Morphing Planar Graphs While Preserving Edge Directions

Therese Biedl, Anna Lubiw, and Michael J. Spriggs

School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, N2L 3G1
{biedl, alubiw, mjspriggs}@uwaterloo.ca

Abstract. Two straight-line drawings P, Q of a graph (V, E) are called *parallel* if, for every edge $(u, v) \in E$, the vector from u to v has the same direction in both P and Q. We study problems of the form: given simple, parallel drawings P, Q does there exist a continuous transformation between them such that intermediate drawings of the transformation remain simple and parallel with P (and Q)? We prove that a transformation can always be found in the case of orthogonal drawings; however, when edges are allowed to be in one of three or more slopes the problem becomes NP-hard.

1 Introduction

The process of drawing a graph is rarely a one-time task devoid of prior geometric information. In many situations we already have a drawing of a graph, and the graph may change or the requirements on the drawing may change. *Dynamic graph drawing* [6] deals with the situation where the graph changes incrementally. The goals—to avoid recomputing the drawing from scratch, and to preserve the user's mental map [22]—are accomplished by altering the drawing as little as possible, which makes it straightforward to animate the changes.

There are situations however, where the graph changes more dramatically or the requirements on the drawing change, and the best approach is to compute a new drawing. Preserving the user's mental map is still desirable, but it is no longer straightforward to animate a continuous transformation from the original drawing to the new drawing [14, 15].

Transforming one geometric object to another in a continuous way is called *morphing*, and is well-studied in graphics [16], where it is often accomplished in *image space* by transforming each pixel. More appropriate for graph drawing applications are *object space* morphs, which operate on geometric objects.

In addition to the visualization applications just mentioned, morphing graph drawings also finds application in the medical imaging problem of creating a 3-dimensional model from 2-dimensional slices obtained e.g. by X-rays [2].

Morphing without maintaining geometric structure is easy but usually unhelpful. The *linear morph*, for example, moves every vertex in a straight line from its position in the source to its position in the target. It has the desirable property of making minimal changes to vertex positions, but has the undesirable property of producing intersections between disjoint objects—for example, you and your dance partner would change places by moving *through* each other.

Besides avoiding intersections, some other criteria for quality morphs are that a vertex should not stray too far from the line between its initial and final positions, and the length and direction of an edge should not deviate radically from the initial and final values. Criteria for evaluating interactive graph drawings also apply—see Bridgeman and Tamassia [7] for the case of orthogonal drawings.

Our aim in this paper is not to develop heuristics to address the many (conflicting) criteria. Rather, we concentrate on morphs that exactly preserve two properties: planarity (i.e. simplicity) and edge directions—we call these *parallel morphs*. The source and target drawings are simple straight-line drawings that represent the same graph embedded the same way, and such that each edge in the source drawing is parallel to its counterpart in the target drawing.

Our main result is an algorithm to find a parallel morph for the case of orthogonal graph drawings. The morphs produced by our algorithm are composed of $O(n)$ linear morphs where n is the size of the graphs. The user's mental model should be well preserved by these morphs. We briefly address the issue of how edge lengths change during the morph. One application of this result arises when VLSI compaction techniques [20] (which preserve edge directions) are used to reduce the area of an orthogonal drawing—our morph provides a continuous motion from the original drawing to the compacted one.

Recently, Lubiw, Petrick and Spriggs [21] devised an algorithm for morphing between two orthogonal drawings of a graph, where in these drawings vertices are points and edges are orthogonal paths. Morphs produced by the algorithm maintain both planarity and orthogonality. The algorithm employs—as a subroutine—the parallel-morphing algorithm described in the present paper.

On the negative side, we show that it is NP-hard to decide whether a parallel morph exists for the case of general planar graph drawings—in fact, in a typical 2-3 dichotomy, the problem is hard for 3 edge directions, and easy for 2.

1.1 Background

There is a broad, rich body of work on transforming one object to another while maintaining some geometric structure. Included are problems of morphing, animation, motion planning, folding, linkage reconfiguration, rigidity theory, etc. We will mention some of the most relevant background.

