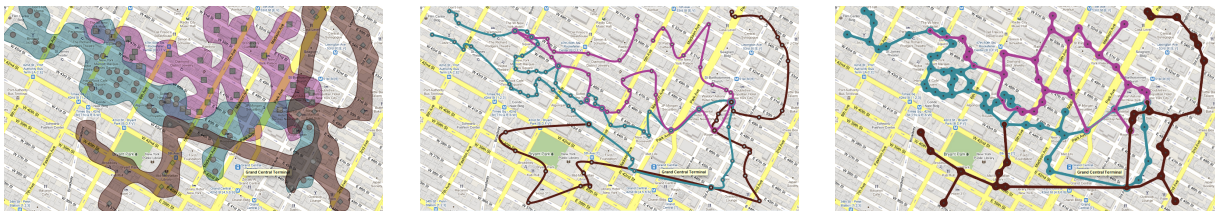
1.2 Problem description

There are basically two different flavours of the set visualisation problem one could study. In the combinatorial setting, elements have no predefined coordinates and can be moved around freely to generate the best visualisation. In linguistics, for example, this is usually a reasonable way of working. In the geometric setting, all elements have a fixed location and cannot be moved. This is usually the case when we work with geographic data, like in the example of restaurants on a map. We focus on the latter variant and study the problem in the geometric setting. Because our research question is focused on the representation of individual sets, we study only one set at a time. Our input thus consists of a set of points in the plane and a plane spanning tree on this point set. We choose to work only with plane spanning trees because they generally result in nicer visualisations than trees containing edge crossings. Our aim is now to transform such a tree into a path through a sequence of small changes, like replacing one edge by another. We set the additional constraints that the tree remains plane and that after every step, the tree should become more path-like, until finally it is indeed a path. Furthermore, we want the trees from two subsequent steps to be compatible. This means that the two trees can have edges in common, but their edges cannot intersect. Note that finding a sequence from a particular tree to a particular path is equivalent to finding a sequence from that same path to that same tree (it is the same sequence in reverse), so we can study the problem in either direction.

1.3 Related work

Set visualisation. With the rise of computerized visualisation, many new set visualisation techniques have been developed in recent years. Perhaps the most common solution when Euler diagrams no longer suffice is to use more complex shapes than ellipses to enclose the sets. Bubble Sets, shown in Figure 1.1a, is a visualisation method using continuous isocontour lines to represent set membership [9].

This can always be done, but when many sets overlap the picture can still become quite unclear. To reduce clutter, some approaches using a minimal geometry have been proposed. The LineSets method uses only a single continuous line to connect the elements of each set [5]. Kelp Diagrams use a very sparse spanning graph, basically a minimum spanning tree with some additional edges [13]. LineSets and Kelp Diagrams are shown in Figure 1.1b and 1.1c, respectively.

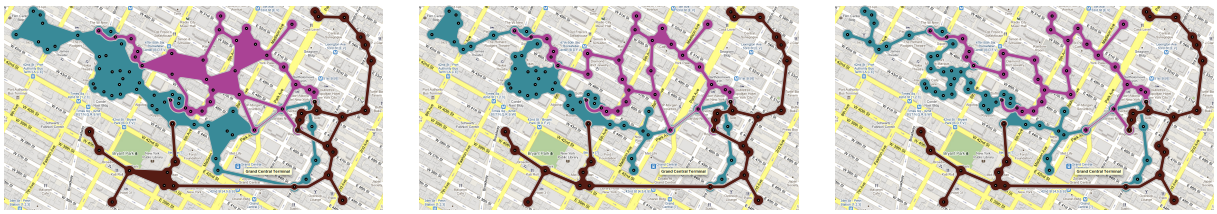


(a) Bubble Sets **(b)** LineSets **(c)** Kelp Diagrams

Figure 1.1: *Examples of some existing visualisation methods. Pictures taken from [21].*

Both LineSets and Kelp Diagrams introduce less clutter than for example Bubble Sets, but if there are some elements very close together it still becomes hard to see what is going on. The hybrid method KelpFusion [21] is based on Kelp Diagrams, but allows the user to adjust the visualisation. On one end of the KelpFusion spectrum, we can get a very dense visualisation that mostly resembles a hull-based approach, where sets are enclosed by some geometric shape. On the other end of the spectrum, KelpFusion can also generate a tree-based visualisation very

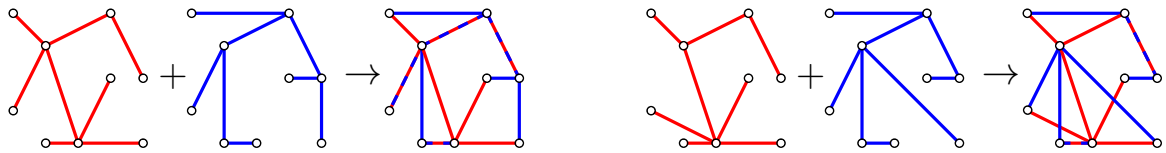
similar to Kelp Diagrams. Figure 1.2 shows an example of three KelpFusion visualisations of the same data with different settings. As can be seen from the figure, a visualisation that is somewhere in between hull-based and tree-based can also give a nice result. There is not too much clutter, but in the areas where many points are close together it is still clear to which sets they belong. If we find a way to transform a tree into path, this could help in developing a similar method to find nice visualisations in between tree-based and path-based approaches, which to the best of our knowledge does not yet exist.



(a) KelpFusion (dense) **(b)** KelpFusion (medium) **(c)** KelpFusion (sparse)

Figure 1.2: *Different settings of the KelpFusion method. Pictures taken from [21].*

Compatibility. Both general compatibility questions and sequences of pairwise compatible graphs have received some previous attention. Two graphs on the same set of points are *compatible* if their simultaneous embedding is crossing-free. Compatible graphs can have edges in common, but their edges cannot intersect. Figure 1.3a shows an example of two compatible trees and their crossing-free simultaneous embedding. Figure 1.3b shows an example of two trees that are not compatible, since their simultaneous embedding contains two crossings.



(a) Compatible **(b)** Incompatible

Figure 1.3: *Two trees that are compatible* ***(a)*** *and two trees that are not compatible* ***(b)****.*

Aronov, Seidel and Souvaine [6] showed that given two simple n-sided polygons, it is not always possible to triangulate these in such a way that the two triangulations are compatible. In the same paper, they proved that if the introduction of $O(n)$ additional points (Steiner points) is allowed, it is always possible to find compatible triangulations. Twenty years later, Aichholzer et al. [2] showed that any two sets of n points that agree on the number of convex hull points and have at most three interior points can be triangulated such that the two triangulations are compatible for some labeling of the points. They also proved that if adding a small number of Steiner points is allowed, a compatible triangulation is possible regardless of the number of interior points.

A *pseudo-triangle* is a simple polygon with exactly three interior angles smaller than π, as shown in Figure 1.4. *Pseudo-triangulations*, as the name suggests, are related to triangulations and use pseudo-triangles in addition to triangles. A pseudo-triangulation is *pointed* if at each vertex, the incident edges span an angle of less than π. Aichholzer, Hackl and Vogtenhuber [16] showed that for any point set S in general position, there exist two compatible pointed pseudo-triangulations whose union

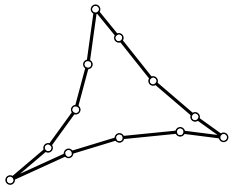


Figure 1.4: *Pseudo-triangle*

is a triangulation of S. However, they also showed that there exist point sets S and pseudo-triangulations T such that there is no pseudo-triangulation of S compatible with but different from T. Aichholzer et al. [3] showed that for any two perfect matchings M and M', there is a sequence $M, \ldots, M'$ of $O(\log n)$ perfect matchings, such that each matching is compatible with the next one in the sequence. Aichholzer, Aurenhammer and Hurtado [1] proved a similar result for spanning trees: for any plane spanning tree T on a set S of n points, there exists a sequence of $O(\log n)$ plane spanning trees starting with T and ending with a minimum spanning tree of S, such that each tree in the sequence is compatible with the next. They show that the number of length-improving, planar edge moves (consisting of taking one edge out and inserting another) necessary to transform a tree into a minimum spanning tree is $O(n \log n)$. Aichholzer and Reinhardt [4] later showed that when only the more restrictive edge slide operation (sliding one endpoint of an edge to a neighbouring vertex) is allowed, two spanning trees can be transformed into each other in $O(n^2)$ steps. The sequence of spanning trees ending with a minimum spanning tree as described in [1] can be found using constrained Delaunay triangulations, so we refer to this approach as the *CDT-method*. A *Delaunay triangulation* is a triangulation in which the circumcircle of each triangle has an empty interior. In a *constrained Delaunay triangulation*, some edges are forced into the triangulation (these are the constraints), but otherwise it is as close as possible to the Delaunay triangulation [8]. The CDT-method can also be used to find a sequence from a path to a minimum spanning tree, and is therefore a possible solution to our problem. We further study this approach and its results in Chapter 4. However, since the CDT-method requires us to take either very large or very many steps during the transformation and does not always make the tree more path-like in every step, we also try to find a faster solution with smaller or nicer steps.

