Morphing Parallel Graph Drawings

by
Michael J. Spriggs

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2007

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

Michael J. Spriggs

I understand that my thesis may be made electronically available to the public.

Michael J. Spriggs

Abstract

A pair of straight-line drawings of a graph is called parallel if, for every edge of the graph, the line segment that represents the edge in one drawing is parallel with the line segment that represents the edge in the other drawing. We study the problem of morphing between pairs of parallel planar drawings of a graph, keeping all intermediate drawings planar and parallel with the source and target drawings. Such a morph is called a parallel morph. Parallel morphs have application to graph visualization.

The problem of deciding whether two parallel drawings in the plane admit a parallel morph turns out to be NP-hard in general. However, for some restricted classes of graphs and drawings, parallel morphability can be efficiently determined.

The main positive result is that every pair of parallel simple orthogonal drawings in the plane admits a parallel morph. We give an efficient algorithm that computes such a morph. The number of steps required in a morph produced by our algorithm is linear in the complexity of the graph, where a step involves moving each vertex along a straight line at constant speed. We prove that this upper bound on the number of steps is within a constant factor of the worst-case lower bound.

We explore the related problem of computing a parallel morph where edges are required to change length monotonically, i.e. to be either non-increasing or non-decreasing in length. Although parallel orthogonally-convex polygons will always admit a monotone parallel morph, deciding morphability under these constraints is NP-hard, even for orthogonal polygons.

We also begin a study of parallel morphing in higher dimensions. Parallel drawings of trees in any dimension will always admit a parallel morph. This is not so for parallel drawings of cycles in 3-space, even if orthogonal. Similarly, not all pairs of parallel orthogonal polyhedra will admit a parallel morph, even if they are topological spheres. In fact, deciding parallel morphability turns out to be PSPACE-hard for both parallel orthogonal polyhedra, and parallel orthogonal drawings in 3-space.

Acknowledgments

I want thank my supervisors, Anna Lubiw and Therese Biedl, for the many interesting and fruitful discussions we have had over the past few years. I thank them also for the direction they have given me throughout the PhD program. Their high standards of writing and critical thought have forced me to raise my own standards.

Thanks is due also to my committee members, Steve Wismath, Alex Lopez-Ortiz, Timothy Chan, and Jochen Konemann. Their insightful comments made the defense enjoyable, and greatly helped me to improve this thesis.

Finally, I thank the family and friends who have encouraged me along the way. Special thanks goes to my wife, Grace, for putting up with me through this long journey. Her support and patience is much appreciated, particularly during the final several months of writing. I dedicate this work to her.

Contents

List of Figures

Chapter 1

Introduction

1.1 Graph Drawing and Morphing

Graphs are combinatorial objects that are used to represent pairwise relationships between members of a set. In the most basic setting, a graph comprises a set of elements, called *vertices*, and a set of unordered pairs of vertices, called *edges*. An edge serves to connect two vertices together. For example, to represent the hyperlink structure of the World Wide Web, one might consider a graph in which a vertex is associated with each webpage, and an edge connects two vertices if and only if there is a hyperlink from one of the webpages to the other. Graphs are ubiquitous in computer science, and find application in many other areas within the natural sciences and engineering. See [22] for an introduction to graph theory.

To understand the relationships encoded by a particular graph, it is often crucial to be able to create a good visual representation of the graph. Such a representation is called a *drawing* of the graph. A common representation is one in which each vertex is associated with a point in the plane (alternatively, a disk or a box) and each edge is associated with an open non self-intersecting curve, terminating at the points that represent the vertices incident on the edge.

Generally, the curve representing an edge is either a line-segment or a polygonal chain, sometimes called a *polyline*. In the former case, the drawing is called a *straight-line* drawing. In the latter case it is called a polyline drawing. It is also common that edges be represented by polylines in which each line segment is parallel with one of the coordinate axes. Such drawings are called *orthogonal drawings* [62].

The problem of automatically generating generating good drawings of graphs has received much attention in recent years; see e.g. [62] for a survey. Graph drawing lies at

the intersection of the fields of algorithms, graph theory, and human-computer interaction. One of the challenges of graph drawing is that it is rarely straightforward to measure whether one drawing of a graph is better than another. Even when criteria for measuring the quality of a drawing have been agreed to, they are often competing. Moreover, for many criteria finding optimal drawings may be too computationally expensive for practical applications.

Several general aesthetic principles have been proposed for good drawings [5], including the following:

- Edges should cross only at a finite number of points, and the number of crossings should be minimal. No edge should cross a vertex, and no two vertices should coincide.
- The area of the drawing should be minimized, subject to a minimum allowable distance between vertices (and bends, if using polyline edges).
- The smallest angle between two edges incident at a common vertex should be maximized.

The first principle is of particular importance. Ideally, we would like drawings without edge crossings. This is not always possible. For example, a graph with five vertices such that an edge connects each pair of vertices (i.e. K_5) cannot be drawn in the plane without crossings. Graphs that can be drawn in the plane without self-intersection are called *planar*. This term is also applied to drawings: a *planar drawing* is a non self-intersecting drawing in the plane [62]. A given graph can be determined to be planar or non-planar in time that is linear in the number of vertices in the graph [39]. Also see [4, 10, 21, 24] for other planarity testing algorithms.

Considerable research has been done developing algorithms for generating planar drawings of planar graphs. It is an old result that every planar graph admits a planar straight-line drawing [25, 56, 69]. Tutte [67, 68] gave one of the earliest practical techniques for generating such drawings. In essence, Tutte's approach (called the *barycenter* method) is to fix coordinates of the vertices of one face of the drawing so that the vertices and edges of this face form a convex polygon. The position of each remaining vertex is set to be a convex combination of the neighboring vertices' coordinates. The coordinates of all vertices can be determined by solving a system of linear equalities.

Although Tutte's method is guaranteed to produce planar drawings of planar graphs, the area of such drawings may be exponential in the number of vertices, assuming a minimum allowable distance between vertices. It is possible to keep the area polynomial, how-

ever. Independently, de Fraysseix, Pach and Pollack [20] and Schnyder [53] have shown that every planar graph admits a planar drawing of $O(n^2)$ area such that each vertex has integer coordinates, where n is the number of vertices.

Work has also been done on the problem of generating planar polyline drawings of graphs. When generating such drawings, one faces dual objectives of making the drawing planar and minimizing the number of bends. A common approach to this problem is to first construct a *visibility representation* of the graph. Such a representation consists of pairwise disjoint horizontal segments and pairwise disjoint vertical segments in the plane. Two horizontal segments—each corresponding to a vertex of the graph—are *visible* to each other if a vertical segment can connect the two without intersecting any other horizontal segment. For each edge of the graph, a vertical segment connects the two mutually visible horizontal segments corresponding to the end-vertices of the edge.

A visibility representation exists for every planar graph, and can be constructed in linear time with respect to the number of vertices of the graph [6, 7, 42, 52, 63]. From a visibility representation, a planar polyline drawing can be generated with $O(1)$ bends per edge in linear time [6, 7].

Visibility representations can also be used to generate planar orthogonal drawings [64]. However, it turns out to be NP-hard to produce a planar orthogonal drawings with the fewest bends possible [30]. However, the hardness only arises when the embedding is not fixed beforehand. Once an embedding is fixed—that is, once the ordering of edges around each vertex is decided for the drawing—there exist polynomial time algorithms based on network flow techniques that generate planar orthogonal drawings with a minimum number of bends for this embedding [27, 59, 60, 61, 65].

In this thesis, we not only concern ourselves with drawings in the plane, but also with drawings in higher dimensional Euclidean spaces. For this reason, a term other than "planar" is preferable to describe drawings without self-intersection. We shall say that a drawing is *simple* if it contains no point of self-intersection.

1.1.1 Dealing with changing drawings

An interesting problem arises when we need to display a drawing that changes over time. Such changes in the drawing may be due either to changes in the underlying abstract graph, or to changing requirements on the drawing itself. Maintaining a visual representation of a changing graph drawing is the problem of *interactive*, or *dynamic*, graph drawing [49, 11, 16, 28]. In an interactive setting, a user is allowed to add and delete vertices and edges from the abstract graph in an online manner. The user might be provided some degree of

control in regard to the placement of the vertices. If edges are represented by polylines, the user might also have some control over the routing of edges.

For interactive graph drawing, it may be inappropriate to use an algorithm that computes a new drawing from scratch every time change is made to the underlying graph. This approach wastes time from a computational viewpoint, but also from the perspective of the user who must spend time studying the new drawing in order to comprehend it. Small changes made to the abstract graph may radically alter the drawing produced by a traditional graph drawing algorithm, thereby destroying the user's mental impression of the graph, or "mental map" [47]. Once destroyed, significant mental effort may be required on the part of the user to reestablish a mental map.

One approach to preserving the user's mental map is to constrain the degree to which the drawing is permitted to change as the underlying graph changes. Bridgeman and Tamassia [12] have examined several distance metrics that might be used to measure the visual similarity of two drawings of a graph. The authors introduce a framework for validating metrics, and provide some experimental analysis. Their goal is to determine the degree to which each metric corresponds to the visual distance between two drawings, as perceived by humans. If it is known which metrics best match human perception, this knowledge may be used as a guide in developing better algorithms for interactive graph drawing.

We can also use morphing to help preserve the user's mental map through changing drawings of a graph [28]. In general terms, a morph is a continuous deformation of one object to another over time, often with the constraint that some geometric structure is preserved throughout. In the context of graph drawing, a morph continuously transforms one drawing of a graph to another drawing of the same graph, such that at each stage of the transformation we have a drawing of the graph. In general, we would like intermediate drawings of the morph to exhibit certain desirable properties, e.g. simplicity.

Typically, if the current drawing of a graph is very similar to the one we want to get to, it is easy to generate a good morph between them. For the class of straight-line drawings, a morph can be described by motions of the vertices of a graph over time. If two straight-line drawings of a graph are similar, it may suffice to move all vertices linearly, taking each vertex from its source coordinates to its target coordinates at constant velocity. On the other hand, if the two drawings are allowed to differ radically, we may need a more sophisticated morph in order to clearly communicate to the viewer the correspondence between the two drawings.

Friedrich and Eades [28] suggest a number of criteria for morphs between straight-line

drawings of a graph. These criteria include:

- The motions of the vertices should be easy to follow.
- The motions of drawing should be structured, e.g. the motions should exhibit uniformity and symmetry.
- Intermediate drawings should satisfy traditional aesthetic principles of good drawings.

In order to evaluate how well a morph adheres to these criteria, Friedrich and Eades identify a number of objectives for which formal metrics may be devised, including:

- Minimize temporary edge crossings.
- Maintain a minimal distance between those vertices which do not move uniformly.
- Maximize structured movements, e.g. maximize uniform motion, maximize symmetry of motion, maximize motion that can be interpreted as movement of a three-dimensional object.
- Minimize the distance traveled by each vertex during the morph.

For some of these objectives, it is NP-hard to generate a morph that is optimal for metrics based on the objective. Often, there does not exist a morph that is optimal for more than one such metric, so generating a morph will involve a trade off between objectives.

One of the objectives discussed by Friedrich and Eades [28] that we have not already mentioned is that when we display a morph, we must take care that the speed of the morph is adequate. In order to display a morph we must obtain a sequence of intermediate drawings of the morph, which are displayed to the user in rapid succession. To the human observer, this is interpreted as movement. If too much changes between subsequent drawings, then the user will not perceive movement. If too little changes, then the user will become impatient and lose interest. We will not consider this issue further. Rather, we will consider only the problem of generating a morph itself.

1.1.2 Parallel drawings and parallel morphs

As mentioned above, Bridgeman and Tamassia [12] have studied distance metrics that might be useful in measuring the visual similarity of two drawings of a graph. The metrics

considered assume that a correspondence is given between points of two drawings, where each point is either a vertex, or a bend in one of the edges in the case that edges are represented by polylines. Each metric falls into one of the following six categories:

- **Distance metrics:** these are based on the distance between each point in one drawing and its partner in the other drawing. Two drawings are likely to be visually similar if the distances between corresponding points of the two drawings are small.
- **Proximity metrics:** these measure the relative distances between pairs of points in one drawing, and the relative distances in the other drawing. Such metrics reflect the idea that if two points are close in one drawing then they should be close in the other drawing also, in order for the drawings to be visually similar.
- **Partitioning metrics:** these are metrics based on some clustering scheme other than proximity relations.
- **Orthogonal ordering:** these are based on the angle between the vector between two points in one drawing, and the corresponding vector in the other drawing.
- **Shape metrics:** these are based on the sequence of directions we must follow as we traverse each edge from one vertex to the other in each of the drawings.
- **Topology metrics:** these are based on the ordering of edges around each vertex in each of the two drawings. Two drawings of a graph with the same topology will generally look more similar than two drawings with different topology.

In their experiments, Bridgeman and Tamassia [12] have discovered that the shape metrics perform as well as or better than most of the other examined metrics at predicting the visual similarity between two drawings of a graph—a notable exception being the orthogonal ordering metrics, which better predict the visual similarity of two drawings in their experiments. The idea behind the shape metrics is that two drawings may look similar if the edges are routed in the same way in both drawings.

In an orthogonal drawing, the shape of an edge is given by the sequence of directions (north, south, east, and west) traveled when following the representative polyline from one vertex to the other. One can define a distance between two representations of an edge based on the edit distance between the two sequences, i.e. the number of insertions, deletions and replacements needed to transform one sequence to the other (alternatively, the normalized edit distance can be used, as defined by Marzal and Vidal [46], which takes

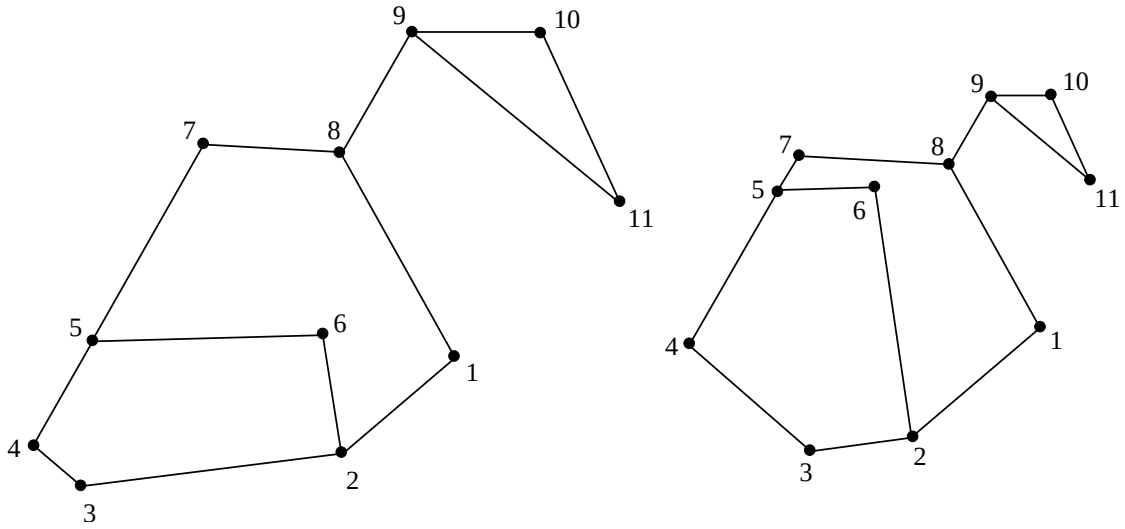

Figure 1.1: A pair of parallel simple drawings of a graph.

into account the number of segments representing the edge). The distance between two drawings is taken to be the sum of the edit distances over all edges.

Consider two orthogonal drawings of a graph for which the distance is zero, as determined by the shape metric. Each edge is given the same shape in each drawing. In essence, such drawings are what we shall call *parallel drawings* in this thesis. However, we will define parallel drawings in the context of straight-line drawings instead of polyline drawings. This is done merely for notational convenience. A pair of orthogonal drawings that have the same shape can always be converted to a pair of parallel straight-line drawings, by replacing bends with vertices.

The notion of parallel drawings can be extended to non-orthogonal drawings also. In general, two straight-line drawings of an abstract graph are *parallel* if for every edge $\{u, v\}$ in the graph, the vector from the point representing u to the point representing v in both drawings has the same direction, and both vectors have non-zero length. Figure 1.1 depicts a pair of parallel simple drawings. In this figure, vertices are represented by dots (we will sometimes omit the dots in illustrations). Vertices are numbered to illustrate the correspondence between the two drawings.

Parallel drawings of a graph will exhibit the same ordering of edges around each vertex. Thus, the distance between two parallel drawings will be zero with respect to the topology metrics described by Bridgeman and Tamassia [12]. As well, for graphs with many edges, we might expect that parallel drawings of the graph will exhibit a small distance with respect to orthogonal ordering metrics. Parallel drawings of a complete graph will have a distance

of zero with respect to an orthogonal ordering metric—recall, these metrics are based on the relative directions of corresponding edge-vectors in the two drawings. However, such drawings are not very interesting since parallel drawings of a complete graph can always be obtained from one another via scaling and translation.

Much greater flexibility occurs for drawings with a small number of edges. Even so, parallel drawings of a graph seem likely to exhibit some degree of visual similarity. Given a drawing of a graph, it may be possible to generate a new drawing that is parallel with the original drawing, such that the new drawing exhibits some desirable property that the original drawing does not. If so, we may wish to use a morph to take us from the current drawing to the new drawing. Morphing between parallel drawings is the topic of this thesis.

In order to use a morph to communicate the correspondence between two drawings, we would like the morph to preserve some of the structure of the given drawings. One way to do this is to insist that all intermediate drawings of the morph are themselves parallel with both of the drawings that we want to morph between. It is not difficult to do this: a morph between parallel drawings that moves all vertices on straight-line paths at constant velocities will have that all of its intermediate drawings are parallel with the original drawings (Lemma 2.1.1, page 27).

One of the most important of these aesthetic principles for graph drawings is to minimize edge crossings. As observed by Friedrich and Eades [28], we should attempt to adhere to the traditional aesthetics of static graph drawing in each intermediate drawing of a morph. A morph that moves each vertex in a straight-line path from its source position to its target position ignores any crossings that may occur. If the drawings that we would like to morph between are simple, then we should try to generate a morph between them that preserves simplicity. On the other hand, if the drawings are not simple, but both exhibit the same crossings on each edge and in the same order, it may be reasonable for a morph to preserve the crossings. In this case, we can replace the crossings by dummy vertices in each drawing. Again, we have the problem of morphing between a pair of parallel simple drawings of a graph.

Define a *parallel morph* as a morph that: (1) preserves simplicity, and (2) keeps all intermediate drawings parallel with the original drawing. In this thesis, we will address problems of the form: given a pair of parallel simple drawings of a graph, can we efficiently decide whether there exists a parallel morph that takes us from one drawing to the other? If so, we would like to generate such a morph. Unsurprisingly, it turns out that the most general version of this problem is computationally intractable.

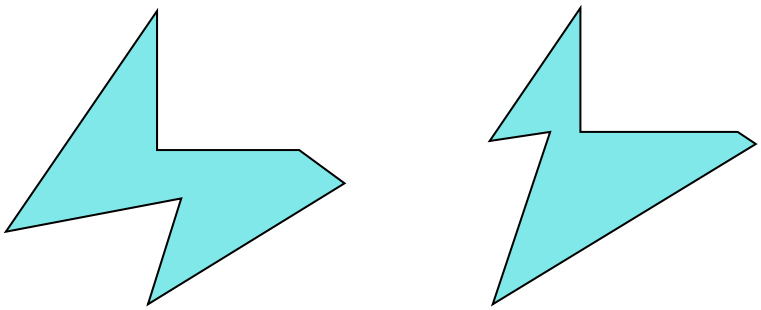

Figure 1.2: A pair of parallel polygons.

In this thesis, we will develop algorithms that generate parallel morphs between pairs of parallel simple drawings for some special classes of drawing—notably, orthogonal drawings in the plane—and establish the hardness of parallel morphing under various constraints on the input drawings. We will explore other topics pertaining to parallel morphs, including the area requirements of a parallel morph, and parallel morphs with the added constraint that edges change length monotonically.

Our exploration of parallel morphing takes us beyond the study of morphs that might be useful for the visualization of graphs. We study parallel morphing in higher dimensional space, and of polyhedra in $\mathbb{R}^3$. The question of morphability becomes one of determining the connectedness of a configuration space, i.e. is the space of all configurations of a particular type of drawing connected? If not, can we efficiently determine whether two drawings lie in a connected region of this configuration space?

We will not concern ourselves with the problem of generating a drawing that is parallel with a given drawing and has various properties that are desirable for graph visualization. Instead, we will assume that we are explicitly given the correspondence between the vertices in two drawings of a graph. Nor do we concern ourselves with the problem of determining whether two drawings are in fact parallel drawings of a graph, when not given the correspondence between vertices of the two drawings. We leave these as issues open to future study.

1.1.3 Background on parallel morphing

There exists only a small body of literature on parallel morphing. Grenander, Chow, and Keenan [33] proved that every pair of parallel polygons will admit a parallel morph; a polygon is a simple drawing of a cycle in the plane (Figure 1.2). Independently, Guibas and Hershberger [35] proved that a pair of n-vertex polygons will admit a parallel morph

composed of $O(n^{4/3+\epsilon})$ steps for any $\epsilon > 0$, where each step is a uniform scaling and translation of a part of the polygon. Hershberger, and Suri [38] improved this bound to $O(n \log n)$ (also see [34]). Earlier, Thomassen [66] had proved that every pair of parallel orthogonal polygons will admit a parallel morph. These algorithms all suffer from the defect that edges in intermediate drawings may shrink to infinitesimal lengths. This fact makes the algorithms unsuitable for graph visualization.