Preserving the Mental Map. Friedrich et al. [14, 15] considered the problem of "animating" the transformation from one graph drawing (not necessarily planar) to another. They do not insist on any geometric structure being strictly maintained, but their goal is to produce an animation that preserves the users mental map, and the criteria they formulate to accomplish this include minimizing temporary edge crossings and maintaining some minimal distance between nodes. Their method uses a combination of rigid motions and linear morphs, with the addition of clustering techniques in the second paper.

Preserving Simplicity. In 1944, long before the word “morph” was coined, Cairns [9] showed that there is a non-intersecting morph from any planar triangulation to any isomorphic one with the same fixed triangle as a boundary. Thomassen [23] strengthened this in two ways: First, he generalized to convex subdivisions and morphs preserving convexity. Secondly, he generalized to straight-line drawings of planar graphs, using the technique of “compatible triangulation” (discovered independently by Aronov et al. [1]) to augment both drawings to isomorphic triangulations, thus reducing to Cairns’ result. These results are constructive, but algorithmic issues are not explored. Although only one vertex moves at a time, the graph is contracted down to a triangle which does nothing for the user’s mental map.

Independently, Floater and Gotsman [13] proved Thomassen’s convex morphing result using an entirely different approach based on Tutte’s method of embedding graphs using barycentric coordinates. Their morph moves all vertices at once, and computes snapshots of the graph at intermediate time points. Combining this result with compatible triangulation [1] gives a different non-intersecting morph for straight line drawings [17]. These morphs can be visually pleasing, but there are no analytical results on the complexity of the vertex trajectories, or the number of time steps required to give the appearance of continuous motion. Erten, Kobourov, and Pitta [11, 12] have implemented the Floater-Gotsman method, with a preliminary phase that attempts to align the two drawings using rigid planar transformations.

Preserving Edge Directions. In addition to preserving simplicity and convexity, Thomassen [23] considered the problem of preserving edge directions. He showed that between any two simple orthogonal cycles with corresponding edges parallel, there is a parallel morph. Thomassen’s morphs shrink edges to infinitesimal lengths. Our main result in this paper generalizes Thomassen’s result to orthogonal graphs, rather than just cycles, and we do not shrink edges to infinitesimal lengths.

Thomassen’s result was extended in a different direction to the case of simple non-orthogonal cycles by Guibas et al. [19], and independently by Grenander et al. [18]. In related work we show that there exists a parallel morph between any two trees in any dimension, but not for orthogonal cycles in 3D even if they represent the trivial knot [3], and not for edge graphs of genus-0 orthogonal polyhedra in 3D [5].

We have also explored the possibility of parallel morphs that change edge lengths monotonically—the most stringent condition for nice edge-length behavior. We show [4] that such morphs are possible for convex and orthogonally convex polygons, but that the decision problem becomes NP-hard for orthogonal polygons.

Preserving Edge Lengths: Linkage Reconfiguration and Rigidity. When a morph must preserve simplicity and edge *lengths* (rather than directions) we arrive at linkage reconfiguration problems, a topic of considerable recent interest—see [10] and references therein. For connections with rigidity theory and parallel redrawings, please see [3].

2 Preliminaries

Let (V, E) be an undirected graph with vertex set V and edge set E, and let $p : V \rightarrow \mathrm{I\!R}^2$. The triple $P = (V, E, p)$ uniquely determines a bend-free straight-line *drawing* of graph (V, E) in the plane. Each edge $(u, v) \in E$ is represented in this drawing by the line segment between $p(u)$ and $p(v)$. We will use $p(u, v)$ to refer to this edge, and $|p(u, v)|$ to denote its length. A drawing $P = (V, E, p)$ is called *simple* if each vertex lies at unique coordinates and each pair of (non-equal) edges may intersect each other only at a common vertex. A drawing is *orthogonal* if each edge of the drawing is parallel with one of the axes.

Two drawings $P = (V, E, p)$ and $Q = (V, E, q)$ of the same graph are called *parallel* if for each edge $(u, v) \in E$, there exists some $\lambda > 0$ such that $p(u)-p(v) = \lambda(q(u) - q(v))$. When this expression holds for a particular edge (u, v), we say that (u, v) has the same *direction* in both P and Q.

