Morphing Planar Graph Drawings via Orthogonal Box Drawings

Therese Biedl
University of Waterloo, Canada

Anna Lubiw
University of Waterloo, Canada

Jack Spalding-Jamieson
University of Waterloo, Canada

Abstract

We give an algorithm to morph planar graph drawings that achieves small grid size at the expense of allowing a constant number of bends on each edge. The input is an n-vertex planar graph and two planar straight-line drawings of the graph on an $O(n) \times O(n)$ grid. The planarity-preserving morph is composed of $O(n)$ linear morphs between successive pairs of drawings, each on an $O(n) \times O(n)$ grid with a constant number of bends per edge. The algorithm to compute the morph runs in $O(n^2)$ time on a word RAM model with standard arithmetic operations – in particular no square roots or cube roots are required.

The first step of the algorithm is to morph each input drawing to a planar orthogonal box drawing where vertices are represented by boxes and each edge is drawn as a horizontal or vertical segment. The second step is to morph between planar orthogonal box drawings. This is done by extending known techniques for morphing planar orthogonal drawings with point vertices.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases morphing, graph morphing, linear morph, planar graph drawing, orthogonal box drawing, orthogonal drawing, algorithm

Digital Object Identifier 10.4230/LIPIcs.GD.2024.40

Related Version This work has appeared as the third author's Master's thesis.
Thesis: `https://hdl.handle.net/10012/20172`
Full Version: `https://arxiv.org/abs/2409.04074`

Funding *Therese Biedl*: Research supported by NSERC FRN RGPIN-2020-03958.
Anna Lubiw: Research supported by NSERC.

1 Introduction

Algorithms to compute a straight-line drawing of a planar n-vertex graph on an $O(n) \times O(n)$ grid have been known since the 1980's [7, 12], but it is an open problem to achieve such straight-line small-grid results for planar graph morphing. To make this precise, let P and Q be two planar straight-line drawings of a graph G that are ***compatible***, meaning that they have the same faces and the same outer face. A ***linear morph sequence*** from P to Q is a sequence of ***explicit intermediate drawings*** of G starting with P and ending with Q. By taking a ***linear morph***, i.e., a linear interpolation of vertex positions, between each successive pair of explicit intermediate drawings, we obtain a continuous ***piece-wise linear morph*** from P to Q indexed by time $t \in [0, 1]$. The morph is ***planarity-preserving*** if the drawing at every time $t \in [0, 1]$ is planar. A straight-line drawing of a graph ***lies on a grid*** if the points representing the vertices lie at grid points; in case edges are drawn as poly-lines with bends, the bends must also lie at grid points. The following problem is open:

▶ **Open Problem.** *For a planar graph G with n vertices and a compatible pair of planar straight-line drawings P and Q of G on an $O(n) \times O(n)$ grid, is there a planarity-preserving piece-wise linear morph from P to Q where each explicit intermediate drawing is a straight-line drawing on an $O(n) \times O(n)$ grid?*

The morphing algorithm of Alamdari, Angelini, Barrera-Cruz, Chan, Da Lozzo, Di Battista, Frati, Haxell, Lubiw, Patrignani, Roselli, Singla, and Wilkinson [1] – which is based on Cairns' edge contraction method – finds a planarity-preserving linear morph sequence of length $O(n)$ but with no guarantee on the grid size of the explicit intermediate drawings. In fact, the vertex coordinates are computed using cube roots on a real RAM model, and vertices may become almost coincident to imitate edge contractions. The recent morphing algorithm of Erickson and Lin [9] achieves the same bound of $O(n)$ linear morphs for the subclass of 3-connected graphs and avoids the edge-contraction paradigm by following Floater's method of interpolating the matrix of barycentric coordinates/weights. However, computing the barycentric weights requires square roots, and, even if that were avoided, the reliance on Tutte/Floater drawings means that the vertex coordinates can require $\Omega(n)$ bits of precision, and thus a grid of size $\Omega(2^n \times 2^n)$ [8]. The open problem was solved for the special case of Schnyder drawings by Barrera-Cruz, Haxell, and Lubiw [2].

Note that the Open Problem asks about the *existence* of a small-grid morph, regardless of algorithmic considerations. Of course, one would also like a fast algorithm to find the morph, and current research has been much more successful with regard to run-time than grid-size. The general morphing algorithm of Alamdari et al. [1] runs in $O(n^3)$ time. This was improved to $O(n^2 \log n)$, and to $O(n^2)$ for 2-connected graphs, by Klemz [10]. The morphing algorithm of Erickson and Lin [9], which involves solving linear systems and only applies to 3-connected graphs, runs in $O(n^{1+\omega/2})$ time, where $\omega < 2.371552$ is the matrix multiplication exponent [14]. Note that a runtime of $O(n^2)$ is optimal if all the explicit intermediate drawings must be given as output, since $\Omega(n)$ such drawings may be required [1].

We approach the Open Problem by insisting that explicit intermediate drawings lie on an $O(n) \times O(n)$ grid, but we relax the condition that edges be drawn as straight line segments, and allow a constant number of bends per edge. Our main result is as follows.

▶ **Theorem 1.** *Let G be a connected planar graph with n vertices. For a compatible pair of planar straight-line drawings P and Q of G on an $O(n) \times O(n)$ grid, there exists a planarity-preserving linear morph sequence from P to Q of length $O(n)$, where each explicit intermediate drawing lies on an $O(n) \times O(n)$ grid and has $O(1)$ bends per edge. Moreover, this sequence can be found in $O(n^2)$ time.*

This result improves the algorithm of Lubiw and Petrick [11] that finds a linear morph sequence of length $O(n^6)$ with explicit intermediate drawings on an $O(n^3) \times O(n^3)$ grid with $O(n^5)$ bends per edge and has run-time $O(n^6)$. Besides the bad bounds, the algorithm introduces bends at non-grid points, which our present result avoids.

For Theorem 1, the way we keep vertices of the explicit intermediate drawings on the grid is to impose an even stronger condition that every edge is drawn as a poly-line with all segments along grid lines (in particular, orthogonal) except for the two segments at its endpoints. The non-orthogonal segments incident to a vertex will live in an orthogonal rectangle called a ***box***. Expressed differently, the first step of the algorithm is to morph each of the input straight-line drawings to a ***planar orthogonal box drawing*** where a vertex is represented by a box, and an edge is drawn as an ***orthogonal poly-line***, a sequence of horizontal and vertical segments, joining two vertex boxes. See Figure 1. In this way, we reduce our problem to the problem of morphing planar orthogonal box drawings. In fact, we only need the special case where the input orthogonal box drawings have no bends.

Our second main result is an algorithm to find a planarity-preserving morph between two compatible planar orthogonal box drawings. For a precise definition of such a morph, see Section 1.1. When the drawings lie on a small grid with few bends, our analysis of grid size, bends, and run-time allows us to prove Theorem 1.