Flip graphs. The smallest transformation we can apply to a graph is to take one edge out, and put another one in. With the additional constraint that the graph remains in the same graph class, this move is called an edge flip, and quite some research has been done on what can be achieved with edge flips. A nice overview of known results is given in the literature survey by Bose and Hurtado [7]. For two given graphs on the same set of vertices with the same number of edges, the minimum number of edge flips needed to transform one into the other can be used as a measure of similarity, which can be varied by constraining the allowable flips [15]. Given a class of graphs $\mathcal{G}$ and a set of flip operations, we can define a flip graph $\mathcal{F}(\mathcal{G})$. Each graph from $\mathcal{G}$ corresponds to a vertex in the flip graph $\mathcal{F}(\mathcal{G})$, and two vertices of $\mathcal{F}(\mathcal{G})$ are connected by an edge if the difference between the graphs they represent is exactly one edge flip. This implies that the flip distance between two graphs is exactly their graph distance in the flip graph. Like for the set visualisation problem, we study the geometrical setting here, in which the graphs we study are straight-line planar embeddings (rather than just combinatorial embeddings). Flip graphs and flip distances have mostly been studied for triangulations, because these are maximal planar graphs.

A *near-triangulation* is a triangulation in which one particular face, called the outer face, is not a triangle. Lawson [20] proved that for any two near-triangulations on the same n points in the plane there exists a sequence of $O(n^2)$ edge flips that transforms one into the other, by showing that any near-triangulation can be transformed into a canonical triangulation in $O(n^2)$ flips. Hurtado, Noy and Urrutia [18] later proved that this quadratic upperbound is tight. The drawback of using a canonical form in the transformation is that the number of flips used in the transformation is not necessarily related to the minimum number of flips required. Hanke, Ottmann and Schuierer [17] attempted to solve this problem for triangulations. They proved that the number of edge flips sufficient to transform one triangulation into another is bounded by the number of intersection between the edges of the two triangulations. They also proved that given two triangulations on the same point set, there always exists an edge flip in one of the

two triangulations that reduces the total number of intersections by at least one, and presented an algorithm based on this fact. We would like to find a similar algorithm to transform spanning trees rather than triangulations.

Delaunay triangulations. As mentioned earlier, the CDT-method relies heavily on (constrained) Delaunay triangulations. Some of our own attempts to transform a tree into a path focus on transforming the *Euclidian minimum spanning tree (EMST)*, the spanning tree on a point set for which the total edge length is minimized. Since the EMST is a subgraph of the Delaunay triangulation, we also studied some related work on the properties of Delaunay triangulations. A *Hamiltonian cycle* is subset of the edges of a graph forming a cycle that visits all vertices exactly once. Dillencourt [10] proved that Delaunay triangulations do not necessarily contain a Hamiltonian cycle. In another paper [12], he showed that finding a Hamiltonian cycle in a Delaunay triangulation is NP-complete. To the best of our knowledge, it is not known whether these statements also hold for Hamiltonian paths. Dillencourt [11] also proved that even if a Delaunay triangulation of a point set S is Hamiltonian, the *travelling salesman tour* of S, which is the shortest Hamiltonian cycle of the complete graph on the point set, is not necessarily a subgraph of the Delaunay triangulation. Floriani et al. [14] proved that we can define a partial order on the triangles of a Delaunay triangulation with respect to a vertex v that is consistent with the visibility of the triangles from v. This means that if we shoot a ray from vertex v in any direction, the order in which it intersects triangles is consistent with the order we defined. This is a very useful property, since it excludes the possibility of cyclic dependencies among the edges and allows for incremental processing of the triangles.

1.4 Thesis overview

To find a transformation in which a tree becomes more path-like in every step, we need to be able to measure how path-like a given tree is. In Chapter 2, we start by analyzing what properties we would like such a measure to have. Then, we discuss a number of existing measures and explain why they are unsuitable for our purposes. Finally, we introduce our own measure, which we use in the remainder of this thesis. If we want to minimize the number of steps in our transformation, the best approach would be to go from a tree to a compatible path in single step. In Chapter 3, we show that this is not always possible, because there exists a tree without a compatible path. Chapter 4 focuses on the CDT-method by Aichholzer et al. We explain how the method works, show what kind of transformation sequences it produces, and give a lowerbound example for which the transformation takes $O(\log n)$ steps. In Chapter 5, we discuss our attempts to develop a new method to transform a tree into a path. Our approach focuses on the use of edge flips. We present a simple solution for the case where the points are in convex position. For points in general position, we discuss some moves that could be used to make a tree more path-like and show that they are not always sufficient. Finally in Chapter 6, we summarize our results and list some open problems.

Chapter 2

Pathliness Measure

In this chapter, we consider a number of ways to measure progress in our transformation from tree to path. This raises one of the main questions in this thesis; how path-like is a tree? We elaborate on this question in Section 2.1. Afterwards, we discuss a number of possible criteria for measuring how path-like a tree is and their drawbacks in Section 2.2. In Section 2.3, we introduce a new measure that we developed, which is also the one we use in the remainder of this thesis.

2.1 How path-like is a tree

Since our objective is to transform a tree into a path in such a way that we make progress in every step, we need to start by defining what we consider to be progress. Our goal is to obtain a path, so we make progress if our tree becomes more path-like. However, it is not always obvious when this is the case. Naturally, the most path-like tree is an actual path, but when we compare two trees that are not paths it is not always clear which one is more path-like. Consider for example the spanning trees $\mathcal{T}_1$, $\mathcal{T}_2$, $\mathcal{T}_3$ and $\mathcal{T}_4$ as shown in Figure 2.1. These are

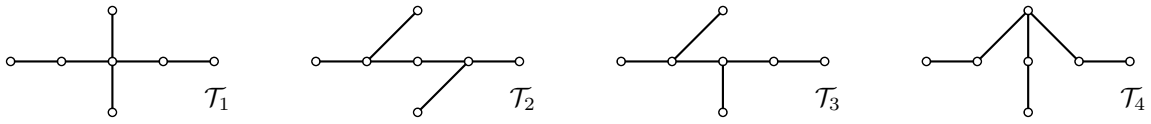


Figure 2.1: *Four spanning trees on the same set of points. It is not obvious which one is most path-like.*

all spanning trees on the same set of points, and the differences between them are small. Here, it is not trivial to decide which ones are more, less, or equally path-like. We could for example claim that $\mathcal{T}_1$ is more path-like than $\mathcal{T}_2$ because it has fewer branching vertices, but we could also claim that $\mathcal{T}_2$ is more path-like than $\mathcal{T}_1$ because its maximum degree is lower. We could also argue that they are equally path-like, for example because they have the same number of vertices not on the longest path. With similar arguments, we could say that $\mathcal{T}_3$ is more path-like than $\mathcal{T}_4$ because it has shorter branches from the longest path, or we could say that it $\mathcal{T}_3$ is less path-like than $\mathcal{T}_4$ because it has more branches. Since there are many properties we could take into account, we can make up multiple arguments like this for any pair of these trees. This is the reason why we need a well-defined measure to determine how path-like a tree is. We call this property the *pathliness* of a tree. We want to find a pathliness measure that adheres to the following criteria:

- The measure induces a partial order on all possible spanning trees on a set of points.

- Any path is a maximal element in the partial order, and any non-path is not a maximal element in the partial order.
- The partial order intuitively makes sense upon visual inspection.
- The measure can be improved with edge flips.

2.2 Unsuitable measures

There are a number of properties of trees and paths that we could use to measure pathliness. We want to find one that gives visually intuitive results, and that we can actually improve with edge flips. We first discuss some of the obvious choices, and explain for each of them why they are not suitable for our objective.

2.2.1 Degree

A path has the property that every vertex has degree 2, except the two endpoins, which have degree 1. In a tree, there can be vertices with degrees higher than 2, and a tree can contain more than two vertices of degree 1. We could therefore base our pathliness measure on degrees. One option is to look at the maximum degree in a tree. This is degree 2 for a path, at least degree 3 for a tree that is not a path, and we could claim that the higher the maximum degree, the less path-like a tree is. The main problem with this approach is that it leads to some counterintuitive results. Figure 2.2 shows two spanning trees on the same set of points. Intuitively, $\mathcal{T}_1$ is much more path-like than $\mathcal{T}_2$, but its maximum degree is actually higher. Since the number of edges

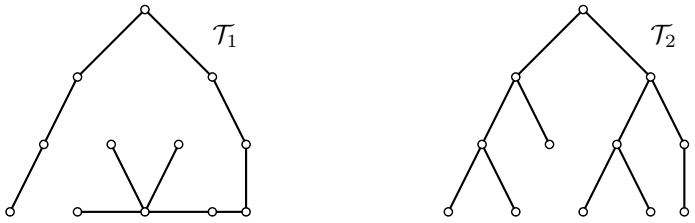


Figure 2.2: *Two spanning trees on the same set of points. Although $\mathcal{T}_1$ has the highest maximum degree, it looks more path-like than $\mathcal{T}_2$.*

is the same for every spanning tree on n points, the sum of degrees or average degree cannot be used as a measure in this context. What we can do is look at the number of vertices of degree 1. This is two for a path, at least three for a tree that is not a path, and we could claim that the more vertices of degree 1 a tree has, the less path-like it is. Another property we could take into account is the number of vertices of degree at least 3. There are none in a path, there is at least one in a tree that is not a path, and we could argue that the more of these vertices a tree contains, the less path-like it is. Combining these two ideas, we could take the number of vertices of degree 2 as a measure for pathliness. For a set of n points, any graph with $n-2$ vertices of degree 2 is a path, a graph with less than $n-2$ vertices of degree 2 is a tree. This measure would work for the example in Figure 2.2: the more path-like tree $\mathcal{T}_1$ has seven vertices of degree 2, while the less path-like tree $\mathcal{T}_2$ has only two. The main problem with this approach is that it is relatively easy to come up with an example where it is not possible to increase the number of degree 2 vertices with a single edge flip.