Closely related to the subject of parallel morphing is the recent work of Streinu [58], who studies *kinetic graphs*, which are straight-line drawings defined over a set of moving vertices. In her study, vertices are assumed to move with constant velocities. A *parallel redrawing graph* is defined as a kinetic graph in which each edge maintains its slope throughout the motion. Of particular interest are planar graphs that maintain non-crossing edges throughout the motion. Unlike our notion of *parallel*, edges are allowed to shrink to zero length, and to reverse direction. Streinu gives a characterization of this class of kinetic graphs.

Although only a small number of papers exist that directly address the topic of parallel morphing, there are connections between parallel morphing and a wide range of other research. We will survey some of these in Section 1.2. In Section 1.3 we will formally define the terms to be used throughout this thesis, and give a few basic lemmas. In Section 1.4 we will overview the contents of each chapter.

1.2 Other Related Work

In this section, we survey work that is closely related to parallel morphing.

Morphing to maintain simplicity

Most morphing algorithms that assume a specified correspondence between the source and target (as we do) are not able to maintain simplicity during the morph. For a general survey on morphing see [31]. For a more theoretical perspective, see the relevant section of [1].

The earliest result is due to Cairns [13], who showed in 1944 that there exists a simplicity-preserving morph from any planar triangulation to any isomorphic one with the same fixed triangle as a boundary. The result was strengthened in 1983 by Thomassen [66] in two ways. First, he showed that any planar subdivision with convex faces can be morphed to any isomorphic one with the same fixed boundary via a morph that preserves convexity. Secondly, he showed that any straight line graph has a non-intersecting morph

to any other straight line graph that is isomorphic and embedded with the same oriented faces. He proved this second result by arguing that the source and target graphs can be simultaneously augmented to isomorphic triangulations with a fixed boundary, a result that is of interest in its own right, and is now known as "compatible triangulation", due to Aronov, Seidel and Souvaine [3]. Both Cairns's and Thomassen's results are constructive, but algorithmic issues are not explored. Thomassen's morphs move only one vertex (or a cluster of vertices that have been contracted together) at a time, along a straight line.

Independently, Floater and Gotsman [26] proved Thomassen's first result (on convex morphing) using an entirely different approach based on Tutte's graph embedding method. In their morph, all vertices move at once, and what can be computed is not the individual trajectories of the vertices, but snapshots of the graph at any intermediate time during the morph. Gotsman and Surazhsky [32, 55] then combined this result with the compatible triangulation result from [3] to show that simple polygons have non-intersecting morphs, thus re-proving Thomassen's second result with a different morphing technique.

Efrat, Har-Peled, Guibas, and Murali [23] have studied the problem of morphing between two simple polygonal open chains in the plane while preserving simplicity. Their idea is to first form a polygon in the plane that has both the source and the target chains on its boundary. Each vertex is routed in the morph along the shortest path through the interior of the polygon to its target.

Recently, Iben, O'Brien, and Demaine [41] have presented an algorithm for generating a simplicity-preserving morph between two simple polygons, such that a correspondence is given between the vertices of the polygons. This algorithm is based on a gradient descent method for minimizing an energy function. The flexibility in the choice of energy function gives the user a good degree of control as to the appearance of a morph.

Parallel redrawings, rigidity and linkage reconfiguration

As defined by Whiteley [70] a drawing of a graph has a *parallel redrawing* if the vertices can be moved such that all edges remain parallel to those in the original drawing, and the resultant drawing is neither a translation nor a scaling of the original. Here, maintaining the simplicity of the drawing is not an issue.

Whiteley [70], and Servatius and Whiteley [54] study questions of the *existence* of a parallel redrawing. This turns out to be directly related to questions in rigidity theory. In rigidity theory, frameworks are composed of rigid bars, idealized as straight-line edges and attached at vertices. Where two bars meet at a vertex, the angle between them is allowed to change freely. The fundamental problem is that of deciding whether or not a framework

is *rigid*, that is, whether there is a non-trivial *infinitesimal motion* that moves the vertices while keeping the lengths of the bars fixed. Simplicity is not an issue. To contrast the two situations: in parallel redrawings edge lengths may change but angles are fixed, whereas in rigidity theory, edge lengths are fixed and angles may change. The answers are the same however: a configuration has a parallel redrawing if and only if it is not rigid, since the vectors orthogonal to an *infinitesimal motion* provide a parallel redrawing. See [54].

Linkage reconfiguration problems also deal with rigid bars and flexible angles, however—in contrast with rigidity theory—simplicity must be maintained, and the questions are about reachability: given two structures that are composed of the same set of rigid bars and with the same combinatorial structure, can we morph from one to the other preserving incidence relationships and the lengths of the bars?

Linkage reconfiguration can be done for a limited class of graphs: between any two simple open chains in the plane with corresponding edges of the same length, there exists a transformation that preserves simplicity and edge lengths [18, 57]. The existence result does not immediately translate into an algorithm. A practical approach to performing the transformation is given in [14]. This approach involves reformulating the problem to the problem of minimizing a suitable energy function. These results extend to cycles (or equivalently, polygons) but not to trees [8].

There is a clear relationship between this pair of problems, and the morphing problems discussed above: linkage reconfiguration problems are to rigidity theory as parallel morphs are to parallel redrawings. In both cases we impose simplicity, and we ask questions of reachability between a pair of configurations, rather than about infinitesimal motions of a single configuration. Although the equivalence between rigidity theory and parallel redrawings does not carry over, this relationship seems tantalizing.

Knot theory

A *knot* is defined as a closed, non-self-intersecting curve embedded in $\mathbb{R}^3$. Two knots are equivalent if one knot can be transformed to the other by continuous deformation without self-intersection. Deciding whether two knots are equivalent is a central problem of *knot theory* (see [50] for an introduction). The complexity of deciding knot equivalence has not yet been completely determined. It has been shown that the problem is in PSPACE [36]. A related problem, that of deciding whether a knot can be deformed to lie in a plane, is in NP. There exist algorithms for both of these problems with running times that are exponential with respect to the number of crossings in an orthogonal projection of the knot(s) [36].

Suppose that we are given non-self-intersecting parallel drawings of a cycle graph in $\mathbb{R}^3$. Each drawing is a closed non-self-intersecting curve (i.e., a knot). If the drawings admit a parallel morph then they correspond to equivalent knots. However, the converse is false, as we show in Section 6.3.

1.3 Preliminaries

1.3.1 Graphs and drawings

A *graph* G comprises a finite set V(G) of *vertices*, and a set E(G) of *edges*, such that each element of E(G) is either an unordered pair $\{u, v\}$ or an ordered pair (u, v) of vertices, where $u, v \in \mathrm{V(G)}$. An unordered pair is called an *undirected* edge, while an ordered pair (u, v) is called a *directed* edge which is oriented from u to v. In a diagram, we will generally indicate the orientation of a directed edge with an arrow. For either $\{u, v\}$ or (u, v), vertices u and v are the *end-vertices* of the edge. An edge is said to be *incident* on its end-vertices.

A *path* of a graph G is an ordered sequence $(v_1, \ldots, v_k)$ of distinct vertices of a graph, such that $\{v_i, v_{i+1}\} \in \mathrm{E(G)}$, for all $i \in \{1, \ldots, k-1\}$. A *cycle* of G is defined similarly, except that in a cycle, the first and last vertices in the sequence are identical.

A *path graph* is a graph with the vertex set $\{v_1, \ldots, v_k\}$ and an edge set containing edges $\{v_i, v_{i+1}\}$ for all $i \in \{1, \ldots, k-1\}$. The edge set of a *cycle graph* with vertex set $\{v_1, \ldots, v_k\}$ contains, in addition, the edge $\{v_1, v_k\}$. We will sometimes simply use the terms "path" and "cycle" to mean "path graph" and "cycle graph", respectively. In all cases, our meaning should be clear from the context.

Given a graph G, a *subgraph* of G is a graph G' such that $\mathrm{V(G')} \subseteq \mathrm{V(G)}$ and $\mathrm{E(G')} \subseteq \mathrm{E(G)}$. The subgraph of G *induced* by a path $(v_1, \ldots, v_k)$ of G is the graph composed of vertices $v_1, \ldots, v_k$ and all edges $\{v_i, v_{i+1}\}$, where $i \in \{1, \ldots, k-1\}$. This graph is a path graph. The subgraph of G induced by a cycle of G is defined similarly.

Definition 1.3.1 *Let d be a positive integer and let G be a graph. A straight-line* drawing *P of G in $\mathbb{R}^d$ is a mapping from:*

- *each vertex $u \in \mathrm{V(G)}$ to a point $P(u) \in \mathbb{R}^d$, and,*
- *each edge $\{v, w\} \in \mathrm{E(G)}$ to the open line segment $P(u, v)$, between $P(u)$ and $P(v)$.*

Often, we will use the term "drawing" to refer to the range of the mapping, i.e. the image of all vertices and edges of the underlying graph. Similarly, when we talk about

a vertex u or an edge $\{v, w\}$ of a drawing P, this should be taken to be the image $P(v)$ or $P(v, w)$, respectively. This usage should not cause any confusion. Unless otherwise noted, throughout this thesis the reader should assume that we are referring to straight-line drawings.

Our definition of a drawing allows for a drawing to map multiple vertices to the same point. The edge between two coincident vertices maps to the empty set.

For a drawing P of a graph G in either the x-y plane or x-y-z space we will use the notation:

$$\begin{aligned} P(v) &= (P_{\mathrm{X}}(v), P_{\mathrm{Y}}(v)) \text{ and} \\ P(v) &= (P_{\mathrm{X}}(v), P_{\mathrm{Y}}(v), P_{\mathrm{Z}}(v)) \end{aligned}$$

to represent the coordinates of v in P, where $v \in \mathrm{V(G)}$. For drawings P in $\mathbb{R}^d$, we use the notation:

$$P(v) = (P_1(v), \ldots, P_d(v))$$

to represent the coordinates of v.

A drawing is called *simple* if no two elements (vertices or edges) map to intersecting elements (points or open line segments, respectively). A drawing that is not simple is called *self-intersecting*. We identify three types of self-intersection, which depends on the types of intersecting elements: a self-intersecting drawing will exhibit at least one of *vertex-vertex*, *vertex-edge*, or *edge-edge* intersection.

A drawing in $\mathbb{R}^d$ is called *orthogonal* if every edge is represented by a line segment that is parallel with one of the d coordinate axes.

Definition 1.3.2 *Let $a, b \in \mathbb{R}^d$ be two points. The L_p-distance between a and b is:*

$$||a, b||_p = \left(\sum_{i=1}^{d} (a_i - b_i)^p \right)^{\frac{1}{p}}$$

where $a = (a_1, \ldots, a_d)$ and $b = (b_1, \ldots, b_d)$.

We will make use mainly of the L_1 (rectilinear) and L_2 (Euclidean) metrics. Unless otherwise specified, assume that we are using the L_2-metric. The other metric that we use is the L_∞-metric.

Definition 1.3.3 *Let $a, b \in \mathbb{R}^d$ be two points. The L_∞-distance between a and b is:*

$$||a, b||_\infty = \max_{i \in \{1, \ldots, d\}} |a_i - b_i|$$

where $a = (a_1, \ldots, a_d)$ and $b = (b_1, \ldots, b_d)$.

In a drawing P of a graph G, the L_p-distance between a pair of vertices $v_1, v_2 \in \mathrm{V(G)}$ is defined as:

$$||P(v_1), P(v_2)||_p \ .$$

The L_p-distance between edge $\{v_3, v_4\}$ and vertex v_5 is:

$$\inf\{\ ||a, P(v_5)||_p \mid a \in P(v_3, v_4)\}.$$

Finally, the L_p-distance between two edges is $\{v_6, v_7\}, \{v_8, v_9\} \in \mathrm{E(G)}$ is:

$$\inf\{\ ||a, b||_p \mid a \in P(v_6, v_7), b \in P(v_8, v_9)\}.$$

Definition 1.3.4 *The* minimum feature size *of a drawing is the minimum L_2-distance in the drawing among:*

1. *two distinct vertices,*
2. *a vertex u and an edge $\{v, w\}$, such that $u \neq v$ and $u \neq w$; and,*
3. *two edges that are not incident at a common vertex.*

Note that the minimum distance between two non-incident edges may be omitted from the definition of minimum feature size for drawings in the plane. It is only necessary for drawings in $\mathbb{R}^d$ where $d \geq 3$.

1.3.2 Parallel drawings and parallel morphs

Definition 1.3.5 *Two drawings P and Q of a graph G are called* parallel *if, for every $\{u, v\} \in E(G)$, there exists some $\lambda > 0$ such that*

$$P(u) - P(v) = \lambda(Q(u) - Q(v)).$$

Since $\lambda > 0$ in the above definition, the vectors going from u to v must have the same direction in each of a pair of parallel drawings. Also, notice that every edge that has zero length in one drawing must have zero length in a second drawing that is parallel with the first.

Definition 1.3.6 *A* morph *M of a graph G is a continuously changing sequence of straight-line drawings of G, that is determined by a continuous motion of the vertices over the interval $[0, 1]$, such that at each instant $t \in [0, 1]$:*

1. *$M(t)$ denotes the drawing of G determined by the vertex positions at time t,*

2. *for all $u \in \mathrm{V(G)}$, $M(t;u)$ denotes the image of u in $M(t)$; and,*

3. *for all $\{v,w\} \in \mathrm{E(G)}$, $M(t;v,w)$ denotes the image of $\{v,w\}$ in $M(t)$.*

Let P and Q be two drawings of a graph G, and let M be a morph of G. If $M(0) = P$ and $M(1) = Q$, we say that M is a morph from P to Q. In this morph, P is called the *source* drawing of M, and Q is called the *target*.

Definition 1.3.7 *A* parallel morph *is a morph M such that $M(t)$ is both simple, and parallel with $M(0)$ and $M(1)$, for all $t \in [0,1]$.*

We will often find it convenient to describe a parallel morph as a composition of a sequence of parallel morphs. In the following lemma, we establish that such a composition of parallel morphs is permissible.

Lemma 1.3.1 *Let P, Q and R be three simple parallel drawings of a graph. If there exists a parallel morph from P to Q and a parallel morph from Q to R, then there exists a parallel morph from P to R.*

Proof: Let M and M' be a parallel morphs from P to Q, and from Q to R, respectively. To prove the existence of a parallel morph M'' from P to R, for all $t \in [0, 1/2]$ let

$$M''(t) = M(2t),$$

and for all $t \in (1/2, 1]$ let

$$M''(t) = M'(2t-1).$$

It is easy to verify that M'' is a parallel morph from P to R, as desired. □

Definition 1.3.8 *An edge is said to change length* monotonically *if it is either non-increasing or non-decreasing in length over the course of the morph.*

Definition 1.3.9 *A* monotone morph *is a parallel morph during which all edges change length monotonically.*

1.3.3 Drawings in the plane

An important class is the class of drawings in the plane. A *face* of a simple drawing P in the plane is a maximally-connected region of the plane that does not intersect P. The unbounded face is called the *exterior* face. Every other face is called an *interior* face.

A *polygon* is a simple drawing of a cycle graph in the plane. If every edge of a polygon is parallel with one of the two coordinate axes, the polygon is called *orthogonal*. An orthogonal polygon is *orthogonally convex* if every line that is parallel with one of the coordinate axes intersects the closed interior face of the polygon in either a single closed segment, or the empty set.

Definition 1.3.10 *The* area *of a drawing in the plane is the area of the smallest axis-aligned box that contains the drawing in its interior.*

Lemma 1.3.2 *Let M be a morph (not necessarily a parallel morph) between two simple straight-line drawings of a graph G in the plane. If some intermediate drawing of M is self-intersecting, then there exists some $t_0 \in (0, 1)$ such that either*

1. *$M(t_0)$ exhibits vertex-vertex intersection, or*
2. *$M(t_0)$ exhibits vertex-edge intersection.*

Proof: To the contrary, assume that no intermediate drawing of M exhibits either vertex-vertex or vertex-edge intersection. Hence, some intermediate drawing exhibits edge-edge intersection. Let $\{u_1, v_2\}$ and $\{u_2, v_2\}$ denote edges whose representative open line segments intersect in an intermediate drawing of M. Let $t_0 \in (0, 1)$ be such that the distance between $\{u_1, v_1\}$ and $\{u_2, v_2\}$ is zero in $M(t_0)$; and, for all t such that $t_0 < t \leq 1$, the distance between these two edges is strictly greater than zero in $M(t)$. Since $M(1)$ is simple and the distance between two edges is a continuous function of time, t_0 must exist.

Drawing $M(t_0)$ exhibits neither vertex-vertex nor vertex-edge intersection. However, the distance between the open segments corresponding to $\{u_1, v_1\}$ and $\{u_2, v_2\}$ is zero. Therefore, the open segments must intersect in $M(t_0)$, i.e. $M(t_0)$ exhibits edge-edge intersection.

If the two intersecting edges are parallel, $M(t_0)$ exhibits either vertex-vertex or vertex-edge intersection. Therefore, these edges are not parallel. Hence, $\{u_1, v_1\}$ and $\{u_2, v_2\}$ intersect in a single point. A sufficiently small continuous motion of the vertices determines a continuous motion of this point of intersection. Therefore, there exists a sufficiently small $\epsilon > 0$ such that in $M(t_0 + \epsilon)$ the distance between the edges is zero. Contradiction. □

1.3.4 Polyhedra

The class of polyhedra is the three-dimensional analogue of the class of polygons. We assume a general familiarity with polyhedra. See [48] for precise definitions.

A polyhedron can be thought of as the boundary of a connected solid in $\mathbb{R}^3$, such that this boundary consists of a finite number of planar polygonal regions (called *faces*) that meet at points (called *vertices*) and line segments (called *edges*) such that the vertices and edges form a drawing of a connected graph. This graph is called the *edge graph* of the polyhedron. Each edge is incident on two exactly two faces and two vertices. Each face is incident on three or more edges. Two faces meet either at a single edge, at a single vertex, or not at all. The intersection of the polyhedron with any sufficiently small open ball centered at a point of the polyhedron is homeomorphic to an open disk, i.e. a polyhedron forms a 2-manifold embedded in $\mathbb{R}^3$.

Two polyhedra are *combinatorially identical* if there exists an isomorphism between the vertices, edges and faces of the polyhedra, such that these elements exhibit the same incidence relationships in both polyhedra. Thus, there is an isomorphism between the edge graphs of the two polyhedra.

Definition 1.3.11 *Two combinatorially identical polyhedra are called* parallel *if the two drawings of the edge graphs determined by the polyhedra are parallel.*

Observation 1.3.1 *If two combinatorially identical polyhedra are parallel, then corresponding faces of the polyhedra reside in parallel planes.*

Proof: Let P and Q be parallel polyhedra with underlying edge graph G. Let f_P be a face of P and let f_Q be the corresponding face in Q. Let $u, v, w \in \mathrm{V(G)}$ be three vertices such that edges $\{u, v\}, \{v, w\} \in \mathrm{E(G)}$ are incident on f_P and f_Q, and the the points at which these vertices reside are not collinear in either P or Q. Since P and Q are parallel, there exist $\lambda_1, \lambda_2 > 0$ such that

$$P(u) - P(v) = \lambda_1(Q(u) - Q(v))$$

and

$$P(v) - P(w) = \lambda_2(Q(v) - Q(w)).$$

Since $P(u)$, $P(v)$ and $P(w)$ are points in $\mathbb{R}^3$, there exists a vector $R \in \mathbb{R}^3$ such that

$$(P(u) - P(v)) \cdot R = 0$$

and

$$(P(v) - P(w)) \cdot R = 0.$$

Therefore,

$$\left(\frac{P(u) - P(v)}{\lambda_1}\right) \cdot R = 0 \quad \Longrightarrow \quad (Q(u) - Q(v)) \cdot R = 0.$$

Similarly, $(Q(v) - Q(w)) \cdot R = 0$. Vector R is perpendicular to both the plane containing f_P and the plane containing f_Q. Hence, the two faces lie in parallel planes. □

Starting with a polyhedron, a *morph* of the polyhedron is specified uniquely by the motion of its vertices over time. In order for vertex motions to serve as a morph of a polyhedron, at each instant the vertex positions must determine a valid polyhedron, in that the vertices and edges bounding each face must form the drawing of a polygon in a single plane in $\mathbb{R}^3$, and the surface determined by the vertex positions is an embedding of a 2-manifold into $\mathbb{R}^3$ (i.e. there is no self-intersection).

Definition 1.3.12 *Given a polyhedron P, a morph of P is called a* parallel morph *if the polyhedron formed at each instant of the morph is parallel with P.*

This thesis discusses mainly *orthogonal* polyhedra. These are polyhedra in which each edge is parallel with one of the coordinate axes. A cube is the simplest example of an orthogonal polyhedron. Observe that each face of an orthogonal polyhedron is perpendicular to one of the coordinate axes.

In general, the discussion of polyhedra in this thesis is limited to those that are homeomorphic to a sphere, i.e. genus-0 polyhedra. Occasionally, polyhedra of higher genus are mentioned, e.g. polyhedra homeomorphic to a torus. Sometimes, when discussing morphs of orthogonal polyhedra it is convenient to refer to the following class of objects, topologically equivalent to a closed disk.