Given two simple, parallel drawings P, Q of a graph (V, E) a *parallel morph* from P to Q is a continuous motion of the vertices that takes us from P to Q such that at all times the positions of the vertices determine a drawing of (V, E) that is both simple and parallel with P and Q. Formally, a parallel morph from P to Q is a continuously changing family of drawings R such that $R(0) = P$, $R(1) = Q$, and for every $t \in [0, 1]$, $R(t) = (V, E, r_t)$ where $r_t : V \rightarrow \mathrm{I\!R}^2$ determines a simple drawing $R(t)$ that is parallel with P and Q.

Given drawings $P = (V, E, p)$ and $Q = (V, E, q)$, the *linear morph* between them is the morph in which each vertex $v \in V$ moves continuously from $p(v)$ to $q(v)$ at constant velocity—i.e. using the notation above, $r_t(v) = tq(v)+(1-t)p(v)$ for each vertex $v \in V$. Notice, by this definition $R(0) = P$ and $R(1) = Q$. One can show easily that a linear morph between two parallel simple drawings keeps each edge parallel with its realization in $R(0)$ and $R(1)$, and changes edge-lengths monotonically. However, it may destroy simplicity. At the heart of our algorithm is the result that a linear morph does maintain simplicity in some situations: when the ordering of the coordinates of the vertices is the same in P and Q; and more generally, when P and Q are *rectangular drawings* as defined in the next section. These results are proved in [5].

3 Morphing Orthogonal Drawings

This section contains our main result—an algorithm to find a parallel morph between any two simple parallel orthogonal graph drawings that are "bend-free"—i.e. in which each edge is a single line segment.

Traditionally, an orthogonal graph drawing represents each edge as a path with bends. We find it more convenient to deal with edges that are single line segments (e.g. for defining "parallel"). Morphing of traditional orthogonal graph drawings can be achieved via our method if each edge has the same number and direction of bends in the source and target drawings—we simply replace each bend by a vertex. Henceforth, "orthogonal drawing" will mean "bend-free orthogonal drawing".

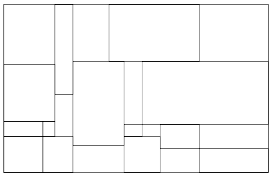

Fig. 1. A rectangular drawing

Theorem 1. *Any two simple parallel orthogonal drawings P, Q of a connected graph (V, E) admit a parallel morph that is composed of $O(|V|)$ linear morphs.*

3.1 Overview of the Morphing Algorithm

A *rectangular drawing* is a drawing in which the boundary of every face—including the outer face—is a rectangle (Fig. 1); the side of a rectangular face may be subdivided by any number of vertices. (A rectangular drawing is a type of *turn-regular* drawing as defined by Bridgeman et al. [8], i.e. no face has "kitty corners".) One can show that for a pair of parallel rectangular drawings, the linear morph is a parallel morph, i.e. it maintains both simplicity and edge directions.

So given two parallel orthogonal drawings P and Q, if they are rectangular drawings, we can morph them by applying a linear morph. Otherwise, our approach is to augment the drawings (by adding vertices, subdividing edges, and/or adding edges) to turn them into parallel rectangular drawings. Clearly, if we can morph two parallel augmented drawings, then we can also morph the original drawings by using the induced morph.

Our algorithm has three stages. The first stage ensures that the boundary of the exterior face of each drawing is a rectangle. Adding a new bounding rectangle around each drawing is easy; the only complication is maintaining connectedness of the graph and keeping the drawings parallel. In the target drawing Q, add a non-intersecting vertical edge between some vertex v and a new vertex u placed along the upper edge of the boundary rectangle. See Fig. 2 (a) and (b). We want the source drawing P to be parallel with the new target. In the source drawing, we can subdivide the upper edge of the bounding rectangle by vertex u, and position it above v, but the line segment (u, v) may cross parts of the drawing. We fix this by performing a parallel morph of the source so that (u, v) can be added, while maintaining simplicity. The fact that such a morph can always be performed on an orthogonal drawing is the key idea underlying our algorithm. Details are given in Sect. 3.2.

This completes the first stage of the algorithm. At this point, we have a new source and new target drawing. The drawings are parallel, the underlying graph is connected, and the the exterior face is bounded by a rectangle.