Lemma 1 *There exists a plane tree for which the number of vertices of degree 2 cannot be increased with a single edge flip.*

Proof. Figure 2.3 shows a set of points and a minimum spanning tree on these points. The graph contains three vertices of degree 1 and one vertex of degree 3. To improve the pathliness of this graph by increasing the number of vertices of degree 2, we first need to remove one of the edges adjacent to the vertex of degree 3, creating a disconnected graph with four vertices of degree 1. Assume without loss of generality that we remove the edge marked in red in the figure. Then, we need to reconnect the graph by inserting an edge between two vertices of degree 1, which are marked in red in the figure. However, none of the vertices of degree 1 can see one of the others, so this is not possible. This implies that for this tree, we cannot increase the number of vertices of degree 2 with a single edge flip. □

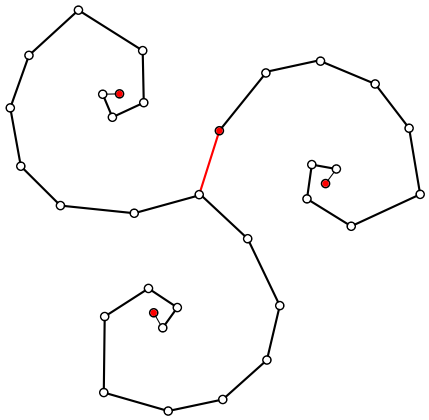


Figure 2.3: *Spanning tree for which the number of vertices of degree 2 cannot be increased with a single edge flip.*

Note that it would not help to consider either the number of vertices of degree 1 or of degree 3 separately, because the same argument holds then. Since our aim is to find a measure of progress that we can actually improve during each step of a transformation, the implications of Lemma 1 make a measure based on degrees an unsuitable solution for our problem.

2.2.2 Geometric length

As explained in Chapter 1, finding a transformation from a tree to a path is equivalent to finding a transformation from a path to a tree. If we assume for the moment that the tree we start with is a Euclidian minimum spanning tree (EMST), we could use the number of EMST edges in the tree as a pathliness measure. Starting from a path, we want to increase the number of minimum spanning tree edges in every step of the transformation, until we arrive at the actual minimum spanning tree. To achieve this, each step in the transformation has to consist of removing a non-EMST edge from the tree, and inserting the EMST edge reconnecting the components. Unfortunately, this is not always possible without violating the planarity of the tree.

Lemma 2 *There exists a plane tree for which it is not possible to remove a non-EMST edge and insert an EMST edge reconnecting the components without introducing a crossing.*

Proof. Consider the example shown in Figure 2.4. The dotted lines represent a densly sampled path, the solid lines are single edges. The solid edges are the only non-EMST edges in the graph. If we remove either of these, the EMST edge reconnecting the components would be the edge crossing the other solid edge. This implies it is not always possible to remove a non-EMST edge and insert the EMST edge reconnecting the components without violating the planarity of the tree. □

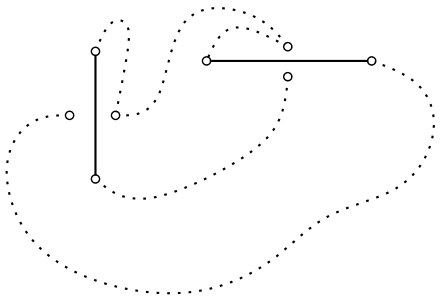


Figure 2.4: *Tree for which it is not possible to remove a non-EMST edge and insert an EMST edge reconnecting the components without introducing a crossing.*

To make the set of allowed transformation steps less restricted, we could consider the total edge length as a measure for pathliness, rather than the number of EMST edges. Instead of always inserting the shortest edge reconnecting the components (which is an EMST-edge by definition), we can then reconnect the components with any edge that is shorter than the one we just removed. This way the total length still decreases in every step, but we have some more freedom in choosing the edges to insert and avoiding crossings. We did not consider this in much detail, because we fear that this approach would lead to a transformation with very many steps.

2.3 Furthest-distance vectors

Since none of the basic options provided us with the desired results, we decided to design a somewhat more sophisticated measure.

2.3.1 Definition

One of the properties of trees and paths we did not consider yet is diameter. The *diameter* of a tree is the length of the longest path it contains. Since we are interested mostly in the structure of the tree, we define the diameter as the number of edges in the longest path, rather than its geometrical length. The maximum diameter possible for a tree on n points is $n-1$, the length of a path on all points. A tree that is not a path always has a diameter strictly less than $n-1$. A longest path in a tree is called a *diametrical path*, and a tree can contain multiple diametrical paths.

A major drawback of most of the approaches described in Section 2.2 is that they are not very sensitive to small changes in the tree. We can perform many edge flips on a tree that do not influence these measures at all. This means that even though they might actually be steps in the right direction, these moves would not be allowed because they are not considered progress by the measure. If we just use the diameter as pathliness measure, we run into the exact same problem. In some cases it would work, such as in the example from Figure 2.2 we considered earlier. The most path-like looking tree $\mathcal{T}_1$ has a diameter of nine, whereas the diameter of $\mathcal{T}_2$ is only six. However, if we look back at the example in Figure 2.1 at the beginning of this chapter, all four spanning trees have the same diameter of four. A measure based only on diameter does not help us to determine which of these is more path-like. We therefore decided to take into account not just the diameter of the tree, but the length of the longest path starting from each of the vertices in the tree. For the endpoints of a diametrical path this equals the diameter of the tree, for the other vertices this number is smaller. We compute the length of the maximum

path from each vertex in the tree, and sort these numbers from large to small. The result is a vector of n numbers for a tree on n points, which we call the *furthest-distance vector*, and we use this vector as pathliness measure. Figure 2.5 shows an example of four spanning trees and their measures.

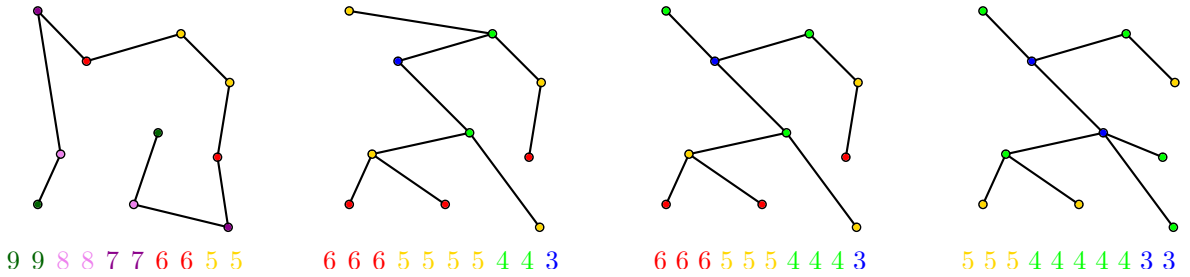


Figure 2.5: *Four spanning trees on the same set of points and their furthest-distance vectors.*

When comparing multiple spanning trees, the lexicographical order on their furthest-distance vectors is the partial order on the trees, and the tree with the highest values in the furthest-distance vector is the most path-like. For the spanning trees in Figure 2.5 this means that they are currently ordered from the most path-like on the left (this is an actual path) to the least path-like on the right. We can now use this measure to sort the spanning trees from the beginning of this chapter. Figure 2.6 shows the same set of trees, this time together with their furthest-distance vectors. We can now see that $\mathcal{T}_2$ is most path-like, $\mathcal{T}_1$ is least path-like, and $\mathcal{T}_3$ and $\mathcal{T}_4$ are equally path-like.

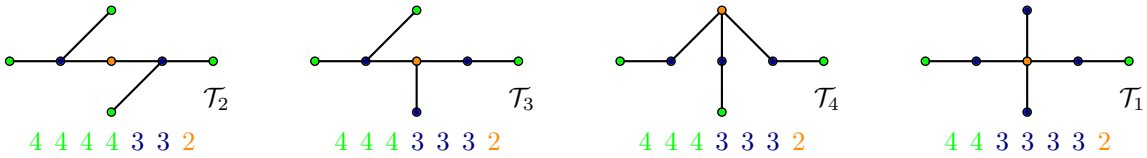


Figure 2.6: *The spanning trees from Figure 2.1, sorted by their furthest-distance vectors.*

2.3.2 Experimental results

Visual intuition. As we stated earlier, one of the properties we want our pathliness measure to have is that it orders trees in a way that intuitively makes sense upon visual inspection. To test whether this holds for furthest-distance vectors, we wrote a program to order either a given or randomly generated set of trees according to this measure. Visual inspection of the output convinced us that the furthest-distance vectors impose an intuitive ordering on the trees. Figure 2.7 shows a set of 24 spanning trees on the same set of points, sorted according to our pathliness measure. The grey boxes indicate sets of trees that are equally path-like, based on their furthest-distance vectors. Because our pathliness measure is based only on combinatorial furthest distances, two trees that are different geometrical embeddings of the same combinatorial structure always have the same furthest-distance vector, and are considered equally path-like. This applies for example to the two paths in the first row of the figure. It is also possible that two trees that are combinatorially different still have the same pathliness measure. This is true for example for the two equally path-like trees in the third row of the figure.