Definition 1.3.13 *An* orthodisk *is an orthogonal genus-0 polyhedron that has been turned into a closed disk by cutting it along an orthogonal drawing of a cycle graph that lies on the surface of the polyhedron.*

The definition of a polyhedron given above severely restricts the class of surfaces representable by polyhedra. This is not a weakness. On the contrary, most of the results in this thesis pertaining to parallel morphs of polyhedra are negative results, i.e., results stating either that there exist pairs of parallel polyhedra for which a parallel morph is impossible, or that it is computationally intractable to decide whether a given pair of parallel polyhedra admit a parallel morph. For such results, restricting the class of polyhedra is a strength.

1.4 Overview of Chapters

Chapter 2: Linear Morphs

A morph between two straight-line drawings of a graph is called a *linear morph* if all vertices move in straight-line paths at constant speed. A linear morph between two parallel drawings will keep all intermediate drawings parallel, though not necessarily simple.

In this chapter, we will discuss some special cases in which a linear morph between a pair of parallel drawings will maintain simplicity, and will therefore be a parallel morph. We prove that if there exists a parallel morph between two simple orthogonal drawings in d-space, then there exists a parallel morph between the drawings that is composed of a sequence of linear morphs. Further, we give an effective procedure (albeit, not polynomial time) to decide whether two orthogonal drawings in d-space will admit a parallel morph.

In the plane, we discover that the linear morph between a pair of parallel polygons will be a parallel morph if one of these polygons is star-shaped. It follows that if a pair of parallel polygons are themselves parallel with a third, star-shaped polygon, then there exists a parallel morph between the original pair. This parallel morph is composed of a sequence of two linear morphs. In order to use this observation for morphing, we need to be able to generate a parallel star-shaped polygon from a given polygon, if one exists. We find that such a polygon can be generated by solving a system of linear inequalities.

Chapter 3: Morphing Orthogonal Drawings in the Plane

From the previous chapter, we know that if a pair of parallel simple orthogonal drawings in d-space admits a parallel morph, then it will admit a parallel morph that is composed of a sequence of linear morphs. The number of linear morphs used in a parallel morph may provide some indication of the visual complexity of the morph. As such, it is interesting to try to use as few linear morphs as possible in a parallel morph.

Our main result in this chapter is that every pair of parallel simple orthogonal drawings of a connected graph in the plane will admit a parallel morph. We give two algorithms for generating parallel morphs of such drawings. The morphs generated by each algorithm are composed of a sequence of linear morphs. These algorithms offer a trade-off between runtime, and the number of linear morphs used in the worst case. Both algorithms generate morphs in which the number of linear morphs can be bounded by a constant factor of the number of vertices in the underlying graph. We are able to prove that the number of linear morphs required in a parallel morph between two parallel simple orthogonal drawings in the plane that is composed of a sequence of linear morphs will need to use a number of

linear morphs proportional to the number of vertices, in the worst case.

While our morphing algorithms produce morphs of connected graphs, we also show that it is not difficult to modify the algorithms to handle parallel orthogonal drawings of a disconnected graph. This situation is complicated slightly by the fact that such drawings will not always admit a simplicity-preserving morph (parallel or otherwise). However, there is an easy-to-check condition that determines whether or not a parallel morph is possible.

Parallel morphs generated by the algorithms of this chapter may suffer from large changes in the lengths of edges. It is possible that an edge grows to a length that is an exponential factor longer (in the number of vertices in the graph) than either its source or target length, while another edge remains constant in length throughout the morph. Such behavior is undesirable for graph visualization, since to display intermediate drawings of a morph, it may be necessary to either scale these drawings to so that some features become indiscernible, or to only show part of the drawing at one time.

Fortunately, it is easy to move vertices of an orthogonal drawing to integer coordinates within a small range. We show how parallel morphs produced by our algorithms can be modified so to avoid exponentially increasing edge lengths.

Chapter 4: Morphing with Monotonically Changing Edge Lengths

This chapter continues our discussion of edge length changes in parallel morphs. We observe that the original algorithms given in Chapter 3 will generate morphs that may be thought of as having two stages: first, an expansive stage in which all edge lengths are non-decreasing, followed by a stage in which all edge lengths change *monotonically*, i.e. a stage in which each edge is either non-increasing or non-decreasing in length. In some applications, it may be desirable that in a parallel morph, edges alternately increase and then decrease in length only a few times. We define a *monotone* morph as a parallel morph in which all edges change length monotonically.

It seems that monotone morphs will always exist for only very restrictive classes of drawings. On the positive side, we have been able to prove that a monotone morph always exists for pairs of parallel *orthogonally-convex* polygons, which are orthogonal polygons such that every line that is parallel with one of the coordinate axes intersects the closed interior in either a single closed line segment, or the empty set. However, even proving this result is surprisingly non-trivial. The result does not extend to graphs with orthogonally-convex faces, nor to orthogonally-convex polyhedra in which every plane that is perpendicular to one of the coordinate axes intersects the closed interior in an orthogonally convex polygon. We give counterexamples for this.

Orthogonal polygons that are convex with respect to only one axis instead of both do not always admit a monotone morph. However, neither have we been able to devise an efficient algorithm for deciding monotone morphability in this case, nor to prove the decision problem to be NP-hard. We give an example in an attempt to illustrate the difficulties that one must face in devising an algorithm for this problem. More generally, for pairs of parallel orthogonal polygons it is NP-hard to decide whether a monotone morph exists. We prove this in Chapter 5.

The algorithms of Chapter 3 produce parallel morphs that will be composed of a sequence of two monotone morphs. We call such morphs *bi-monotone*. Interestingly, every pair of drawings that admits a parallel morph will admit a bi-monotone morph; note that drawings need not be orthogonal or even in the plane for this result to hold. However, for orthogonal drawings in the plane, we prove that the area of an intermediate drawing of certain classes of bi-monotone morph must be exponentially larger than the area of the source and target drawings, in the worst case.

Chapter 5: The Hardness of Morphing in the Plane

In this chapter, we study the problem of deciding whether a pair of parallel orthogonal polygons in the plane will admit a parallel morph under the additional constraint that a given subset of the edges remains static in length throughout the morph. For a morph to exist, it is necessary that the set of edges to be held static during the morph has identical lengths in both the source and target drawings of the input. We prove this problem to be NP-hard via a reduction from boolean satisfiability.

Although interesting on its own, the main reason that we choose to establish NP-hardness for this problem is that, following a minor augmentation of the proof, we obtain a proof that it is NP-hard to decide monotone morphability; a problem raised in Chapter 4. Further, the problem of deciding parallel morphability with static edges reduces easily to the problem of deciding parallel morphability for a pair of parallel (but not necessarily orthogonal) drawings of a connected graph in the plane. In total, we establish NP-hardness for three problems in this chapter.

Chapter 6: Paths, Trees, and Polyhedra

When we leave the plane and move into higher dimensions, we are faced with a host of interesting problems. From Chapter 5, we know that it is already NP-hard to decide parallel morphability for drawings of connected graphs in the plane. Deciding parallel morphability in higher dimensions must also be NP-hard: given a drawing of a connected

graph in which all edges lie in a single plane embedded in $\mathbb{R}^d$ such that $d \geq 3$, all edges must remain coplanar during a parallel morph.

So, we examine special cases in which the problem of deciding parallel morphability is tractable. We begin by restricting our attention to drawings of a tree graph. We prove that every pair of parallel simple drawings of a tree in d-space will admit a parallel morph. These drawings need not be orthogonal. There is a straightforward algorithm for generating a morph, which uses a number of linear morphs which are proportional to the length of the longest path through the tree.

In contrast, deciding parallel morphability for drawings of a cycle in $\mathbb{R}^3$ seems much more challenging. One of the difficulties is that two drawings of a cycle may correspond to different topological knots, precluding the possibility of any simplicity-preserving morph between them. In particular, we give a pair of parallel orthogonal drawings of a cycle that do not admit a parallel morph, such that these drawings correspond to the trivial knot. However, we have not been able to prove it NP-hard to decide whether a parallel pair of drawings of a cycle will admit a parallel morph. Neither have we been able to give an efficient algorithm for this problem, even if we know that the drawings correspond to the same knot.

We suspect that deciding parallel morphability for parallel drawings of a cycle in $\mathbb{R}^3$ is computationally intractable. Nonetheless, due to our success with orthogonal drawings in the plane, we had hoped that it might be easier to decide parallel morphability for orthogonal polyhedra with the same topology as a sphere. Unfortunately, this intuition turned out to be false. We give examples of such parallel orthogonal polyhedra that do not admit a parallel morph. One of the polyhedra we devise is useful to us again in Chapter 7, where we prove it to be PSPACE-hard to decide parallel morphability for orthogonal polyhedra.

Chapter 7: PSPACE-Hardness Results

In this chapter, we again study the hardness of problems related to deciding parallel morphability. Our goal is to prove hardness results for problems of deciding parallel morphability for orthogonal drawings and orthogonal polyhedra in $\mathbb{R}^3$. However, as it can be difficult to describe drawings in higher dimensions, we find it most convenient to begin this topic by examining a related problem in the plane.

Consider the following problem: given a pair of parallel simple orthogonal drawings of a graph in the plane and a distinguished subset of the vertices, called the *static vertices*, decide whether there exists a parallel morph between the drawings such that all static

vertices remain stationary during the morph. Note that it is necessary that every static vertex has the same coordinates in both drawings, otherwise it must move during any morph between the drawings. We prove that this problem is PSPACE-hard.

We reduce from the problem of deciding parallel morphability with static vertices, to prove that it is PSPACE-hard to decide whether a pair of parallel orthogonal drawings of a connected graph in $\mathbb{R}^3$ will admit a parallel morph. We reduce from the same problem to prove that it is PSPACE-hard to decide whether a pair of parallel orthogonal polyhedra will admit a parallel morph, where the polyhedra are topological spheres. In the latter of these reductions, we adapt a polyhedron developed in Chapter 6, which acts as a sort of loop under a parallel morph.

Unfortunately, we have been unable to establish the complexity of deciding parallel morphability for parallel drawings of a cycle graph in $\mathbb{R}^3$.

Chapter 8: Conclusion

In our final chapter, we first give a brief summary of our results. We conclude with open problems and mention directions for future research in the study of parallel morphing.

Chapter 2

Linear Morphs

2.1 Introduction

Consider a morph between two drawings of a graph in which every vertex moves from its initial coordinates to its target coordinates at constant velocity. To obtain the coordinates of a vertex at some intermediate drawing, we linearly interpolate between the source and target coordinates of the vertex. We call such a morph a *linear morph*—see below for a precise definition.

A linear morph is, in a sense, the most straightforward morph possible. Shortly, we will prove that in a linear morph between a pair of parallel drawings, every intermediate drawing is parallel with the source and target drawings. However, linear morphs are not necessarily parallel morphs since they do not always maintain simplicity; see Figure 2.1. They do maintain simplicity when one drawing is a scaling and/or translation of the other.

Linear morphs are used implicitly by Guibas, Hershberger and Suri [34] in their "parallel move" operation, which consists of a uniform scaling and/or translation operation performed on a portion of a polygon. The algorithm that they develop computes a se-

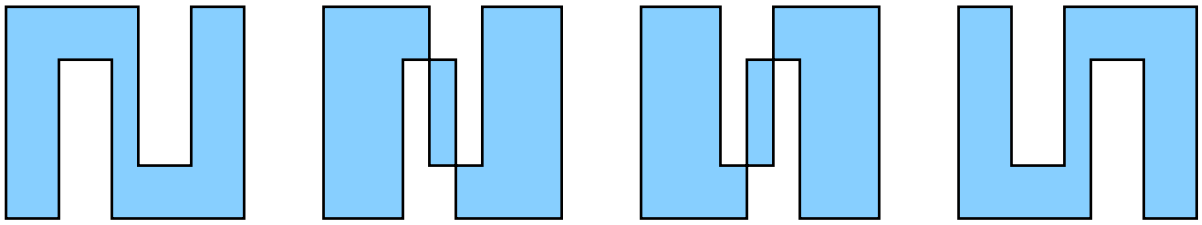

Figure 2.1: A sequence of snapshots of a linear morph between a pair of parallel orthogonal polygons; this morph does not maintain simplicity.

quence of parallel moves that together describe a parallel morph between given parallel polygons. Streinu [57] studies linear morphs in the context of parallel redrawings. Linear morphs have also been used to morph graph drawings in the absence of parallel and simplicity constraints (see e.g. [28]).

Friedrich and Eades [28] discuss a number of measures that might be used to compare the quality of morphs between drawings of a graph. Their aim is to develop a means to evaluate how well a morph preserves a user's "mental map" [47]. One of the measures they identify is the length of the paths traveled by a vertex during the morph. It is suggested that in order to preserve the user's mental map, the lengths of the vertex paths should be minimized. In the absence of other measures—such as the number of temporary edge crossings that appear during the morph, for example—a linear morph is optimal.

One of our goals in this chapter is to prove that, for every pair of orthogonal drawings that admit a parallel morph, the drawings will admit a parallel morph that is composed of a finite sequence of linear morphs. Such a morph can be represented by:

1. the abstract graph,

2. the start and end times of each linear morph; and,

3. the *terminal* (i.e. source and target) drawings of each linear morph in the sequence.

Every algorithm that we present in this thesis for computing a parallel morph will generate a parallel morph composed of a sequence of linear morphs. For each algorithm, we bound the number of linear morphs that may be required in the worst case. The number of linear morphs comprising a morph (parallel or otherwise) provides us with a measure of the visual complexity of the morph.

The second main result of this chapter is that, given a pair of parallel polygons such that one of these polygons is star-shaped, the linear morph from one polygon to the other maintains simplicity and is a parallel morph. Therefore, if we are given a pair P and Q of parallel polygons such that there exists a third, star-shaped polygon R that is also parallel with each of the given polygons, then there exists a parallel morph from P to Q via R. This parallel morph is composed of a sequence of two linear morphs. We end the chapter by showing that R can be generated (or, shown not to exist) by computing a feasible solution to a system of linear equalities and inequalities.

Let us proceed by giving a formal definition of a linear morph.

Definition 2.1.1 *Let P and Q be a pair of d-dimensional drawings of a graph G that are not necessarily parallel. The* linear morph *from P to Q is the morph M such that for all*

$t \in [0,1]$ *and* $v \in V(G)$,

$$M(t;v) = tQ(v) + (1-t)P(v).$$

In a morph, edge lengths are said to change *monotonically* if every edge is either non-increasing or non-decreasing in length as the morph proceeds. Mindful of the goal of preserving the user's mental map, we expect that a morph which exhibits only small changes in edge lengths—measured, for example, as the sum of the absolute values of the increases and decreases in length of each edge—is preferable to a morph in which large changes occur in edge lengths. This measure is minimized in a morph in which edge lengths change monotonically. Edge length monotonicity in parallel morphs is the topic of Chapter 4.

Lemma 2.1.1 *Every intermediate drawing of a linear morph between parallel drawings is parallel with the source and target drawings. Further, edge lengths change monotonically.*

Proof: Let P and Q be a pair of parallel drawings in $\mathbb{R}^d$, and let M be the linear morph from P to Q. We must prove that for every $\{u,v\} \in E(G)$ there exists a function $\lambda : [0,1] \to (0,\infty)$ such that for every $t \in [0,1]$

$$M(t;u) - M(t;v) = \lambda(t)(P(u) - P(v)). \tag{2.1}$$

Since P and Q are parallel drawings, there exists some $\lambda' > 0$ such that

$$\lambda'(P(u) - P(v)) = Q(u) - Q(v).$$

From the definition of a linear morph, we have that

$$\begin{aligned} M(t;u) - M(t;v) &= tQ(u) + (1-t)P(u) - tQ(v) - (1-t)P(v) \\ &= t(Q(u) - Q(v)) + (1-t)(P(u) - P(v)) \\ &= t\lambda'(P(u) - P(v)) + (1-t)(P(u) - P(v)) \\ &= (t(\lambda' - 1) + 1)(P(u) - P(v)). \end{aligned}$$

We observe that $(t(\lambda' - 1) + 1) > 0$ for all $t \in [0,1]$. Therefore, we can satisfy (2.1) by setting

$$\lambda(t) = (t(\lambda' - 1) + 1).$$

To see that the length of $\{u,v\}$ changes monotonically, it suffices to observe that $\lambda(t)$ changes linearly in t. □

In Section 2.2 we prove that, if a pair of parallel orthogonal drawings of a graph in $\mathbb{R}^d$ admits a parallel morph, then this pair admits a parallel morph composed of a finite sequence of linear morphs. In Section 2.3, we prove that if we are given a pair of parallel polygons, one of which is star-shaped, then the linear morph is a parallel morph between these. With this result, we proceed to show that if a pair of parallel polygons happens to be parallel with a third, star-shaped polygon, they will admit a parallel morph composed of two linear morphs. We make concluding remarks in Section 2.4.

2.2 Parallel Morphs of Orthogonal Drawings

Throughout this section, we restrict our attention to orthogonal drawings in $\mathbb{R}^d$, i.e. those drawings in which each edge is parallel with one of the d coordinate axes. Our aim is to prove that whenever a parallel morph exists between two orthogonal drawings, there exists a parallel morph between these drawings that is composed of a finite number of linear morphs. Moreover, there exists a finite-time algorithm to generate such a parallel morph as a sequence of drawings. Along the way, we will encounter definitions and lemmas that will aid us in later chapters.

To prove our main result, we will assume that we are given a parallel source and target drawings, which are simple orthogonal drawings in $\mathbb{R}^d$ that admit a parallel morph. We partition the set of all drawings that are parallel with the source and target into a finite number of classes, based on the orderings of the vertices with respect to each coordinate axis. Starting with a particular parallel morph from the source to the target, we can extract a sequence of classes. We prove that for any pair of simple orthogonal drawings whose classes are adjacent in this sequence, a linear morph is a parallel morph.

Next, we generate a new parallel morph that is composed of a sequence of linear morphs. To do so, we choose a representative drawing from each class and perform a sequence of linear morphs between representative drawings that takes us through the same sequence of classes as does the original parallel morph. Each class needs to be visited at most once, therefore the new parallel morph is composed of a finite sequence of linear morphs.

Definition 2.2.1 *Given a pair P and Q of drawings of a graph G in $\mathbb{R}^d$, we say that the drawings have equal* vertex orderings *if, for all $u, v \in V$ and $i \in \{1, \ldots, d\}$, either:*

1. *$P_i(u) < P_i(v)$ and $Q_i(u) < Q_i(v)$,*
2. *$P_i(u) > P_i(v)$ and $Q_i(u) > Q_i(v)$; or*

3. $P_i(u) = P_i(v)$ *and* $Q_i(u) = Q_i(v)$.

We note similarity of our "vertex ordering" and the "orthogonal ordering" defined by Misue et al. [47]. In their paper, they stress the value of maintaining the orthogonal ordering of vertices (represented by boxes) when adjusting the layout of a graph drawing, when the goal is preservation of the user's mental map.

The notion of the vertex ordering of a drawing enables us to classify drawings; two drawings belong to the same class if their vertex orderings are equal.

Lemma 2.2.1 *If two orthogonal drawings of the same graph in* $\mathbb{R}^d$ *have equal vertex orderings, then the drawings are parallel with each other.*

Proof: Let P and Q be two orthogonal drawings of a graph G in $\mathbb{R}^d$, such that P and Q have equal vertex orderings. To prove that these drawings are parallel with each other, we must prove that every (oriented) edge has the same direction in both drawings.

So, let $\{u, v\} \in E(G)$ be an arbitrary edge. By the definition of vertex ordering, the two vertices either coincide in both P and Q, or else they coincide in neither drawing. Therefore, the length of $\{u, v\}$ in P is zero if and only if its length in Q is zero. Assume below that $\{u, v\}$ has non-zero length in both P and Q.

Since the drawings are orthogonal, there exists exactly one coordinate axis $i \in \{1, \ldots, d\}$ such that

$$P_i(u) - P_i(v) \neq 0.$$

Since P and Q have the same vertex ordering, it follows that

$$Q_i(u) - Q_i(v) \neq 0.$$

Further, for all $j \in \{1, \ldots, d\}$ such that $j \neq i$,

$$Q_j(u) - Q_j(v) = P_j(u) - P_j(v) = 0.$$

Define a value λ such that

$$\lambda = \frac{P_i(u) - P_i(v)}{Q_i(u) - Q_i(v)} .$$

Since P and Q have the same vertex ordering, either

1. $P_i(u) - P_i(v) > 0$ and $Q_i(u) - Q_i(v) > 0$; or
2. $P_i(u) - P_i(v) < 0$ and $Q_i(u) - Q_i(v) < 0$.

In both cases, $\lambda > 0$. It is straightforward to verify that

$$P(u) - P(v) = \lambda(Q(u) - Q(v))$$

and hence, P and Q are parallel drawings. □

Lemma 2.2.2 *If an orthogonal drawing of a graph in $\mathbb{R}^d$ self-intersects, then all orthogonal drawings of the graph with the same vertex ordering self-intersect.*

Proof: Let P be an orthogonal drawing in $\mathbb{R}^d$ of a graph G, such that P self-intersects. Let Q be an arbitrary orthogonal drawing with the same vertex ordering as P. We will prove below that Q self-intersects.

A drawing that self-intersects will exhibit at least one of the following types of self-intersection: vertex-vertex, vertex-edge, and edge-edge. We treat each of these separately.

