The Point-set embeddability Problem for Plane Graphs*

Therese Biedl† Martin Vatshelle‡

Technical Report CS-2011-27

Abstract

In this paper, we study the Point-set embeddability-problem, i.e., given a planar graph and a set of points, is there a mapping of the vertices to the points such that the resulting straight-line drawing is planar? This problem is NP-hard if the embedding can be chosen, but becomes polynomial for triangulated graphs of treewidth 3. We show here that in fact it can be answered for all planar graphs with a fixed embedding that have constant treewidth and constant face-degree.

We also prove that as soon as one of the conditions is dropped (i.e., either the treewidth is unbounded or some faces have large degrees), Point-set embeddability with a fixed embedding becomes NP-hard. The NP-hardness holds even for a 3-connected planar graph with constant treewidth, triangulated planar graphs, or 2-connected outer-planar graphs.

*The authors would like Dimitrios Thilikos for helpful input.

†David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, ON N2L 1A2, Canada. *biedl@uwaterloo.ca*. Research supported by NSERC and by the Ross and Muriel Cheriton fellowship.

‡Institute for Informatik, University of Bergen, Norway. *vatshelle@ii.uib.no*. Research done while visiting the University of Waterloo.

1 Introduction

Planar graph drawing is the art of drawing a graph in the two-dimensional plane such that no two edges cross. It has been known for many decades that any planar graph can be drawn with straight lines without crossing [25, 9, 22]. In fact, one of the first problems studied in the graph drawing community was how to achieve a straight-line drawing such that coordinates of points are small. Multiple constructions were given to achieve an $O(n) \times O(n)$-grid for a planar graph with n vertices [19, 10].

One line of research in this field has worked on restricting the placement of the vertices further. In particular, is it possible to restrict the placement of vertices to a given set of points? This is the so-called Point-set embeddability problem: Given a planar graph G and a set P of points in the plane, test whether there exists a mapping of the vertices of G onto the point-set P that results in a planar straight-line drawing of G. This problem is the topic of the present paper.

1.1 Existing results

It has been discovered multiple times [13, 8, 5] that Point-set embeddability is easy for outer-planar graphs, presuming the points are in general position and the drawing that we want is the "standard" one (with all points on the outer-face). Namely, *any* outer-planar graph can be embedded in *any* set of n points in general position so that the drawing is planar.

On the other hand, for arbitrary planar graphs a point-set embedding is not possible for all point sets. Cabello [7] proved that it is NP-hard to decide whether a planar graph can be embedded in a given set of points, even if the graph is 2-outer-planar. However, his proof crucially uses that the embedding of the graph is not fixed; in particular NP-hardness remained open for 3-connected graphs.

Since then, many variations of Point-set embeddability have been investigated. For example, authors studied version with further restrictions as to which vertices may be mapped to which points [2], or where only a subgraph is restricted to points [12]. Other papers considered Point-set embeddability for special graph classes. In two papers, Nishat et al. [16] and Moosa and Rahman [14] gave algorithms to test Point-set embeddability in planar 3-trees (i.e., triangulated planar graphs of treewidth 3).

1.2 Our results

Our investigation was motivated by the last two papers. If Point-set embeddability is polynomial for triangulated graphs of treewidth 3, then what can be said about triangulated graphs of treewidth k, for some constant k? We showed that Point-set embeddability is polynomial for all these, and we can even relax the triangulation restriction. Specifically we show:

> Point-set embeddability is polynomial for any graph of constant treewidth and constant face-degree if the embedding of the graph must be respected.

On the other hand, if the embedding can be chosen freely, then Cabello's NP-hardness result shows NP-hardness even for graphs of constant treewidth (his constructed graphs have treewidth 5) and for constant face-degree (only one of his faces has degree greater than 4, and this can be overcome by modifying his constructiion a bit.)

This leaves open the case of graphs that have faces with arbitrarily large degree, or that have large treewidth, but the embedding is fixed. Here we prove:

POINT-SET EMBEDDABILITY is NP-hard for 2-connected outer-planar graphs if the embedding is fixed.

POINT-SET EMBEDDABILITY is NP-hard for 3-connected planar graphs, even if the treewidth is constant.

POINT-SET EMBEDDABILITY is NP-hard for triangulated planar graphs.

Our paper is organized as follows. After giving definitions, we present the algorithm for graphs of constant treewidth and face-degrees in Section 3. We prove the NP-hardness of POINT-SET EMBEDDABILITY in various scenarios in Section 4 and conclude with open problems.

2 Background

We assume familiarity with graph-theoretic concepts such as graphs, degrees, and connectivity. Throughout this paper, let $G = (V, E)$ denote a graph with n vertices and m edges. G is always assumed to be *planar*, i.e., it can be drawn in the plane without crossing. In such a planar drawing, the connected pieces that remain when removing the drawing are called *faces*, with the unbounded piece called *outer-face*. All our input graphs are assumed to be simple, i.e., to have neither loops nor multiple edges. Any simple planar graph (even if disconnected) has $m \leq 3n - 3$ edges.

Combinatorial embeddings To describe drawings and faces without using geometry, we need two ingredients. A *rotational system* assigns to each vertex of G a circularly ordered list of the edges of G incident to v. A planar drawing *respects the rotational system* if for every vertex the order in the rotational system is the counter-clockwise (ccw) order of incident edges in the drawing. We assume throughout this paper that a rotational system of G has been fixed in such a way that it can be respected by some planar drawing.

Such a rotational system determines all planar drawings of G (up to choice of the outer-face and topological deformations) if G is connected. If G is not connected, then we need more information to determine its drawing uniquely. We define two kinds of *angle* in a graph. One kind of angle is an ordered triple $\{e_1, v, e_2\}$ such that e_1 and e_2 both end at v, and the ccw order at v contains e_1 followed by e_2. If v has degree 1 then e_1 and e_2 are the same. In a planar drawing, the area ccw between e_1 and e_2 and v belongs to some face f; we say that f is *incident to this angle*. The other kind of angle occurs at a vertex that has degree 0, and is described simply by the vertex itself. In any drawing, the face that is adjacent to this vertex is called the face *incident to this angle*.

A *combinatorial embedding* of a graph now consists of a planar graph with a rotational system, a finite set F, one special element o of F, and a mapping of angles to elements of F. We say that a planar drawing *respects the combinatorial embedding* if it respects the rotational system, its faces are in 1-1 correspondence with F with the outer-face mapped to o, and the face incident to each angle is as prescribed by the mapping. In this paper we always assume that a combinatorial embedding is fixed and only consider drawings that respect it. Therefore, we will often use geometric terms (such as face and outer-face) interchangeably with combinatorial terms (such as F and o.)

The *degree* $\deg(f)$ of a face f is the number of angles to which f is incident. If the edges incident to f form a simple cycle (as they do for 2-connected graphs), then the degree of a face equals the number of edges (or number of vertices) that are incident to f. But the degree may be higher than that if there are cut-vertices or vertices of degree 0 incident to f.