▶ **Theorem 2.** *Let G be a connected planar graph with n vertices. If P and Q are a compatible pair of planar orthogonal box drawings of G on an $O(n) \times O(n)$ grid with $O(1)$ bends per edge, then there exists a planarity-preserving linear morph sequence from P to Q of length $O(n)$ where each explicit intermediate drawing is an orthogonal box drawing that lies on an $O(n) \times O(n)$ grid with $O(1)$ bends. Moreover, this sequence can be found in $O(n^2)$ time.*

We prove Theorem 2 by building on algorithms to morph between planar orthogonal drawings where a vertex is drawn as a point and an edge is drawn as an orthogonal poly-line, a more limited setting since vertices must have degree at most 4. For these algorithms, n is the number of vertices plus bends. The algorithm of Biedl, Lubiw, Petrick, and Spriggs [4] uses $O(n^2)$ linear morphs and runs in $O(n^3)$ time. Van Goethem, Speckmann, and Verbeek [13] improved the number of linear morphs to $O(n)$ by performing operations simultaneously, however without a run-time analysis. We follow the two phases of Biedl et al.: (1) morph so that for each directed edge, its sequence of directions of edge segments (ignoring segment lengths) is the same in both drawings; and (2) morph between such "parallel" orthogonal graph drawings. We use the second phase as-is. However, we must do some work to extend the first phase to the setting of orthogonal box drawings, and to improve the number of linear morphs to $O(n)$ by performing many operations simultaneously as in Van Goethem et al., while keeping the grid size small and the run-time fast.

1.1 Preliminaries

Most graph drawing methods represent the vertices as points. In a ***straight line*** drawing an edge is drawn as a line segment between the vertex points, whereas in a ***poly-line*** drawing an edge is drawn as a simple poly-line – a non-self-intersecting path of line segments joined at ***bends***.

In a ***linear morph*** from drawing P to Q, the position of a ***defining point*** representing a vertex or bend is linearly interpolated between its position in P and in Q, so the point travels on a straight line at constant speed. A ***unidirectional morph*** is a special case of a linear morph where the lines along which the points move are all parallel. In a ***horizontal [vertical] morph***, these lines are horizontal [vertical].

Our algorithm works with poly-line drawings, and the algorithm may add ***degenerate bends*** at existing vertices/bends or at a grid point along an edge segment. The algorithm may also delete such degenerate bends. We separate the steps of a linear morph sequence into: steps that add/delete degenerate bends; and steps that linearly interpolate between drawings that have the same number of bends along every edge, in which case their orders of appearance along the edge in the two drawings determines their correspondence.

In an ***orthogonal point drawing*** vertices are represented by points and edges by orthogonal poly-lines. In an ***orthogonal box drawing*** a vertex is represented by a positive area ***box*** with two horizontal sides and two vertical sides, and edges are represented by orthogonal poly-lines. The point where an edge attaches to a vertex box is called a ***port***. An orthogonal box drawing is ***planar*** if there are no extraneous feature intersections: no coincident ports, no intersecting vertex boxes, no intersections between an edge and another edge, no intersection between an edge and a vertex box except at a port. One exception is that we will allow a port at a corner of a vertex box. For orthogonal box drawings, there is one more type of ***degenerate bend*** in addition to those defined above: a bend at a port.

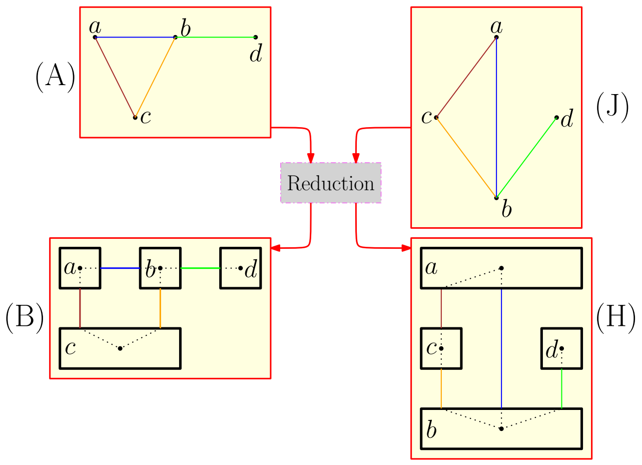


Figure 1 The two straight-line point drawings (A) and (J) are (respectively) morphed to admitted drawings of the orthogonal box drawings (B) and (H), by introducing two bends to each edge. A morph from (B) to (H) (see the relevant part of Figure 2) then induces a morph from (A) to (J).

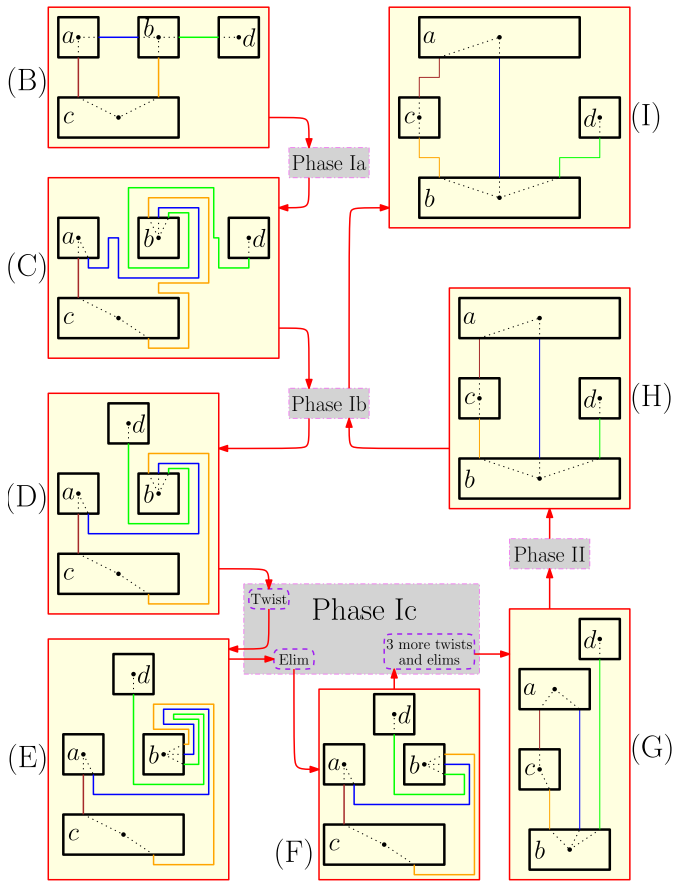


Figure 2 Morphing between orthogonal box drawings where one drawing has bends. The input drawings are (B) and (I). Phase Ia morphs (B) to (C), port-aligned with (I). Phase Ib morphs (C) to (D) and (I) to (H), eliminating zig-zags. In (D) we need 4 clockwise twists of b to match spirality with (H). Twisting once morphs (D) to (E), and eliminating zig-zags gives (F). Three more twists and zig-zag eliminations give (G), which is morphed to the parallel drawing (H) in Phase II.