First, suppose that P exhibits vertex-vertex intersection. There exist vertices $u, v \in \mathrm{V}(G)$ such that $P(u) = P(v)$. By the definition of vertex ordering, $Q(u) = Q(v)$. Hence, Q self-intersects in this case.

Now suppose that P exhibits vertex-edge intersection. There exists a vertex $u \in \mathrm{V}(G)$ and edge $\{v, w\} \in \mathrm{E}(G)$ such that $P(u)$ intersects $P(v, w)$. Since P is an orthogonal drawing, there exists a coordinate axis $i \in \{1, \ldots, d\}$ such that either:

$$P_i(v) < P_i(u) < P_i(w) \quad \text{or} \quad P_i(v) > P_i(u) > P_i(w).$$

Since P and Q have the same vertex ordering, either:

$$Q_i(v) < Q_i(u) < Q_i(w) \quad \text{or} \quad Q_i(v) > Q_i(u) > Q_i(w).$$

For all $j \in \{1, \ldots, d\}$ such that $j \neq i$,

$$P_j(v) = P_j(u) = P_j(w)$$

hence

$$Q_j(v) = Q_j(u) = Q_j(w).$$

It follows that $Q(u)$ intersects $Q(v, w)$.

Finally, let us consider the possibility of an edge-edge intersection in P. Suppose that there exists a pair of edges $\{u_1, v_1\}, \{u_2, v_2\} \in E(G)$ such that $P(u_1, v_1)$ intersects $P(u_2, v_2)$; we may assume that each edge has non-zero length, otherwise edge-edge intersection cannot occur between these two edges. Define $i_1 \in \{1, \ldots, d\}$ as the coordinate axis that is parallel

with $P(u_1, v_1)$, and define $i_2 \in \{1, \ldots, d\}$ as the coordinate axis parallel with $P(u_2, v_2)$. If $i_1 = i_2$ (i.e. the two edges are parallel and intersecting) then P must exhibit vertex-edge intersection. We have handled this case already, so assume that $i_1 \neq i_2$.

For all $i \in \{1, \ldots, d\}$ such that $i \neq i_1$ and $i \neq i_2$,

$$P_i(u_1) = P_i(v_1) = P_i(u_2) = P_i(v_2).$$

Thus,

$$Q_i(u_1) = Q_i(v_1) = Q_i(u_2) = Q_i(v_2).$$

Observe that since $i_1 \neq i_2$,

$$P_{i_1}(u_2) = P_{i_1}(v_2) \text{ and } P_{i_2}(u_1) = P_{i_2}(v_1).$$

Without loss of generality, assume that

$$P_{i_1}(u_1) < P_{i_1}(v_1) \text{ and } P_{i_2}(u_2) < P_{i_2}(v_2).$$

Since $P(u_1, v_1)$ and $P(u_2, v_2)$ intersect, and $i_1 \neq i_2$,

$$P_{i_1}(u_1) < P_{i_1}(u_2) = P_{i_1}(v_2) < P_{i_1}(v_1)$$

and

$$P_{i_2}(u_2) < P_{i_2}(u_1) = P_{i_2}(v_1) < P_{i_2}(v_2).$$

Notice that P and Q have the same vertex ordering. It follows that

$$Q_{i_1}(u_1) < Q_{i_1}(u_2) = Q_{i_1}(v_2) < Q_{i_1}(v_1)$$

and

$$Q_{i_2}(u_2) < Q_{i_2}(u_1) = Q_{i_2}(v_1) < Q_{i_2}(v_2).$$

Therefore, $Q(u_1, v_1)$ intersects $Q(u_2, v_2)$. □

Suppose that we are given a pair of parallel simple orthogonal drawings of a graph, such that the drawings have equal vertex orderings. Throughout a linear morph between the two drawings, the vertex ordering will not change. By Lemma 2.2.2, this linear morph is a parallel morph. So, morphing between drawings with equal vertex orderings is easy. We need to determine when we can morph between drawings with different orderings.

Definition 2.2.2 *Given a pair of drawings P and Q in $\mathbb{R}^d$ of a graph G, the drawings have* consonant vertex orderings *if for all $u, v \in V(G)$ and $i \in \{1, \ldots, d\}$, either:*

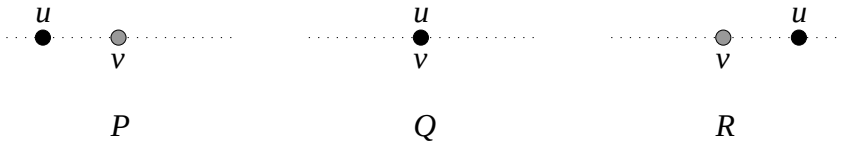

Figure 2.2: Drawings P and Q have consonant orderings, as do Q and R. The orderings of P and R are not consonant.

1. $P_i(u) \leq P_i(v)$ *and* $Q_i(u) \leq Q_i(v)$*; or*
2. $P_i(u) \geq P_i(v)$ *and* $Q_i(u) \geq Q_i(v)$.

While the property of having the same vertex ordering is a transitive property of drawings, the property of having consonant vertex orderings is not. To illustrate the difference between "same" and "consonant" vertex orderings, imagine three drawings P, Q, R on the real line, involving two vertices, u and v and no edges:

$$P(u) = -1,\ Q(u) = 0,\ R(u) = 1$$
$$P(v) = Q(v) = R(v) = 0.$$

See Figure 2.2.

Drawings P and Q have consonant vertex orderings, as do Q and R, while P and R do not. However, no two of P, Q, R have the same vertex ordering. In general, two drawings with the same vertex ordering may be said to have consonant orderings.

The following lemma reveals the importance of consonant vertex orderings. We will refer to this lemma in subsequent chapters. Before stating the lemma, let us recall the definition of *minimum feature size* of a drawing (Definition 1.3.4, page 15) as the infimum of distances among: (1) a pair of vertices, (2) a vertex and a non-incident edge, and (3) a pair of edges.

Lemma 2.2.3 *Let P and Q be a pair of parallel simple orthogonal drawings of a graph G in $\mathbb{R}^d$ with consonant vertex orderings and a minimum feature size of $\delta > 0$. The linear morph from P to Q is a parallel morph that maintains a minimum feature size of $\frac{\delta}{\sqrt{d}}$.*

Proof: Let M denote the linear morph from P to Q. By Lemma 2.1.1, every intermediate drawing of M from P to Q is parallel with P and Q. Below, we prove that M maintains a minimum feature size of at least $\frac{\delta}{\sqrt{d}}$. Hence, M maintains simplicity and is a parallel morph.

First, we will prove that M maintains a minimum feature size of δ when we measure distance using the L_1 (rectilinear) metric; recall Definition 1.3.2 (page 14), which states that the L_1-distance between two points $a, b \in \mathbb{R}^d$ is

$$||a, b||_1 = \sum_{i=1}^{d} |a_i - b_i|.$$

Subsequently, we will prove that this implies that the minimum feature size, when measured using the L_2 (Euclidean) metric, is at least $\frac{\delta}{\sqrt{d}}$.

To obtain the minimum feature size of a drawing, we must consider vertex-vertex distances, vertex-edge distances, and edge-edge distances. We begin by examining the L_1-distance between pairs of vertices in intermediate drawings of M. Let $u, v \in V(G)$. Without loss of generality, assume that for all $i \in \{1, \ldots, d\}$

$$P_i(u) \geq P_i(v) \quad \text{and} \quad Q_i(u) \geq Q_i(v). \tag{2.2}$$

For all $t \in [0, 1]$,

$$||M(t; u), M(t; v)||_1 = \sum_{i=1}^{d} |tQ_i(u) + (1-t)P_i(u) - tQ_i(v) - (1-t)P_i(v)|$$

which, by the order assumption of (2.2),

$$\begin{aligned} &= \sum_{i=1}^{d} t|Q_i(u) - Q_i(v)| + (1-t)|P_i(u) - P_i(v)| \\ &= t||Q(u), Q(v)||_1 + (1-t)||P(u), P(v)||_1 \\ &\geq t\delta + (1-t)\delta = \delta. \end{aligned}$$

Next, we prove that M maintains a minimum L_1-distance of δ between every vertex and non-incident edge of G. Let $\{u, v\} \in \mathrm{E}(G)$ and let $w \in \mathrm{V}(G)$, where $u \neq w \neq v$. Let $i_0 \in \{1, \ldots, d\}$ denote the coordinate axis that is parallel with $P(u, v)$. Without loss of generality, assume that $P_{i_0}(u) < P_{i_0}(v)$ and $Q_{i_0}(u) < Q_{i_0}(v)$.

Observation 2.2.1 *If $P_i(v) \leq P_i(w)$ and $Q_i(v) \leq Q_i(w)$, then, for all $t \in [0, 1]$,*

$$\begin{aligned} M_i(t; v) &= tQ_i(v) + (1-t)P_i(v) \\ &\leq tQ_i(w) + (1-t)P_i(w) \\ &= M_i(t; w). \end{aligned}$$

In other words, consonance is preserved by a linear morph between consonant drawings. Since P and Q have consonant vertex orderings, one of the following will hold:

1. $M_{i_0}(t; v) \leq M_{i_0}(t; w)$, for all $t \in [0, 1]$.
2. $M_{i_0}(t; w) \leq M_{i_0}(t; u)$, for all $t \in [0, 1]$.
3. $M_{i_0}(t; u) \leq M_{i_0}(t; w) \leq M_{i_0}(t; v)$, for all $t \in [0, 1]$.

For all $i \in \{1, \ldots, d\} - \{i_0\}$ and $t \in [0, 1]$, the distance between $\{u, v\}$ and w in $M(t)$ with respect to the i-th axis is equal to

$$|M_i(t; w) - M_i(t; u)| = |M_i(t; w) - M_i(t; v)|.$$

In the first two cases above, the L_1-distance between w and $\{u, v\}$ at time t is (respectively) equal to either:

$$\sum_{i \in \{1,\ldots,d\}} |M_i(t; w) - M_i(t; v)| = ||M(t; v), M(t; w)||_1 \geq \delta$$

or

$$\sum_{i \in \{1,\ldots,d\}} |M_i(t; w) - M_i(t; u)| = ||M(t; w), M(t; u)||_1 \geq \delta.$$

In the third case, the L_1-distance between $\{u, v\}$ and w in $M(t)$ is zero with respect to the i_0-th axis. Therefore, the L_1 distance between w and $\{u, v\}$ is:

$$\sum_{i \in \{1,\ldots,d\} - \{i_0\}} |M_i(t; u) - M_i(t; v)|.$$

This equals the L_1-distance between w and $\{u, v\}$ when we project $M(t)$ orthogonally into the $(d-1)$-dimensional space in which $\{u, v\}$ appears as a single point. Thus, we must bound the minimum L_1-distance between two points in $(d-1)$-dimensional space during the linear morph between drawings with consonant vertex orderings, such that the vertices are separated by at least distance δ in the terminal drawings of this linear morph. We have already seen that such a linear morph will maintain a minimum feature size of at least δ, under the L_1-metric.

We can use the same trick to bound the minimum L_1-distance between two edges of G that are not incident at a common vertex. Let $\{u_1, v_1\}, \{u_2, v_2\} \in \mathrm{E}(G)$ such that all of these vertices are distinct. Let $i_1, i_2 \in \{1, \ldots, d\}$ denote the coordinate axes that are parallel with $P(u_1, v_1)$ and $P(u_2, v_2)$, respectively.

If the distance between $\{u_1, v_1\}$ and $\{u_2, v_2\}$ is zero with respect to either i_1 or i_2 in both P and Q, then this distance must be zero for all $M(t)$. In this case, we can project the drawings orthogonally into the $(d-1)$-dimensional space in which either $\{u_1, v_1\}$ or $\{u_2, v_2\}$ (or both, if $i_1 = i_2$) is a point. The L_1-distance between the two edges in $M(t)$ is equal to the L_1-distance between the point and edge (or, two points) in this $(d-1)$-dimensional space. We have already seen that this distance will be at least δ.

Otherwise, there exists no axis for which the distance between the edges is zero in both P and Q. Then it must be that the infimum L_1-distance between the edges is equal to the minimum distance between two of the vertices of these edges, for all $M(t)$. Again, this distance is at least δ.

So, we have established that M maintains a minimum feature size of δ, when measured using the L_1-metric. Let us now bound the minimum feature size using the L_2-metric. As a special case of the Cauchy-Schwarz inequality, for points $x, y \in \mathbb{R}^d$

$$\left(\sum_{i=1}^{d} x_i y_i\right)^2 \leq \left(\sum_{i=1}^{d} x_i^2\right)\left(\sum_{i=1}^{d} y_i^2\right).$$

Given two points $a, b \in \mathbb{R}^d$. For all $i \in \{1, \ldots, d\}$, let $x_i = |a_i - b_i|$ and $y_i = 1$.

$$||a,b||_1 = \sum_{i=1}^{d} |a_i - b_i| \leq \sqrt{\left(\sum_{i=1}^{d} (a_i - b_i)^2\right)\left(\sum_{i=1}^{d} 1^2\right)} = ||a,b||_2 \sqrt{d}\ .$$

It follows that M maintains a minimum feature size of at least $\frac{\delta}{\sqrt{d}}$. □

Theorem 2.2.1 *Let P and Q be a pair of parallel orthogonal drawings of a graph G in $\mathbb{R}^d$. If there exists a parallel morph from P to Q, then there exists such a morph, composed of a sequence of fewer than*

$$(2^{n-1} n!)^d$$

linear morphs, where n is the number of vertices in G.

Proof: Construct a graph H in which every vertex is associated one-to-one with a distinct ordering of the vertices of G (in $\mathbb{R}^d$). The ordering associated with each vertex of H corresponds to a class of parallel drawings (Lemma 2.2.1). Either every drawing in this class is simple, or else every drawing self-intersects (Lemma 2.2.2). Remove from H each vertex for which the associated ordering corresponds to a class of drawings that is either: (1) self-intersecting; or, (2) not parallel with P and Q. Add an edge to H between every pair of vertices for which the associated vertex orderings are consonant.

The vertex orderings of P and Q are associated with either one or two vertices of H. Both orderings are associated with a single vertex if the orderings are the same. In this case, Lemma 2.2.3 tells us that the linear morph from P to Q is a parallel morph.

We will prove that P and Q admit a parallel morph if and only if there exists a path (i.e. a sequence of distinct vertices of $\mathrm{V}(H)$, such that adjacent vertices in this sequence are connected by an edge) in H from the vertex associated with the vertex ordering of P to the vertex associated with the vertex ordering of Q. Hence, to decide whether P and Q admit a parallel morph, we need only to determine whether there exists a path in H between two vertices.

Claim 2.2.1 *If there exists a path in H between the vertices associated with the orderings of P and Q, then there exists a parallel morph from P to Q, composed of a sequence of at most $|V(H)| - 1$ linear morphs.*

Proof: Let $(h_1, \ldots, h_k)$ be a path in H, such that h_1 and h_k are associated with the vertex orderings of P and Q, respectively. Each vertex in this path is associated with a (non-empty) class of simple orthogonal drawings of G that are parallel with P and Q. Let P^i denote some drawing of G chosen arbitrarily from the class associated with h_i, where $i \in \{2, \ldots, k-1\}$.

Adjacent drawings of the sequence $(P, P^2, \ldots, P^{k-1}, Q)$ have consonant vertex orderings. Hence, by Lemma 2.2.3, the morph that is generated by performing linear morphs between adjacent drawings in this sequence is a parallel morph. Clearly, at most $|\mathrm{V}(H)|-1$ linear morphs are required. □

Next, we prove in the other direction.

Claim 2.2.2 *If there exists a parallel morph from P to Q, then there exists a path in H between the vertices associated with the ordering of P and Q.*

Proof: Let M be a parallel morph from P to Q. Extract from M the (possibly infinite) sequence of times $T = (t_1, t_2, \ldots)$ where, for all $t_i \in T$:

1. $t_i \in [0, 1]$,
2. $t_i < t_{i+1}$ (if $t_{i+1} \in T$); and,
3. t_i is the time at which some pair of vertices goes from having unequal coordinates to having equal coordinates with respect to one of the coordinate axes.

To make this third criterion more precise: some time $t \in [0,1]$ is an element of the sequence T if and only if there exists two vertices $u, v \in \mathrm{V}(G)$ and some coordinate axis $j \in \{1, \ldots, d\}$ such that

$$M_j(t; u) = M_j(t; v)$$

and there exists a value $\kappa \in \mathbb{R}$ such that for all $\epsilon < \kappa$,

$$M_j(t - \epsilon; u) \neq M_j(t - \epsilon; v).$$

Since the drawing $M(t)$ is simple and parallel with P and Q, for all $t \in [0,1]$, there exists a vertex of H associated with the vertex ordering of $M(t)$. However, the sequence of vertices of H determined by the sequence of drawings

$$(M(0), M(t_1), M(t_2), \ldots, M(1))$$

is not necessarily a path, since the same vertex might be encountered multiple times in the sequence. However, by removing cycles, we obtain a sequence of vertices of H that contains each vertex at most once.

In order to prove that—after removing cycles—the resulting sequence is indeed a path, we need only to establish that an edge exists in H between every pair of adjacent vertices in the sequence. This is true if the drawings $M(t_i)$ and $M(t_{i+1})$ have consonant vertex orderings, for all $t_i, t_{i+1} \in T$.

Observe that if $M(t_i)$ and $M(t_{i+1})$ do not have consonant vertex orderings, then there exist vertices $u, v \in \mathrm{V}(G)$ and $j \in \{1, \ldots, d\}$ such that:

$$M_j(t_i; u) < M_j(t_i; v) \quad \text{and} \quad M_j(t_{i+1}; u) > M_j(t_{i+1}; v).$$

Since M changes vertex coordinates continuously, there must exist some $t \in T$ where $t_j < t < t_{j+1}$, such that

$$M_j(t; u) = M_j(t; v).$$

Therefore, $M(t_i)$ and $M(t_{i+1})$ have consonant vertex orderings. We conclude that there exists a path in H between the vertices associated with the vertex orderings of P and Q. □

By the above two claims, if P and Q admit a parallel morph, then there exists a path in H between the vertices associated with the vertex orderings of P and Q, and therefore, there exists a parallel morph from P to Q composed of at most $|\mathrm{V}(H)| - 1$ linear morphs.

To complete our proof of Theorem 2.2.1, it remains only to bound the number of vertices that might appear in H.

We can construct a vertex ordering as follows: for each of the d coordinate axes, choose some sequence of the n vertices. Assign a relation of either '$<$' or '$=$' between each ordered pair of vertices that are adjacent in the sequence. For each axis, there are exactly $n!$ possible sequences and $2^{(n-1)}$ possible assignments of relations. Note that this approach strictly overestimates the number of possible vertex orderings. Overall, there are fewer than $|\mathrm{V}(H)| \leq (2^{n-1}n!)^d$ possible vertex orderings. □

Theorem 2.2.1 suggests an algorithm for deciding whether or not a given pair P, Q of parallel orthogonal drawings admits a parallel morph: construct the graph H, and determine whether or not there exists a path between the two vertices associated with the vertex orderings of P and Q. Clearly, such an algorithm is prohibitively expensive both in terms of time and space requirements. It is an interesting open problem to determine whether deciding parallel morphability for orthogonal drawings is in PSPACE.

In Chapter 3 it is shown that there always exists a parallel morph between parallel orthogonal drawings in the plane. However, for parallel orthogonal drawings in $\mathbb{R}^3$ it is PSPACE-hard to decide parallel morphability. This hardness result is established in Chapter 7.

2.3 Linear Morphs from Star-Shaped Polygons

Consider parallel source and target polygons in the plane such that one of these polygons is star-shaped. In this section, we establish that for such polygons the linear morph from the source to the target is a parallel morph. It follows that if neither the source nor the target is star-shaped but there exists a star-shaped polygon that is parallel with both polygons, then there exists a parallel morph from the source to the target, i.e. perform a linear morph from the source polygon to the star-shaped polygon, followed by a linear morph from the star-shaped polygon to the target.

For a given polygon, suppose that there exists a star-shaped polygon that is parallel to this one. Then, such that a star-shaped can be generated by computing a feasible solution to a system of linear constraints.

2.3.1 Preliminaries

A *polygon* is a simple drawing of a cycle graph in the plane. A polygon partitions the plane into two regions: the *interior* and the *exterior* of the polygon.

Definition 2.3.1 *A polygon P is* convex *if, for every pair of points $a, b \in \mathbb{R}^2$ in the open*

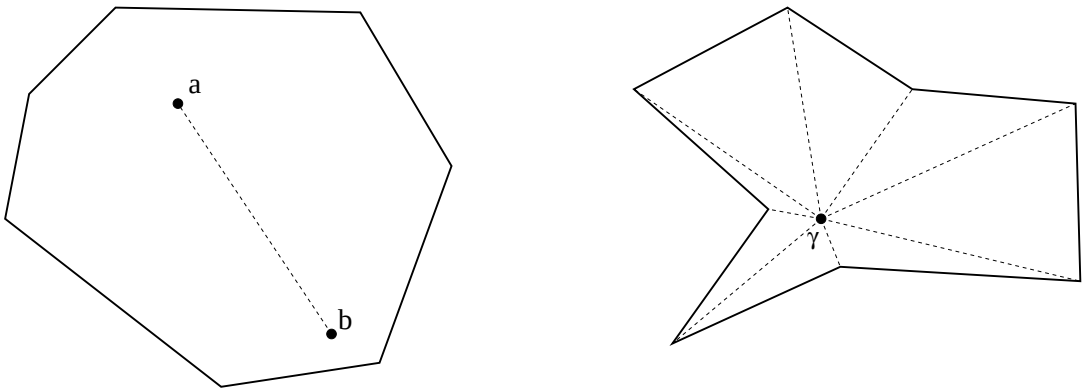

Figure 2.3: *Left:* a convex polygon. *Right:* a star-shaped polygon.