The second stage of the algorithm further modifies the drawings obtained in the first stage so that the boundary of each interior face is a rectangle. Until

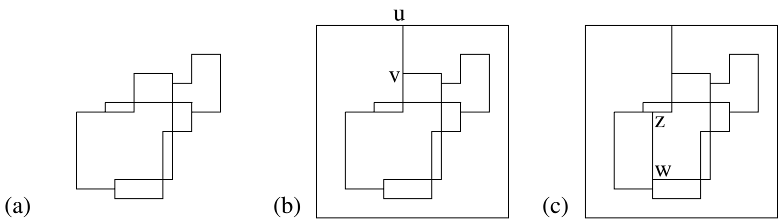

Fig. 2. Modifying the target drawing: (a) The original target. (b) The target following the first stage. (c) The target after adding an edge in the second stage.

every face of the target is a rectangle, iterate as follows. Pick a face f that is not a rectangle, and add a vertical edge from one reflex vertex w of f to the nearest edge e of f, which we subdivide by a new vertex z; see Fig. 2(c). To maintain parallel drawings, subdivide edge e by vertex z in the source drawing. Then, morph the source so that the vertical edge (w, z) can be added to it while maintaining the simplicity of the drawing; refer again to Sect. 3.2 for details.

The third stage of the algorithm is a linear morph between the rectangular source and rectangular target drawings. With that, the morph is complete.

3.2 Morphing to Add a New Edge

The first two stages of our morphing algorithm depend on the ability to morph the source drawing to a parallel drawing that admits a non-intersecting vertical edge between two given vertices. The idea is to draw a non-intersecting orthogonal path between the two vertices, and then morph the drawing (including the path) in order to straighten the path until it has no bends—at which point it forms the desired edge.

Not every orthogonal path can be straightened. Let Φ be a simple orthogonal drawing of a path. Φ is *balanced* if we encounter an equal number of left and right turns as we follow the path from one end to the other. In the remainder of this section we show that a balanced path can be straightened, and that in the above situation we can always find a balanced path between the two vertices we wish to join by an edge. Together with an analysis of the number of morphing steps, this will complete the proof of Theorem 1.

Straightening a Balanced Path. In this section we show that a balanced path of m bends can be straightened using $O(m)$ linear morphs.

Suppose that P and Φ are drawings. We define $P \cup \Phi$ in the natural way, noting that any vertex common to P and Φ must be in the same location in both drawings.

Lemma 1. *Let $P = (V, E, p)$ and $\Phi = (V_\Phi, E_\Phi, \phi)$ be simple orthogonal drawings with $v_\alpha, v_\beta \in V \cap V_\Phi$ such that Φ is a balanced drawing of a path with end-vertices v_α and v_β, and $P \cup \Phi$ is simple. There exists a parallel morph from P to a drawing $P' = (V, E, p')$ such that $p'(v_\alpha)$ and $p'(v_\beta)$ can be connected by a horizontal or vertical line segment whose interior does not intersect P'. Further,*

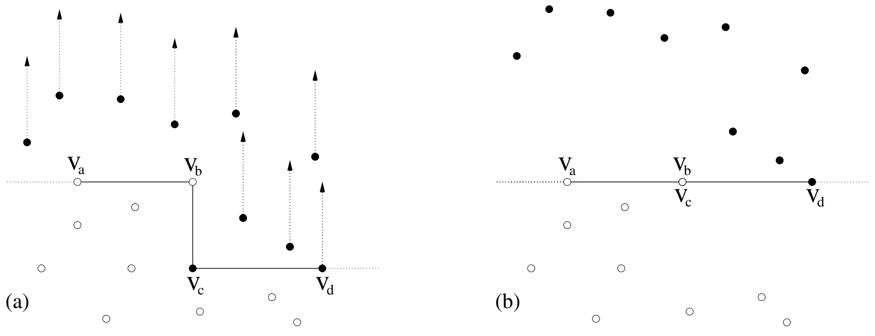

Fig. 3. The arrangement of the path vertices

the morph is composed of a sequence of $O(m)$ linear morphs, where m is the number of vertices in Φ.