If G is 2-connected, then the boundary of each face f is a simple cycle and hence there is a natural order of vertices around each face f of G. If G has cutvertices or is disconnected, then

for some faces the boundary is not simple. To be able to handle this, we assume that every face f stores a fixed order $\rho(f)$ of vertices that are adjacent to f. The precise order of $\rho(f)$ is not important as long as it has been fixed.

A *triangulated graph* is a planar graph for which all faces have degree 3. Such a graph is 3-connected.

Drawings and Point-set embeddability In this paper, we only consider straight-line drawings, i.e., drawings where edges are straight-line segments between the points assigned to their endpoints. Therefore, we can describe a *proper drawing* of a planar graph G by giving only the mapping from vertices to points, and demand that this mapping satisfies: (a) No two vertices are mapped to the same point. (b) The interior of the line-segment of each edge contains no points of other vertices or edges. (c) The rotational system and outer-face are respected. (d) The face-angle incidences are respected. We will also consider drawings of a subgraph G' of G, and consider them to be proper if the conditions are satisfied for all elements of G' that also exist in G. So in particular a proper drawing of of G' means a drawing that satisfies (a) and (b) above, respects the induced rotational system, and respects the face-angle incidences for every face f and angle α of G' that also existed in G.

Given a mapping of the n vertices to points, we can check in $O(n \log n)$ time whether this mapping corresponds to a proper drawing; the bottleneck being the time for testing intersections of line segments and finding the face-incidences for disconnected components.

The Point-set embeddability problem is defined formally as follows. Given a planar graph $G = (V, E)$ with a combinatorial embedding. Further given a set S of points in the plane, where we may assume $|S| \geq n = |V|$. Is there a mapping of V to S such that the resulting drawing of G is proper? Note that we allow S to have more points than necessary to draw G. We also do not require S to be in general position.

Crucial for solving Point-set embeddability on some graphs will be to consider the dual graph. For a graph G with a combinatorial embedding, let the *dual graph* G^* consist of assigning a vertex v_f to each face $f \in F$. For each edge e adjacent to two faces f_1, f_2, add an edge from v_{f_1} to v_{f_2} in the dual graph. The dual graph (at least in our definition) ignores all vertices of degree 0; our algorithm will take care of them separately since they are included in the list $\rho(f)$ for the face f that they are incident to. The dual graph is always connected even if G was not. It may have loops or multiple edges even if G was simple.

Treewidth and pathwidth The treewidth of a graph is a graph parameter that has proved very useful both in theoretical studies (for the Graph Minor Theorem [17]) and for algorithmic development and fixed-parameter tractable algorithms (see e.g. [3].) We will not give the (somewhat complicated) definition of treewidth as we do not use it directly.

Computing the treewidth of a graph is NP-hard in general [1], but can be done in polynomial time if the treewidth is a constant [4]. Graphs of treewidth t are also called *partial t-trees*. Any graph of treewidth 2 is planar, and is also called a *series-parallel graph*. A special case of series-parallel graphs are *outer-planar graphs*, which can be drawn such that all vertices are on the outer-face.

To give an upper bound on the treewidth, it suffices to analyze the *pathwidth* defined as follows: A graph has pathwidth k if its vertices can be enumerated as $v_1, \ldots, v_n$ such that for any $1 \leq i \leq n$, at most k vertices in $\{v_1, \ldots, v_i\}$ have a neighbour in $\{v_{i+1}, \ldots, v_n\}$. Any graph of pathwidth k also has treewidth at most k, but not vice versa.

3 Algorithms for point-set embeddability

In this section, we give an algorithm for Point-set embeddability if some graph parameters are small. Specifically, we aim to show:

Theorem 3.1. *Let G be a planar graph with n vertices and a fixed combinatorial embedding. If the dual graph G^* of G has treewidth at most t and maximum degree at most Δ, then* Point-set embeddability *can be solved for G on any point set S in $O(nt\Delta \log(t\Delta) \cdot |S|^{1.5(t+1)\Delta})$ time and space.*

We first give an outline of the proof. Rather than using the treewidth directly, we use the bound on treewidth and maximum degree to bound another graph parameter of the dual graph G^* known as the *carving width* and defined precisely below. Translating back to G, the carving-width roughly corresponds to the number of edges needed on a cycle to separate the faces of G into parts (and recursively in the subgraphs). Since the carving width is bounded, we can brute-force enumerate all possible ways in which such a cycles could be mapped to the given point set S. This therefore gives rise to a dynamic programming approach. In the base case, for each face f of G find all possible sets of points in S that could possibly hold the boundary of f. Then in the recursive steps consider all possible ways of gluing faces together to obtained a bigger subgraph; this is polynomial if the carving width is a constant.

3.1 Carving-width

The carving width of a graph (not necessarily planar) was first introduced by [20] and can be defined as follows:

Definition 3.2 (carving-width)**.** *A* carving-decomposition *of width k is a rooted binary tree T and a bijection δ between V and the leaves of T such that for any arc a of T, there are at most k edges e in G that* cross *arc a, i.e., the endpoints of e are mapped to different components of $T - a$. The carving-width of a graph $G = (V, E)$ is the minimum width over all carving-decompositions.*

The carving-width is a generalization of the *cut-width*, where we require the tree T to be a caterpillar, see [23, 24] for an overview. Computing the carving-width of a graph is NP-hard in general, but becomes polynomial if the graph is planar [20]. (It is also fixed-parameter tractable in the carving width [23].)

There is a close relationship between the carving width and the treewidth and the maximum degree Δ. In particular, any graph of treewidth t has carving width at least $\frac{1}{3}t$ [23]. Also, for graphs without loops the carving width is at least Δ, since the arc from the leaf that stores vertex v is crossed by $\deg(v)$ edges. The carving width can also be upper-bounded in terms of treewidth and maximum degree.

Lemma 3.3. *If G has treewidth t and maximum degree Δ, then the carving width of G is at most $(t + 1) \cdot \Delta$.*

Proof. It is known that the carving width of G is at most $b \cdot \Delta$, where b is the so-called *branchwidth* of G [15]. The branchwidth in turn is bounded by $t + 1$ [18], so the result holds. □

3.2 Embedding one face

We now solve POINT-SET EMBEDDABILITY for any graph G for which the dual graph G^* has carving width at most k; this proves Theorem 3.1 by the previous lemma. We will also assume that $k \geq \Delta$; this always holds if G^* is simple, but even if G^* has loops this can be assumed since Theorem 3.1 only demands a bound of $\Delta(t+1)$.

So assume we have a carving decomposition T of G^* that has width $k \geq \Delta$. Presume also that we are given a set of S of $s \geq n$ points into which we want to embed G. We solve POINT-SET EMBEDDABILITY using a bottom-up dynamic programming algorithm.

Recall that we assumed T to be rooted. The leaves of T are in bijection with the vertices of G^*, hence the faces of G. Let f be one such face of G. Recall that $\rho(f)$ is some enumeration of the vertices of f, and $|\rho(f)| \leq \deg(f)$.