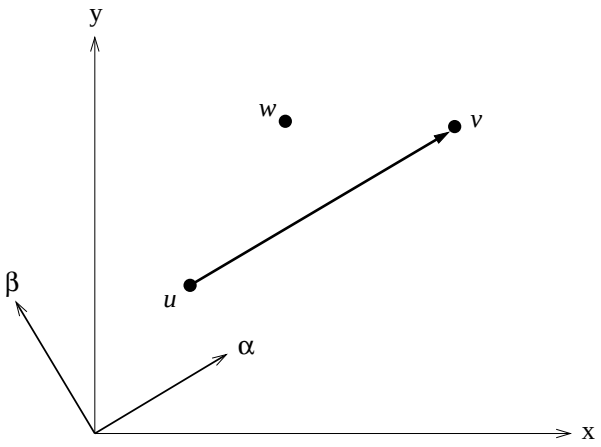

Figure 2.4: Vertex w is to the left of the oriented edge (u, v).

interior of P, the open line segment between a and b lies entirely within the open interior of P.

Define visibility for polygons in the following strict sense.

Definition 2.3.2 *Two points $a, b \in \mathbb{R}^2$ are* properly visible *to each other with respect to a polygon P if the open line segment between a and b does not intersect P.*

Definition 2.3.3 *A polygon P is* star-shaped *if there exists a point $\gamma \in \mathbb{R}^2$ in the open interior of P such that γ is properly visible to every point of P.*

Examples of convex and star-shaped polygons are given in Figure 2.3. Clearly, every convex polygon is also star-shaped.

Let u, v, w be three vertices of a graph G, mapped by a drawing P to distinct points in the plane. We say that w is to the *left* of the oriented edge (u, v) if the points $P(u), P(v), P(w)$ are not collinear, and (u, v, w, u) is a counterclockwise traversal of the triangle composed of vertex set $\{u, v, w\}$ and edge set:

$$\{\{u, v\}, \{v, w\}, \{w, u\}\}.$$

We say that w is to the *right* of (u, v) if the points $P(u), P(v), P(w)$ are not collinear and (u, v, w, u) is a clockwise traversal of this triangle.

Lemma 2.3.1 *Let P and Q be a pair of drawings of a graph G in the plane such that $u, v, w \in V(G)$ and $\{u, w\} \in E(G)$, and $\{u, v\}$ is parallel in both drawings. Let M be a linear morph from P to Q.*

1. *If w is to the left of (u, v) in both P and Q, then w is to the left of (u, v) in $M(t)$ for all $t \in [0, 1]$.*

2. *If w is to the left of (u, v) in P, and collinear with (u, v) in Q, then w is to the left of (u, v) in $M(t)$ for all $t \in [0, 1)$.*

Proof: Rotate both P and Q clockwise about the origin by the same angle such that $\{u, v\}$ is parallel with the y-axis and v has a larger y-coordinate than u. Only the first claim is proved below. The second claim can be proved similarly.

In the first case,

$$P_{\mathrm{x}}(w) < P_{\mathrm{x}}(u) = P_{\mathrm{x}}(v)$$

and

$$Q_{\mathrm{x}}(w) < Q_{\mathrm{x}}(u) = Q_{\mathrm{x}}(v).$$

Hence, for all $t \in [0, 1]$,

$$\begin{aligned} M_{\mathrm{x}}(t; w) &= tQ_{\mathrm{x}}(w) + (1-t)P_{\mathrm{x}}(w) \\ &< tQ_{\mathrm{x}}(u) + (1-t)P_{\mathrm{x}}(u) \\ &= M_{\mathrm{x}}(t; u) = M_{\mathrm{x}}(t; v). \end{aligned}$$

Similarly, for all $t \in [0, 1]$,

$$\begin{aligned} M_{\mathrm{y}}(t; u) &= tQ_{\mathrm{y}}(u) + (1-t)P_{\mathrm{y}}(u) \\ &< tQ_{\mathrm{y}}(v) + (1-t)P_{\mathrm{y}}(v) \\ &= M_{\mathrm{y}}(t; v). \end{aligned}$$

In summary, for all $t \in [0,1]$, w is to the left of (u,v) in $M(t)$. □

An analogous result holds for the case in which w lies to the right of (u,v) in P, and w lies either to the right of (u,v), or collinear with (u,v) in Q.

2.3.2 A linear morph from a star-shaped polygon

In this section, we prove the following.

Theorem 2.3.1 *Given a pair of parallel polygons such that one of these polygons is star-shaped, the linear morph from one polygon to the other is a parallel morph.*

Suppose that P and Q are a pair of parallel polygons such that one of these is star-shaped. By Lemma 2.1.1, intermediate drawings of the linear morph from P to Q are pairwise parallel. It remains only to prove that all intermediate drawings are simple.

To this end, vertices and edges are added to P, thereby decomposing the plane into a number of disjoint faces. Edges and vertices are similarly added to Q, generating a decomposition of the plane. The two decompositions are combinatorially identical, in that they form two (not necessarily parallel) drawings of a graph. It is shown that no face collapses during a linear morph between the two drawings, i.e. the elements on the boundary of each face remain pairwise non-intersecting during the morph. The following lemma then implies that the linear morph between P and Q preserves simplicity.

Lemma 2.3.2 *Let P be a drawing of a graph G such that every face (including the exterior face) is bounded by a polygon. Let Q be another drawing of G. Let M be a morph from P to Q. For each face of P, if all elements of the face remain pairwise non-intersecting throughout M, then M preserves simplicity.*

Proof: (By contradiction.) Assume that, although all elements of each face of P remain pairwise non-intersecting throughout M, morph M does not preserve simplicity. There exists some unique $t_0 \in \mathbb{R}$ where $0 < t_0 < 1$ such that $M(t_0)$ self-intersects, and $M(t)$ is simple for all $t \in \mathbb{R}$ such that $0 \leq t < t_0$. It follows from Lemma 1.3.2 that $M(t_0)$ exhibits either vertex-vertex or vertex-edge intersection. Let v be a vertex such that v intersects another element e in $M(t_0)$, where e is either a vertex or edge.

Since elements on each face of P remain pairwise non-intersecting throughout M, v and e cannot lie on the boundary of the same face in P. Since vertices move in continuous paths, there must exist some sufficiently small $\epsilon > 0$ such that in $M(t_0 - \epsilon)$ vertex v lies in the open interior of a face which has e on its boundary. Clearly, in P vertex v does not lie

in the face corresponding to this face of $M(t_0 - \epsilon)$. Hence, v must have earlier intersected some element as it passed from the exterior of the face to the interior. This contradicts the assumption that $M(t_0)$ is the first non-simple drawing of M. □

Definition 2.3.4 *Given a simple orthogonal drawing P of a graph G, an* augmentation step *is one of the following changes made to P:*

1. *Add a new vertex to P.*
2. *Add an edge to P between two vertices that are already present in the drawing.*
3. *Subdivide an edge by a vertex: for some $\{u, v\} \in \mathrm{E}(G)$, add a new vertex w to P such that $P(w) \in P(u, v)$, and replace $\{u, v\}$ by edges $\{u, w\}$ and $\{w, v\}$.*

The reverse process is deletion and removal of degree-2 vertices.

Definition 2.3.5 *A drawing P' is an* augmentation *of a drawing P if P' can be obtained from P via some sequence of augmentation steps.*

Observation 2.3.1 *Let P and Q be a pair of simple drawings of a graph. Let P' be an augmentation of P, and let Q' be an augmentation of Q. If P' and Q' are drawings of the same graph and the linear morph from P' to Q' preserves simplicity, then the linear morph from P to Q preserves simplicity.*

Proof: Let M denote the linear morph from P to Q. Let M' denote the linear morph from P' to Q'. By the definition of augmentation, if a point $x \in \mathbb{R}^2$ belongs to more than one element of $M(t)$, where $t \in \mathbb{R}$ and $0 \leq t \leq 1$, then x also belongs to more than one element of $M'(t)$. Therefore, if M' preserves simplicity, then so does M. □

The idea behind the proof of Theorem 2.3.1 is to show that P and Q can be augmented such that the two augmentations are combinatorially equivalent, and the elements of each face of the augmentation of P remain pairwise non-intersecting during the linear morph to the augmentation of Q. Then, Lemma 2.3.2 can be employed to prove that the linear morph from P to Q preserves simplicity.

This scheme is complicated slightly by the fact that that the augmentations of P and Q defined are themselves not simple. Instead, they are devised such that for all sufficiently small $\epsilon > 0$, $M(\epsilon)$ and $M(1 - \epsilon)$ are simple, where M is the linear morph from P to Q. It can be shown that the elements of each face of $M(\epsilon)$ remain pairwise non-intersecting

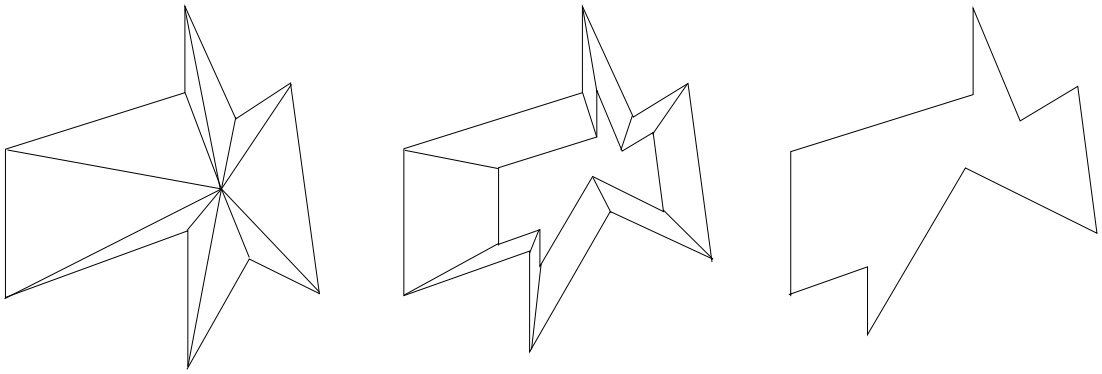

Figure 2.5: *Left:* drawing P. *Center:* $M(t)$ for some $t \in (0,1)$. *Right:* drawing Q.

during the linear morph to $M(1-\epsilon)$. Using Lemma 2.3.2 and the fact that P and Q are simple drawings, it follows that the linear morph from P to Q is a parallel morph.

It is convenient to discuss decompositions of the interiors of P and Q first, followed by decompositions of the exteriors.

Interior decomposition

Let G denote the underlying graph of P and Q. Without loss of generality, assume that P is star-shaped. G is a *cycle graph* in that it is connected and all vertices have degree two. Modify G using the following procedure:

1. for all vertices $v \in \mathrm{V}(G)$, add a *copy* $c(v)$ of v to $\mathrm{V}(G)$,
2. for all vertices $v \in \mathrm{V}(G)$, add $\{v, c(v)\}$ to $\mathrm{E}(G)$; and,
3. for all edges $\{u, v\} \in \mathrm{E}(G)$, add $\{c(u), c(v)\}$ to $\mathrm{E}(G)$.

Also, modify P and Q so that they are drawings of the modified graph G. The resulting drawings are augmentations of the original drawings P and Q. For all $v \in \mathrm{V}(G)$, define

$$P(c(v)) = \gamma \quad \text{and} \quad Q(c(v)) = Q(v)$$

where $\gamma \in \mathbb{R}^2$ is a point in the open interior of P that is properly visible to every point on P. Since P is star-shaped, γ must exist.

Following modification, G is no longer a cycle, and P and Q are no longer polygons. Moreover, the drawings are not even simple: in P all copies of the original vertices coincide, and in Q each vertex coincides with its copy; see Figure 2.5 (left and right, respectively).

Let M denote the linear morph from P to Q. Figure 2.5 (center) illustrates an intermediate drawing of M.

Call a cycle of G a *facial cycle* if the open interior of the face of P (or Q) bounded by this cycle contains no other element of P (respectively, Q). This interior may be empty. It is shown below that every facial cycle is simple in every intermediate drawing of the linear morph from P to Q, excluding the two terminal drawings.

There are two types of face to consider; these are treated separately. First, consider the facial cycle of G that contains all copies of the original vertices, and none of the original vertices. In P, all vertices of the facial cycle coincide at γ. In Q, the drawing of this facial cycle is identical to the original polygon Q.

Lemma 2.3.3 *The drawing of the facial cycle of G that contains all vertices $c(v) \in V(G)$ is a simple polygon in $M(t)$, where $t \in (0,1)$.*

Proof: For all $\{c(u), c(v)\} \in \mathrm{E}(G)$ and $t \in [0,1]$, observe that:

$$\begin{aligned} M(t;c(u)) - M(t;c(v)) &= ((1-t)\gamma + tQ(c(u))) - ((1-t)\gamma + tQ(c(v))) \\ &= (1-t)\gamma + tQ(u) - (1-t)\gamma - tQ(v) \\ &= (1-t)(Q(u) - Q(v)). \end{aligned}$$

Hence, for all $t \in (0,1)$, the drawing of the facial cycle of G that consists of all copies of the original vertices is a (non-zero) scaling and translation of the original polygon Q, and is therefore simple. □

Next, consider a facial cycle of G, with vertices

$$u,\ v,\ c(u),\ c(v)$$

where $\{u, v\} \in \mathrm{E}(G)$. The edges in this facial cycle are

$$\{u, v\},\ \{v, c(v)\},\ \{u, c(u)\},\ \{c(u), c(v)\}.$$

Lemma 2.3.4 *The drawing of the facial cycle of G that consists of vertices $u, v, c(u)$ and $c(v)$ is a polygon in $M(t)$, for all $t \in (0,1)$.*

Proof: Suppose that (u, v) is the counterclockwise orientation of $\{u, v\}$ with respect to the original polygon P. Since $P(c(u)) = P(c(v)) = \gamma$, both $c(u)$ and $c(v)$ are to the left of (u, v) in P. In Q, all four vertices are collinear. By Lemma 2.3.1, both $c(u)$ and $c(v)$ are to the left of (u, v) in $M(t)$, for all $t \in (0,1)$.

By Lemma 2.3.3, edges $\{u, v\}$ and $\{c(u), c(v)\}$ are parallel with each other in $M(t)$, for all $t \in (0,1)$. It follows that the drawing of the facial cycle with vertices $u, v, c(u), c(v)$ is a polygon in $M(t)$, where $t \in (0,1)$. □

Exterior decomposition

We now decompose the exteriors of P and Q. Instead of using edges (i.e. open line segments) in this decomposition, we find it convenient to use rays. For this decomposition, let us assume that G is the original cycle graph, prior to the modifications applied for the interior decomposition.

Definition 2.3.6 *Given vectors $a, b \in \mathbb{R}^2$ such that $a \neq (0,0)$, an* open ray $\mathcal{R}(a,b)$ *is:*

$$\mathcal{R}(a,b) = \{x \mid x = \lambda a + b, \lambda > 0\}.$$

Definition 2.3.7 *Two open rays $\mathcal{R}(a_1,b_1)$ and $\mathcal{R}(a_2,b_2)$ are* parallel *if there exists a real value $\lambda > 0$ such that $a_1 = \lambda a_2$.*

We can extend our definition of a drawing to include open rays. An open ray may be thought of as representing an edge (as does an open line segment) such that one of the two vertices incident on this edge resides at infinity. A drawing is simple if its elements are pairwise non-intersecting, where an element is either a vertex, an edge, or an open ray.

For every ray we add to P from a vertex v, we will add a parallel ray to Q from vertex v. We define a linear morph between two drawings with parallel rays in the obvious way: if parallel rays reside at a vertex v in drawings P and Q, then in every intermediate drawing of the linear morph from P to Q, a ray resides at v that is parallel with the corresponding rays in P and Q.

Recall that γ is a point in the interior of the polygon P, and is properly visible to every point in the closed interior of P.

For all $u \in \mathrm{V}(G)$, add the open ray $r_P(u)$ to P, where

$$r_P(u) = \mathcal{R}(P(u) - \gamma, P(u)).$$

See Figure 2.6 (left). We observe that for all $u, v \in \mathrm{V}(G)$,

$$r_P(u) \cap r_P(v) = \emptyset.$$

That is, no two rays added to P will intersect each other. Further, no ray will intersect the interior of P. Thus, by adding these rays to P we decompose the exterior face of P into a number of disjoint faces, each of which is bounded by a drawing consisting of two vertices, two open rays, and one edge.

In a similar manner, let us decompose the exterior of Q. For every vertex $u \in \mathrm{V}(G)$ add to Q the open ray $r_Q(u)$, where:

$$r_Q(u) = \mathcal{R}(P(u) - \gamma, Q(u)).$$

See Figure 2.6 (right). Observe that $r_Q(u)$ is parallel with $r_P(u)$.

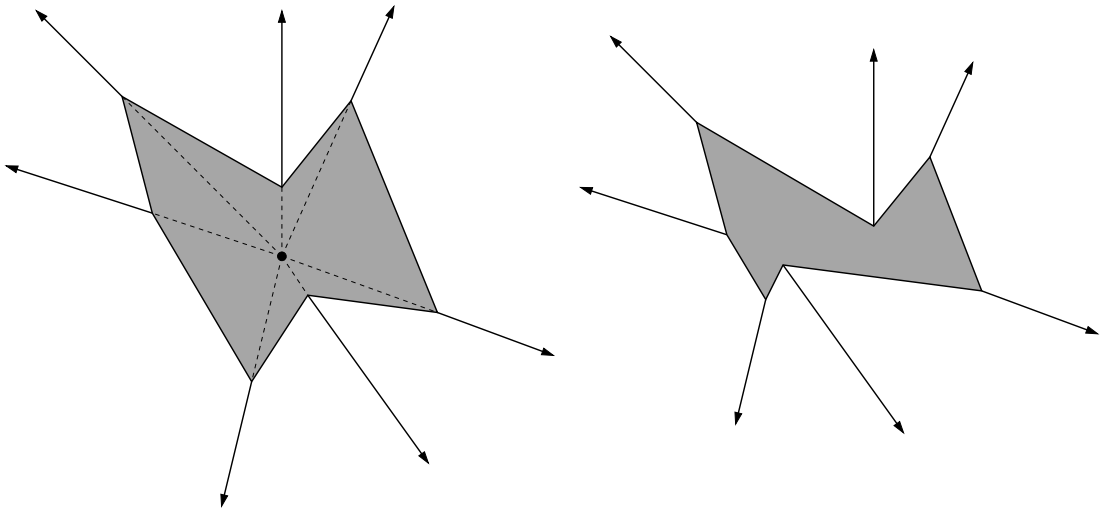

Figure 2.6: *Left:* rays added to the exterior of P. *Right:* parallel rays added to Q.

Lemma 2.3.5 *Following the addition of open rays to Q, no open ray will intersect either a vertex, an edge, or another open ray of Q.*

Proof: Let $(v_1, \ldots, v_n, v_1)$ denote the clockwise cycle of vertices in polygons P and Q, and let $v_{n+1} = v_1$ and $v_0 = v_n$. We will prove that for all $i, j \in \{1, \ldots, n\}$ where $i \neq j$, the following conditions hold:

1. $r_Q(v_i) \cap r_Q(v_j) = \emptyset$,
2. $r_Q(v_i) \cap Q(v_j) = \emptyset$, and
3. $r_Q(v_i) \cap Q(v_{j-1}, v_j) = \emptyset$.

This is proved by induction on the number of vertices encountered (including v_i and v_j) when traversing the cycle clockwise from v_i to v_j.

Since corresponding open rays and edges are parallel in P and Q, the three conditions hold whenever $j = i+1$. So, assume that the conditions are satisfied for all $i, j \in \{1, \ldots, n\}$ where $i \neq j$ when the number of vertices encountered when going clockwise from v_i to v_j is fewer than k.

Consider the case in which k vertices are encountered when going clockwise from v_i to v_j. Assume to the contrary that $r_Q(v_1)$ intersects at least one of: (1) the open ray $r_Q(v_k)$, (2) the vertex $Q(v_k)$, or (3) the edge $Q(v_{k-1}, v_k)$.