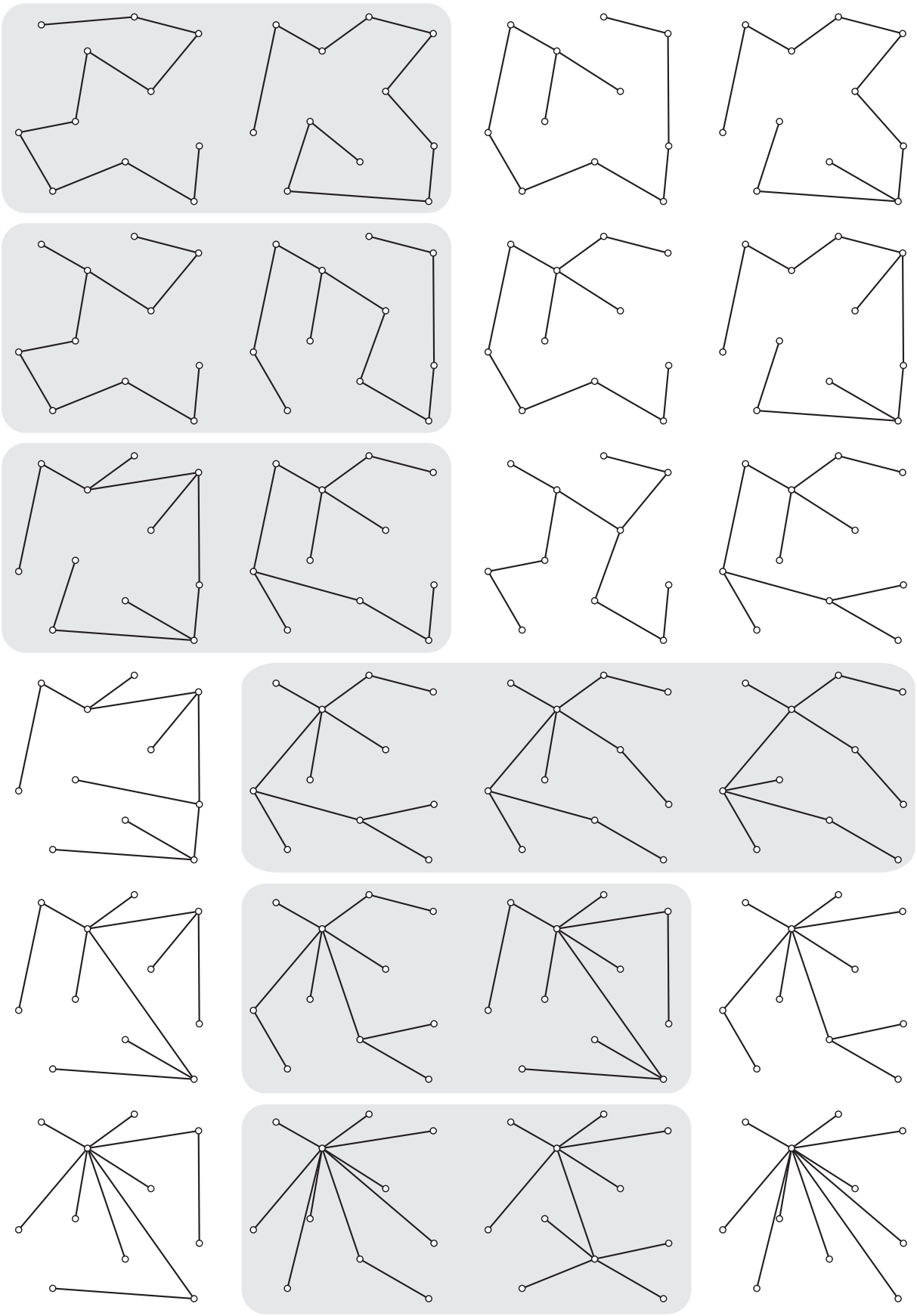


Figure 2.7: *Trees sorted by our pathliness measure, boxes indicate equally path-like trees.*

Basic path-to-EMST transformation. The most basic way to transform a path into a Euclidian minimum spanning tree (EMST) is to repeatedly take out the longest non-EMST edge and insert the EMST edge reconnecting the components. We have shown in Section 2.2 that this cannot always be done without introducing a crossing, and therefore it is not a suitable solution to our problem. However, we still wanted to see how our pathliness measure would change during such a transformation. We implemented a method to apply this basic transformation on any given tree. To avoid situations in which no moves are possible, we allow the introduction of crossings. We performed the experiment on one hundred random paths and one hundred monotone paths. Figure 2.8 shows how the pathliness changed during the transformations. We mapped the furthest-distance vectors to values by treating them as n-nary

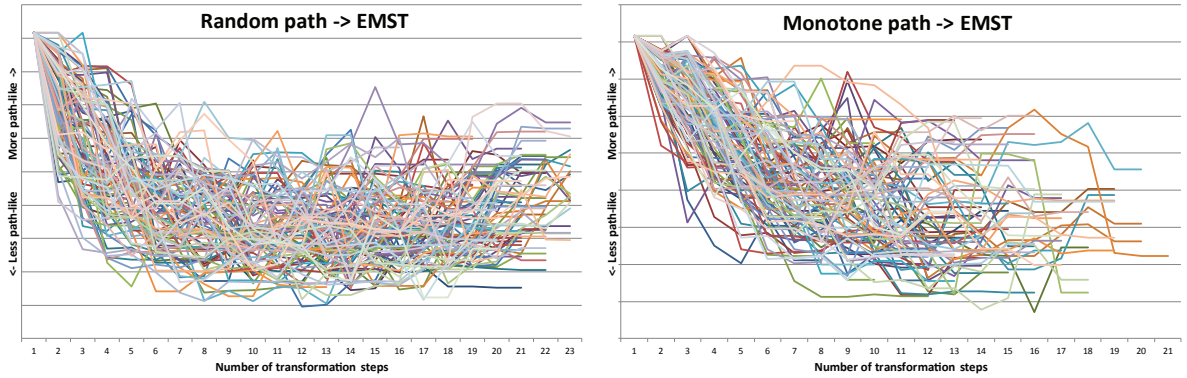


Figure 2.8: *Changes in pathliness during the transformation from 100 random paths (left) and 100 monotone paths (right) to a Euclidian minimum spanning tree (EMST).*

numbers to generate this plot. The figure shows that the pathliness generally decreases in the first few steps of the transformation, but after that it changes in all possible directions. A closer look at just a few of the transformations, shown in Figure 2.9, reveals the same pattern. We aim to find a transformation for which a plot of the pathliness would be monotone.

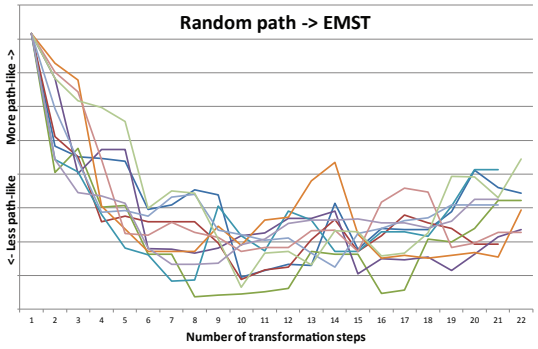


Figure 2.9: *Changes in pathliness during the transformation from 10 random paths to a Euclidian minimum spanning tree (EMST).*

Chapter 3

Compatible Path

The fastest way, with respect to the number of steps, to transform a tree into a path is to find a path compatible with the original tree in a single step. In this chapter, we prove that this is not always possible, since not every tree has a compatible path.

3.1 Tóth-tree

The tree shown in Figure 3.1, which we call the *Tóth-tree*, was suggested to us by Csaba Tóth in private communication as a component that could possibly be used to construct a tree without a compatible path. The tree was published in a different context in a paper by Hurtado and Tóth [19]. An important property of this tree is that only six of its vertices can be seen from the outside of the convex hull. The *convex hull* of a tree is the smallest convex polygon enclosing all vertices. We call the vertices of the Tóth-tree that can be seen from outside the convex hull the *outer vertices*, and the remaining vertices the *inner vertices*. In Figure 3.1, the outer vertices are marked in red. The remaining vertices are hidden from the outside by edges of the tree. A single Tóth-tree does have a number of compatible paths, some examples of which are shown in Figure 3.2. We observe that none of the compatible paths shown in the figure have outer vertices for both their endpoints. It turns out that this is not a coincidence: such a compatible path does not exist.