We use the following relationship between box drawings and poly-line drawings. Any planar orthogonal box drawing has a corresponding planar ***admitted poly-line drawing***, where the vertex box is replaced by a vertex point at the center of the box joined by straight segments to the ports, which become bends in the edges. Thus an edge drawn with k bends in the orthogonal box drawing becomes an edge with $k + 2$ bends in the admitted drawing. Figure 2 shows extra segments for the admitted drawings with dotted lines.

A morph of an orthogonal box drawing D is specified in terms of its ***defining points*** which are the vertex box corners, the ports, and the edge bends. In a ***linear morph*** between two orthogonal box drawings the positions of the defining points are linearly interpolated. We restrict to ***structure-preserving*** linear morphs that meet the following conditions:

1. At every time during the morph the positions of the four corners of a vertex box determine a ***vertex rectangle*** which is a rectangle of positive area, though not necessarily with horizontal and vertical sides, and ports stay attached to their vertex rectangle.
2. For an edge from u to v, every segment along the edge remains orthogonal (horizontal or vertical) and does not change its direction (upward, downward, rightward or leftward). If a segment has length zero at a strictly intermediate time during the morph, then it has length zero throughout the morph.

By allowing non-axis-aligned vertex rectangles in condition (1) we can "twist" a square vertex box as shown in Figure 3[1]. The figure shows a clockwise twist where each corner moves to the position of its clockwise neighbouring corner, so two of the corners move horizontally and the other two move vertically. Ports and bends may move non-orthogonally. Twists will be defined formally in Section 4.3.1. Apart from twists, all our morphs of orthogonal box drawings will in fact be horizontal or vertical.

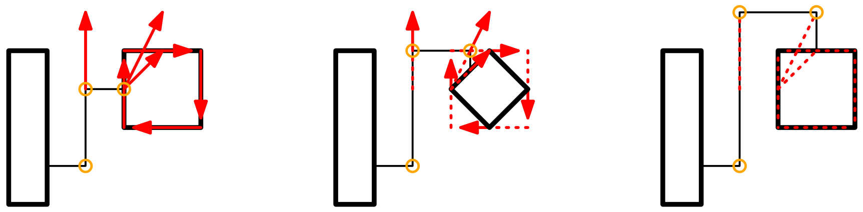


Figure 3 An example of a twist at a vertex box. Red arrows show motion to come; dashed portions show past motion.

A linear morph of an orthogonal box drawing is ***planarity-preserving*** if the drawing is planar at every time during the morph. When we say a ***planarity-preserving linear morph of an orthogonal box drawing*** we always mean that the morph is also structure-preserving. We prove in Section 3 that a planarity-preserving linear morph of an orthogonal box drawing induces a planarity-preserving linear morph of its admitted poly-line drawing.

We morph orthogonal box drawings using a sequence of intermediate goals. Two orthogonal box drawings of the same graph are ***port-aligned*** if, for each vertex v, when we compare the two vertex boxes representing v, then for every side (top, bottom, right left) the sequence of edges attached to that side of the vertex box is the same in both drawings. Two drawings without degenerate bends are ***parallel*** if they are port aligned and for every directed edge the two directed poly-lines representing the edge have the same sequence of left/right turns. To make drawings parallel, we use twists and zig-zag eliminations (to be defined), where a ***zig-zag*** is a sequence of two opposite turns, left then right, or right then left.

[1] This is similar to the "rotation" steps used to morph rectangular duals [5].

2 Algorithm Overview

This section gives the main steps of the algorithm that will prove Theorem 1. The input is a pair of compatible straight-line planar drawings of a graph G on n vertices.

▶ **Algorithm 1** (Reduction to orthogonal box drawings).

Morph each of the straight-line drawings to an admitted poly-line drawing of an orthogonal box drawing in which edges are drawn as single horizontal/vertical segments. See Figure 1 for an example. We prove in Lemma 3 that a morph of an orthogonal box drawing induces a morph of its admitted poly-line drawing, so this step reduces the problem to that of morphing orthogonal box drawings with zero bends per edge. Details of the reduction are in Section 3.

▶ **Algorithm 2** (Morphing between planar orthogonal box drawings).

The input consists of a pair of compatible planar orthogonal box drawings P and Q of a graph G on n vertices. P and Q may have bends, which is more general than the output of Algorithm 1. We describe steps of the morph "from both ends", modifying P and Q until they meet in the middle, which is legitimate since linear morphs are reversible.

Phase I: Morph P and Q to become parallel. Building on the approach of Biedl et al. [4] for the case of orthogonal point drawings, we use the following steps:

Phase Ia: Morph P to become port-aligned with Q. This adds bends in edges of P.

Phase Ib: Morph P and Q to eliminate zig-zags. (In the situation required to prove Theorem 1, Q has no bends and thus no zig-zags.)

Phase Ic: Morph P so that for every directed edge the difference between its number of left and right turns (the "spirality") is the same in P as in Q, while preserving port-alignment. This is accomplished via twists, which are interspersed with zig-zag eliminations. We use the property that if P and Q are port-aligned with matching spirality and no zig-zags or degenerate bends, then they are parallel.

Details of Phase I can be found in Section 4.

Phase II: Morph between the two parallel orthogonal box drawings. To do this, we imagine ports, corners, and bends as vertices and appeal directly to the result of Biedl et al. [4] for morphing parallel orthogonal point drawings. See the full version for details.

See Figure 2 for a sketch of all these steps. Although most of the detailed proofs that our morphs are planarity-preserving appear only in the full version, it is worth noting that, apart from twists (which are handled in Section 4.3.1) and the reduction to orthogonal box drawings, all of our morphs are horizontal or vertical, which simplifies the task. We prove in the full version that a horizontal morph between planar orthogonal box drawings P and Q is planarity-preserving so long as every horizontal line intersects the same sequence of defining points and edge/box segments.

3 Reduction to Orthogonal Box Drawings

In this section we morph a straight-line planar graph drawing to an admitted poly-line drawing of an orthogonal box drawing, and we prove that a morph between two orthogonal box drawings induces a morph between their admitted drawings. In all cases "morph" means satisfying all our conditions, as detailed below. We begin with the second result:

▶ **Lemma 3.** *Let P and Q be planar orthogonal box drawings of the same graph. Let P' and Q' be the admitted planar poly-line drawings of P and Q, respectively. Suppose there is a planarity-preserving linear morph from P to Q. Then the linear morph from P' to Q' is also planarity-preserving.*

The idea is that a linear combination is preserved throughout a linear morph, and the center of a vertex box/rectangle is a linear combination (the average) of the four corners. For details, see the full version.