Assume for the moment that $r_Q(v_1)$ intersects $Q(v_k)$, as illustrated in Figure 2.7. Remove all open rays from Q, along with vertices $v_{k+1}, \ldots, v_n$ (and incident edges) and add

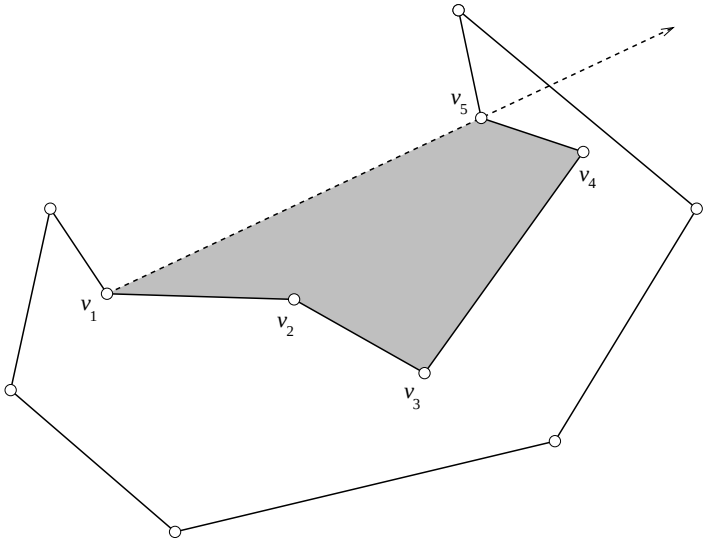

Figure 2.7: Polygon Q. Ray $r_Q(v_1)$, represented by a dashed arrow, intersects $Q(v_5)$.

the edge $\{v_1, v_k\}$. From the inductive assumption, it follows that this operation generates a new polygon such that the sequence $(v_1, \ldots, v_k, v_1)$ is a counterclockwise traversal of this polygon; the interior of such a polygon is shown shaded in Figure 2.7. If instead $r_Q(v_1)$ intersects $r_Q(v_k)$ or $Q(v_{k-1}, v_k)$, generate a polygon by adding an extra vertex at the intersection of $r_Q(v_1)$ and one of the intersecting elements, and adding edges as appropriate.

The open ray $r_Q(v_2)$ is directed into the interior of the new polygon, and therefore must intersect it. Since $r_Q(v_2)$ does not intersect $r_Q(v_1)$, neither does $r_Q(v_2)$ intersect edge $\{v_1, v_k\}$ of the new polygon. There must exist some value $l \in \{3, \ldots, k\}$ such that $r_Q(v_2)$ intersects either: the open ray $r_Q(v_l)$, the vertex $Q(v_l)$, or the edge $Q(v_{l-1}, v_l)$.

This contradicts the inductive assumption. Therefore, the three conditions hold whenever there are k vertices in the clockwise traversal from v_i to v_j. Thus, the lemma is proved. □

For each $\{u, v\} \in \mathrm{E}(G)$, there exists a face of P bounded by $P(u)$ and $P(v)$, open rays $r_P(u)$ and $r_P(v)$, and edge $P(u, v)$. There exists a corresponding face of Q with vertices $Q(u)$ and $Q(v)$, open rays $r_Q(u)$ and $r_Q(v)$, and edge $Q(u, v)$ on the boundary. The edges and rays that bound each of these faces will remain parallel during a linear morph from P to Q. It is clear that no two elements of a face will intersect during the linear morph. Thus, each face of the exterior decomposition survives the linear morph.

This completes our proof of Theorem 2.3.1.

2.3.3 Morphing with an intermediate star-shaped polygon

From Theorem 2.3.1 we have the following corollary.

Corollary 2.3.2 *Given a pair of parallel polygons P and Q such that both are parallel to a third star-shaped polygon R, there exists a parallel morph from P to Q.*

Proof: By Theorem 2.3.1, the morph composed of the linear morph from P to R, followed by the linear morph from R to Q, is a parallel morph. □

This corollary prompts the question: how do we compute a star-shaped polygon that is parallel with a given polygon? Below, we formulate a linear system of equalities and inequalities based on a given polygon P such that any feasible solution determines a star-shaped polygon that is parallel with P. The system consists of $5n$ variables, $2n$ equality constraints, and $2n$ inequalities, where n is the number of vertices in P.

Let P be a polygon and let $v_1, \ldots, v_n \in V(G)$ such that the cycle $(v_1, \ldots, v_n, v_1)$ represents the counterclockwise order of vertices around the polygon. For all $i \in \{1, \ldots, n-1\}$, let

$$\alpha_i = P(v_{i+1}) - P(v_i).$$

Let β_i denote the unit vector that is orthogonal to α_i, such that β_i points left with respect to the direction of vector α_i.

For all $i \in \{1, \ldots, n\}$, we use the vector of variables $x_i \in \mathbb{R}^2$ to represent the coordinates of vertex v_i of the star-shaped polygon. If this star-shaped polygon exists, the origin is properly visible to every point of this polygon. We use the variable $\lambda_i \in \mathbb{R}$ to represent the length of edge $\{v_i, v_{i+1}\}$ in the star-shaped polygon. Finally, we use variables $a_i \in \mathbb{R}$ and $b_i \in \mathbb{R}$ to represent the coordinates of the origin of an alternate coordinate system whose basis vectors are α_i and β_i, respectively, and whose origin is x_i.

We define a linear system with the following variables and constraints:

$$\begin{array}{ll} \text{Variables:} & x_i \in \mathbb{R}^2 \\ & \lambda_i, a_i, b_i \in \mathbb{R} \\ \text{Constraints:} & x_{i+1} - x_i = \lambda_i \alpha_i \\ & x_i + a_i \alpha_i + b_i \beta_i = (0, 0) \\ & \lambda_i, b_i \geq 1 \end{array}$$

In this system, assume that $x_{n+1} = x_1$.

We shall prove that if there exists a feasible solution to this system, then the solution determines a polygon in which each vertex v_i resides at point x_i. Moreover, the polygon

contains the origin in its interior, and the origin is properly visible to every point of the polygon. Therefore, a star-shaped polygon that is parallel with a given polygon can be computed in polynomial time by solving a linear program, using e.g. Karmarkar's algorithm [44], which runs in $O(n^{3.5}L^2 \ln L \ln \ln L)$ time, where L is the number of bits in the input coefficients.

Note that the 1 in the constraints $\lambda_i \geq 1$ and $b_i \geq 1$ can be replaced by any rational constant strictly greater than zero.

Theorem 2.3.3 *There exists a feasible solution to the above system if and only if there exists a star-shaped polygon Q that is parallel with P, where $Q(v_i) = x_i$ for all $i \in \{1, \ldots, n\}$.*

Proof: Assume that there exists a star-shaped polygon Q that is parallel with P. Without loss of generality, assume that the origin is properly visible to all points of Q, and that every edge of Q is at least as long as the corresponding edge in P. Further, assume that there is at least unit distance from the origin to each edge of Q. If not, scale and translate Q to make this so.

To prove that there exists a feasible solution to the linear system, for all $i \in \{1, \ldots, n\}$, let $x_i = Q(v_i)$ and let

$$\lambda_i = \frac{||Q(v_i), Q(v_{i+1})||_2}{||P(v_i), P(v_{i+1})||_2} .$$

Thus, the constraint $\lambda_i \geq 1$ is satisfied. Moreover,

$$\begin{aligned} x_{i+1} - x_i &= Q(v_{i+1}) - Q(v_i) \\ &= ||Q(v_i), Q(v_{i+1})||_2 \left(\frac{P(v_{i+1}) - P(v_i)}{||P(v_i), P(v_{i+1})||_2} \right) \\ &= \lambda_i \alpha_i. \end{aligned}$$

Thus, the first equality constraint of the linear system is satisfied.

Since Q contains the origin and the origin is properly visible to all points of Q, the origin lies to the left of each counterclockwise-oriented edge (v_i, v_{i+1}) in Q; see Figure 2.8. Also, by an earlier assumption, there is at least unit distance from each edge of Q to the origin. Hence, the origin lies in the set

$$\{x_i + \alpha_i a_i + \beta_i b_i \mid a_i \in (-\infty, \infty), b_i \in [1, \infty)\}$$

Thus, the second equality constraint of the system is satisfied, as is the constraint $b_i \geq 1$.

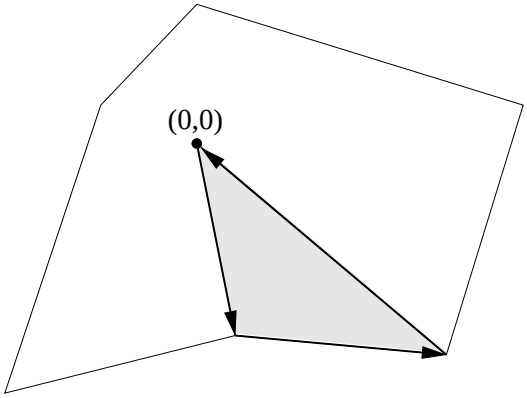

Figure 2.8: The origin is to the left of every counterclockwise-oriented edge in Q.

Therefore, if there exists a star-shaped drawing Q that is parallel with P, then there exists a feasible solution to the system in which $x_i = Q(v_i)$ for all $i \in \{1, \ldots, n\}$. Conversely, assume that there exists a feasible solution to the system. Let Q be the drawing of G such that $Q(v_i) = x_i$. It remains to prove that Q is a star-shaped polygon that is parallel with P. This is established by the following two claims.

Claim 2.3.1 *Q is parallel with P.*

Proof: From the first equality constraint of the system, for all $i \in \{1, \ldots, n\}$:

$$\begin{aligned} Q(v_{i+1}) - Q(v_i) &= x_{i+1} - x_i \\ &= \lambda_i \alpha_i \\ &= \lambda_i (P(v_{i+1}) - P(v_i)). \end{aligned}$$

Since $\lambda_i \geq 1$, it follows that Q is parallel with P. □

Claim 2.3.2 *Drawing Q is simple, and every point of Q is properly visible to the origin.*

Proof: For each $i \in \{1, \ldots, n\}$, let θ_i denote the inner angle of P at vertex v_i. Since the angles of a triangle sum to π, and every n-vertex polygon admits a decomposition into $n - 2$ triangles [15], it follows that

$$\sum_{i=1}^{n} \theta_i = (n-2)\pi.$$

From the constraints

$$x_i + \alpha_i a_i + \beta_i b_i = (0, 0)$$

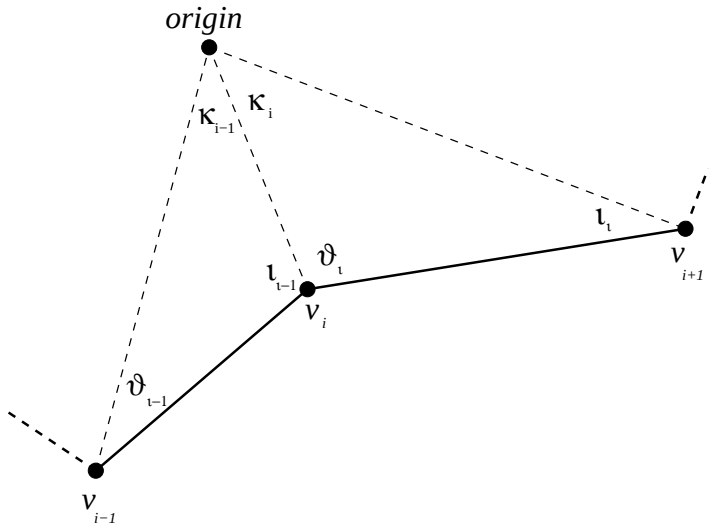

Figure 2.9: The interiors of adjacent triangles $\triangle_{i-1}$ and $\triangle_i$ of Q are disjoint.

and

$$b_i \geq 1$$

it follows that for all $i \in \{1, \ldots, n\}$, the origin lies strictly to the left of the line through $Q(v_i, v_{i+1})$ with respect to the direction of vector $Q(v_{i+1}) - Q(v_i)$. A triangle $\triangle_i$ is obtained by adding edges between the origin and points $Q(v_i)$ and $Q(v_{i+1})$, such that

$$Q(v_i), Q(v_{i+1}), (0, 0)$$

represents a counterclockwise traversal of $\triangle_i$.

Let ϑ_i denote the interior angle of $\triangle_i$ at $Q(v_i)$, let ι_i denote the interior angle of $\triangle_i$ at $Q(v_{i+1})$, and let κ_i denote the interior angle of $\triangle_i$ at the origin; see Figure 2.9. Since P and Q are parallel, for all $i \in \{1, \ldots, n\}$,

$$\theta_i = \vartheta_i + \iota_{i-1}.$$

To prove that Q is simple and that every point is properly visible to the origin, it suffices to prove that the interiors of adjacent triangles are pairwise disjoint, and that the sum of the angles at the origin of these triangles is equal to 2π.

For all $i \in \{1, \ldots, n\}$, since P and Q are parallel polygons, edges $Q(v_{i-1}, v_i)$ and $Q(v_i, v_{i+1})$ are non-intersecting and each has non-zero length. Since the origin lies to the left of both the line through $Q(v_{i-1}, v_i)$ with respect to the direction of vector $Q(v_i) - Q(v_{i-1})$, and the line through $Q(v_i, v_{i+1})$ with respect to the direction of $Q(v_{i+1}) - Q(v_i)$, it follows that the interiors of $\triangle_{i-1}$ and $\triangle_i$ are disjoint; recall Figure 2.9.

Finally, to see that the interior angles of the triangles at the origin sum to exactly 2π, observe that

$$\sum_{i=1}^{n} \kappa_i = \sum_{i=1}^{n} \pi - \vartheta_i - \iota_{i+1} = \pi n - \sum_{i=1}^{n} \theta_i = \pi n - \pi(n-2) = 2\pi.$$

□

In conclusion, there exists a feasible solution to the system if and only if there exists a simple, star-shaped polygon Q that is parallel with P, where $Q(v_i) = x_i$ for all $i \in \{1, \ldots, n\}$. This completes the proof of Theorem 2.3.3. □

2.4 Conclusion

We began this chapter with a discussion of linear morphs in $\mathbb{R}^d$, proving that for every pair of orthogonal drawings that admits a parallel morph, there exists a parallel morph that is a finite sequence of linear morphs. Each linear morph can be efficiently specified by its source and target drawings. For this reason, linear morphs serve as good "elementary moves" of parallel morphs. The number of linear morphs involved in a morph provides a measure of the complexity of the morph.

However, we have established only a trivial upper bound on the number of linear morphs required in a parallel morph between two orthogonal drawings in $\mathbb{R}^d$. In Chapter 7 we shall see that deciding parallel morphability for orthogonal drawings in $\mathbb{R}^3$ is PSPACE-hard. It is left as an open problem to determine whether deciding parallel morphability is in PSPACE. If the problem is not in PSPACE, then for any n it must be possible to construct a pair of parallel drawings in $\mathbb{R}^3$ with n vertices such that the number of distinct vertex orderings encountered in morphing from one drawing to the other is super-polynomial in n. Does such a family of pairs of drawings exist?

In Section 2.3, we studied linear morphs of polygons in the plane, proving that for every pair of parallel polygons, if one of these is star-shaped, then the linear morph maintains simplicity. As a consequence of this, if we are given a pair of parallel polygons such that there exists a third star-shaped polygon parallel with these two, the polygons will admit a parallel morph composed of two linear morphs. We prove that if a star-shaped polygon exists that is parallel to a given polygon, it can be determined by computing a feasible solution to a linear system of equalities and inequalities.

This result does not hold for polygons in general. It is easy to construct a pair of parallel polygons that do not admit a parallel morph comprising two linear morphs. On the other

hand, Guibas, Hershberger and Suri [34] have shown that every pair of parallel polygons will admit a parallel morph composed of $O(n \log n)$ linear morphs. Can it be efficiently decided whether a pair of parallel polygons admits a parallel morph composed of a sequence of two (or, in general, k) linear morphs? What is the complexity of approximating the minimum number of linear morphs required?

In Chapter 5 we prove it to be NP-hard to decide whether a pair of parallel (non-orthogonal) drawings in the plane admits a parallel morph. The proof is via a reduction from 3SAT. Parallel drawings are constructed that admit a parallel morph if and only if a given boolean expression is satisfiable. If the expression is satisfiable, then a parallel morph exists composed of five linear morphs. Therefore, deciding whether two drawings admit a parallel morph comprising five linear morphs is NP-hard. It may be that with a slight modification to our construction, fewer linear morphs suffice. We leave it as an open problem to determine whether this is so.

Chapter 3

Morphing Orthogonal Drawings in the Plane

3.1 Introduction

In this chapter, we will see that every pair of parallel simple orthogonal drawings in the plane admits a parallel morph. Our proof relies on the fact that, if we can augment an orthogonal drawing of a graph by adding an orthogonal drawing of a path between two of its vertices such that this path drawing contains an equal number of left and right turns, then there exists a parallel morph of the drawing that straightens the path drawing to a single edge. This technique is particular to the plane: generalizing to three dimensions would involve "flattening" an orthogonal surface via a parallel morph. However, such flattening is not always possible, as we prove in Chapter 6.

Recall, from Chapter 2 that if two drawings admit a parallel morph, then they will admit a parallel morph that is composed of a sequence of linear morphs. Since any linear morph is uniquely represented by its two terminal drawings, we can represent such a parallel morph by a sequence $(P^0, \ldots, P^k)$ of drawings, where P^0 and P^k are the source and target drawings; and, for each $i \in \{0, \ldots, k-1\}$, the linear morph between P^i and P^{i+1} is a parallel morph.

Our main result is that every pair of parallel simple orthogonal drawings of a connected graph will admit a parallel morph. Moreover, such drawings will admit a parallel morph composed of $O(n)$ linear morphs, where n is the number of vertices in the underlying graph. We present two polynomial-time algorithms that generate such a parallel morph—given a pair of simple orthogonal drawings, each algorithm generates a sequence of $O(n)$ parallel drawings that represents a parallel morph between the drawings.

The two algorithms have different runtime bounds, as well as different bounds on the number of drawings generated. Our first algorithm (see Section 3.2) has an asymptotically smaller time bound than our second algorithm (see Section 3.3). However, the bound on the number of drawings generated by the second algorithm is a constant factor smaller than the bound on the number generated by the first algorithm.

It is not clear which of these two algorithms provides a more visually appealing morph. Friedrich and Eades [28] identify "symmetrical movement" as a desirable trait in a morph for graph visualization. Our first algorithm generates a sequence linear morphs with a very simple form: in each linear morph all vertices move at the same speed in one of two opposite directions. This motion is highly uniform and symmetrical. In contrast, morphs produced by the second algorithm do not have this property. Instead, at any point in time each vertex may have a unique velocity (i.e. speed and direction). Thus, although morphs generated by the second algorithm may consist of fewer linear morphs, the symmetrical motions of vertices in morphs produced by the first algorithm may better preserve a user's "mental map".

To determine whether one of these algorithms is significantly better than the other for graph visualization seems to require experimentation. A suitable experiment might involve performing a sequence of the following tests: two identical drawings are displayed side-by-side to a user. The drawing on the right undergoes a parallel morph (generated by one of the two morphing algorithms) to a target drawing, while the drawing on the left remains static. Immediately upon completion of the morph, the user is asked to identify the vertices in the source drawing corresponding to a subset of vertices in the target, or vice versa. The time taken to answer these queries may indicate how well the user's mental map is preserved throughout the morph. Clearly, setting up such an experiment is fraught with difficulty. Even the production of suitable drawings for the experiment does not seem straightforward. Further exploration of this topic is beyond the scope of this thesis.

Let us briefly discuss the space requirements of our algorithms. Throughout this chapter, we assume that the coordinates of a single vertex can be stored in $O(1)$ space. Therefore, $O(n)$ space suffices to store a mapping from n vertices to n points in the plane. Since an orthogonal drawing in the plane has at most four edges incident on each vertex, a straight-line drawing with n vertices can be stored in $O(n)$ space.

The space required by each algorithm depends on the application. Both algorithms compute a sequence of drawings $(P^0, \ldots, P^k)$. In the *online* setting, as soon as each drawing P^i is computed it is sent directly to output. We are not required to store drawings that have been previously sent to output. In the *offline* setting, all intermediate drawings are

computed and are returned as output simultaneously when the algorithm terminates. In the offline setting, the morphing algorithm is required to store all of the drawings $(P^0, \ldots, P^k)$.

The online setting is most appropriate for an application in which we want to *animate* the morph in real time, i.e. to produce a sequence of intermediate drawings of the morph that are displayed in rapid succession, so as to give the appearance of motion. To animate a linear morph, we need only to know the two terminal drawings of the linear morph, and the start and end times of the linear morph. A snapshot of the intermediate drawing at any time during a linear morph can be obtained by linear interpolation. Thus, to animate the morphs produced by our parallel morphing algorithms, we need only keep two drawings of the sequence in memory at any one time. Preceding drawings of the sequence may be discarded. For the algorithms that we describe in this chapter we give space bounds for the online setting.

The algorithms of Sections 3.2 and 3.3 will compute parallel morphs between drawings of a connected graph. A pair of parallel simple orthogonal drawings do not necessarily admit a parallel morph when the underlying graph is disconnected. In Section 3.4 we give an easily-tested condition that determines whether or not a pair of parallel simple orthogonal drawings of a disconnected graph will admit a parallel morph. If a parallel morph exists, it can be efficiently computed.

In Section 3.5 we prove a lower bound on the number of linear morphs required in a parallel morph between two orthogonal drawings, and in Section 3.6 we discuss how edges change in length during a parallel morph produced by our algorithms. Concluding remarks appear in Section 3.7.

3.1.1 Preliminaries

Before introducing the morphing algorithms, we describe some concepts that will be useful to us throughout this chapter.

Definition 3.1.1 *A* rectangular drawing *is a simple orthogonal drawing in which every face—including the outer face—is bounded by a rectangle (Figure 3.1); the side of a rectangle may be subdivided by any number of vertices.*

Lemma 3.1.1 *The linear morph between parallel rectangular drawings is a parallel morph.*

Proof: Let P, Q denote a pair of parallel rectangular drawings. Let M denote the linear morph from P to Q. By Lemma 2.1.1 the intermediate drawings of M are pairwise parallel.