Now consider any ordered set π of $|\rho(f)|$ points in S. If we map vertices in $\rho(f)$ to points of π in this order, then this gives one possible way of drawing f (or more precisely, the graph G_f that is formed by face f) on the point set S. Of course, for most sets π, this will not be a proper drawing.[1]

For each such π, test whether the drawing of G_f is proper. If G can be drawn on S, then for some ordered sets π we will succeed, and by trying all possible ordered sets π, we can find all possible ways to draw G_f in S. We hence compute and store, for every face f and every ordered set π of $|\rho(f)|$ points of S:

$$M_b(f,\pi) = \begin{cases} 1 & \text{if mapping } \rho(f) \text{ to } \pi \text{ in this order gives a proper drawing of the graph} \\ & \text{formed by face } f. \\ 0 & \text{otherwise} \end{cases}$$

The computation time for all these values $M_b(.,.)$ is $O(n \cdot k \log k \cdot s^k)$, since there are $O(n)$ faces f, at most $k!\binom{s}{k} \leq s^k$ ordered sets of length $|\rho(f)| \leq \deg(f) \leq \Delta \leq k$, and we spend $O(k \log k)$ time to test whether the drawing of f is proper.

3.3 Subgraphs and boundaries

Now we consider some larger subgraphs of G. To explain how to combine such subgraphs, we need a definition. Let S be a subset of faces of G. The *boundary* B of S is the set of all edges e for which the face on one side belongs to S and the face on the other side does not belong to S. We say that a vertex belongs to the boundary if one of its incident edges is on the boundary.

We specifically need boundaries for sets of faces defined by the carving decomposition T of G^*. For every node $w \in T$, let $F(w)$ be all vertices of G^* (hence faces of G) mapped to a leaf below w in T. Let $B(w)$ be the boundary of $F(w)$. By definition of boundary and $F(w)$, any edge in $B(w)$ has one side on a face below w in T and the other side on a face not below w in T. Therefore, the dual of any edge in $B(w)$ crosses the arc from w to its parent in T. By definition of carving width, therefore $|B(w)| \leq k$.

Similarly as for faces, we assume that for every node w of T we have fixed an order $\rho(w)$ of the vertices that are on the boundary $B(w)$. We also need the notation $G(w)$ for the graph formed by faces $F(w)$, i.e., the graph consisting of all vertices and edges that are on at least one face in $F(w)$.

[1] We could improve the run-time of our algorithm by only considering point sets for which the simple cycles that form the boundary of f are crossing-free. A set of s points has at most $O(86.89^s)$ crossing-free spanning cycles [21], so this would eliminate many candidates (but still be exponential.)

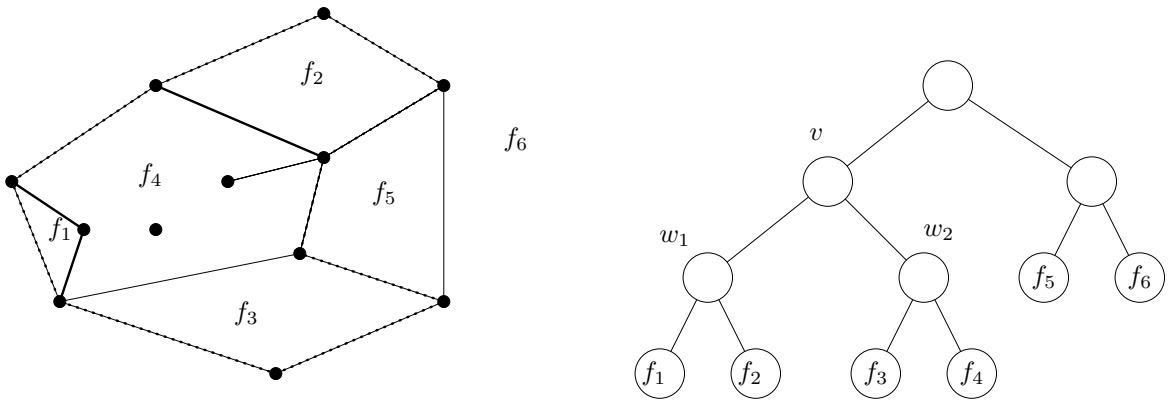

Figure 1: An example of a planar graph and a carving decomposition of its dual graph. Face f_4 has degree 9 (this includes the isolated vertex.) The boundary $B(v)$ is shown dotted, and E_σ (for v) is shown bold.

We now aim to compute the following function for all nodes w of T and all ordered sets π of $|\rho(w)|$ points of S:

$$M(w,\pi) = \begin{cases} 1 & \text{if mapping } \rho(w) \text{ to } \pi \text{ in this order gives a proper drawing of } B(w), \text{ and} \\ & \text{it can be expanded to a proper drawing of } G(w) \text{ on } S. \\ 0 & \text{otherwise} \end{cases}$$

Note that in the previous subsection we computed a very similar function for any node ℓ that is a leaf of T (hence corresponds to a face f.) To compute $M(\ell,\pi)$ for a leaf ℓ, we only need to find all entries $M_b(f,\pi')$ that are 1. Observe that $\rho(\ell)$ is a subset of $\rho(f)$ (it ignores all vertices that are inside f, such as vertices of degree 0 or perhaps cutvertices.) So π' induces an ordered set π of length $\rho(\ell)$ for which $M(\ell,\pi) = 1$, and this is the only way in which $M(\ell,\pi)$ can be 1. So we can compute the values $M(\ell,.)$ for any leaf ℓ in time $O(ks^k)$.

3.4 Combining subgraphs of the carving decomposition

Now consider two siblings w_1 and w_2 in T with their parent v. We can use the entries in $M(w_1,\pi_1)$ and $M(w_2,\pi_2)$ to compute the entries for $M(v,\pi)$ as follows. Let E_σ be all edges in $B(w_1) \cap B(w_2)$, therefore $B(v) = B(w_1) \cup B(w_2) - E_\sigma$. Let V_σ be all those vertices in $\rho(w_1) \cup \rho(w_2)$ that are not in $\rho(v)$. A vertex in V_σ "disappears" from the boundary, which is possible only if it has at least two incident edges in E_σ; hence $|V_\sigma| \leq |E_\sigma|$. For ease of notation, we will assume that the vertices in V_σ listed *last* in both enumerations $\rho(w_1)$ and $\rho(w_2)$ of vertices in $B(w_1)$ and $B(w_2)$.

Note that if we have an ordered set π of points in S assigned to vertices in $\rho(v)$, then this fixes the points assigned to all vertices of $\rho(w_1) - V_\sigma$ and $\rho(w_2) - V_\sigma$. To compute $M(v,\pi)$ therefore we only need to decide on how to expand π to add points for the vertices in V_σ, and test whether the subgraphs can be drawn in the resulting areas. So we compute $M(v,\pi)$ as follows:

Initialize $M(v,\pi) = 0$.
Test whether assigning $\rho(v)$ to π draws (v) properly.
    Break if not.

Let π_1 be the assignment of points to $\rho(w_1) - V_\sigma$ implied by π.
Let π_2 be the assignment of points to $\rho(w_2) - V_\sigma$ implied by π.
For any ordered set π_σ of $|V_\sigma|$ points[2]
    If $M(w_1, \pi_1 \cup \pi_\sigma) = 1 = M(w_2, \pi_2 \cup \pi_\sigma)$
        Set $M(v, \pi) = 1$ and break.