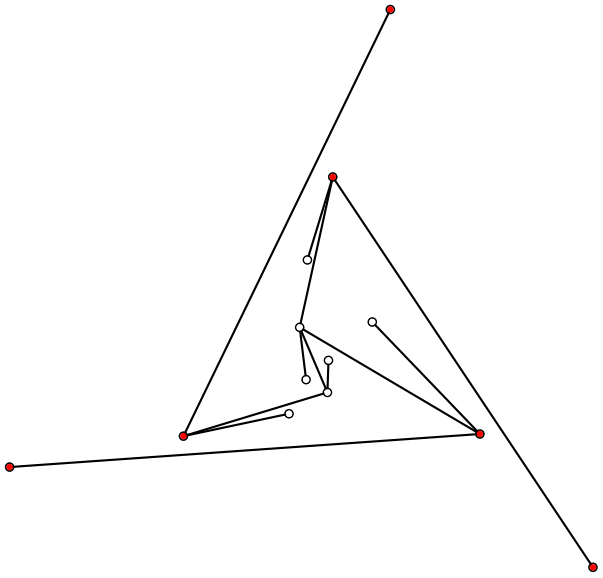


Figure 3.1: *Tóth-tree, the vertices visible from the outside are marked in red*

Lemma 3 *The Tóth-tree does not have a compatible path for which both endpoints are outer vertices.*

Proof. We proved this lemma with computer assistance. We wrote a program that exhaustively tried all Hamiltonian paths starting and ending at outer vertices and checked whether they were crossing free and compatible with the original tree. No such path could be found. □

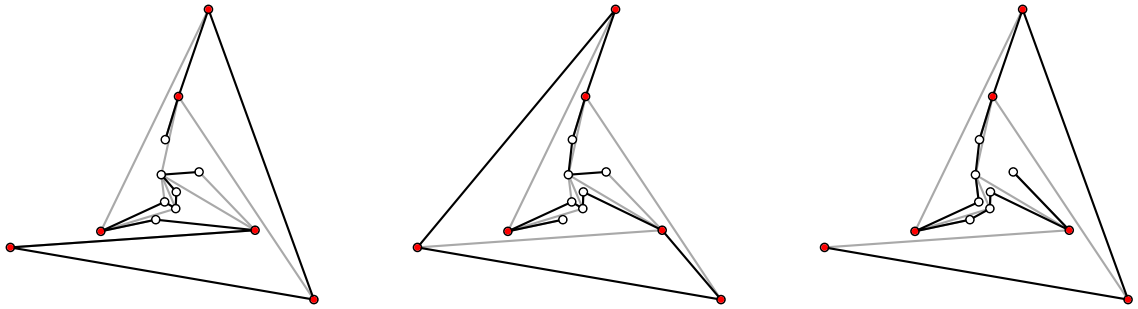


Figure 3.2: *Examples of paths compatible with the Tóth-tree*

3.2 Counterexample

Inspired by Lemma 3, we constructed a tree that does not have a compatible path.

Lemma 4 *There exists a tree that does not have a compatible path.*

Proof. Consider the tree shown in Figure 3.3, in which the blue components are Tóth-trees. Each of the Tóth-trees can be accessed from only two vertices of the larger tree. If we want to construct a compatible path for this tree, there are two options for each Tóth-tree. Either it has to be traversed entirely, with a path starting and ending at outer vertices, or one of its inner vertices is one of the endpoints of the compatible path. A path has only two endpoints, and we have three Tóth-trees here, which implies that at least one of the Tóth-trees has to be traversed entirely, from one outer vertex to another. However, we know by Lemma 3 that there exists no such path that is compatible with the Tóth-tree. Therefore, the tree in Figure 3.3 does not have a compatible path. □

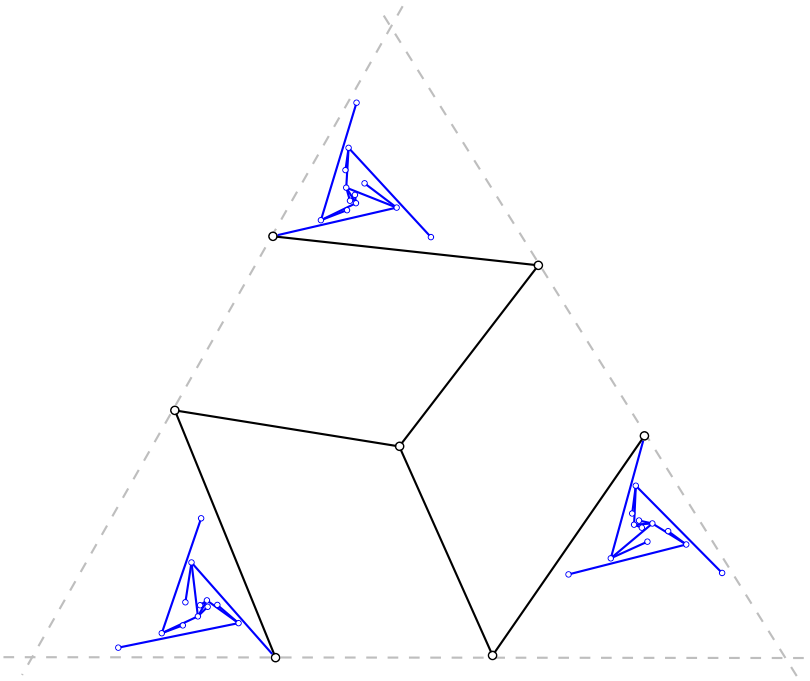


Figure 3.3: *Tree without a compatible path.*

The counterexample from Figure 3.3 is not a minimum spanning tree. Some of its properties, such as the very sharp angles, cannot even occur in a minimum spanning tree. To the best of our knowlegde, the question whether or not a minimum spanning tree always has a compatible path is still an open problem. The fact that there are no known examples of minimum spanning trees without a compatible path is one of the reasons why we mainly focus on transforming a path to a minimum spanning tree, rather than just any tree, in the remainder of this thesis.

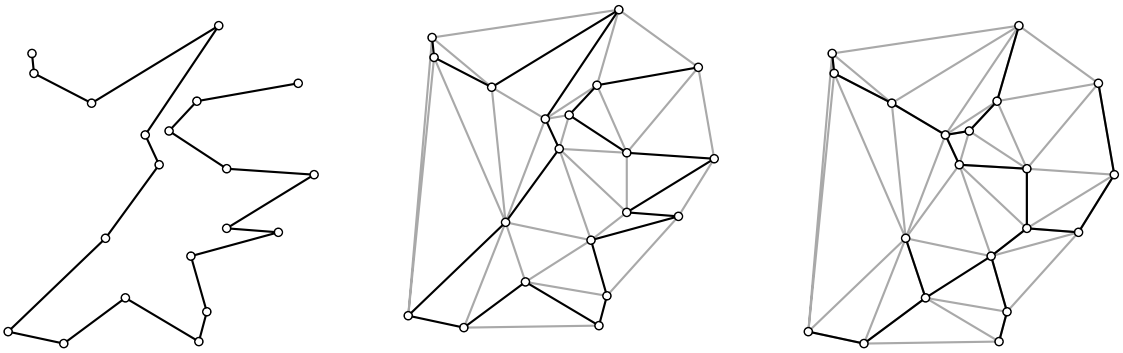
Chapter 4

Constrained Delaunay Triangulations

In this chapter we focus on the CDT-method by Aichholzer et al. [1], which uses constrained Delaunay triangulations to transform any tree into a minimum spanning tree. We first explain in some more detail how the method works and show some examples of the sequences it generates. Furthermore, we present an example which proves that the lowerbound for the number of steps described by Aicholzer et al. also holds if we start with a path.

4.1 Method

Since we are interested in transformations from tree to path, we want to use the CDT-method to transform a path (rather than just any tree) into a Euclidian minimum spanning tree (EMST). Reversing the sequence then gives us the transformation we were looking for. Each move in the CDT-transformation consists of two steps. First, we compute the constrained Delaunay triangulation on our current tree. Note that in the first step, this tree is a path. Then, we compute the minimum spanning tree on this constrained Delaunay triangulation, and that is our new tree. This process is illustrated in Figure 4.1. Figure 4.1a shows a set of n points and the path we start with. Figure 4.1b shows the constrained Delaunay triangulation on this path, and the minimum spanning tree of this triangulation is shown in Figure 4.1c.



(a) *Path*

(b) *Constrained Delaunay triangulation*

(c) *Minimum spanning tree*

Figure 4.1: *One step in the CDT-transformation*

In this example, the new tree that we created is already the minimum spanning tree of the point set, so the transformation is complete after a single step. When this is not the case, we repeat the same process with the new tree as our starting point. Figure 4.2 shows an example of a

transformation that takes three steps to reach the minimum spanning tree. As can be seen in the Figure, the first step is a very large one, in which a lot of edges change. In the remaining steps, only one edge changes at a time. This is a pattern that we see in most of the transformations based on this method: the first step introduces a big change, the remaining steps are either non-existent or only introduce minor changes. A constrained Delaunay triangulation is by definition planar. Since we insert only edges that are in the constrained Delaunay triangulation of the previous tree, two subsequent trees in this transformation are always compatible.

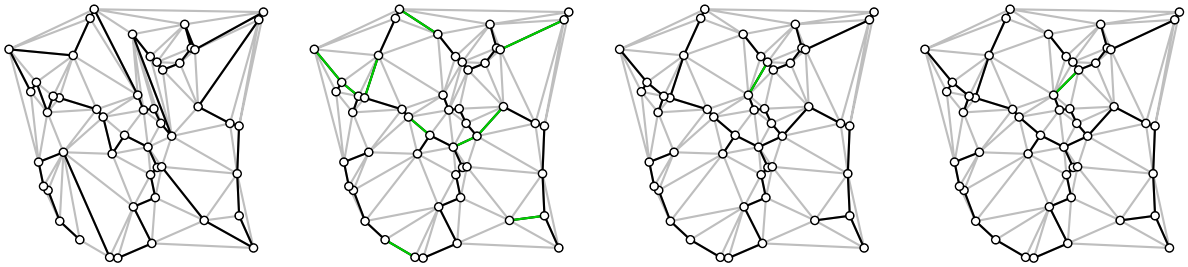


Figure 4.2: *Transformation from path to minimum spanning tree in three steps. New edges are marked in green.*

4.2 Lowerbound

Aichholzer at al. give an upperbound of $O(\log n)$ on the number of steps needed for a CDT-transformation, and also give an example for which this bound is asymptotically tight. In this example, the transformation starts with a tree, while we want to start with a path. However, we also found an example of a path that requires $O(\log n)$ transformation steps. The construction of this lowerbound example, illustrated in Figure 4.3, is based on the idea that the transformation cannot be completed as long as there are edges in our tree which cross an EMST edge. Consider an edge that is part of the EMST, shown in Figure 4.3a. Now construct a path that crosses this edge, but does not contain it, as shown in Figure 4.3b. For this path, we need at least two transformation steps. In the first step, the edge crossing the EMST edge is removed, and in the second step the EMST edge can be inserted. Let e be the edge of the path crossing the EMST edge. Insert two more edges parallel to e, one on either side, and make them at least three times as long as e. The edges can be connected into a path as shown in Figure 4.3c.

