Morphing Planar Graph Drawings with Bent Edges

Anna Lubiw[*][†] and Mark Petrick[*][‡]

Abstract

We give an algorithm to morph between two planar drawings of a graph, preserving planarity, but allowing edges to bend during the course of the morph. The morph uses a polynomial number of elementary steps, where each elementary step is a linear morph that moves each vertex in a straight line at uniform speed. Although there are planarity-preserving morphs that do not require edge bends, it is an open problem to find polynomial-size morphs. We achieve polynomial size at the expense of edge bends.

1 Introduction

A *morph* from one drawing of a planar graph to another is a continuous transformation from the first drawing to the second that maintains planarity. Developments in the theory of morphing run parallel to the developments in graph drawing, though they lag behind. In particular, the milestones in the history of planar graph drawing are: the existence results for straight line planar drawings due to Wagner, Fáry and Stein; Tutte's algorithm to construct such drawings; and, in 1990, the polynomial time algorithms of de Frayssiex, Pach, Pollack [2] and independently Schnyder [5] to construct a straight-line drawing of an n-vertex planar graph on an $O(n) \times O(n)$ grid.

Mirroring these, the first result on morphing planar graph drawings was an existence result: between any two planar straight-line drawings there exists a morph in which every intermediate drawing is straight-line planar. This was proved for triangulations, by Cairns [1] in 1944, and extended to planar graphs by Thomassen [6] in 1983. Both proofs are constructive—they work by repeatedly contracting one vertex to another. Unfortunately, they use an exponential number of steps, and are horrible for visualization purposes since the graph contracts to a triangle and then re-emerges.

The next development was an algorithm to morph between any two planar straight-line drawings, given by Floater and Gotsman [3] in 1999 for triangulations, and extended to planar graphs by Gotsman and Surazhsky [4] in 2001. The morphs are not given by means of explicit vertex trajectories, but rather by means of "snapshots" of the graph at any intermediate time t. By choosing sufficiently many values of t, they give good visual results,

[*]David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, ON, Canada, N2L 3G1

[†]alubiw@uwaterloo.ca

[‡]mdtpetrick@uwaterloo.ca

but there is no proof that polynomially many steps suffice. Furthermore, the morph suffers from the same drawbacks as Tutte's original planar graph drawing algorithm in that there is no nice bound on the size of the grid needed for the drawings.

The history of morphing planar graph drawings has not progressed to the analogue of the small grid results of de Fraysseix et al.: *It is an open problem to find a polynomial size morph between two given drawings of a planar graph.*

In this paper we solve this problem provided that edges are allowed to bend during the course of the morph. We give a polynomial-time algorithm to find a planarity preserving morph between two drawings of a planar graph on n vertices, where the morph is composed of a sequence of $O(n^6)$ linear morphs. During the course of the morph, an edge becomes a polygonal path with $O(n^5)$ bends.

Terminology. A *planar drawing* of a graph $G = (V, E)$ assigns to each vertex $v \in V$ a distinct point $p(v)$ in the plane, and to each edge $e = (u, v)$ a path from $p(u)$ to $p(v)$ in such a way that paths intersect only at a common endpoint. A *plane graph* is one that has a planar drawing. A *combinatorial* or *planar embedding* of a plane graph is a cyclic order of edges around vertices consistent with a planar drawing. We will consider drawings in which edges are drawn as polygonal paths. A point where such a polygonal path changes direction is called a *bend*.

A *morph* from a drawing P of a graph G to a drawing Q of the graph is a continuous family of drawings $P(t)$, indexed by time $t \in [0, 1]$ where each $P(t)$ is a drawing of G, and $P(0) = P$ and $P(1) = Q$. A morph *preserves planarity* if each $P(t)$ is planar.

2 The Morphing Algorithm

We give an algorithm that takes two planar straight line drawings P and Q representing the same combinatorial embedding of a graph G, and finds a planarity-preserving edge-bending morph from P to Q using a polynomial number of elementary steps.

Conceptually, the morph is simple: if $v_1, v_2, \ldots, v_n$ are the vertices of Q ordered by x-coordinate, then we first locate v_1 in P—it must be on the outer face—and "pull it out" of the drawing until it is at the far left, allowing the edges of P to bend in compensation. We then repeat with v_2, v_3 and so on until the vertices of P appear in the same x-ordering as those of Q. If the pulling out is done with care, the edges of P will now be polygonal paths monotone in the x direction. We then perform a linear morph to adjust all vertices and bends to the correct y coordinate, thus straightening all the polygonal paths.

We now discuss the algorithm in more detail, beginning with the set-up phase. We discretize by adding a vertical line through each vertex of P. We also add n vertical lines $L_i, 1 \leq i \leq n$ to the left of the drawing of P. Line L_i will be the eventual home of vertex v_i, and when all vertices v_i reach their line L_i then the vertices of P are in the same x-order as those of Q. We plan ahead of time the route that each vertex of P will follow as we pull it to its eventual position at the left. Specifically, we augment Q with an extra vertex v_0 at the far left, and augment with extra straight-line edges so that every vertex $v_i, 1 \leq i \leq n$ is joined by some edge e_i to a vertex earlier in the ordering. We augment P to match the augmented Q and its embedding by routing each new edge as a polygonal path.

The main body of the algorithm is an iteration on $i = 1, \dots, n$. In iteration i vertex v_i is pulled along the path of e_i until it reaches line L_i. This is accomplished by repeatedly moving v_i from the vertical line on which it lies to an adjacent vertical line along the last segment of e_i. We call this the *main step* of the algorithm, and give details below. Each main step is preceded by a *straightening step* that modifies the *incoming* edges to v_i. *Incoming* and *outgoing* edges to a vertex are defined wrt the x-ordering in Q. In P, the incoming edges need not come from the left but they are contiguous in the cyclic order of edges. The straightening step, whose details we defer to the full version of the paper, enforces:

Property 1. All incoming edges to v_i enter v_i from the same side of the vertical line l through v_i. If l' is the adjacent vertical line on that side, there are no other vertices/bends along l' between the incoming edges.