Note that our algorithm only tests whether $G(w_1), G(w_2)$ and $B(v)$ are drawn properly before setting $M(v, \pi)$ to be 1. It is not too hard to see that this also ensures that all of $G(v)$ is drawn properly. In particular, $B(w_1)$ and $B(w_2)$ are drawn in disjoint regions (except at the common edges E_σ), otherwise $B(v)$ would self-cross. Since $B(w_1)$ and $B(w_2)$ consist of the disjoint union of circuits, they define regions which must contain the proper drawings of $G(w_1)$ and $G(w_2)$. This gives a drawing of $G(v)$, and the only way for it not to be proper would be if the circular order of edges were violated at some vertex in $B(w_1) \cap B(w_2)$. But we test for this when testing whether $B(v)$ is drawn properly. Hence this gives a proper drawing of $G(v)$ as desired.

3.5 Putting it all together

To finish the proof of the theorem, we now do the obvious dynamic programming in the tree T: Compute $M(\ell, \pi)$ for all leaves ℓ and all ordered lists π of $\rho(\ell)$. Then go bottom-up in the tree and compute $M(v, \pi)$ for every node v using the values of its two children. Note that for the root r the boundary $B(r)$ is empty. Hence $B(r, \emptyset)$ will tell us whether the whole graph G can be embedded in the given point set.

It remains to analyze the running time. Constructing the dual graph and computing its carving-decomposition is polynomial (and for all but the smallest values of k it is dominated by the other operations.) For every leaf ℓ with corresponding face f we spend $O(k \log k \cdot s^k)$ time to compute $M_b(f, .)$, and then again as much to compute $M(\ell, .)$.

Now consider a node v higher up in a tree. Given one fixed ordered set of points π, we can test whether this gives a proper drawing of the boundary in $O(k \log k)$ time since $|\rho(v)| \leq |B(v)| \leq k$. Computing π_1 and π_2 takes $O(k)$ time. There are $O(s^{|V_\sigma|})$ ways to do choose π_σ, and for each one of them we look up two values.[3] So we spend $O(k \log k + s^{|V_\sigma|})$ time for one ordered set π of length $\rho(v)$. Summing over all such sets gives run-time $O(s^{|\rho(v)|} \cdot \big(k \log k + s^{|V_\sigma|}\big))$.

To bound this, we need to bound $|\rho(v)| + |V_\sigma| \leq |B(v)| + |E_\sigma|$. Notice that $|B(v)|, |B(w_1)|$ and $|B(w_2)|$ all are at most k. Also, $|B(w_1)| + |B(w_2)| = |B(v)| + 2|E_\sigma|$, therefore $2|B(v)| + 2|E_\sigma| \leq k + |B(v)| + 2|E_\sigma| \leq k + |B(w_1)| + |B(w_2)| \leq 3k$. So we spent $O(k \log k \cdot s^{|B(v)|+|E_\sigma|}) = O(k \log k \cdot s^{1.5k})$ time per node v. Note that this bound also holds for the leaves and dominates the terms needed for initialization. Since T has $O(n)$ nodes, therefore the total run-time is $O(nk \log k \cdot s^{1.5k})$, which finishes the proof.

Corollary 3.4. Point-set embeddability *is polynomial for any triangulated graph G of bounded treewidth.*

[2]We do not explicitly restrict which points are used by the subgraphs, and so we even test point sets that are clearly wrong (such as assigning points to V_σ that are outside the circuits formed $B(v)$.) One could certainly decrease the number of ordered sets tested, at the expense of spending more time to select the sets. We opted for the simpler code.

[3]We presume $M(w, \pi)$ is stored so that they it be looked up in constant time. More space-saving data structures are possible, but they have larger lookup time.

Proof. Since G is triangulated, the dual graph has maximum degree 3. Since G has bounded treewidth, so does the dual [6]. Therefore by the previous theorem POINT-SET EMBEDDABILITY is polynomial for G. □

It would be desirable to find an algorithm for POINT-SET EMBEDDABILITY that is not only polynomial if the treewidth and face-degree are constant, but that is fixed-parameter tractable in these parameters. In other words, if the dual graph has carving width k, can we solve POINT-SET EMBEDDABILITY in time $O(f(k)p(n, |S|, k))$ for some polynomial p and some function f that does not depend on n or $|S|$? This remains open. The main obstacle is that we would need to restrict the set of points in S to only $f(k)$ points (independent of $|S|$) that could possibly be used for boundaries; this seems unlikely to be feasible without considering the structure in G in much more detail than our algorithm did.

4 NP-hardness

In this section, we show that POINT-SET EMBEDDABILITY is NP-hard. This was already shown by Cabello [7], but our proof improves on his in some ways. Most importantly, Cabello's NP-hardness proof uses that the embedding of the planar graph is not fixed: flipping the order among the neighbours is the crucial decision-making gadget in his proof. As opposed to this, our proof is for graphs where the embedding is fixed, and we can generalize it to planar graphs that are triangulated and hence have a unique embedding. Secondly, Cabello's proof uses points some of which are on a line, whereas our proof can use points in general position. (His proof could be modified to achieve this as well.) Finally, the graph in Cabello's reduction is 2-outer-planar and has treewidth 3. For our construction, the graph is outer-planar and hence has treewidth 2. We give multiple versions of our NP-hardness proof. The first version is very simple, but uses a graph that is not connected and assumes that the combinatorial embedding is fixed. The second version adds layers of circles and connects them up to make the graph connected. The last three versions add even more edges between layers to achieve higher connectivity and other properties.

All reductions are from the *3-partition* problem defined as follows: Given $3n$ positive integers $a_1, \ldots, a_{3n}$ with $\sum_{i=1}^{3n} a_i = n \cdot B$, partition the numbers into n groups such that the numbers within each group sum to B. It is well-known that 3-partition is strongly NP-hard [11].

4.1 NP-hardness for disconnected graphs

So assume we are given an instance $\{a_1, \ldots, a_{3n}\}$ of 3-partition, with $\sum_{i=1}^{3n} a_i = n \cdot B$. We construct the point set P and the (disconnected) graph G as follows: The convex hull of our point set P consists of $4n + 8$ points placed on an oval (see also Figure 2.) The leftmost point of the oval is at the origin, and the rightmost point is at $(2n + 4, 0)$. There are four more points where the oval intersects the vertical line with x-coordinate $\varepsilon > 0$ and $2n+4-\varepsilon$ (where $\varepsilon > 0$ is a small constant). On the lower hull, we have points at x-coordinates $2j-\varepsilon, 2j+\varepsilon$, for $j = 1, \ldots, n+1$, we call these b_j and d_j for $j = 1, \ldots, n+1$. On the upper hull, we have points at x-coordinates $2j+1-\varepsilon, 2j+1+\varepsilon$, for $j = 1, \ldots, n$; we use b'_j and d'_j for these points.

The outer-face of graph G has $4n + 8$ vertices in a cycle. Any mapping of G on the point set P that respects the outer-face must place the outer-face of G on the convex hull of P.