Proof. As we follow Φ from v_α to v_β, we pass an equal number of left and right turns. We prove the lemma by induction on the number of left turns in Φ. If Φ contains no left turns, then it contains no right turns either and must be a line segment, and we are done. So assume that Φ contains $k > 0$ left turns. Since Φ is balanced, somewhere a left turn must be followed by a right turn or vice versa; so assume that $v_a, v_b, v_c, v_d \in V_\Phi$ is a sub-path with a right turn at v_b followed by a left turn at v_c. We will show below how to remove these two turns with a linear morph; this proves the lemma by induction.

Assume w.l.o.g. that the arrangement of $\phi(v_a)$, $\phi(v_b)$, $\phi(v_c)$, $\phi(v_d)$ is as shown in Fig. 3(a). Let $\mathcal{V} \subset V \cup V_\Phi$ be those vertices that lie either:

1. Strictly above the ray originating at $\phi(v_b)$ and going leftward; or
2. On or above the ray originating at $\phi(v_c)$ and going rightward.

The vertices in $\mathcal{V}$ are shown black in Fig. 3, while the others are drawn white.

Let R be the linear morph from $P \cup \Phi$ in which each $v \in \mathcal{V}$ moves upward at a uniform rate a distance of $|\phi(v_b, v_c)|$ while other vertices remain fixed; see Fig. 3(b). Let $P' = (V, E, p')$ denote the drawing of graph (V, E) following this linear morph. Notice, R reduces the distance between v_b and v_c to zero. Simplify the path graph (V_Φ, E_Φ) by removing v_b and v_c and adding the horizontal edge (v_a, v_d). The resulting path Φ' has one fewer left turn and one fewer right turn than Φ, so it is a balanced path between v_α and v_β with fewer than k left turns.

To complete the proof we must show that R keeps edges parallel, and—excepting vertices v_α and v_β—maintains simplicity. This is proved easily (we omit details) by observing the following properties of our morph: (1) Vertices move only vertically and upward. (2) If a vertex moves, then any vertex vertically above it (with the exception of v_b) moves by exactly the same amount. (3) By simplicity of $\Phi \cup P$, no horizontal edge of P has one vertex in $\mathcal{V}$ and the other vertex outside $\mathcal{V}$. $\square$

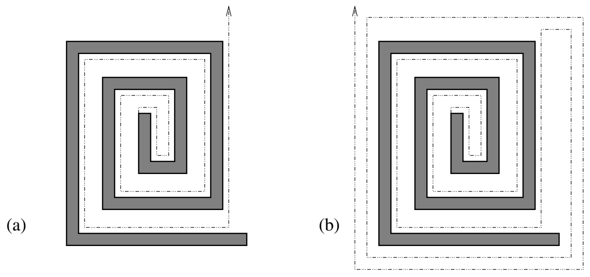

Fig. 4. A path from a vertex in P to infinity: In (a) the path has an excess of eight left turns, and in (b) the path is balanced

Finding a Balanced Path in the First Stage. Recall that for the first stage we want a balanced path in the source drawing between vertex v lying on the original outer face and a vertex u on the upper edge of the bounding rectangle. It suffices to show that we can build a simple balanced path from v that ends in an upward-directed vertical ray.

Lemma 2. *Let $P = (V, E, p)$ be an orthogonal drawing of a connected graph and let $v \in V$ be a vertex that has no incident vertical segment above it, such that the face immediately above $p(v)$ is the outer face. Drawing P admits a balanced simple path Φ of complexity $O(|V|)$ that starts at $p(v)$, goes upward, and ends with an upward-directed vertical ray.*

Proof. We construct a path that goes upward some small distance ϵ from $p(v)$ and then walks around the boundary of the outer face until we reach a point where an upward-directed ray does not intersect P. If this path is balanced we are done. Otherwise, add the appropriate number of turns of opposite direction, as illustrated in Fig. 4(b). □

Lemma 1 and Lemma 2 together prove that the first stage of the algorithm runs correctly, and is composed of $O(n)$ morphing steps.