We now turn to the problem of morphing a straight-line planar drawing to an admitted drawing of an orthogonal box drawing.

▶ **Theorem 4.** *Let P be a planar straight-line drawing of a graph G with n vertices drawn on an $O(n) \times O(n)$ integer grid. Then there is a planar orthogonal box drawing D of G on an $O(n) \times O(n)$ grid with no bends, and there is a planarity-preserving linear morph sequence of length $O(n)$ from P to P', the admitted poly-line drawing of D, where each explicit intermediate drawing lies on an $O(n) \times O(n)$ grid. Moreover, this sequence can be computed in $O(n^2)$ time.*

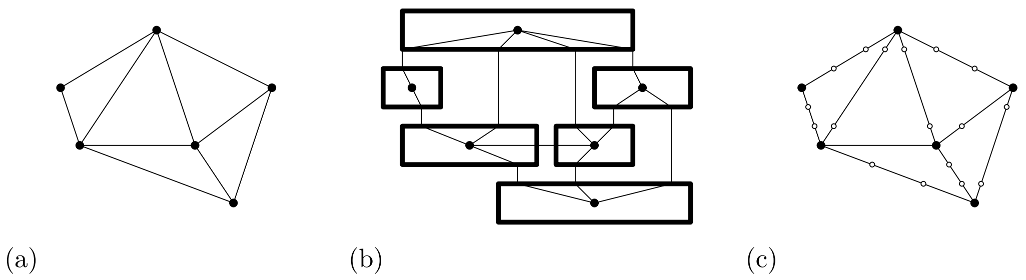


Figure 4 (a) a straight-line drawing P; (b) the corresponding orthogonal box drawing D and admitted drawing P' with every vertex at the same y-coordinate as in P; (c) adding two bends to each edge of P would allow a horizontal morph to P', but this is forbidden since the bends are not on the grid.

We prove Theorem 4 in two steps: (1) construct an orthogonal box drawing D with certain properties as described below; (2) show how to morph from P to P'.

For (1) a vertex v of P at y-coordinate c will be replaced in D by a vertex box that is skinny in the y-direction: we add a new grid line just below c and a new grid line just above c and construct a vertex box between these new grid lines. Adding the grid lines ***refines*** the grid in the y direction by a factor of 3. See Figure 4. The properties for D are that edges have no bends, each vertex has the same y-coordinate in P and in the refined grid of P', and every horizontal line intersects the same sequence of edges in P and P'. We construct D in $O(n \log n)$ time by turning an existence result due to Biedl [3] into an efficient algorithm. Her result says that any planar straight-line drawing can be redrawn so that each vertex becomes a horizontal segment (or "bar") with the same y-coordinate, and each non-horizontal edge becomes a vertical segment incident to the bars representing its endpoints, while maintaining the order of edges and vertices crossed by any horizontal line. Thickening the vertex bars to boxes gives the result we need. For details, see the full version.

For (2), we show how to morph from P to the admitted drawing P'. It would be straightforward to add degenerate bends where each edge crosses the newly added horizontal grid lines (see Figure 4(c)), and perform a single horizontal morph to move those bends to the position of the corresponding ports. However, our rule is that new bends can only be added at grid points, so we are forced to a more complicated solution. Imagine P' drawn strictly to the left of P. Order the edges and vertices of P left-to-right, with priority given to edges. One by one, in this order, pull the edges and vertices of P to their positions in P'. To handle one edge of P we add two new bends at its endpoints, and morph them to the positions of the corresponding ports. See Figure 5 for a sketch of the algorithm and the full version for more details.

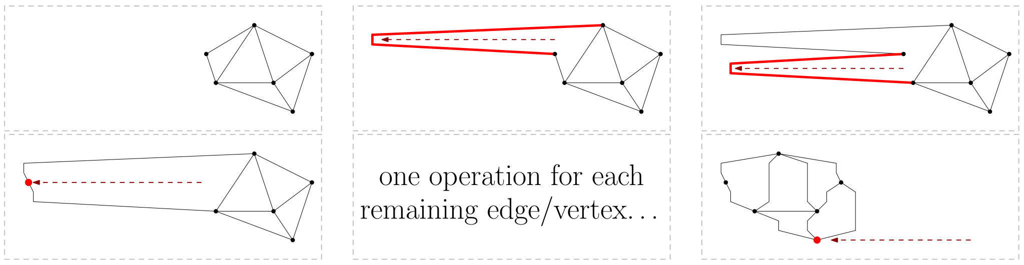


Figure 5 Morphing from a straight-line drawing on the right to (the admitted drawing of) an orthogonal box drawing on the left by moving edges/vertices one-by-one. Each pane shows the newly morphed edge/vertex in red.

4 Morphing Orthogonal Box Drawings

This section contains the algorithm to morph between orthogonal box drawings (Algorithm 2). The algorithm works for any two compatible planar orthogonal box drawings, but the bounds needed to prove our main theorem (number of linear morphs, run-time, grid size, and number of bends) apply only when the input drawings lie on an $O(n) \times O(n)$ grid and have $O(1)$ bends per edge. Informally, we call these ***small*** drawings but lemma statements are precise. Let $\boldsymbol{f(D)}$ denote the number of defining points of orthogonal box drawing D. A count of box corners, ports, and bends shows that $f(D)$ is $O(n)$ if the number of bends is $O(n)$.

Recall from Section 2 that the algorithm has two phases. Phase I morphs the drawings P and Q to become parallel, and is described in this section. Phase II morphs between the resulting parallel drawings, and is deferred to the full version, since it follows directly from Biedl et al. [4].

Phase Ia, port alignment, is in Section 4.1. Port aligned drawings are parallel if each directed edge has the same sequence of left/right turns in both drawings. Following Biedl et al. [4], we define the ***spirality*** of a directed orthogonal poly-line to be its number of left turns minus its number of right turns[2] ignoring degenerate bends. An edge drawn without zig-zags has only left turns or only right turns. Thus if the two drawings of an edge have the same spirality, no degenerate bends, and no zig-zags, then they have the same sequence of left/right turns. We eliminate zig-zags in Section 4.2 and adjust spirality in Section 4.3. A main novel contribution is the twist operation for boxes in Section 4.3.1.

4.1 Phase Ia: Port Alignment

We morph P to be port-aligned with Q by successively "turning" a port around the corner of a vertex box as shown in Figure 6. Each turn operation uses $O(1)$ planarity-preserving linear morphs, and adds 3 bends and 3 grid lines. We prove in the full version that $O(1)$ such turn

[2] This is different from the definition of spirality in [13].