Next, add a point p_a at $(\varepsilon/2, 0)$. In the graph, pick three arbitrary consecutive vertices c_1, c_2, c_3 of the outer-face and add edge (c_1, c_3). Add a new vertex v_a inside the resulting triangle $\{c_1, c_2, c_3\}$

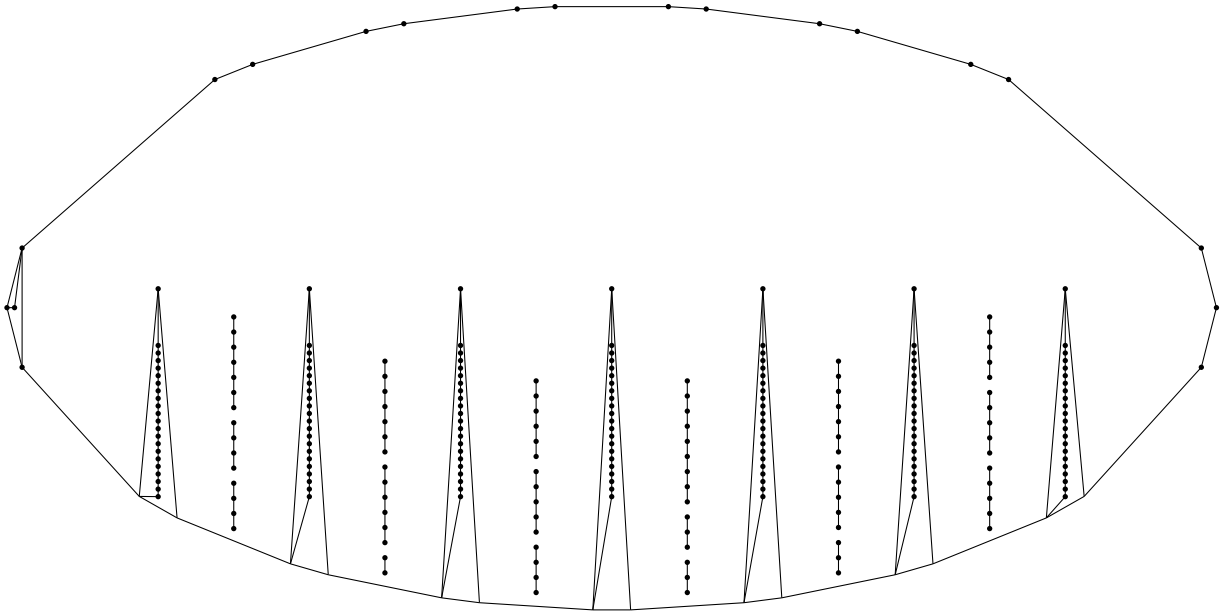

Figure 2: An example of NP-completeness construction for disconnected graphs with $n = 6$

and connect it to c_1 and c_2. Thus in any realization of G on P, vertices c_1, c_2, c_3 must be mapped to consecutive points on the convex hull such that the triangle $\{c_1, c_2, c_3\}$ contains a point. We will not add any such points *except* p_a. Therefore, in any realization c_2 must be mapped to $(0, 0)$, which fixes the mapping of the entire outer-face of G to the points of the convex hull of P.

Next, for $j = 1, \ldots, n + 1$, add $M + 1$ points on the vertical line with x-coordinate $2j$, for M sufficiently large ($M > 2n$ suffices). We call these the *jth lower tooth points*. The topmost point (which we call p_j) is above the y-axis. All other points are below the y-axis, and "sufficiently far away from the convex hull", i.e., not inside any triangle defined by three consecutive points on the convex hull. [4]

By the above vertex v_a at point p_a fixes which outer-face vertex of G is mapped to which point; hence for $j = 1, \ldots, n + 1$ let vertices b_j, d_j in G be those that are mapped to points b_j, d_j in P. We introduce a new vertex t_j that is adjacent to b_j and d_j. Inside the triangle $\{t_j, b_j, d_j\}$ place an M-path (a path with M vertices) and make one end adjacent to t_j and the other to b_j. We can realize this by placing t_j and the M-path vertices at p_j and the M points below it. Vice versa, assume we have an mapping of the graph on the point set. Then t_j must be placed so that the triangle $\{b_j, t_j, d_j\}$ contains M points. But the base of this triangle has width 2ε, so it corresponds to a line that passes through the line segment (b_j, d_j) and has M points within distance ε. When adding points to P later, we will ensure that there is no such line except the vertical line that defined the jth lower tooth. It follows that in any realization, t_j must be placed on the jth lower tooth (and it must be at p_j so that $\{b_j, t_j, d_j\}$ has M points inside.)

We complete the point set by placing B points on (or near) the vertical line with x-coordinate $2j + 1$, for $j = 1, \ldots, n$. We call these points the jth *blob*, and all points in it have negative

[4] There is no need for these points to be exactly on the line, as long as the triangle $\{p_j, b_j, d_j\}$ contains them all. In particular, points could be put in general position.

y-coordinate and are sufficiently far away from the convex hull.

We complete the graph by adding $3n$ paths inside the face adjacent to $t_1, \ldots, t_{n+1}$. The ith path has a_i vertices. The resulting graph is easily seen be outer-planar if we change the combinatorial embedding; in particular it has treewidth 2.

If the instance of 3-partition has a solution, then we can realize the graph on the point set by drawing these a_i-paths on the blobs according to the partition into groups. Vice versa, assume G can be realized on the point sets. Then all points not in the blobs are occupied by the outer-face and the teeth as discussed above. Any two blobs are separated by the line segment (b_j, t_j). So by planarity no two points on two different blobs can have an edge between them. Therefore, each of the $3n$ paths of G must be entirely realized within one blob. Since there are B points in a blob, this gives a partition of $a_1, \ldots, a_n$ into groups summing to B, as desired.

Note that our construction used exactly $|V|$ points for P. But NP-hardness holds even for a point set with $s \geq |V|$ points, for any s, as follows: Replace point p_a (the anchor point near $(0, 0)$) by $s - |V| + 1$ points that all are within the triangle $\{c_1, c_2, c_3\}$. Then v_a must be mapped to one of these points, and the triangle $\{c_1, c_2, c_3\}$ enforces that the other $s - |V|$ points cannot be used anywhere in the construction. The rest of the NP-hardness proof hence is not affected by having extra points.

4.2 Adding layers

For disconnected graphs, our decision-making gadget was to move the disconnected paths around. Trying to make the construction connected but still keeping all paths adjacent to the same face would make such moving around change the order of the vertices around the face and hence changing the embedding. So in order to make the graph connected, we must put each path into its own face. We do this by introducing lots of cycles (with the a_i-paths squeezed between them), and more teeth and other devices to force those cycles into specific locations.

First we add n *upper teeth*, one at each edge $\{b'_j, d'_j\}$. This is done exactly as for lower teeth, except that the points are placed on the vertical line with x-coordinate $2j + 1$ ($j = 1, \ldots, n$), with positive y-coordinates except the bottommost one. So there is an upper tooth between each pair of consecutive lower teeth.