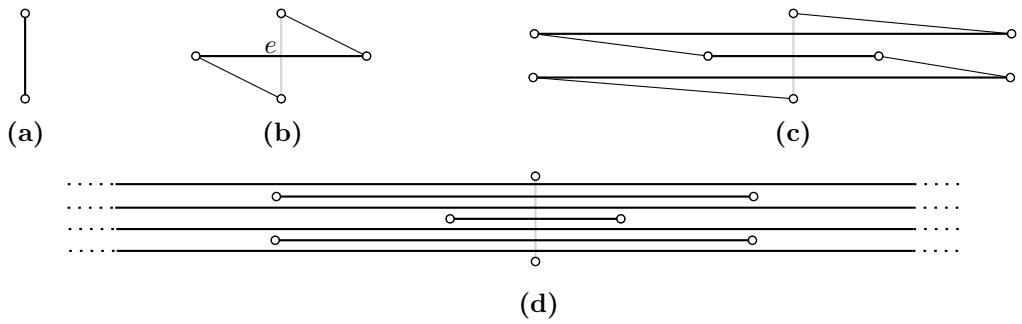


Figure 4.3: *Construction of the lowerbound example.*

In the constrained Delaunay triangulation of this path, all other edges incident to one of the endpoints of edge e are longer than e, due to the long edges we inserted. We can prove by contradiction that this implies that e is part of the minimum spanning tree of the triangulation.

Assume we have a minimum spanning tree $\mathcal{T}$ of the constrained Delaunay triangulation that does not use e. Both endpoints of e must be connected to $\mathcal{T}$, so they both have an incident edge longer than e. That means we can decrease the length of $\mathcal{T}$ by removing one of these edges and inserting edge e instead. This contradicts our assumption that $\mathcal{T}$ is a minimum spanning tree. Therefore, edge e has to be in the minimum spanning tree of the triangulation and cannot be removed in the first transformation step. Only the longer edges are removed, such that in the next step, edge e can be removed, and in the third step, the EMST edge can be inserted. We extend the example by repeatedly adding more parallel edges that are at least three times as long as the currently longest between all the existing edges, as shown in Figure 4.3d. In every step of the transformation, only the longest edges intersecting the EMST edge can be removed. These are at most half of the edges, so it takes $O(\log n)$ transformation steps before the EMST edge can be inserted.

The lowerbound example we showed here is not a very natural path to start with, since it is very long and has a lot of sharp angles. It remains an open problem whether the upperbound of $O(\log n)$ transformation steps is also tight if we start for example with the 2-approximation of the traveling salesman path or the actual traveling salesman path.

Chapter 5

Edge Flips

In this chapter, we focus on transforming a tree into a path by means of edge flips. In Section 5.1, we present a simple solution for the case where the vertices are in convex position. In Section 5.2, we focus on points in general position. We consider mainly transformations of Euclidian minimum spanning trees (EMSTs) here. We review some properties of EMSTs that we would like to maintain, and we discuss a number of possible approaches for transforming them into paths.

5.1 Convex position

The *convex hull* of a set of points is the smallest convex polygon enclosing all points. A point set is in *convex position* if all of the points are vertices of the convex hull. Convex point sets have the nice property that a polygon formed by a subset of the points is always empty: there can be no other points in the interior of such a polygon. We use this property to prove that we can always increase the diameter of a tree on a set of points in convex position with a compatible edge flip.

Lemma 5 *For any plane tree that is not a path on a set of points in convex position, we can increase the diameter by one with a compatible edge flip.*

Proof. The tree is not a path, so there is some vertex v with degree at least three on the diametrical path. Two of the edges incident to v are on the diametrical path, and at least one edge incident to v is not on the diametrical path. We call the neighbours of v on the diametrical path u_1 and u_2. Consider the circular order of neighbours around v. There has to be a neighbour of v not on the diametrical path that comes right next to either u_1 or u_2 in this order. Assume without loss of generality that vertex w is the neighbour of v not on the diametrical that comes right after u_1 in the counterclockwise circular order of the neighbours around v. This situation is illustrated in Figure 5.1 on the left.

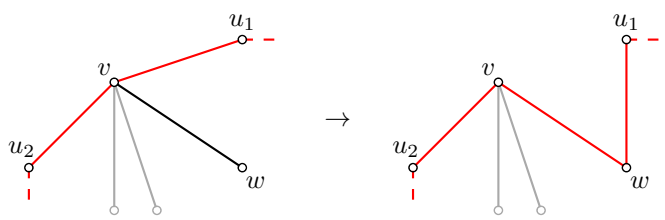


Figure 5.1: *Edge flip increasing the length of the diametrical path (red).*

We now know that the triangle u_1vw is empty. There can be no other points in this triangle because the points are in convex position, and there can be no other edges incident to v in this triangle because u_1 and w are next to each other in the circular order of neighbours around v. Remove the edge u_1v from the tree, and insert the edge wu_1, as illustrated in Figure 5.1. This does not introduce a crossing or compatibility problem, since triangle u_1vw was empty. We now increased the diameter of the tree by one, because the diametrical path now traverses two edges between v and u_1 rather than one. □

Note that the move described in Lemma 5 improves our pathliness measure. It increases the diametrical path, and the first two numbers in the furthest-distance vector always equal the length of the diametrical path. Therefore, we can use this move to transform a tree into a path, improving the measure in every step. The diametrical path of any tree on three or more points has a length of at least 2, and the length of a path on n points is $n - 1$. Since we increase the length of the diametrical path in every step, this transformation terminates after at most $n - 3$ steps. This leads us to the following theorem.

Theorem 6 *For n points in convex position, we can transform any plane tree into a plane path using $n - 3$ compatible edge flips, improving the pathliness in every step.*

5.2 General position

For points in general position, we focus on finding transformations from a path to a Euclidian minimum spanning tree (EMST), rather than from any tree to a path. We discussed in Chapter 3 that EMSTs might have a compatible path, which is not true for all trees, and EMSTs have some nice properties that could help us find a transformation.

5.2.1 Visibility order

Since the EMST is a subset of the Delaunay triangulation of a point set, we can make use of some properties of Delaunay triangulations. Floriani et al. [14] proved that given a Delaunay triangulation of set of points in the plane, we can define a partial order on the triangles with respect to a certain vertex v that is consistent with the two-dimensional visibility of the triangles from v. This means that for a ray emanating from v in any direction, the order in which it intersects the triangles is consistent with the partial order. Using the partial order on the triangles, we can also define a partial order on the edges.

Lemma 7 *There exists a partial order on the edges of a Delaunay triangulation with respect to any vertex v, such that the order in which a ray emanating from v in any direction intersects edges is consistent with the partial order.*

Proof. This follows directly from the result by Floriani et al. Consider a Delaunay triangulation and a partial order on its triangles with respect to a vertex v. Represent every triangle by the edge(s) through which a ray emanating from v could enter this triangle. The edges adjacent to v are excluded from the partial order, since a ray emanating from v can never properly intersect these. The convex hull edges are exactly the maximal elements of the partial order. □

Since the edges of an EMST are a subset of the edges of a Delaunay triangulation, we can also define an order as described in Lemma 7 on the edges of an EMST. We call such a partial order a *visibility order*. We say that a vertex v can *see* an edge e if the triangle defined by v and the endpoints of e is empty. An edge e is *visible* from a vertex v if v can see e. Any partial

order can be completed to a consistent full order. To simplify our arguments, we assume for the remainder of this chapter that the visibility order is a full order. Visibility orders have the following property.

Lemma 8 *Given a graph G and a visibility order with respect to vertex v, if we remove the first $i-1$ edges in the visibility order from the graph, vertex v can see edge i.*

Proof. By the definition of a visibility order, a ray emanating from v in any direction can only intersect edges with positions $< i$ in the order before intersecting the edge with position i. Since all the edges with positions $< i$ have been removed from the graph, a ray emanating from v will not intersect anything before intersecting edge i, which means v can see this edge. □

To apply a move similar to the one we used for the convex case, we need vertices not on the diametrical path that can see an edge of the diametrical path. A visibility order guarantees us that each vertex can see an edge it is not adjacent to.

Lemma 9 *Given a graph G and a visibilty order with respect to a vertex v, vertex v can see at least one edge it is not adjacent to.*

Proof. This follows directly from Lemma 8. If we do not remove any edges, v can at least see the first edge in the visibility order. Since edges adjacent to v are excluded from the visibility order, this implies that v can see at least one edge it is not adjacent to. □

The property shown in Lemma 9 is very helpful if we want to perform a transformation with edge flips. It excludes the possibility of cyclic dependencies among the edges, and guarantees that there is always some edge flip we can perform (although it might not be the one we want to perform). Therefore, we would like to maintain a visibility order on our tree during the transformation.