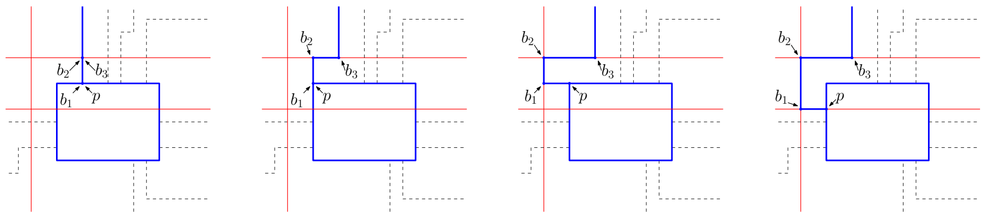


Figure 6 A sketch of how a port can be morphed around a corner by introducing 3 bends b_i and 3 new grid lines (in red).

operations per edge suffice. For small drawings, we use in total $O(n)$ linear morphs, and remain on an $O(n) \times O(n)$ grid with $O(1)$ bends per edge. The run-time is $O(n^2)$. Note that a port may be coincident with a corner of a vertex box in explicit intermediate drawings. However, after Phase Ia, we never allow a port at a corner.

4.2 Phase Ib: Zig-zag Elimination

Zig-zag elimination is used in Phases Ib and Ic. It performs a sequence of horizontal and vertical morphs to get rid of all the zig-zags in an orthogonal box drawing, and produces a drawing D with grid size $f(D)$. This is $O(n)$ if each edge has $O(1)$ bends, but the exact bound $f(D)$ will be important when we apply zig-zag elimination $O(n)$ times in Phase Ic.

In Phase Ib, we apply zig-zag elimination once to P, and, if necessary, once to Q. Note that during Phase I, the drawing Q is altered only in Phase Ib, and is not altered at all in the situation arising from Theorem 1.

A zig-zag is ***horizontal*** [***vertical***] if the segment between the two turns is horizontal [vertical]. Biedl et al. [4] were the first to show that a horizontal linear morph can be used to eliminate one horizontal zig-zag in an orthogonal point drawing. Figure 7 shows the idea as applied to an orthogonal box drawing. Van Goethem et al. [13] observed that all horizontal zig-zags in a drawing P can be eliminated simultaneously with a single horizontal linear morph to a drawing P'. Their result also applies to orthogonal box drawings, but we must improve the grid size and runtime, see below.

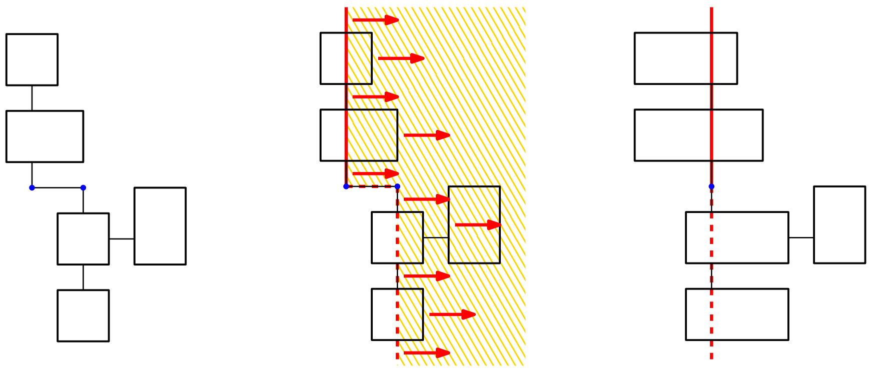
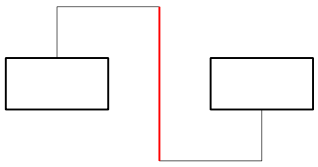


Figure 7 An orthogonal box drawing with a horizontal zig-zag, and the zig-zag elimination that converts its segment into two coinciding degenerate bends.

One could determine P' by pretending to perform $O(n)$ individual zig-zag eliminations in P, but this would be too slow and result in a large grid. Instead we compute P' via *compaction techniques*. We encode the relative x-order of all maximal vertical segments of


(a) A drawing P with a vertical zig-zag and no horizontal zig-zags.

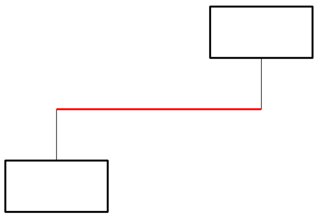

(b) A drawing P' with a horizontal zig-zag resulting from a zig-zag elimination on P.

Figure 8 An example of a vertical zig-zag elimination that results a new horizontal zig-zag.

a drawing P in an auxiliary directed acyclic graph $G_T(P)$ of size at most $f(P)$. Then we modify $G_T(P)$ to capture that the segments of horizontal zig-zags must vanish, and determine the target-drawing P' simply by computing longest paths in $G_T(P)$ to get the x-coordinates of defining points (y-coordinates remain unchanged). Graph $G_T(P)$ can be computed in $O(n \log n)$ time using a so-called trapezoidal map; the run-time can be reduced to $O(n)$ when P represents a connected graph using Chazelle's triangulation results [6]. The computed x-coordinates are at most the number of defining points of P', not counting degenerate bends (which we are about to eliminate). More details of this morph to eliminate all horizontal zig-zags are given in the full version.

By repeatedly eliminating horizontal zig-zags and then eliminating vertical zig-zags we can eliminate all zig-zags, which yields the main result of this section:

▶ **Lemma 5.** *Let G be a planar graph with n vertices. Let P be a planar orthogonal box drawing of G on an $O(n) \times O(n)$ grid with $O(1)$ bends per edge. Then there is a planarity-preserving linear morph sequence of length $O(1)$ from P to a zig-zag-free orthogonal box drawing P' with the same port alignment and edge spiralities as P such that P' is drawn on an $f(P') \times f(P')$ (hence $O(n) \times O(n)$) grid. Furthermore, during the linear morph sequence, the number of bends never increases, and the width and height of the grid may increase but never beyond $f(P)$. Finally, the linear morph sequence can be computed in $O(n)$ time.*

Proof. Repeat the following four steps (a "round") until no bends have been removed for an entire round:

1. eliminate all horizontal zig-zags with one horizontal morph as described above,
2. remove degenerate bends,
3. eliminate all vertical zig-zags with one vertical morph,
4. remove degenerate bends.

Note that we need multiple rounds in general because eliminating vertical zig-zags may create horizontal ones, and vice versa (see Figure 8). Except for the last round, each round eliminates at least one zig-zag – and thus at least two bends – from every edge that has zig-zags. No bends are ever added. Thus the number of rounds (hence also the number of morphs) is $O(1)$. Each drawing in the linear morph sequence can be computed in $O(n)$ time, so the run-time to compute all of them is also in $O(n)$.