Next, we add more points near what we call the *vertical anchor lines*, which have x-coordinates 1 and $2n + 3$. We place M points on each of these vertical lines. We also place (above, below, and outside) a triplet of points that encloses the M points. M is large enough that the only way to enclose M points without intersecting a tooth is to use points on the anchor lines, and not both triangles can use the same anchor line. [5] Likewise, in the graph we define two triangles, each of which has an M-path inside that connects to two vertices of the triangle. In any realization, these *anchor triangles* hence must contain points on the two anchor lines. See Figure 3.

Finally, for some $\ell \geq (3n+1)(4n+9)$, we add ℓ *layers*, which are cycles of length $4n+8$. These are all inside each other; the outermost cycle is in the face adjacent to all teeth, and the innermost cycle is around the two anchor triangles. After every $2n - 1$ such layers, we place one of the paths for a_i, $i = 1, \ldots, 3n$. To make everything connected, we connect each a_i-path to some vertex on each of the adjacent layers, we connect two consecutive layers without an a_i-path between them by adding one arbitrary edge. Finally we connect the outermost layer by adding an edge to t_1, and connect the innermost layer to the anchor triangles at diametrically opposed vertices of the layer.

[5] It suffice to have $M > B + 2\ell$, where ℓ will be defined below.

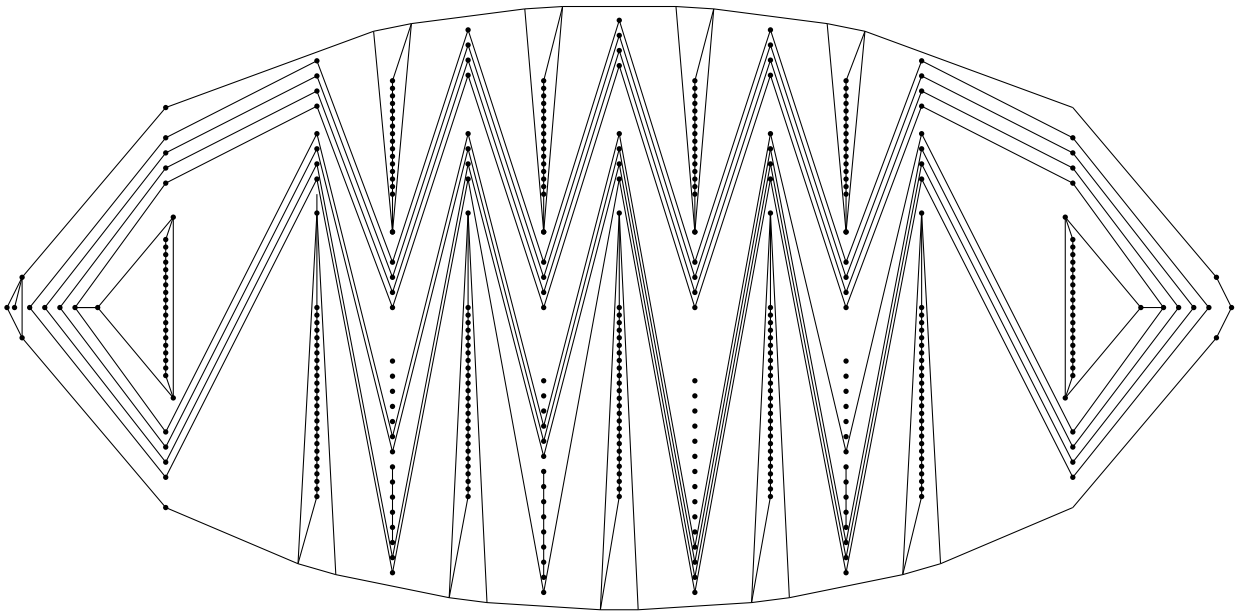

Figure 3: The construction for a connected graph.

This finishes the construction of the connected graph.

To finish the construction of the point set, we must add more points for the layers as follows: The vertical line at a lower tooth obtains 2ℓ additional points, all above the tooth. The vertical line at an upper tooth obtains 2ℓ additional points, all below the tooth. The vertical line at an anchor triangle obtains 2ℓ additional points, of which ℓ are above and ℓ are below. Finally we also add ℓ points on the x-axis to the left of the left anchor-triangle, and ℓ points on the x-axis to the right of the right anchor-triangle. This finishes the construction of the point set. Figure 3 shows only 4 of the layers (there should be at least $13 \cdot 25 = 325$ layers for $n = 4$.)

Now assume G is embedded in this point set. As explained earlier this fixates the place of the upper and lower teeth (the points for the layers cannot help, since the outer-face vertices are fixed and the triangle $\{b_j, d_j, t_j\}$ must have exactly M points in it.) Also, each anchor triangles must enclose points on the anchor line, and in particular, one anchor triangles is to the left of all teeth and the other is to the right.

The crucial observation is that any layer has both anchor triangles *inside* in the fixed combinatorial embedding. But it also must avoid all teeth by planarity. So in essence there is only one way to route a layer: Zig-zag from left to right between the teeth, go around the anchor triangle, then zig-zag from right to left between the teeth, then go around the other anchor triangle. So each of the ℓ layers uses up two points below each upper tooth, two points above each lower tooth, and one point each below, above and outside of each anchor triangle. In particular this uses up all the points, except for B points below each of the upper teeth. Hence as before, each a_i-path (for $i = 1, \ldots, 3n$) must be placed entirely below one of the upper teeth, thereby giving a solution to 3-partition.

We also need to argue that G can be placed in P if 3-partition has a solution. To do this, we route the outer-face, teeth and anchor triangles as constructed. Place each layer using the route

described above (always using the outermost point not used by an earlier layer or an a_i-path). Observe that an edge between two layers could be drawn by connecting the corresponding points of the routes (i.e., horizontally or vertically), but we also have the option of drawing this edge slanted to the next point of the next layer's route. There are at least $4n+9$ layers between a_i-paths and each layer has length $4n+8$, so we can move to any other position for the next a_i-path by shifting zero or one unit forward with each layer. We can hence place the a_{i+1}-path in the blob that corresponds to the group containing a_{i+1} in the 3-partition solution regardless of which blob had the a_i-path. The first and last $4n+9$ layers can be used to shift routes so that the connections to the outer-face (resp. anchor triangles) can be obtained regardless of the blob that contains a_1 and a_{3n}. So if 3-partition has a solution, then we can realize G in P, which finishes the NP-hardness proof for connected graphs.

4.3 Higher connectivity

We now give three different ways of how to connect layers further, depending on which properties we want to achieve.

Outer-planar 2-connected graphs: The graph as constructed is already outer-planar, and the only cutvertices occur at the connections between two consecutive layers. To make the graph 2-connected, we add a second edge between two consecutive layers (and also from the outermost layer to the outer-face and from the innermost layer to the anchor triangle). If we make sure that the two connections between layers end at two consecutive vertices of the layer, then the resulting graph is still outer-planar as illustrated in Figure 4.