Finding a Balanced Path in the Second Stage. We augment the target drawing by $\Theta(n)$ edges to produce a rectangular drawing, and, for each such edge, find a corresponding orthogonal path in the source which we then straighten by morphing. If we were to add the target edges in arbitrary order, each of the $\Theta(n)$ paths in the source might have $\Theta(n)$ bends to straighten, for a total of $\Theta(n^2)$ morphing steps in this stage. We can avoid this by choosing the new target edges carefully. We use only vertical edges. Each new vertical edge cuts a face in two. We choose an edge s.t. one of the new faces is a rectangle. In the source drawing, we find a balanced path with $O(1)$ turns by walking just inside the perimeter of this rectangular face. Straightening this balanced path takes $O(1)$ morphing steps, for a total of $O(n)$ morphing steps in the second stage.

This finishes the proof of Theorem 1. We note here that while only $O(n)$ linear morphs are needed, each of them might require $\Omega(n)$ time for updating the coordinates of vertices (which are needed for computing later morphs correctly). Hence the total time to perform all morphs is $O(n^2)$.

4 Edge Lengths in Morphing Orthogonal Drawings

In this section we explore how edge lengths change during parallel morphs between orthogonal drawings. There seems to be a trade-off between the number of times an edge increases and decreases in length, and the amount by which an edge deviates from its lengths in the source and target. Morphs produced by the algorithm of Sect. 3 are well-behaved with respect to the first measure, but not the second. In these morphs, each edge is non-decreasing in length until the third stage when a linear morph to the target is performed. If, prior to the final linear morph, we scale up the drawing so that every edge is longer than its target length, we obtain a two-phase morph where edges are non-decreasing in the first phase, and non-increasing in the second phase. Call this a $(+,-)$-morph. We can prove any $(+,-)$-morph will, in some cases, dramatically alter edge lengths.

For a parallel morph $R(t) = (V, E, r_t)$, define the *stretch factor* $\Delta(R)$ as:

$$\Delta(R) = \max_{(u,v)\in E} \left\{ \frac{\max_{t\in[0,1]}\{|r_t(u,v)|\}}{\max\{|r_0(u,v)|, |r_1(u,v)|\}}, \frac{\min\{|r_0(u,v)|, |r_1(u,v)|\}}{\min_{t\in[0,1]}\{|r_t(u,v)|\}} \right\} \quad (1)$$

The stretch factor is the largest factor by which some edge of the graph deviates from the range delimited by its lengths in the source and target drawings.

Theorem 2. *For any positive integer n there exists a pair of parallel orthogonal drawings with n vertices such that for any $(+,-)$-morph R between the drawings, $\Delta(R) \geq 2^{\Omega(n)}/n$.*

Due to space limitations we omit the proof, but an example of the construction is given in Fig. 5. Curiously, we have been unable to construct situations where $(-,+)$-morphs have such bad stretch factors. If we allow more fluctuations in edge lengths we can do much better.

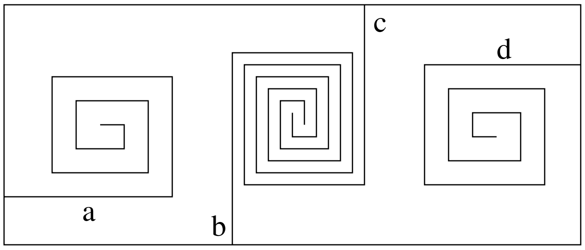

Fig. 5. The source drawing for Theorem 2. The target drawing is similar, except that spirals b and c are "disentwined" while spirals a and d are "entwined."

Theorem 3. *There exists a parallel morph R between any two simple orthogonal drawings P, Q of a graph (V, E) such that $\Delta(R) \leq n - 1$, where $n = |V|$.*

By Theorem 1 there exists a parallel morph R' from P to Q. The idea—for proving Theorem 3—is to decompose R' by a sequence of breakpoints such that between breakpoints no two vertices change order in w.r.t. a coordinate axis. The drawings at the breakpoints can be realized on nice-sized grids, and a new parallel morph R can be generated by a sequence of linear morphs between successive drawings on these nice-sized grids. Edge lengths are well-behaved on the grid, and linear morphs change edge-lengths monotonically. R has a linear stretch factor and each edge will alternately expand and shrink $O(n^3)$ times.