Each morph preserves port alignment and edge spiralities. Steps 1 and 2 do not change the grid height and result in a drawing of width at most the (current) number of defining points, which is at most the initial number $f(P)$. Symmetrically, steps 3 and 4 do not change the grid width and result in a drawing of height at most the current number of defining points, which is at most $f(P)$. At the end, the final drawing P' lies on an $f(P') \times f(P')$ grid since the last round did not decrease the number of defining points. ◀

4.3 Phase Ic: Adjusting Spirality

In Phase Ic we morph P so that the spirality of each edge matches its spirality in Q. The basic operation we use is a "twist", a morph that essentially turns all the ports around a vertex box: in a clockwise twist the ports on the top of the box become ports on the right, while ports on the right become ports on the bottom, and so on around the box. (Note that doing this by turning edges around corners as in Section 4.1 would require a linear number of morphs.) We describe twists in Section 4.3.1.

How do twists adjust spirality? We will see that a clockwise twist of a vertex box adds two left turns and one right turn to all edges leaving the vertex box. This guides the calculation of how many twists to apply at each vertex box. We also need a multiple of four twists for each vertex box in order to maintain port alignment. The question of how many twists to apply and what order to apply them in is addressed in Section 4.3.2. We also keep the drawing zig-zag free, and on a small grid.

4.3.1 Performing twists

In this section we focus on how to perform one twist simultaneously on a set of vertex boxes. We need a preliminary set-up step that makes all the vertex boxes square and clears some space around them. More precisely, the ***k-proximal region*** of the vertex box of vertex v consists of all points within L_∞ distance $k \cdot \deg(v)$ from the box, where $\deg(v)$ is the degree of v (thus the k-proximal region is a rectangle formed by enlarged the vertex box by $k \cdot \deg(v)$ on each side). The vertex boxes of a drawing are ***k-spaced*** if their k-proximal regions are disjoint and the only edge segments intersecting a k-proximal region are those incident to the vertex box. With a constant number of linear morphs we can make all vertex boxes square and 2-spaced (as shown in the first pane of Figure 9), while keeping the drawing port-aligned, and preserving the spirality of each edge. See the full version.

We next describe how to take an orthogonal box drawing that has square 2-spaced vertex boxes, and perform one twist of one vertex box. The twist will only alter the drawing inside the 2-proximal region of the vertex box, so we can twist multiple vertices simultaneously, see Lemma 7 below.

A ***clockwise twist*** operation is defined as follows, see Figure 9. (Counterclockwise twists are defined analogously.) Let v be a vertex of degree $\deg(v)$ and let S be its square vertex box. In the first step of the morph we add 3 degenerate bends in each edge incident to S. The second step is the actual linear morph. We give exact coordinates for the bends added to the i^{th} edge e_i (in clockwise order) that is attached to the top side of S, say at port p_i. Other coordinates are analogous. Bend $b_{i,1}$ is placed at p_i. Bends $b_{i,2}$ and $b_{i,3}$ are placed on the edge e_i, at distance $\deg(v) + i$ above p_i.

During the linear morph, each corner of S moves to the next clockwise corner. Each port on S is a linear combination of two box corners; this is maintained throughout the morph. It remains to describe the positions of the bends. As before, we give exact coordinates only for edge e_i attached to the top side. Let p'_i be the new position for port p_i. Bend $b_{i,3}$ does not move, so $b'_{i,3} = b_{i,3}$. Bend $b_{i,1}$ moves to a point $b'_{i,1}$ that is i units to the right of p'_i. Bend $b_{i,2}$ moves horizontally to a point $b'_{i,2}$ directly above $b'_{i,1}$.

▶ **Lemma 6.** *The clockwise twist is structure-preserving and planarity-preserving.*

Proof outline. We need only focus on the 2-proximal region of S. We first show that the twist is structure-preserving. Because S is square, it remains square throughout the morph. Also, each port stays attached to its side of S. The edge segment $p_i b_{i,1}$ remains horizontal,

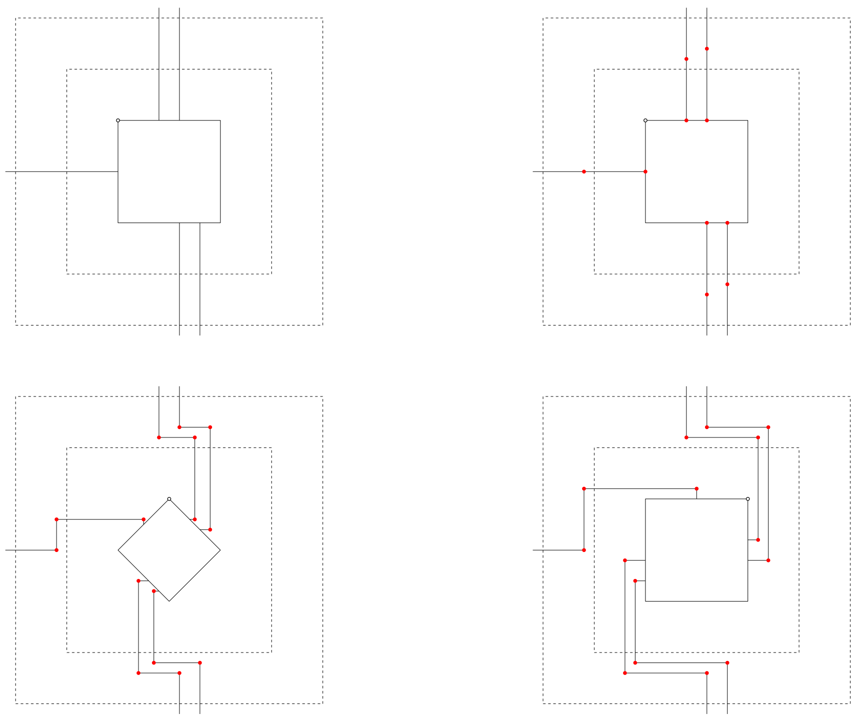


Figure 9 Performing a clockwise twist. Bends are drawn as small red circles. The top left box corner (marked with a hollow circle) moves to the top right.

starting at length zero and ending at length i. Segment $b_{i,1}b_{i,2}$ remains vertical of positive length, and segment $b_{i,2}b_{i,3}$ remains horizontal, starting at length zero and ending at positive length.