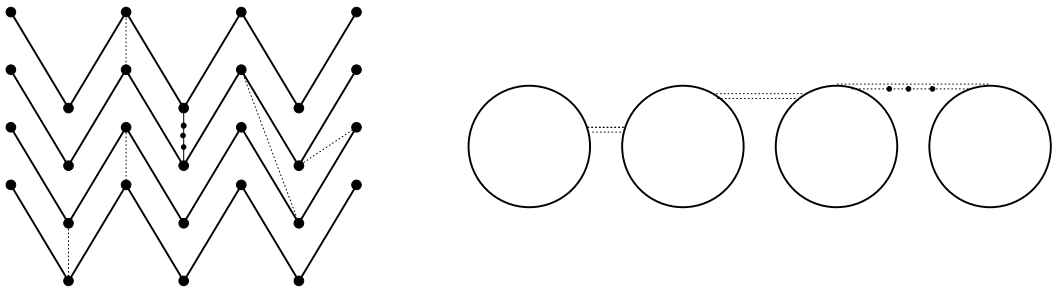

Figure 4: A close-up (not to scale) of making the graph 2-connected, and a sketch of the combinatorial embedding that achieves outer-planarity.

Inner triangulated graphs:
If we add more edges, then we make all inner faces triangles as follows:

- Fix one arbitrary vertex of the outermost layer to be placed at the leftmost point of the outermost route. This does not affect embeddability of G, since there are enough layers to shift such that the a_1-path can be in any of the blobs.

  Then triangulate any face outside the outermost layer in such a way that the edges can be drawn without creating crossings. In other words, in the fixed drawing of the outer-face, teeth and outermost layer, triangulate the polygons that represent faces and add the corresponding edges to G.

- Similarly fix the innermost layer and triangulate the faces inside it.
- Let the vertices of two consecutive layers be $v_1, \ldots, v_N$ and $w_1, \ldots, w_N$, respectively. If these layers have no a_i-path between them, then triangulate the faces between the layers by adding the zig-zag path $v_1, w_1, v_2, w_2, \ldots, v_N, w_N, v_1$. This still allows to shift the route one over in one direction: Either we draw all (v_i, w_i) vertically or horizontally and all (w_i, v_{i+1}) slanted, or vice versa. See also the bottom two layers in Figure 5.
- If the two layers have an a_i-path between them, then don't add edge (v_2, w_2); instead connect the a_i-path to v_2 and w_2, and connect all vertices on it to v_1 and w_3. The a_i-path has to be drawn vertically for this to be realizable, but we can add this restriction since there are enough layers between a_i-paths to reach the next blob already earlier.

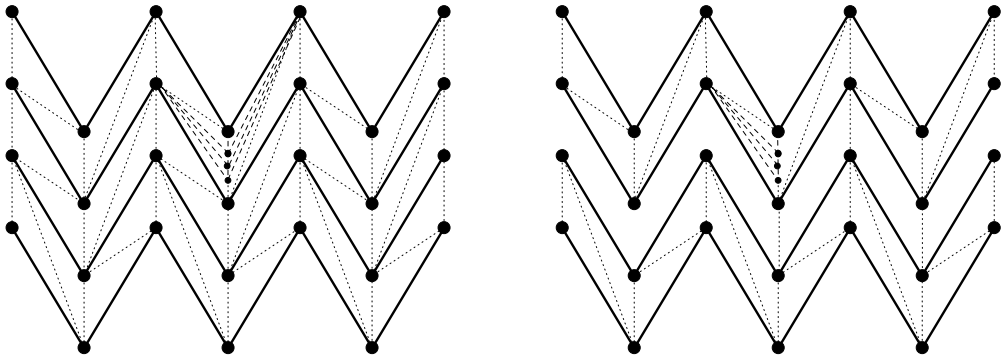

Figure 5: Making the graph (left) inner triangulated and (right) 3-connected with constant path-width.

Observe that most vertices of the graph have constant maximum degrees. Exceptions occur only at vertices that are adjacent to all the vertices of a K-path (where $K = M$ or $K = a_i$). We can avoid these high degrees by choosing points in P more carefully: Rather than placing points on a line, we place them in two not-quite parallel lines that bend away from each other,forming a)(shape. We can then triangulate the interior of the K-path and leave only a quadrangle as the outer-face, leading to degree at most 7 at all vertices. We sketch this placement in Figure 6 (which is not to scale; the points must be much closer to a line to ensure that the layers can leave exactly a_i points free between them).

Triangulated graphs: The previous construction gave an inner triangulated graph. We can triangulate the outer-face as well (see Figure 6) by adding points on circles whose convex hull contains all previously added points and each circle halves the number of points on the outer face. Stop as soon as the convex hull or the points have at most 48 points and hence at least 24 points. Extending the graph to a triangulation by making the convex hull of each layer a cycle and connecting each point to 3 consecutive points on the previous layer will give a graph with max degree 7. Now make a triangle containing all points and connect each vertex on the previous outer-face to one or two vertices of the triangle. This new graph can be drawn on the new point-set if the old graph could be drawn on the old point-set.

If we can argue that the old outer face has to be fixed at the same set of points as before we are done. Since the graph is triangulated any face could theoretically be the outer face, but since

the convex hull of all but the three outer points has at least 24 points and all but 3 vertices has degree at most 7, there is only one triangle whose deletion would leave a big enough outer face. One sees that each time one fix one layer, next layer is fixed because of the size of the convex hull of the unused points. So the problem remains hard for triangulated graphs with constant maximum degree 11(that is the degree of the 3 vertices on the outer face), even if the outer-face is not fixed.

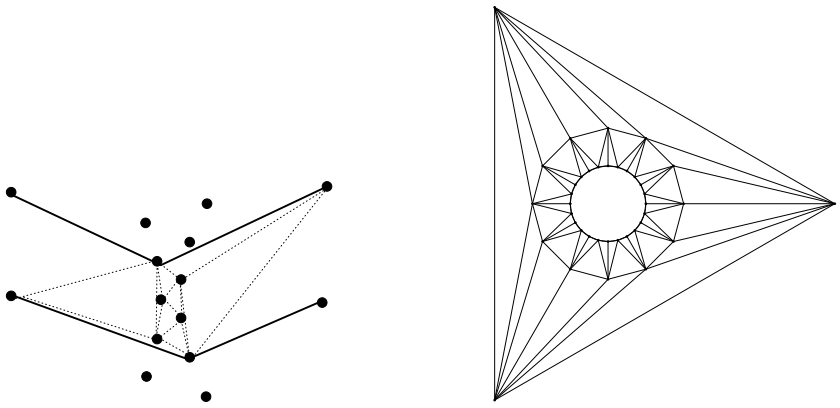

Figure 6: (Left) With carefully chosen points near a line, we can add edges within K-paths to triangulate while keeping the maximum degree small. (Right) Triangulation of the outer face; the length of the cycles is not to scale.

3-connected graphs with small treewidth: The construction of the inner triangulated graph has unbounded treewidth (since the layers and connections between them form a large grid). This is to be expected (recall that Point-set embeddability is polynomial if the treewidth is constant and the graph is triangulated). We show now that the treewidth (and even the pathwidth) can be constant if we replace "triangulated" by "3-connected."