5 Non-orthogonal Morphing Is NP-Hard

Previous sections deal with orthogonal drawings. We now consider general drawings, and prove that it is NP-hard to decide whether parallel non-orthogonal drawings of a graph admit a parallel morph—even if there are only three possible edge directions. We note that the algorithm of Sect. 3 together with a shear can be used to morph any parallel graphs drawn using two edge directions.

Our NP-hardness reduction is from a closely related problem called *Parallel Morphing with Static Edges* (PM-Static):

- Given parallel orthogonal polygons $P = (V, E, p)$ and $Q = (V, E, q)$ and a subset $\mathcal{E} \subset E$ such that for each edge $(u, v) \in \mathcal{E}$, $|p(u, v)| = |q(u, v)|$,
- does P, Q admit a parallel morph such that all edges in $\mathcal{E}$ remain of fixed length throughout the morph?

We call the edges in $\mathcal{E}$, *static edges*, and the remaining edges of E are called *elastic edges*. The proof that PM-Static is NP-hard appears in [5]; we use a similar reduction to prove it NP-hard to decide whether two parallel orthogonal polygons admit a monotone morph [4].

Theorem 4. *Given two parallel drawings of a graph, it is NP-hard to decide whether there exists a parallel morph between them—even in the case where edges can only be horizontal, vertical, or of slope 1.*

Proof. We reduce from PM-Static. Let $P = (V, E, p)$ and $Q = (V, E, q)$ be a pair of parallel orthogonal polygons and let $\mathcal{E} \subseteq E$ be a set of static edges, whose lengths in P and Q are equal. Assume w.l.o.g. that both P and Q are embedded on a unit grid, i.e., all vertices are located at integer coordinates; one can show (details omitted) that this can be done with coordinates polynomial in $n = |V|$.

Construct a drawing P' from P as follows. Fix a value $\epsilon = (4n)^{-1}$. For each vertex $v \in V$, include a drawing of an $\epsilon \times \epsilon$-square in P', centered at $p(v)$, with a diagonal edge between the lower-left and upper-right corners. Observe that such a square permits only translation and scaling during a parallel morph.

For each edge $(u, v) \in E$, in P' connect the diagonalized squares corresponding to u and v as follows. An elastic edge of P is encoded in P' by two parallel axis-aligned edges, and a static edge is encoded by a series of diagonalized squares; see Fig. 6. The encoding of a static edge in P' permits only translation and scaling in a parallel morph, while the encoding of an elastic edge also permits changes

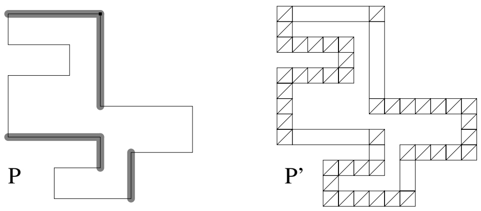

Fig. 6. An orthogonal polygon P and corresponding drawing P'

to the length of the two parallel edges. We construct Q' from Q in the same way. One can easily verify that P' and Q' are simple, and also show (details are omitted here) that they admit a parallel morph if and only if P and Q admit a parallel morph that does not change the length any static edge. □

6 Conclusion

This paper addressed the problem of morphing one planar graph drawing to another when corresponding edges have the same direction and the morph should maintain this property. We showed how to morph orthogonal graph drawings; our morphs are computationally and visually well-behaved. However, as soon we allow edges to have one of three slopes the problem becomes NP-hard. We conclude with some open problems.

The morphing algorithm of Sect. 3 works for orthogonal *point*-drawings (vertices are points), but not necessarily for orthogonal *box*-drawings (vertices are disjoint boxes that must remain of the same dimensions throughout the morph). What is the complexity of this problem? In more practical situations, corresponding edges will not be parallel in the source and target drawings. A morph should not change edge directions more than necessary. Is it possible to design morphs that minimize changes to edge directions, or to angles? Even the following is open: given two polygons, is there a non-intersecting morph between them that preserves convexity/non-convexity of angles?