We next show that the twist is planarity-preserving. It follows from the above that edges do not intersect the vertex box during the morph. It remains to show that no two edges intersect. For edges incident to the same side (say the top side) of S, observe that: the vertical segments $b_{i,1}b_{i,2}$ maintain their left-to-right ordering; the horizontal segments $b_{i,2}b_{i,3}$ remain at constant, distinct y-coordinates, and the horizontal segments $p_ib_{i,1}$ maintain their top-to-bottom ordering. For edges incident to different sides of S, we define four disjoint subregions of the 2-proximal region of S as shown in Figure 10, and argue that edges remain in their subregion as we morph the edges and the subregions (details are in the full version). ◀

Because a twist does not change the drawing outside the 2-proximal region of a vertex box, we can indeed (as promised before) twist multiple vertices simultaneously, which immediately gives the following:

▶ **Lemma 7.** *Let G be a connected planar graph with n vertices. Let P be a planar orthogonal box drawing of G drawn on an $O(n) \times O(n)$ grid with O(1) bends per edge. Let $t : V(G) \to \{-1, 0, 1\}$ specify the twists to be performed at vertices, where -1 indicates a clockwise twist and 1 indicates a counterclockwise twist. Then there is a planarity-preserving sequence of length $O(1)$ from P to an orthogonal box drawing P' that effects the twists of t, where each explicit intermediate drawing is drawn on an $O(n) \times O(n)$ grid with $O(1)$ bends per edge. Moreover, the linear morph sequence can be computed in $O(n^2)$ time.*

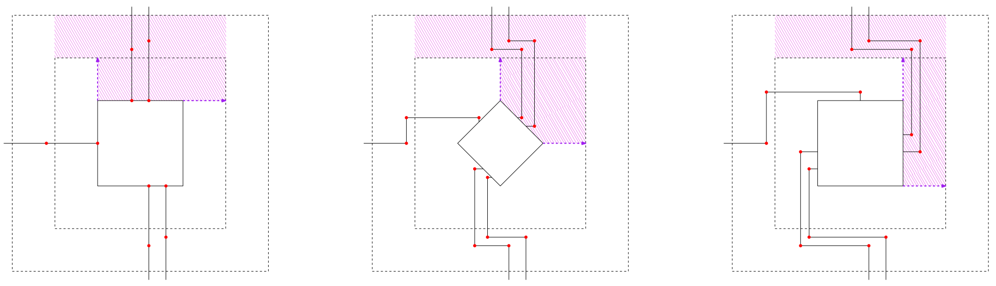


Figure 10 The subregion of the 2-proximal region of S corresponding to the top side of an initial box, throughout the morph.

4.3.2 Planning twists

There are two aspects to planning twists in P: (1) decide how many twists to perform at each vertex box; and (2) schedule the twists into rounds where each round twists each vertex box at most once, so that one round can be performed simultaneously as in Lemma 7. For (1) we solve a set of equations, the same as in Biedl et al. [4]. Step (2) is new, as are the improved bounds on the number of linear morphs and the grid size that complete Phase Ic.

Numbers of twists. We need some notation. For edge $e = uv$ and any orthogonal box drawing D, let $s_D(u, v)$ denote the spirality of the edge in D. Our goal is to morph P (while leaving Q unchanged) until for every edge uv its ***spirality difference***, $\Delta s_P(u, v) := s_P(u, v) - s_Q(u, v)$, becomes zero. The final P must also be port-aligned with Q (equivalently, port-aligned with the initial P), although this need not be true of the explicit intermediate drawings. These two conditions together constitute the requirements for our twists.

Observe that a clockwise twist (as defined in the previous section) at u's vertex box adds two left turns and one right turn to every edge leaving u, for a net change of $+1$ in the spirality of edges uv leaving u, and a net change of -1 for edges vu entering u. These are reversed for a counterclockwise twist. For example, if $\Delta s_P(u, v)$ is positive, we should apply counterclockwise twists to u and/or clockwise twists to v. For vertex v, let $t(v)$ be the number of twists to be performed on v's vertex box in P, where a positive value indicates counterclockwise twists and a negative value indicates clockwise twists. Necessary and sufficient conditions to meet the requirements for our twists are:

$$\text{for each edge } uv \quad \Delta s_P(u, v) = t(u) - t(v) \tag{1}$$

$$\text{for each vertex } v \quad t(v) \equiv 0 \bmod 4 \tag{2}$$

▶ **Lemma 8.** *A solution to the above equations with $|t(v)| \in O(n)$ can be found in $O(n)$ time.*

Proof outline. The lemma was proved by Biedl et al. [4] for orthogonal point drawings. As a main ingredient, they prove that it suffices to impose condition (2) for a single vertex v_0, and to impose condition (1) for the edges of a spanning tree T rooted at v_0. The idea is that a non-tree edge e creates a cycle with T which corresponds to a face of the drawing. Condition (1) is then proved for e using the fact that a planar orthogonal cycle has 4 more right turns than left turns when traversed clockwise. We prove that this main ingredient carries over to orthogonal box drawings – the difference is that a face of the point drawing has edges meeting at vertices, whereas a face of a box drawing contains portions of box sides. See the full version.

Given this main ingredient, we assign $t(v_0) := 0$, and assign $t(v)$ to be $t(v_0)$ plus the sum of the spirality differences on edges of the path from v_0 to v in T. This takes $O(n)$ time, and gives $t(v) \in O(n)$ since the spirality difference of each edge is constant. ◀

Allocating twists to rounds. We allocate the twists $t(v)$ to rounds r_i, where i ranges from 1 to $k := \max_v |t(v)|$ so that each vertex box is twisted at most once in each round. For vertex v, we put its twists into rounds 1 through $|t(v)|$.

Algorithm for Phase Ic. Let $P'_0 := P$. For $i = 1, \ldots, k$, perform the simultaneous twists of round r_i on input P'_{i-1} according to Lemma 7 to obtain P_i, and then do zig-zag elimination on P_i according to Lemma 5 to obtain P'_i. Output P'_k.

Analyzing the algorithm. Note that the algorithm does $O(n)$ linear morphs. It remains to bound the grid size, the number of bends, and the run-time for Phase Ic. Since zig-zag elimination is called $O(n)$ times it becomes important to give absolute bounds to avoid hiding a "growing constant" inside the $O(n)$ bound on grid size. The main idea is to analyze the number of bends, since that determines the grid size after zig-zag elimination.

▶ **Lemma 9.** *There are constants c and d, independent of i, such that P_i and P'_i, $i = 1, \ldots, k$ lie on a grid of size $cn \times cn$ with at most d bends per edge.*

Proof. We first prove a bound on bends for P'_i for $i = 0, \ldots, k$. The drawing P'_i is zig-zag-free so the number of bends on edge uv is the absolute value of its spirality, which is the same as in P_i since zig-zag elimination preserves spirality (Lemma 5). Our method of allocating twists to rounds ensures that the spirality of an edge stays the same except during a constant number of rounds where it improves. More formally:

▷ Claim 10 (Proof in the full version). After each round of simultaneous twists, the spirality of an edge remains the same or gets closer to its spirality in Q, i.e., $s_{P_i}(u,v)$ lies in the closed interval between $s_{P_{i-1}}(u,v)$ and $s_Q(u,v)$.