If two layers $v_1, \ldots, v_N$ and $w_1, \ldots, w_N$ are connected by an alternating path, then the subgraph induced by them is 3-connected and has pathwidth 4 (use $v_1, w_1, \ldots, v_N, w_N$ as enumeration; then only v_1, w_1, v_i, w_i or only v_1, w_1, w_i, v_{i+1} have later neighbours for any i). If two layers have an a_i-path between them, then adding the edges between them as described above also gives a subgraph of pathwidth 4 (use w_1, v_2, v_3, the vertices of the a_i-path,w_2, $w_3, v_4, w_4, \ldots, v_N, w_N, v_1$ as enumeration). Similarly the graphs outside the outer-most layer and inside the anchor triangles can be made to have constant pathwidth by triangulating suitably.

The main idea is now to *not* triangulate the faces between every other pair of consecutive layers. Instead, we use only three edges between three consecutive vertices on each of the layers; see also Figure 5. This means that the pathwidth of the resulting graph is at most 3 more than the pathwidth of the components, which was constant. So the resulting graph has constant pathwidth. By choosing points for K-paths as before, we can even make it have constant maximum degree. [6]

All our reductions are polynomial in n and B, and we have hence proved:

Theorem 4.1. Point-set embeddability *is NP-hard, even in the following situations:*

- *For a* 2*-connected outer-planar graph if the combinatorial embedding must be respected.*
- *For a triangulated planar graph with maxiumum degree* 11*.*

[6] We leave exact determination of pathwidth and maximum degree to the reader; we estimate both to at most 7.

- *For a 3-connected planar graph with constant pathwidth and maximum degree.*

5 Conclusion

In this paper, we studied the Point-set embeddability problem, especially for the case where the combinatorial embedding of the graph is fixed (and hence previous results on NP-hardness and solving it for outer-planar graphs don't apply.) We showed that the problem is polynomial if the treewidth and the face-degrees are bounded, but becomes NP-hard as soon as one of these conditions is dropped. NP-hardness holds even for the special case of a 2-connected outer-planar (with a fixed combinatorial embedding, which doesn't have all vertices on the outer-face), or a triangulated planar graph (which hence has only one combinatorial embedding), or a 3-connected graph with constant treewidth (which also has only one combinatorial embedding.)

A few open problems remain:

- Our algorithm is polynomial if the treewidth and the face-degrees are constant, but it is not fixed-parameter tractable in these parameters. Does a fixed-parameter tractable algorithm exist, or is the problem $W[1]$-hard (say, with respect to the carving width of the dual graph)?
- What is the right kind of graph parameter to characterize when Point-set embeddability is polynomial if the combinatorial embedding is not fixed? It is not hard to see that the graph used in the reduction of Cabello [7] has treewidth 3, and one can modify it so that it has constant face-degrees. So these parameters alone do not suffice. Under what conditions (other than outer-planarity) is Point-set embeddability polynomial when the combinatorial embedding can be chosen?

References

[1] S. Arnborg, D.G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM J. Alg. Disc. Meth.*, 8:277–284, 1987.

[2] M. Badent, E. Di Giacomo, and G. Liotta. Drawing colored graphs on colored points. *Theor. Comput. Sci.*, 408(2-3):129–142, 2008.

[3] H. Bodlaender. Treewidth: algorithmic techniques and results. In *Mathematical Foundations of Computer Science (MFCS 1997)*, volume 1295 of *Lecture Notes in Computer Science*, pages 19–36. Springer-Verlag, 1997.

[4] H. L. Bodlaender and Ton Kloks. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *J. Algorithms*, 21(2):358–402, 1996.

[5] P. Bose. On embedding an outer-planar graph in a point set. *Computational Geometry: Theory and Applications*, 23(3):303–312, 2002.

[6] V. Bouchitté, F. Mazoit, and I. Todinca. Chordal embeddings of planar graphs. *Discrete Mathematics*, 273(1-3):85–102, 2003.

[7] S. Cabello. Planar embeddability of the vertices of a graph using a fixed point set is NP-hard. *Journal of Graph Algorithms and Applications*, 10(2):353–363, 2006.

[8] N. Castañeda and J. Urrutia. Straight line embeddings of planar graphs on point sets. In *Canadian Conference on Computational Geometry (CCCG '96)*, pages 312–318, 1996.

[9] I. Fáry. On straight line representation of planar graphs. *Acta Scientiarum Mathematicarum (Szeged)*, 11:229–233, 1948.

[10] H. de Fraysseix, J. Pach, and R. Pollack. How to draw a planar graph on a grid. *Combinatorica*, 10:41–51, 1990.

[11] M.R. Gary and D.S. Johnson. *Computers and Intracatablity: A Guide to the Theory of NP-completeness*. Freeman, 1979.

[12] E. Di Giacomo, W. Didimo, G. Liotta, H. Meijer, and S. Wismath. Constraint point-set embeddability of planar graphs. *Intl. Journal of Computational Geometry and Applications*, 20(5):577–600, 2010.

[13] P. Gritzmann, B. Mohar, J. Pach, and R. Pollack. Embedding a planar triangulation with vertices at specified positions. *American Mathematic Monthly*, 98:165–166, 1991.

[14] T.M. Moosa and M. Sohel Rahman. Improved algorithms for the point-set embeddability problem for plane 3-trees. In *COCOON*, volume 6842 of *Lecture Notes in Computer Science*, pages 204–212. Springer, 2011.

[15] N. Nestoridis and D. Thilikos. Square roots of minor closed graph classes, 2011. Presented at EuroComb'11.

[16] R.I. Nishat, D. Mondal, and Md. S. Rahman. Point-set embeddings of plane 3-trees. *Comput. Geom.*, 45(3):88–98, 2012.

[17] N. Robertson and P. Seymour. Graph minors. XVIII. Tree-decompositions and well-quasi-ordering. *J. Combin. Theory Ser. B*, 89(1):77–108, 2003.

[18] N. Robertson and P.D. Seymour. Graph minors. X. Obstructions to tree-decomposition. *J. Comb. Theory, Ser. B*, 52(2):153–190, 1991.

[19] W. Schnyder. Embedding planar graphs on the grid. In *ACM-SIAM Symposium on Discrete Algorithms (SODA '90)*, pages 138–148, 1990.

[20] P. D. Seymour and R. Thomas. Call routing and the ratcatcher. *Combinatorica*, 14(2):217–241, 1994.

[21] M. Sharir and E. Welzl. On the number of crossing-free matchings, (cycles, and partitions). In *Symposium on Discrete Algorithms (SODA 2006)*, pages 860–869. ACM Press, 2006.

[22] S. Stein. Convex maps. In *American Mathematical Society*, volume 2, pages 464–466, 1951.

[23] D. M. Thilikos, M. Serna, and H. L. Bodlaender. Cutwidth i: A linear time fixed parameter algorithm. *Journal of Algorithms*, 56:1–24, 2005.

[24] D. M. Thilikos, M. Serna, and H. L. Bodlaender. Cutwidth ii: Algorithms for partial ω-trees of bounded degree. *Journal of Algorithms*, 56:25–49, 2005.

[25] K. Wagner. Bemerkungen zum Vierfarbenproblem. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 46:26–32, 1936.