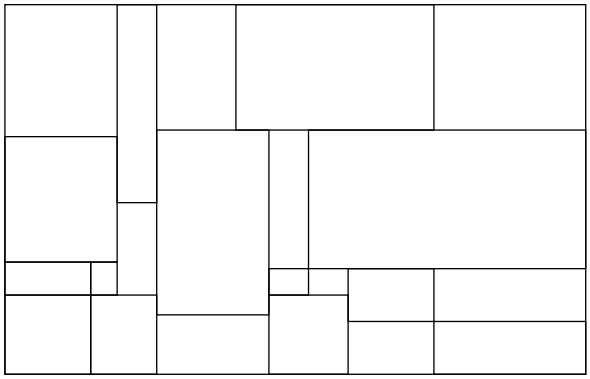

Figure 3.1: A rectangular drawing.

Each face (including the exterior face) of P is bounded by a rectangle, i.e. a star-shaped polygon. By Theorem 2.3.1, for each face of the source drawing, the elements of the drawing of the boundary of the face remain pairwise non-intersecting throughout M. Therefore, by Lemma 2.3.2 morph M preserves simplicity and is thus a parallel morph. □

If parallel drawings are rectangular, then generating a parallel morph is trivial. If not, then the morphing algorithms discussed in this chapter morph and *augment* the drawings to make them parallel and rectangular. Recall *augmentation* from Definition 2.3.5.

Lemma 3.1.2 *Let P and Q be parallel orthogonal drawings of a graph G, and let P' and Q' be parallel drawings of a graph G' that are augmentations of P and Q, respectively. Suppose that M' is a parallel morph from P' to Q'. Let M be the morph of G in which*

$$M(t; v) = M'(t; v)$$

for all $t \in [0, 1]$ and $v \in \mathrm{V}(G)$. Then, M is a parallel morph from P to Q.

Proof: Observe that no augmentation step either removes a vertex or changes its coordinates. Therefore, $\mathrm{V(G)} \subseteq \mathrm{V(G')}$. Hence, $M(t)$ is uniquely defined for all $t \in [0, 1]$. Further, $M(0) = P$ and $M(1) = Q$. Since M' moves each vertex along a continuous path over time, the same is true of M. Therefore, M is a morph from P to Q.

Let us prove that M keeps edges parallel. Suppose that $P(u, v)$ is a horizontal edge (i.e. parallel with the x-axis) such that $P_{\mathrm{x}}(u) < P_{\mathrm{x}}(v)$. By the definition of augmentation, there exists a sequence of one or more horizontal edges

$$P'(u, u_1), P'(u_1, u_2), \ldots, P'(u_k, v)$$

where

$$P'_{\mathrm{x}}(u) < P'_{\mathrm{x}}(u_1) < \ldots < P'_{\mathrm{x}}(u_k) < P'_{\mathrm{x}}(v).$$

Suppose that $M(t; u, v)$ is not parallel with $M(0; u, v)$ for some $t \in [0, 1]$. This implies that not all edges $M'(t; u, u_1), \ldots, M'(t; u_k, v)$ are parallel with their representations in $M'(0)$, contradicting the assumption that M' is a parallel morph. Hence, $M(t; u, v)$ is horizontal for all $t \in [0, 1]$. An analogous result holds for vertical edges. Therefore, M keeps edges parallel throughout. To prove that M is a parallel morph, it remains only to prove that M maintains simplicity.

For a proof-by-contradiction, assume that there exists some $t \in [0, 1]$ such that $M(t)$ self-intersects. By Lemma 1.3.2 (page 17), there exists an intermediate drawing in which one of the intersecting elements is a vertex. Hence, we may assume that there exists some $u, v, w \in \mathrm{V}(G)$ such that $M(t; w)$ intersects either $M(t; u)$ or $M(t; u, v)$. It follows that $M'(t; w)$ must intersect either one of the vertices

$$M'(t; u), M'(t; u_1), \ldots, M'(t; u_k), M'(t; v)$$

or edges

$$M'(t; u, u_1), M'(t; u_1, u_2), \ldots, M'(t; u_k, v)$$

where these vertices and edges are as described above. This contradicts the assumption that M' is a parallel morph. Therefore, M maintains simplicity and is thus a parallel morph. □

More than once in this chapter we use the technique of repeatedly morphing and augmenting source and target drawings until they become identical. By Lemma 3.1.2 an algorithm that can compute such a sequence of morphs and augmentations will immediately give us a parallel morph.

In Section 3.4, we use the technique of morphing and augmenting in both the source and target drawings until finally we arrive at a common drawing R. The final morph from the source to the target is then composed of the forward morph from the source to R, followed by the reverse of the morph from the target to R.

3.2 The First Algorithm

In this section, we prove the following theorem.

Theorem 3.2.1 *In the plane, every pair of parallel simple orthogonal drawings of a connected graph admits a parallel morph comprising a sequence of at most* $\frac{16n}{3} + O(1) \approx 5.33n + O(1)$ *linear morphs, where* n *is the number of vertices in the graph. The morph can be computed in* $O(n^2)$ *time and* $O(n)$ *space.*

We begin with an overview of the algorithm in Section 3.2.1 and then introduce our basic morphing step—the *slide*—in Section 3.2.2. A more detailed description of the algorithm is found in Section 3.2.3.

3.2.1 Overview

Our first algorithm works as follows. To begin, we augment the target drawing (via some sequence of augmentation steps) to make the target rectangular. In order to keep the source and target drawings parallel, each augmentation step in the target is reflected in the source: whenever a vertex is added to the target, or an edge is subdivided, we perform a corresponding augmentation step in the source drawing; and, whenever a new edge is added to the target, we first perform a parallel morph of the source drawing so that this edge can be added to the source. Then, we add this edge to the source, producing a new source that is parallel with the current target.

This process continues until the source and the target are rectangular. Then, a single linear morph suffices to take us the rest of the way from the source to the target. By Lemma 3.1.2, the sequence of parallel morphs performed on augmentations of the original source drawing gives us a parallel morph from the original source to the original target.

We note that our algorithm morphs only the source. In the online setting, at each stage of the computation we only need to maintain the current target and the current source. Every time we compute a new source that is the result of a linear morph of the previous source, we output this new source and discard the previous one. If instead, the algorithm morphed in both the source and the target, then it would need to store all previously computed drawings of the target, so that the entire sequence of drawings that determines the morph may be output in the correct order.

Let us consider the algorithm in greater detail. There are three stages. The first stage ensures that the boundary of the exterior face in both the source and target is a rectangle. We begin by adding a non-intersecting axis-aligned bounding box to the target, i.e. an axis-aligned box that contains the target in its interior face. Subdivide the upper horizontal edge of the bounding box by a vertex u_0 such that for some $v_0 \in \mathrm{V}(G)$, $\{u_0, v_0\}$ can be added to the drawing as a vertical edge without destroying simplicity. Figure 3.2 (left) shows an example of a target; Figure 3.2 (center) illustrates the target following the first stage of the algorithm.

To keep the source and target parallel, add a non-intersecting bounding box to the source—the vertices of this bounding box should correspond to the vertices of the bounding box added to the target. Subdivide the upper horizontal edge of the bounding box by vertex

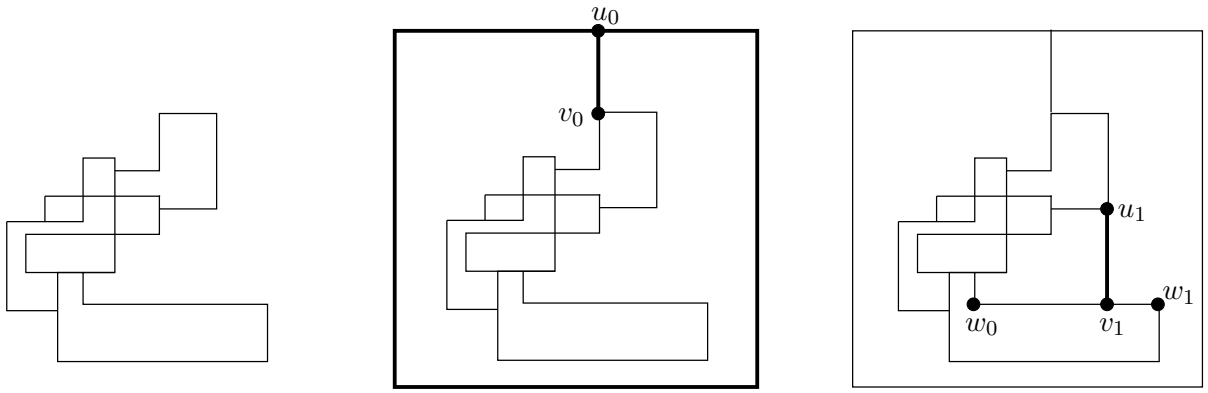

Figure 3.2: *Left:* the original target. *Center:* the target following the first stage. *Right:* the target after augmenting with one additional edge in the second stage.

u_0. At this point, we would like to add $\{u_0, v_0\}$ to the source as a non-intersecting vertical edge, as we have done in the target. This may not be immediately possible. In general, we will need to perform a parallel morph of the source before $\{u_0, v_0\}$ can be added. The fact that such a morph always exists for an orthogonal drawing is key to our algorithm. We describe this result in detail in Section 3.2.2.

This completes the first stage. At this point, the source and target drawings are parallel, the underlying graph is connected, and the boundary of the exterior face in both drawings is a rectangle.

In the second stage, the algorithm further modifies the source and target so that the boundary of each interior face is a rectangle. Until every face of the target is a rectangle, iterate as follows: augment the target by adding a non-intersecting vertical edge between some vertex $u_1 \in \mathrm{V}(G)$ and a horizontal edge $\{w_0, w_1\}$ of the target; see Figure 3.2 (right). As a special case, we might add a non-intersecting vertical edge between two vertices of V(G) with the same horizontal coordinate.

Adding the vertical edge between u_1 and $\{w_0, w_1\}$ in the target involves first subdividing $\{w_0, w_1\}$ by a vertex v_1 that has the same horizontal coordinate as u_1. It is not difficult to see that it requires only finitely many vertical edges to make the target rectangular.

To keep the source and target parallel, we partition $\{w_0, w_1\}$ by v_1 in the source drawing. We then perform a parallel morph of the source so that $\{u_1, v_1\}$ can be added to it as a vertical edge that does not destroy simplicity. This morph is performed in a manner similar to the morph performed in the first stage, and uses the procedure described in Section 3.2.2.

Upon completion of the second stage, both the source and the target are parallel simple orthogonal rectangular drawings. The third stage of the algorithm consists of a linear

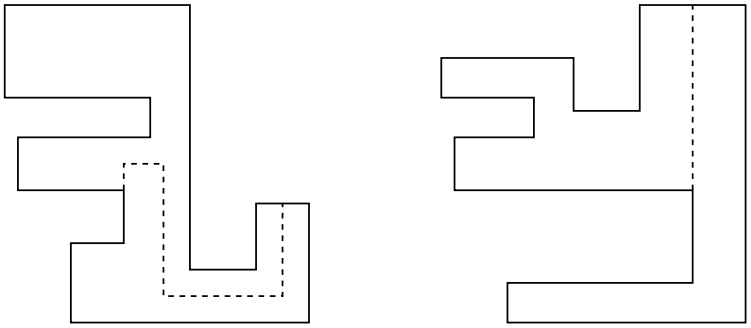

Figure 3.3: *Left:* A drawing of an orthogonal polygon (solid curve), with an orthogonal path (dashed curve). *Right:* the result of a parallel morph that straightens the dashed curve.

morph from the source to the target, i.e. the algorithm simply outputs the target. By Lemma 3.1.1, the linear morph applied in the third stage is a parallel morph. With that, we are done.

In the first two stages of the algorithm the vertex motions are *expansive* in the sense that edges are non-decreasing in length over time. This will be clear when we describe our morphing technique in the next section. In the final linear morph, each edge changes length monotonically; recall Lemma 2.1.1. Further discussion of edge-lengths and monotonicity in parallel morphs appears in Section 3.6 and in Chapter 4.

3.2.2 Slides

The first two stages of our morphing algorithm rely on the fact that whenever we add a vertical edge to the target, we can perform a parallel morph from the source to a new drawing in which the same vertical edge can be added without destroying simplicity. We describe a procedure performing such a morph in this section.

The idea behind this procedure is the following. Whenever a vertical edge $\{u, v\}$ is added to the target, we will compute in the source a non-intersecting orthogonal drawing of a path graph with end-vertices u and v—this path is generated in the source drawing by tracing around the boundary of the face that corresponds to the face containing $\{u, v\}$ in the target. Then, we perform a parallel morph of the union of source and path drawings in order to straighten the path to a single edge; Figure 3.3.

The morph itself is composed of a sequence of elementary motions that we call *slides*, which are themselves a type of linear morph. Each slide allows us to remove two vertices from the path. Hence, the number of slides required is proportional to the number of

vertices in the path drawing to be straightened. Further, slides are expansive in the sense that every edge is non-decreasing in length during a slide.

Not every drawing of a path can be straightened. Below, we describe a class of paths that is amenable to straightening. Let G_ϕ be a path graph such that $\mathrm{V(G_\phi)} = \mathrm{v_1}, \ldots, \mathrm{v_k}$ and $\mathrm{E(G_\phi)} = \{\mathrm{v_1}, \mathrm{v_2}\}, \ldots, \{\mathrm{v_{k-1}}, \mathrm{v_k}\}$, and let ϕ be a simple orthogonal drawing of G_ϕ. We call v_1 and v_k the *terminal vertices* of ϕ, and we call $\{v_1, v_2\}$ and $\{v_{k-1}, v_k\}$ the *terminal edges* of ϕ. We will refer to the other edges and vertices as *non-terminal*.

Suppose that we follow ϕ from v_1 to v_k. At each vertex $v_i \in \{v_2, \ldots, v_{k-1}\}$, we encounter either a left turn, a right turn, or no turn (i.e. the two edges incident on v_i are parallel). If we follow ϕ in the opposite direction, from v_k to v_1, each left turn becomes a right turn, and vice versa.

Definition 3.2.1 *A drawing ϕ of a path graph G_ϕ is* balanced *if we encounter an equal number of right and left turns as we traverse the drawing from one end to the other.*

Let P be an orthogonal drawing of a graph G. We define the union of two drawings in the natural way. We note that $P \cup \phi$ is itself a drawing only if every vertex that is common to both G and G_ϕ maps to the same coordinates in both drawings.

Definition 3.2.2 *Let P be a simple orthogonal drawing of a graph G and let $u, v \in \mathrm{V}(G)$, where $u \neq v$. A simple orthogonal drawing ϕ of a path graph G_ϕ with terminal vertices u and v is called a* straightening path *of P if the following hold:*

1. $\mathrm{V(G)} \cap \mathrm{V(G_\phi)} = \{\mathrm{u}, \mathrm{v}\}$;
2. *$P \cup \phi$ is a simple drawing; and*
3. *ϕ is balanced and a turn occurs at every non-terminal vertex.*

It follows from the definition of a straightening path that when we traverse a straightening path from one end to the other, the first and last edges have the same direction.

We say that a straightening path is *trivial* if it contains exactly two vertices. Otherwise, it is *non-trivial*.

Definition 3.2.3 *Let ϕ be a non-trivial straightening path of a simple orthogonal drawing, with underlying graph G_ϕ. A sequence $(v_\mathrm{a}, v_\mathrm{b}, v_\mathrm{c}, v_\mathrm{d})$ of vertices of $\mathrm{V(G_\phi)}$ is a* zigzag *of ϕ (see Figure 3.4) if:*

1. *$v_\mathrm{a}, v_\mathrm{b}, v_\mathrm{c}, v_\mathrm{d} \in \mathrm{V}(G_\phi)$ are four distinct vertices;*

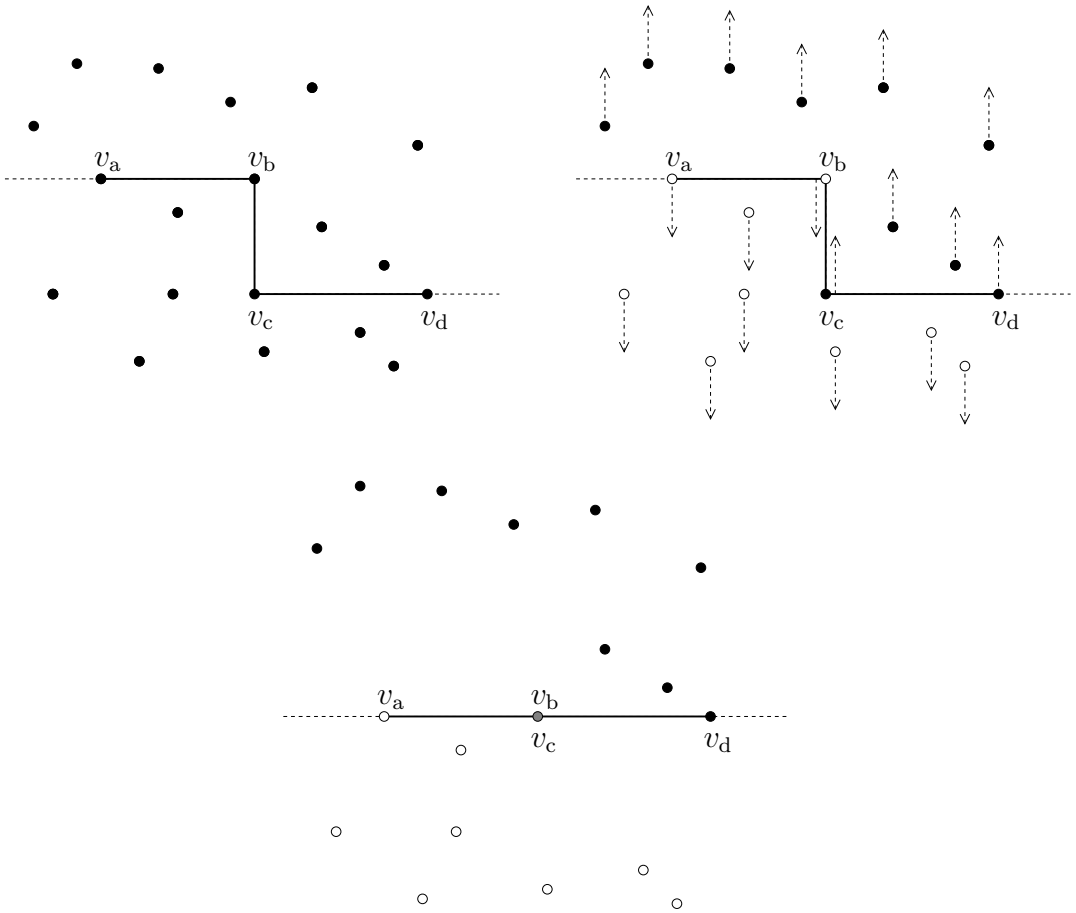

Figure 3.4: The slide operation on a zigzag.

2. $\{v_a, v_b\}, \{v_b, v_c\}, \{v_c, v_d\} \in E(G_\phi)$; *and,*

3. *as we follow* ϕ *from* v_a *to* v_d*, the turns we encounter at* v_b *and* v_c *comprise both a left and a right turn.*

Observation 3.2.1 *Every non-trivial straightening path contains at least one zigzag.*

Proof: Since every straightening path has a turn at each vertex and an equal number of left and right turns, if a straightening path is non-trivial, it contains a left turn and a right turn at adjacent vertices. □

We are now ready to define our *slide* operation. Let P be a simple orthogonal drawing and let ϕ be a non-trivial straightening path of P. Let (v_a, v_b, v_c, v_d) be a zigzag of ϕ. Without loss of generality, assume that $\phi(v_b, v_c)$ is a vertical edge, and that $\phi(v_b)$ resides above $\phi(v_c)$; see Figure 3.4 (top-left). If either $\phi(v_b, v_c)$ is vertical and $\phi(v_c)$ lies above

$\phi(v_\mathrm{b})$, or $\phi(v_\mathrm{b}, v_\mathrm{c})$ is a horizontal edge, then rotate and/or reflect the plane to obtain the same configuration as in this figure.

Note that Figure 3.4 depicts vertices of the drawing $P \cup \phi$ as dots; the only edges illustrated are those of the zigzag.

Let G and G_ϕ denote the graphs underlying P and ϕ, respectively. For every $v \in \mathrm{V}(G) \cup \mathrm{V}(G_\phi)$, we color v black if either:

1. $(P \cup \phi)(v)$ lies above the horizontal ray originating at $\phi(v_\mathrm{b})$ and directed leftward; or,

2. $(P \cup \phi)(v)$ lies on or above the horizontal ray originating at $\phi(v_\mathrm{c})$ and directed rightward.

Otherwise, we color v white; see Figure 3.4 (top-right).

Definition 3.2.4 *The* slide *of $P \cup \phi$ determined by zigzag $(v_\mathrm{a}, v_\mathrm{b}, v_\mathrm{c}, v_\mathrm{d})$ is the linear morph of $P \cup \phi$ in which the black vertices move upward a distance of*

$$\frac{||\phi(v_b), \phi(v_c)||_2}{2}$$

and the white vertices move downward this same distance.

Following a slide the vertices v_b and v_c coincide, and the edge $\{v_\mathrm{b}, v_\mathrm{c}\}$ has zero length; see Figure 3.4 (bottom).