The final phase of the algorithm is a linear morph from P to Q. We prove that it preserves planarity by proving that, after the main body of the algorithm, the trapezoidizations of P and Q are combinatorially the same. Details are defered.

2.1 Main Step of Algorithm

In the main step of the algorithm vertex v_i is moved from its current vertical line l to an adjacent line l' along the path of edge e_i. This step takes place repeatedly during iteration i of the algorithm. Iterations $1, \dots, i-1$ have already moved vertices $v_1, \dots, v_{i-1}$ to lines $L_1, \dots, L_{i-1}$ respectively.

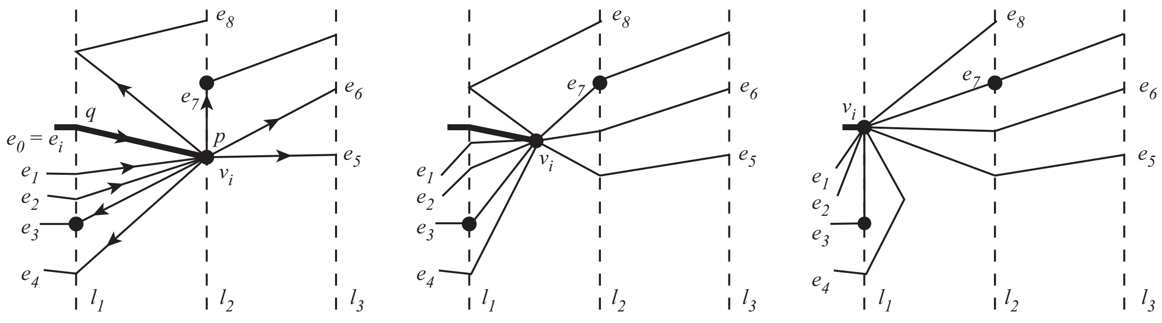

Figure 1: The main step of the algorithm moves v_i along one segment of e_i.

The operation is local, altering only the position of v_i and of bends and vertices joined to v_i by a straight segment. These bends and vertices all lie on line $l = l_2$, its predecessor l_1 and its successor l_3. Since bends and vertices only occur on vertical lines, portions of edges between adjacent vertical lines are straight, which makes it possible to use linear morphs. By Property (1) all incoming edges to v_i are contiguous and arrive from the same side; assume without loss of generality that they arrive from l_1. Also by Property (1) there are no vertices/bends along l_1 between the incoming edges. The situation is as shown in Figure 1. We morph as follows:

vertex v_i: Move vertex v_i along e_i from its current point p on l_2 to point q where e_i intersects l_1.

incoming edges: These arrive from bends on l_1. Move these bends along l_1 to q. (See edges e_1 and e_2 in Fig. 1.)

outgoing edges: There are several cases: Any edge to l_3 acquires a new bend on l_2, initially at p, and with final positions nicely spaced on l_2. (See e_5 and e_6.) There may be an edge to a vertex on l_2—leave the vertex fixed. (See e_7.) Finally, consider an edge to a bend/vertex t on l_1. If t is a bend, and *the interval along l_1 between q and t contains only bends connected to v_i* (*) then move t to q. (See e_8.) If t is a vertex and property (*) holds then t stays fixed and the edge (v_i, t) morphs to lie along l_1. (See e_3.) Otherwise (*) fails which means that there is an *intervening* bend/vertex along l_1 between q and t. In this case t stays fixed and the segment (v_i, t) morphs to a two-segment path that bends around the intervening point(s). (See e_4.) We defer further details and justification of planarity to the full paper.

Each main step of the algorithm can be accomplished via two linear morphs, the dividing point being when the new bends appear.

3 Analysis

Note that the final case of the main step of the algorithm (see the right-hand pane of Fig. 1) introduces new bends. Each bend requires a new vertical line, and each crossing of the vertical line with an existing edge counts as a new bend. This blow-up in the number of bends is potentially dangerous, since the number of main steps performed by the algorithm depends on the number of bends. To circumvent the danger, we focus on *turns*, which are bends where the path changes x-direction, and we examine more closely the straightening step and the main step of the algorithm, and argue that turns are *progagated* rather than created. Looking at the big picture, the intuition is that, since iteration i of the algorithm causes each outgoing edge of v_i to follow the path of the incoming edge e_i, thus, at worst, we copy the turns of e_i onto all the outgoing edges. We begin by bounding the number of turns introduced during the set-up phase. Further details of the following theorem are defered.

Theorem 1 *The algorithm uses $O(n^6)$ linear morphs.*

References

[1] S. S. Cairns. Deformation of plane rectilinear complexes. *American Mathematical Monthly*, 51:247–252, 1944.

[2] H. de Fraysseix, J. Pach, and R. Pollack. How to draw a planar graph on a grid. *Combinatorica*, 10(1):41–51, 1990.

[3] M. S. Floater and C. Gotsman. How to morph tilings injectively. *Journal of Computational and Applied Mathematics*, 101:117–129, 1999.

[4] C. Gotsman and V. Surazhsky. Guaranteed intersection-free polygon morphing. *Computers and Graphics*, 25:67–75, 2001.

[5] W. Schnyder. Embedding planar graphs on a grid. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, pages 138–148, 1990.

[6] C. Thomassen. Deformation of plane graphs. *Journal of Combinatorial Theory Series B*, 34:244–257, 1983.