5.2.2 Increasing the diametrical path

Recall that the diametrical path of a tree is the longest path in the tree. In the remainder of this section, we use *dp-edge* as abbreviation for an edge on the diametrical path, and *non-dp-edge* for an edge that is not on the diametrical path. Similarly, a *dp-vertex* is on the diametrical path and an *non-dp-vertex* is not.

The easiest way to improve our pathliness measure is to increase the length of the diametrical path. The following move, similar to the one we used for points in convex position, is a way to do this with a single edge flip.

Move M1. Find a pair of vertices v and u such that:

- vertex v is a dp-vertex of degree at least three;
- vertex u is an non-dp-vertex and a neighbour of v;
- vertex u can see at least one of the dp-edges adjacent to v.

Let v' be the second endpoint of the dp-edge that can be seen by u. If u can see both dp-edges adjacent to v, pick v' such that the angle $\angle uvv'$ is minimized. The move is an edge flip; remove the edge (v, v') and insert the edge (u, v'). As illustrated in Figure 5.2, this increases the length of the diametrical path by one.

Lemma 10 *Move $M1$ preserves the property that we can define a visibility order on the edges of a tree with respect to any vertex in the tree.*

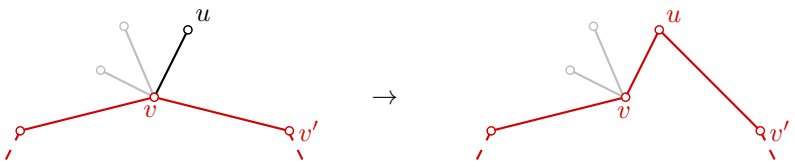


Figure 5.2: *Illustration of move $M1$. The diametrical path is shown in red.*

Proof. The move consists of two steps: adding the edge uv' to the graph, and removing the edge vv' from the graph. We show that both these actions can be performed while maintaining a visibility order with respect to any vertex in G. First, consider the insertion of edge uv'. We divide the graph around triangle uvv' into four areas, as shown in Figure 5.3.

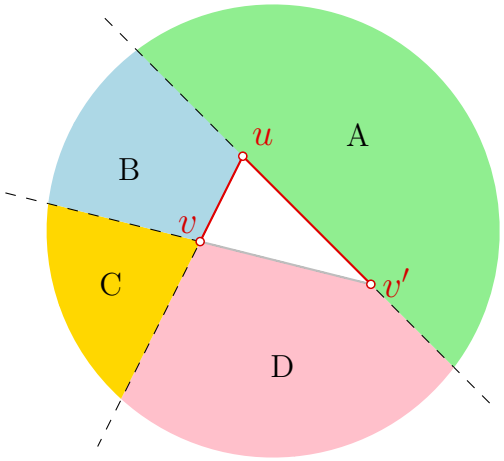


Figure 5.3: *We divide the graph around triangle uvv‘ in four areas*

For any vertex v_A in area A, a ray emating from v_A and intersecting the new edge uv' would intersect either uv or vv' immediately afterwards. This means that uv' can safely be inserted in the partial order right before whichever one of these two came first. For any vertex v_B in area B, a ray emanating from v_B and intersecting uv' would intersect uv immediately before that, so uv' can be inserted in the partial order right after uv. For any vertex v_C in area C, a ray emanating from v_C and intersecting uv' would intersect either uv or vv' immediately before that. This means uv' can be inserted in the partial order right after whichever one of these two came last. Finally, for any vertex v_D in area D, a ray emanating from v_D and intersecting uv' would intersect vv' immediately before that, so uv' can be inserted in the partial order right after vv'. Removing edge vv' does not cause any problems either, we can just remove the edge from the visibility order. □

Given an EMST, move $M1$ can always be applied. However, move $M1$ is not sufficient to transform any tree into a path. Consider the example in Figure 5.4. The red edges represent the diametrical path, and vertex v is the only vertex of degree 3. To apply move $M1$, we have to remove the edge vw and connect w to one of the neighbours of v on the diametrical path. This is not possible without introducing a crossing, since w cannot see either of these neighbours. A situation like this cannot exist in an EMST, but even if we start with an EMST it might still occur after a few transformation steps.

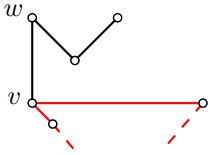


Figure 5.4: *Tree to which move M1 cannot be applied.*

In the example from Figure 5.4 it is still possible to increase the length of the diametrical path with a single edge flip. To achieve this, we should not insist on connecting w to one of the neighbours of v, but instead allow the connection of any vertex from the branch to a neighbour of v.

Removing the edge connecting v to this neighbour then gives us a new tree, with a diametrical path that is two or more edges longer than in the original tree. Figure 5.5 shows the result of applying this generalised version of move $M1$ to the tree from Figure 5.4. In this example the length of the diametrical path was increased by three. The main problem with this generalised move is that, although it occasionally works well, it is very hard to prove any of its properties. Since we have no guarantees on when we can apply it, how much it improves our diametrical path, and whether it maintains the visibility order, we decided not to use this move. Instead, we maintain the original definition of move $M1$ and add a second move. Like move $M1$, this move increases the length of the diametrical path by one. The main difference is that we do not do this with a single edge flip, but with a double edge flip: we remove two edges, and insert two new ones.

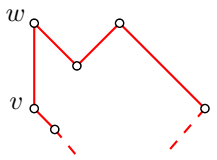


Figure 5.5: *The tree from Fig. 5.4 after application of the generalised move $M1$.*

Move M2. Find a vertex w and an edge e such that:

- vertex w is a non-dp-vertex;
- edge e is a dp-edge;
- vertex w can see both endpoints of edge e.

Since vertex w is not on the diametrical part, it is part of a branch that is connected to the diametrical path in some dp-vertex v of degree at least three. Let edge e' be the edge connecting the branch containing w to v. This move is a double edge flip: remove the edges e and e' and insert the edges connecting w to both endpoints of e. As illustrated in Figure 5.6, this increases the length of the diametrical path by one.

Figure 5.6: *Illustration of move $M2$. The diametrical path is shown in red.*

With moves $M1$ and $M2$, we can transform a tree into a path in such a way that the length of the diametrical path increases by one in each step. This means that the transformation takes at most $n - 3$ steps, because the initial diametrical path has at least length 2 (for 3 or more vertices) and the final path has length $n - 1$. Since each step involves at most two edge flips, the total number of edge flips in such a transformation is at most $2n - 6$. Figure 5.7 shows an example of a transformation with moves $M1$ and $M2$. The diametrical path is shown in red, and the figure shows step by step how a branch of the tree is included in the diametrical path.

Recall that we wanted to maintain a visibility order on the vertices of our tree during the transformation. We were unable to prove whether or not this holds for move $M2$. However, we discovered that the visibility order does not guarantee that we can always perform move $M2$. As explained in Section 5.2.1, a visibility order guarantees that each vertex can see at least one edge it is not adjacent to. To perform move $M2$, however, we need a non-dp-vertex that can see a dp-edge. A visibility order does not guarantee that such a vertex exists.

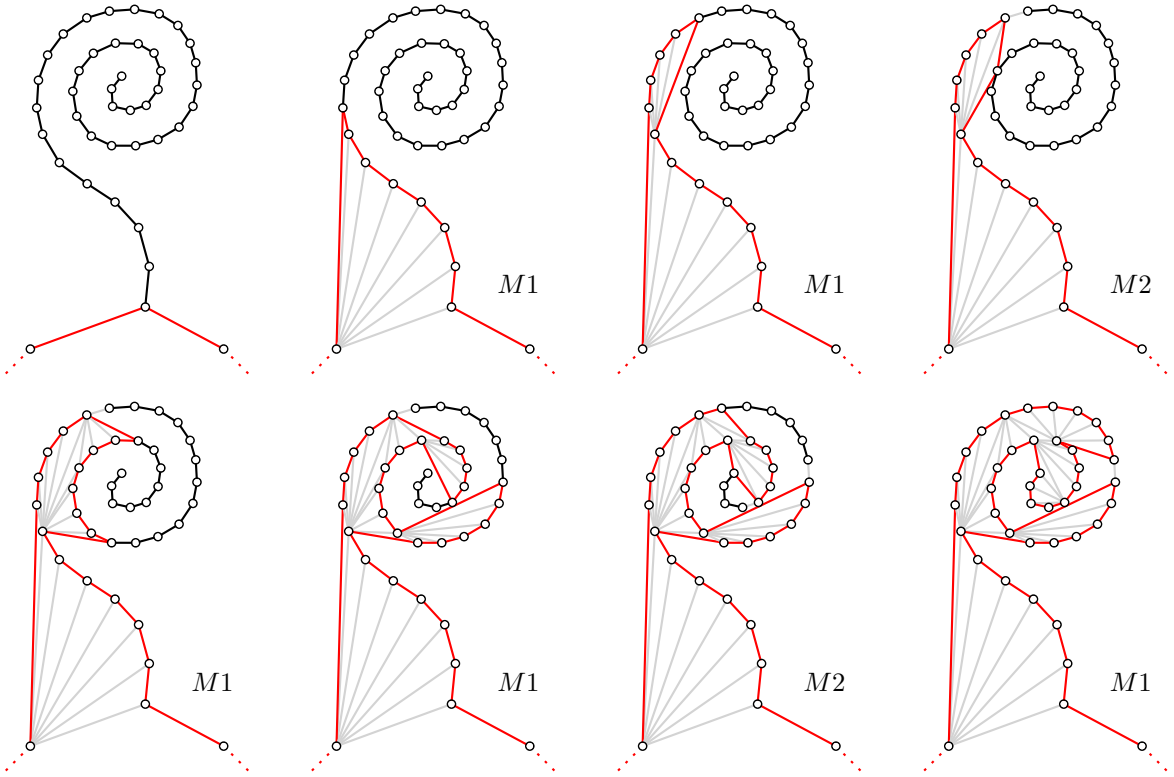


Figure 5.7: *Transformation of a branch of the diametrical path with moves $M1$ and $M2$.*