References

1. B. Aronov, R. Seidel, and D. Souvaine. On compatible triangulations of simple polygons. *Computational Geometry: Theory and Applications*, 3:27–35, 1992.
2. G. Barequet and M. Sharir. Piecewise-linear interpolation between polygonal slices. *Comput. Vis. Image Underst.*, 63(2):251–272, 1996.
3. T. Biedl, A. Lubiw, and M. Spriggs. Parallel morphing of trees and cycles. In *15th Canadian Conference on Computational Geometry (CCCG)*, pages 29–32, 2003.
4. T. Biedl, A. Lubiw, and M. J. Spriggs. Angles and lengths in reconfigurations of polygons and polyhedra. *Proc. Mathematical Foundations of Computer Science (MFCS'04)*, pages 748–759, 2004.
5. T. Biedl, A. Lubiw, and Michael J. Spriggs. Angles and lengths in reconfigurations of polygons and polyhedra (long version), 2005. See *http://arxiv.org/.*

6. J. Branke. Dynamic graph drawing. In D. Kaufmann and D. Wagner, editors, *Drawing Graphs – Methods and Models, Lecture Notes in Computer Sciecne 2025*, pages 228–246. Springer, 2001.
7. S. Bridgeman and R. Tamassia. Difference metrics for interactive orthogonal graph drawing algorithms. *Journal of Graph Algorithms and Applications*, 4 (3):47–74, 2000.
8. S. S. Bridgeman, G. Di Battista, W. Didimo, G. Liotta, R. Tamassia, and L. Vismara. Turn-regularity and optimal area drawings of orthogonal representations. *Computational Geometry*, 16(1):53–93, 2000.
9. S.S. Cairns. Deformations of plane rectilinear complexes. *American Math. Monthly*, 51:247–252, 1944.
10. J. Cantarella, E.D. Demaine, H. Iben, and J. O'Brien. An energy-driven approach to linkage unfolding. In *20th Annual ACM Symposium on Computational Geometry*, pages 134–143, 2004.
11. C. Erten, S. Kobourov, and C. Pitta. Intersection-free morphing of planar graphs. In G. Liotta, editor, *Graph Drawing 2003, Lecture Notes in Computer Science 2912*, pages 320–331. Springer, 2004.
12. C. Erten, S. Kobourov, and C. Pitta. Morphing planar graphs. In *20th Annual ACM Symposium on Computational Geometry*, pages 451–452, 2004.
13. M. Floater and C. Gotsman. How to morph tilings injectively. *Journal of Computational and Applied Mathematics*, 101:117–129, 1999.
14. C. Friedrich and P. Eades. Graph drawing in motion. *Journal of Graph Algorithms and Applications*, 6 (3):353–370, 2002.
15. C. Friedrich and M.E. Houle. Graph drawing in motion II. In P. Mutzel, M. Jünger, and S. Leipert, editors, *Graph Drawing 2001, Lecture Notes in Computer Science 2265*, pages 220–231. Springer, 2002.
16. J. Gomes, L. Darsa, B. Costa, and L. Velho. *Warping and Morphing of Graphical Objects*. Morgan Kaufmann, 1999.
17. C. Gotsman and V. Surazhsky. Guaranteed intersection-free polygon morphing. *Computers and Graphics*, 25:67–75, 2001.
18. U. Grenander, Y. Chow, and D. M. Keenan. *Hands: A Pattern Theoretic Study of Biological Shapes (Appendix D)*. Springer-Verlag, 1991.
19. L. Guibas, J. Hershberger, and S. Suri. Morphing simple polygons. *Discrete and Computational Geometry*, 24(1):1–34, 2000.
20. T. Lengauer. *Combinatorial Algorithms for Integrated Circuit Layout*. Wiley, 1990.
21. A. Lubiw, M. Petrick, and Michael J. Spriggs. Morphing orthogonal planar graph drawings. In *Proceedings ACM-SIAM Symposium on Discrete Algorithms*, 2006. To appear.
22. K. Misue, P. Eades, W. Lai, and K. Sugiyama. Layout adjustment and the mental map. *J. Visual Lang. Comput.*, 6 (2):183–210, 1995.
23. C. Thomassen. Deformations of plane graphs. *Journal of Combinatorial Theory*, Series B: 34:244–257, 1983.