Thus $|s_{P_i}(u,v)| \leq \max\{|s_P(u,v)|, |s_Q(u,v)|\}$ which is bounded by some constant d' since edges in P and Q each have a constant number of bends.

We now turn to the grid size. For $i = 0, \ldots, k$ drawing P'_i is the result of applying Lemma 5 and hence its grid size is $f(P'_i) \times f(P'_i)$, where $f(P'_i)$ is the number of defining points of P'_i. Counting box corners, ports and bends, gives $f(P'_i) \leq 4n + 2m + d'm$, where G has $m \leq 3n - 6$ edges, so $f(P'_i) \leq c'n$ for some $c' \leq 10 + 3d'$. P_i is obtained from P'_{i-1} by applying one round of simultaneous twists as in Lemma 7. This may add a constant term to the bound d' on the number of bends, and may increase the bound $c'n \times c'n$ on the grid size by a constant factor, which gives the final values of d and c. ◀

We summarize Phase Ic as follows:

▶ **Lemma 11.** *Let G be a connected planar graph with n vertices. Let P and Q be compatible port-aligned zig-zag-free planar orthogonal box drawings of G on an $O(n) \times O(n)$ grid with $O(1)$ bends per edge. Then there is a planarity-preserving linear morph sequence of length $O(n)$ from P to a zig-zag-free orthogonal box drawing P' with the same port-alignment and edge spiralities as Q (thus P' and Q are parallel) such that all explicit intermediate drawings are on an $O(n) \times O(n)$ grid with $O(1)$ bends per edge. Moreover, the linear morph sequence can be computed in $O(n^2)$ time.*

5 Conclusions

We have now assembled all the ingredients for proving our two main theorems. In summary, Theorem 2 (morphing planar orthogonal box drawings) is proved by the steps of Algorithm 2: Phase Ia (port-alignment) is sketched in Section 4.1, with details in the full version; Phase Ib (zig-zag elimination in both drawings) is handled by Lemma 5; Phase Ic (spirality) is handled by Lemma 11; and Phase II (morphing parallel drawings) simply appeals to [4] (details are in the full version).

Theorem 1 (morphing straight-line drawings on a small grid) is proved by the reduction to morphing orthogonal box drawings (see Theorem 4 in Section 3) plus Theorem 2.

In conclusion, our main result is an algorithm to morph compatible planar straight-line drawings with a linear number of linear morphs for which all explicit intermediate drawings lie on small grids, have few bends per edge, and can be computed quickly. Extending the result to disconnected graphs involves further details, and is left for future work.

The biggest remaining open problem is the one from the introduction: Is there a piece-wise linear morph with small explicit intermediate drawings *without* bends? If so, can we limit to a short linear morph sequence? And if not, can we at least limit the number of bends per edge to a very small constant, such as 1 or 2?

References

1 Soroush Alamdari, Patrizio Angelini, Fidel Barrera-Cruz, Timothy M Chan, Giordano Da Lozzo, Giuseppe Di Battista, Fabrizio Frati, Penny Haxell, Anna Lubiw, Maurizio Patrignani, Vincenzo Roselli, Sahil Singla, and Bryan T Wilkinson. How to morph planar graph drawings. *SIAM Journal on Computing*, 46(2):824–852, 2017. `doi:10.1137/16M1069171`.

2 Fidel Barrera-Cruz, Penny Haxell, and Anna Lubiw. Morphing Schnyder drawings of planar triangulations. *Discrete & Computational Geometry*, 61:161–184, 2019. `doi:10.1007/s00454-018-0018-9`.

3 Therese Biedl. Height-preserving transformations of planar graph drawings. In Christian Duncan and Antonios Symvonis, editors, *Graph Drawing: 22nd International Symposium, GD 2014, Würzburg, Germany, September 24-26, 2014*, pages 380–391, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg. `doi:10.1007/978-3-662-45803-7_32`.

4 Therese Biedl, Anna Lubiw, Mark Petrick, and Michael Spriggs. Morphing orthogonal planar graph drawings. *ACM Transactions on Algorithms (TALG)*, 9(4):1–24, 2013. `doi:10.1145/2500118`.

5 Steven Chaplick, Philipp Kindermann, Jonathan Klawitter, Ignaz Rutter, and Alexander Wolff. Morphing rectangular duals. In *Graph Drawing and Network Visualization: 30th International Symposium, GD 2022, Tokyo, Japan, September 13–16, 2022*, pages 389–403. Springer, 2023. `doi:10.1007/978-3-031-22203-0_28`.

6 Bernard Chazelle. Triangulating a simple polygon in linear time. *Discrete & Computational Geometry*, 6(3):485–524, 1991. `doi:10.1007/BF02574703`.

7 Hubert De Fraysseix, János Pach, and Richard Pollack. How to draw a planar graph on a grid. *Combinatorica*, 10:41–51, 1990. `doi:10.1007/BF02122694`.

8 Giuseppe Di Battista and Fabrizio Frati. From Tutte to Floater and Gotsman: on the resolution of planar straight-line drawings and morphs. In *Graph Drawing and Network Visualization: 29th International Symposium, GD 2021, Tübingen, Germany, September 14–17, 2021*, pages 109–122. Springer, 2021. `doi:10.1007/978-3-030-92931-2_8`.

9 Jeff Erickson and Patrick Lin. Planar and toroidal morphs made easier. *Journal of Graph Algorithms and Applications*, 27(2):95–118, 2023. `doi:10.7155/jgaa.00616`.

10 Boris Klemz. Convex Drawings of Hierarchical Graphs in Linear Time, with Applications to Planar Graph Morphing. In Petra Mutzel, Rasmus Pagh, and Grzegorz Herman, editors, *29th Annual European Symposium on Algorithms (ESA 2021)*, volume 204 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 57:1–57:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. `doi:10.4230/LIPIcs.ESA.2021.57`.

11 Anna Lubiw and Mark Petrick. Morphing planar graph drawings with bent edges. *Journal of Graph Algorithms and Applications*, 15(2):205–227, 2011. URL: `https://jgaa.info/index.php/jgaa/article/view/paper223/2737`, `doi:10.7155/JGAA.00223`.

12 Walter Schnyder. Embedding planar graphs on the grid. In *SODA '90: Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 138–148, 1990. URL: `https://dl.acm.org/doi/pdf/10.5555/320176.320191`.

13 Arthur van Goethem, Bettina Speckmann, and Kevin Verbeek. Optimal morphs of planar orthogonal drawings. *Journal of Computational Geometry*, 13(1):263–297, 2022. `doi:10.20382/jocg.v13i1a11`.

14 Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou. New bounds for matrix multiplication: from alpha to omega. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3792–3835, 2024. `doi:10.1137/1.9781611977912.134`.