Lemma 11 *Given a plane tree, the presence of a visibility order for each vertex does not guarantee that there exists a non-dp-vertex that can see a dp-edge.*

Proof. Consider the tree in Figure 5.8, where the diametrical path is shown in red. There exists a visibility order with respect to each vertex of this tree. Still, all vertices not on the diametrical path can see only edges that are not on the diametrical path either. □

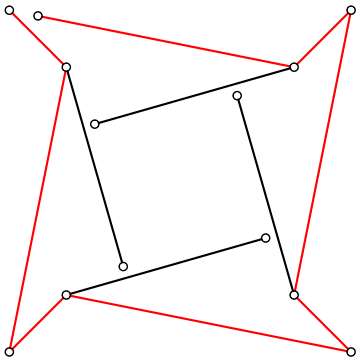


Figure 5.8: *In this tree there exists a visibility order, but none of the non-dp-vertices can see a dp-edge.*

The example in Figure 5.8 is a tree on which we can define a visibility order, but which we cannot transform using moves $M1$ and $M2$. So, we cannot transform any tree into a path using these moves. If we consider an EMST, rather than just any tree, we know that we can always apply move $M1$. However, after we applied some transformation steps, the tree is no longer an EMST, and we no longer have that guarantee. The question whether or not moves $M1$ and $M2$

are sufficient to transform an EMST into a path while increasing the length of the diametrical path in every step therefore remains an open problem.

5.2.3 Increasing pathliness

In the previous section, we focused on edge moves directly increasing the length of the diametrical path. However, what we actually want is to increase the pathliness of the tree, and this does not require the length of the diametrical path to increase. For the tree in Figure 5.8, for example, it is not possible to increase the length of the diametrical path with a single edge flip, but it is possible to increase the pathliness with a single edge flip. Unfortunately, this is not always the case.

Lemma 12 *There exists a tree for which the pathliness measure cannot be improved with a single edge flip.*

Proof. Consider the example in Figure 5.9, with only one vertex w that does not lie on the diametrical path. Let v be the neighbour of w on the diametrical path. There are two ways to increase the pathliness here: either include w in the diametrical path, or connect w to a vertex closer to one of the endpoints of the diametrical path than v. To include w in the diametrical path with a single edge flip, we need to connect it to one of the neighbours of v. This is not possible, since w cannot see either of these neigbours. The alternative is to remove the edge vw, and insert an edge connecting w with a dp-vertex that is closer to one of the endpoints of the diametrical path than v. However, w cannot see any of these points either. Therefore, it is not possible to increase the pathliness of this tree with a single edge flip. □

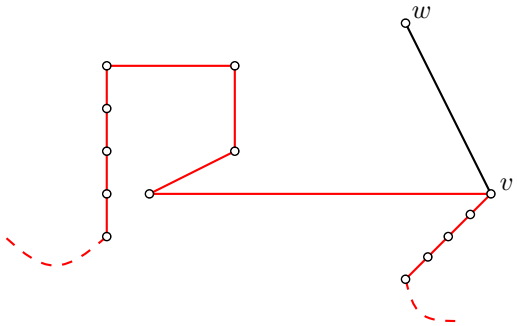


Figure 5.9: *The measure of this tree cannot be improved with a single edge flip.*

The example of Lemma 12 is not an EMST, and a situation like this cannot occur in an EMST. As mentioned in the previous section, an EMST that is not a path always allows move $M1$. This implies that the pathliness of an EMST can always be improved with a single edge flip. It remains an open question whether or not we can keep improving pathliness with edge flips if we start from an EMST.

Chapter 6

Conclusion

In this thesis, we studied compatible transformations from tree to path. Our aim was to find a transformation for which each pair of subsequent steps is compatible and the tree becomes more path-like in every step. We designed a pathliness measure to determine how path-like a tree is. The measure is based on the furthest distance within the graph from each of the vertices. These distances are larger for a path than for any other tree. We proved that a compatible transformation in a single step is not always possible, because there exist trees without a compatible path. It remains an open problem whether or not a Euclidian minimum spanning tree always has a compatible path. We analyzed the results of an existing transformation method by Aichholzer et al., based on constrained Delaunay triangulations. This method generates compatible transformation steps, but has the disadvantage that many edges change in a single step. We presented an example to show that even if we start with a path, the transformation can take $O(\log n)$ steps. It is still an open problem whether this can also happen if we start with (the 2-approximation of) the traveling salesman path. We also studied transformations based on edge flips. For n points in convex position, we proved that we can transform a tree into a path in at most $n - 3$ compatible, pathliness improving edge flips. For points in general position, we discussed two moves based on edge flips that improve the length of the diametrical path. We showed that these moves are not sufficient to transform any tree into a path. Whether or not they are sufficient to transform an EMST into a path remains an open problem. We also considered edge flips that increase only the pathliness, but not necessarily the length of the diametrical path. We proved that there exists a tree for which we cannot improve the pathliness with a single edge flip. It is still an open problem whether we would ever reach such a tree if we start from an EMST.

Bibliography

[1] Oswin Aichholzer, Franz Aurenhammer, and Ferran Hurtado. Sequences of spanning trees and a fixed tree theorem. *Computational Geometry*, 21(1-2):3–20, 2002.

[2] Oswin Aichholzer, Franz Aurenhammer, Ferran Hurtado, and Hannes Krasser. Towards compatible triangulations. *Theoretical Computer Science*, 296(1):3–13, 2003.

[3] Oswin Aichholzer, Sergey Bereg, Adrian Dumitrescu, Alfredo García Olaverri, Clemens Huemer, Ferran Hurtado, Mikio Kano, Alberto Márquez, David Rappaport, Shakhar Smorodinsky, Diane L. Souvaine, Jorge Urrutia, and David R. Wood. Compatible geometric matchings. *Computational Geometry*, 42(6-7):617–626, 2009.

[4] Oswin Aichholzer and Klaus Reinhardt. A quadratic distance bound on sliding between crossing-free spanning trees. *Computational Geometry*, 37(3):155–161, 2007.

[5] Basak Alper, Nathalie Henry Riche, Gonzalo Ramos, and Mary Czerwinski. Design study of LineSets, a novel set visualization technique. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2259–2267, 2011.

[6] Boris Aronov, Raimund Seidel, and Diane L. Souvaine. On compatible triangulations of simple polygons. *Computational Geometry*, 3:27–35, 1993.

[7] Prosenjit Bose and Ferran Hurtado. Flips in planar graphs. *Computational Geometry*, 42(1):60–80, 2009.

[8] L. Paul Chew. Constrained delaunay triangulations. *Algorithmica*, 4(1):97–108, 1989.

[9] Christopher Collins, Gerald Penn, and Sheelagh Carpendale. Bubble Sets: revealing set relations with isocontours over existing visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1009–1016, 2009.

[10] Michael B. Dillencourt. A non-Hamiltonian, nondegenerate Delaunay triangulation. *Information Processing Letters*, 25(3):149–151, 1987.

[11] Michael B. Dillencourt. Traveling salesman cycles are not always subgraphs of Delaunay triangulations or of minimum weight triangulations. *Information Processing Letters*, 24(5):339–342, 1987.

[12] Michael B. Dillencourt. Finding Hamiltonian cycles in Delaunay triangulations is NP-complete. *Discrete Applied Mathematics*, 64(3):207–217, 1996.

[13] Kasper Dinkla, Marc J. van Kreveld, Bettina Speckmann, and Michel A. Westenberg. Kelp Diagrams: point set membership visualization. *Computer Graphics Forum*, 31(3):875–884, 2012.

[14] Leila De Floriani, Bianca Falcidieno, George Nagy, and Caterina Pienovi. On sorting triangles in a Delaunay tessellation. *Algorithmica*, 6(4):522–532, 1991.

[15] Wayne Goddard and Henda C. Swart. Distances between graphs under edge operations. *Discrete Mathematics*, 161(1-3):121–132, 1996.

[16] Thomas Hackl, Oswin Aichholzer, and Birgit Vogtenhuber. Compatible pointed pseudo-triangulations. In *Canadian Conference on Computational Geometry*, pages 91–94, 2010.

[17] Sabine Hanke, Thomas Ottmann, and Sven Schuierer. The edge-flipping distance of triangulations. *Journal of Universal Computer Science*, 2(8):570–579, 1996.

[18] Ferran Hurtado, Marc Noy, and Jorge Urrutia. Flipping edges in triangulations. *Discrete & Computational Geometry*, 22(3):333–346, 1999.

[19] Ferran Hurtado and Csaba D Tóth. Plane geometric graph augmentation: a generic perspective. In *Thirty Essays on Geometric Graph Theory*, pages 327–354. Springer, 2013.

[20] Charles L. Lawson. Transforming triangulations. *Discrete Mathematics*, 3(4):365 – 372, 1972.

[21] Wouter Meulemans, Nathalie Henry Riche, Bettina Speckmann, Basak Alper, and Tim Dwyer. Kelpfusion: A hybrid set visualization technique. *IEEE Transactions on Visualization and Computer Graphics*, 19(11):1846–1858, 2013.