Lemma 3.2.1 *Let P be a simple orthogonal drawing and let ϕ be a non-trivial straightening path of P. The slide of $P \cup \phi$ determined by a zigzag $(v_\mathrm{a}, v_\mathrm{b}, v_\mathrm{c}, v_\mathrm{d})$ of ϕ keeps every edge except $\{v_\mathrm{b}, v_\mathrm{c}\}$ non-zero in length and parallel with its representation in $P \cup \phi$, and every pair of elements except for v_b and v_c non-intersecting.*

Proof: Assume that $(v_\mathrm{a}, v_\mathrm{b}, v_\mathrm{c}, v_\mathrm{d})$ appears in $P \cup \phi$ in the same configuration shown in Figure 3.4. If not, perform a reflection and/or rotation of the plane. The lemma is established by the following claims.

Claim 3.2.1 *The pair v_b and v_c is the only pair of vertices to intersect during the slide.*

Proof: Two vertices can intersect only if they have equal horizontal coordinates in $P \cup \phi$, and have been assigned different colors. Further, the white vertex must reside above the black vertex. It is not difficult to see that the only pair that meets these criteria is v_b and v_c. Therefore, v_b and v_c are the only vertices that will intersect during the slide. □

Claim 3.2.2 *Each edge except $\{v_b, v_c\}$ remains non-zero in length and parallel with its image in $P \cup \phi$ throughout the slide.*

Proof: Let $\{u, v\}$ be an edge of $P \cup \phi$ such that $\{u, v\} \neq \{v_b, v_c\}$. Suppose that $(P \cup \phi)(u, v)$ is vertical. Hence, the horizontal coordinates of u and v are equal in $P \cup \phi$. Since vertices move only with respect to the vertical axis, in all intermediate drawings of the slide the horizontal coordinates of u and v are equal. By Claim 3.2.1, u and v do not intersect during the slide, thus $\{u, v\}$ does not attain zero length. Therefore, if $(P \cup \phi)(u, v)$ is vertical, edge $\{u, v\}$ remains parallel in all intermediate drawings of the slide, and non-zero in length.

Next, suppose that $(P \cup \phi)(u, v)$ is horizontal. Since horizontal coordinates of vertices do not change during the slide, $\{u, v\}$ remains non-zero in length during the slide. Suppose that $\{u, v\}$ does not remain horizontal. It must be that u and v have been assigned different colors. Observe that this can happen only if $(P \cup \phi)(u, v)$ intersects the closed segment between $\phi(v_b)$ and $\phi(v_c)$, contradicting the assumption that $P \cup \phi$ is simple. Therefore, $\{u, v\}$ remains horizontal throughout the slide, and non-zero in length. □

Claim 3.2.3 *No two elements of $P \cup \phi$ intersect during the slide, except for v_b and v_c.*

Proof: By Lemma 1.3.2 (page 17), if a pair of elements intersect during the morph, then some intermediate drawing of the slide exhibits either vertex-vertex or vertex-edge intersection. We have already established that vertex-vertex intersection cannot occur. Below, we prove that vertex-edge intersection cannot occur either.

Let us assume that for some vertex u where $v_b \neq u \neq v_c$, and some edge $\{v, w\}$, there exists an intermediate drawing of the slide in which u intersects $\{v, w\}$. Since these elements intersect during a vertical motion of the vertices, it must be that the vertical line through $(P \cup \phi)(u)$ intersects $(P \cup \phi)(v, w)$.

If $(P \cup \phi)(v, w)$ is vertical, then u must intersect either v or w at some point during the slide strictly before it intersects $\{v, w\}$. By Claim 3.2.1 this cannot happen. Therefore, $(P \cup \phi)(v, w)$ cannot be vertical.

So, assume that $(P \cup \phi)(v, w)$ is horizontal. The vertical line through $(P \cup \phi)(u)$ intersects $(P \cup \phi)(v, w)$. We can subdivide $\{v, w\}$ by a vertex that has the same horizontal coordinate as $(P \cup \phi)(u)$ without affecting the simplicity of the slide. By Claim 3.2.1, u will not intersect this new vertex during the slide. Therefore, $(P \cup \phi)(u, v)$ cannot be horizontal. □

With that, we have completed our proof of Lemma 3.2.1. □

Having introduced the slide operation, we now state our key morphing lemma.

Lemma 3.2.2 *Let P be a simple orthogonal drawing of a graph G such that $u, v \in \mathrm{V}(G)$, and let ϕ be a straightening path of P with terminal vertices u and v, such that the terminal edges of ϕ are vertical. There exists a parallel morph from P to a drawing R such that:*

1. *$\{u, v\}$ can be added to R as a vertical edge without destroying simplicity; and*
2. *the parallel morph from P to R is composed of at most $(k-2)/2$ slides, where k is the number of vertices in ϕ.*

The parallel morph can be computed in $O(nk)$ time and $O(n+k)$ space.

Proof: For the moment, let us ignore the algorithmic requirements of computing a parallel morph. We prove by induction on k that there exists a parallel morph from P to R that is composed of $(k-2)/2$ slides.

Suppose that $k = 2$. In this case ϕ is trivial, hence $P(u)$ and $P(v)$ have equal horizontal coordinates so no slides are required, and $(2-2)/2 = 0$. Assume that $(k-4)/2$ slides suffice for straightening paths of $k-2$ vertices, where the terminal vertices of the straightening path are u and v, and the terminal edges are vertical.

Since straightening paths are balanced and have a turn at each non-terminal vertex, ϕ contains an even number of vertices. Thus we may assume that k is an even number greater than or equal to four. By Observation 3.2.3, ϕ contains a zigzag $(v_{\mathrm{a}}, v_{\mathrm{b}}, v_{\mathrm{c}}, v_{\mathrm{d}})$. Perform the slide of $P \cup \phi$ determined by this zigzag. By Lemma 3.2.1, the slide keeps all edges of $P \cup \phi$ parallel and non-zero in length and all elements are pairwise non-intersecting, except for the vertices v_{b} and v_{c} which coincide at the very end of the slide—recall Figure 3.4 (bottom). Let P' denote the drawing of G following the slide, and let ϕ' denote the drawing of the path graph of ϕ following the slide.

We observe that $\phi'(v_{\mathrm{a}}, v_{\mathrm{b}})$ and $\phi'(v_{\mathrm{c}}, v_{\mathrm{d}})$ are parallel, and that $\phi'(v_{\mathrm{b}}) = \phi'(v_{\mathrm{c}})$. If either $\{v_{\mathrm{a}}, v_{\mathrm{b}}\}$ or $\{v_{\mathrm{c}}, v_{\mathrm{d}}\}$ is a terminal edge of ϕ then both of these edges are vertical in ϕ'. Remove v_{b} and v_{c} from ϕ', along with the three edges of the subpath $\{v_{\mathrm{a}}, v_{\mathrm{b}}\}$, $\{v_{\mathrm{b}}, v_{\mathrm{c}}\}$ and $\{v_{\mathrm{c}}, v_{\mathrm{d}}\}$. Add the edge $\{v_{\mathrm{a}}, v_{\mathrm{d}}\}$. Clearly, $\phi'(v_{\mathrm{a}}, v_{\mathrm{d}})$ is either horizontal or vertical.

At this point, ϕ' is a straightening path of P' with terminal vertices u and v, such that ϕ' has $k-2$ vertices in total. Further, the terminal edges of this straightening path are vertical—or, in the case that only a single edge remains, the single edge is vertical. By our inductive assumption, there exists a parallel morph from P' to R composed of $(k-4)/2$ slides. Therefore, a total of $(k-4)/2 + 1 = (k-2)/2$ slides suffice for a parallel morph from P to R.

The above proof-by-induction immediately implies an algorithm for computing a parallel morph from P to R. Let us consider the time and space requirements of this algorithm.

For each slide, we can compute a new drawing from a previous one in time and space proportional to the number of vertices in the union of the drawing and straightening path, i.e. $O(n+k)$. The only complication is deciding which zigzag to eliminate next from the straightening path.

Trivially, we can decide which pair to eliminate next in $O(k)$ time by scanning the entire straightening path for adjacent turns of opposite orientation. Since there are $O(k)$ slides in total, with this approach the morph can be computed in $O(nk+k^2)$ time.

A more efficient approach is to maintain a queue of zigzags of the straightening path. To determine which zigzag to eliminate next, remove the zigzag at the head of the queue, and update the queue as needed. Each update can be performed in constant time. Hence, with this approach the morph can be computed in $O(nk)$ time.

In the online model, we only need to store a constant number of drawings at every point of the computation. Therefore, the morphing algorithm runs in $O(n+k)$ space. □

3.2.3 Algorithmic details

In this section, we complete the proof of Theorem 3.2.1. In particular, we show how to compute straightening paths and prove bounds on their complexity.

First stage

Recall that in the first stage the algorithm augments both the source and target drawings with the addition of a bounding box. In the target, subdivide the upper horizontal edge of the bounding box with a vertex u, and add a vertical edge $\{u, v\}$ between u and some $v \in \mathrm{V}(G)$. To make the source and target parallel, subdivide the upper horizontal edge of the bounding box in the source drawing by u and perform a parallel morph from the source so that in the resulting drawing $\{u, v\}$ can be added as a vertical edge without destroying simplicity.

By Lemma 3.2.2, the source drawing admits the required parallel morph if a straightening path can be added with terminal vertices u and v such that the terminal edges of this straightening path are vertical. Below, we prove that such a path always exists. First, let us introduce a concept that will find use several times in this chapter.

Definition 3.2.5 *Let P be a simple orthogonal drawing and let $\epsilon > 0$ be a real number. The* ϵ-shell *of a given face of P is the locus of all points $x \in \mathbb{R}^2$ that lie within the open interior of the face, such that the L_∞ distance between x and the closest point of P is exactly ϵ.*

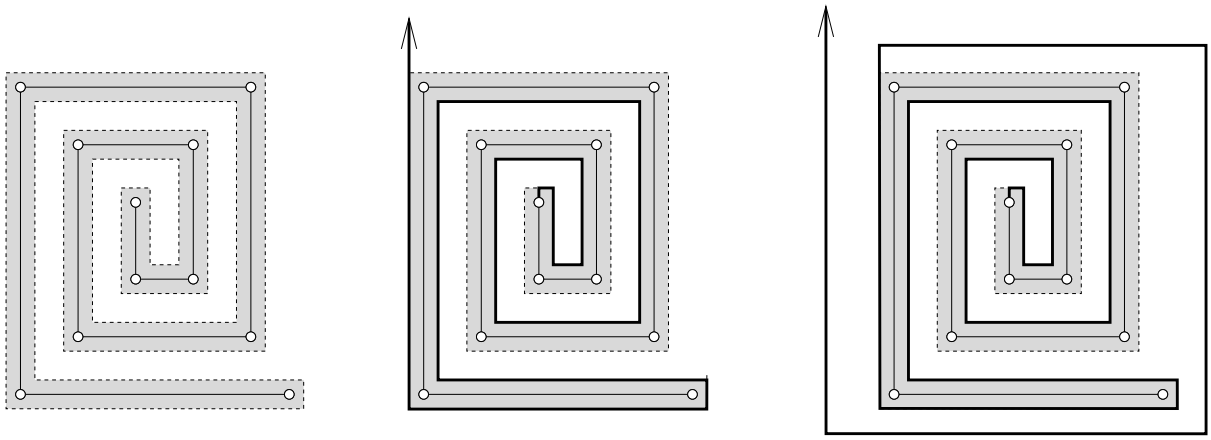

Figure 3.5: *Left:* a drawing with its shell; the interior of the shell is shown shaded and the white dots are vertices of the drawing. *Center:* a path clockwise around the shell with an excess of four left turns. *Right:* a balanced path drawing.

Figure 3.5 (left) illustrates a simple orthogonal drawing of a path graph. The drawing has only a single (exterior) face. The solid segments are edges and the white dots are vertices of this drawing and the dashed curve is an ϵ-shell.

Observation 3.2.2 *For every face of a simple orthogonal drawing P and every $\epsilon \in (0, \delta/2)$, where δ is the minimum feature size of P, the ϵ-shell is non-empty and unique, and does not intersect P.*

An ϵ-shell may be considered to be a simple orthogonal drawing of a cycle graph, i.e. an orthogonal polygon. Thus, it is permissible to speak of edges and vertices of an ϵ-shell.

Lemma 3.2.3 *In the target drawing, let $\{u, v\}$ be the vertical edge added in the first stage of the algorithm, such that v is a vertex of the input graph G, and u subdivides the upper horizontal edge of the bounding box. Then, in the source, the upper horizontal edge of the bounding box can be subdivided by u such that there exists a straightening path of the source with the following properties:*

1. *the straightening path has terminal vertices u and v;*
2. *the terminal edges of the straightening path are vertical; and*
3. *the straightening path contains at most $\frac{8n}{3} + O(1)$ vertices, where n is the number of vertices in the original graph prior the addition of the bounding box.*

Such a straightening path can be computed in $O(n)$ time.

Proof: Let P and Q denote the original source and target drawings, respectively, prior to the addition of the bounding boxes. Let δ be the minimum feature size of P. Fix ϵ such that $\epsilon \in (0, \delta/2)$.

Trace out a drawing of a path graph as follows: beginning at $P(v)$, move upward distance ϵ. This puts us at a point on the ϵ-shell of the exterior face of P. Follow the edges of this ϵ-shell either clockwise or counterclockwise until we reach one of the highest vertices of the ϵ-shell. The decision to go either clockwise or counterclockwise around the ϵ-shell depends on which of these two options results in the fewest turns in the path.

A vertical edge can be extended upward from any of the highest vertices of the ϵ-shell without intersecting either P or the ϵ-shell. Therefore, in the presence of the bounding box the path can be extended upward from any highest vertex of the ϵ-shell to the bounding box. Locate u on the bounding box at the point at which it is hit by the path. Figure 3.5 (center) illustrates a drawing of a path that is the result of tracing clockwise around the ϵ-shell (the bounding box is not shown).

The drawing of a path graph described above is simple and has terminal vertices u and v. Its terminal edges are vertical. Further, the union of P with this path drawing is itself a simple drawing. Even so, the path drawing might not be a straightening path since it is not necessarily balanced, i.e. we might encounter an excess of either left or right turns in the path drawing as we traverse it from v to u. For example, the path illustrated in Figure 3.5 (center) has an excess of eight left turns when we traverse it in the direction of the arrow.

The excess number of turns in the path drawing will be some multiple of four, since the path is traced out starting and ending in the same direction, and each turn changes the direction by an angle of $\pi/2$. If the number of excess turns is zero, then the path is balanced. Otherwise, we turn it into a balanced path drawing by adding a number of turns equal in number to the excess, but of opposite orientation. The path is extended around P either clockwise or counterclockwise as appropriate, hugging the bounding box; see Figure 3.5 (right).

Let ϕ denote the balanced path drawing generated by the above procedure. It is clear that ϕ is a straightening path of P. The ϵ-shell of the outer face of P can be computed in $O(n)$ time. Hence, this straightening path can also be computed in $O(n)$ time.

Claim 3.2.4 *The number of vertices in any ϵ-shell of P is at most $\frac{8n-4}{3}$, where $n \geq 2$ is the number of vertices in P.*

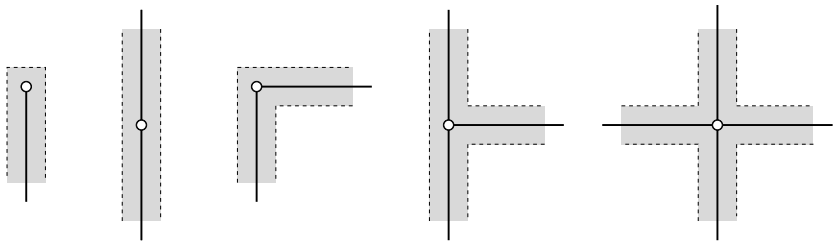

Figure 3.6: The vertices contributed to the ϵ-shell by each vertex of G.

Proof: It suffices to consider the number of vertices in the ϵ-shell of a drawing for which the underlying graph is a tree. To the contrary, suppose that G contains a cycle and let e be an edge of this cycle. There exist two faces f_1, f_2 of P with e on the boundary of each. Removing e from P results in a face f whose closed interior is identical to the union of the closed interiors of f_1 and f_2. The ϵ-shell of f has no fewer vertices than the larger of the ϵ-shells f_1 and f_2. So, let us assume that G is a tree.

The number of vertices in the ϵ-shell of P that are contributed by a particular vertex of G depends on the degree of the vertex; see Figure 3.6. Vertices of degree one, two and three each contribute (at most) two vertices to the shell, while those of degree four each contribute four vertices to the shell.

Suppose that the tree G of n vertices contains n_4 vertices of degree four, and $(n - n_4)$ vertices of lower degree. Since a tree of n vertices contains $(n - 1)$ edges, it follows that

$$4n_4 + (n - n_4) \leq 2(n - 1).$$

Therefore,

$$n_4 \leq \frac{n-2}{3}.$$

So the number of the vertices in the ϵ-shell is at most

$$4n_4 + 2(n - n_4) = 2n_4 + 2n \leq 2\left(\frac{n-2}{3}\right) + 2n = \frac{8n-4}{3}.$$

□

The bound in Claim 3.2.4 is tight: for any n such that $\frac{n-2}{3}$ is an integer, there exists a drawing of a tree graph with n vertices, such that for sufficiently small ϵ, the ϵ-shell contains $\frac{8n-4}{3}$ vertices. For example, the drawing illustrated in Figure 3.7 has 11 vertices and the ϵ-shell contains $\frac{8 \cdot 11 - 4}{3} = 28$ vertices.

Recall that ϕ may traverse the ϵ-shell of P in either a clockwise or counterclockwise direction. To minimize the size of ϕ, it should go the direction that results in a smaller

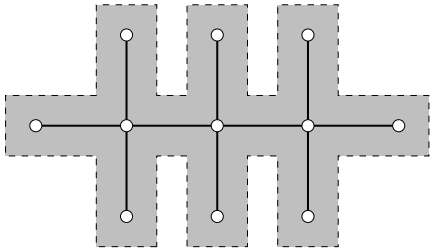

Figure 3.7: A drawing of a tree for which the bound in Claim 3.2.4 is tight.

path. Hence, the number of vertices of ϕ that lie on the ϵ-shell of P is at most

$$\frac{1}{2}\left(\frac{8n-4}{3}\right) + O(1) = \frac{4n}{3} + O(1).$$

Drawing ϕ also contains vertices that are not on the ϵ-shell of P. In particular, there are the two terminal vertices, u and v; and, the vertices that serve to balance ϕ. The number of balancing vertices is at most equal to the number of vertices of ϕ that lie on the ϵ-shell. Therefore, the total number of vertices in the straightening path ϕ can be bounded by the expression

$$2\left(\frac{4n}{3} + O(1)\right) + 2 = \frac{8n}{3} + O(1).$$

□

Lemma 3.2.4 *The first stage of the algorithm uses at most* $\frac{4n}{3} + O(1)$ *slides. The morph can be computed in* $O(n^2)$ *time and* $O(n)$ *space.*

Proof: This follows trivially from Lemma 3.2.2 and Lemma 3.2.3. □

It might be possible to improve on the bound of $\frac{8n}{3} + O(1)$ on the number of vertices in a straightening path, as given in Lemma 3.2.3. It is easy to see that for spiral drawings such as the one illustrated in Figure 3.5, with the right target drawing the algorithm generates a straightening path having $2n + \Theta(1)$ vertices. We know of no class of drawings for which the algorithm generates asymptotically larger straightening paths.

Although the bound on the number of vertices in an ϵ-shell given in Claim 3.2.4 is tight, drawings whose ϵ-shells meet this bound (e.g. the drawing in Figure 3.7) necessarily have several degree-1 vertices. For each degree-1 vertex, there exists at least one edge in the ϵ-shell that is the middle edge of a zigzag, i.e. it has turns of opposite orientation at its vertices. The vertices of such an edge do not cause additional balancing vertices to be added in the straightening path. This observation may be important in reducing the bound on the number of vertices in a straightening path generated by our algorithm.

The goal in generating a straightening path is to generate a straightening path with the fewest vertices possible. For this purpose, one might instead employ the algorithm of Das and Narasimhan [19] as a subroutine. This algorithm generates a minimum-vertex orthogonal path between two points of a orthogonal drawing in $O(n \log n)$ time, i.e. a path that exhibits *minimum link distance*. In our proof, we have used an algorithm that intersects an ϵ-shell (except in the terminal edges of the path) because this approach aids in bounding the size of straightening paths generated. Perhaps by using a different straightening-path algorithm, a better worst-case bound on the size of a straightening path can be obtained.

Second stage

In the second stage, while there exists an interior face in the target drawing that is not rectangular, the target is augmented with the addition of a vertical edge between a vertex and a horizontal edge (or, possibly between two vertices). To keep the source and target drawings parallel, it is generally necessary to morph from the current source drawing to a new source such that the same vertical edge that was added to the target can be added to the source. Note that the algorithm could instead augment the drawings using both horizontal and vertical edges. However, using only edges of one orientation simplifies the subsequent analysis.

Each parallel morph of the source drawing works in a manner similar to the morph of the source performed in the first stage: first generate a straightening path, then use the morphing procedure described in Lemma 3.2.2 to straighten the straightening path.

When adding edges to the target drawing, the algorithm should extend a vertical edge from a vertex only if each of the two faces generated has strictly fewer corners than does the original face. Observe that we cannot extend a vertical edge from a degree-4 vertex and we will never need to extend an edge from a degree-3 vertex; see Figure 3.8. However, we might need to extend vertical edges from the degree-1 and degree-2 vertices. In particular, from each degree-1 vertex we will extend either one or two edges, while from each degree-2 vertex we will extend at most one edge.

It is easy to see that we will not need to extend a vertical edge from any of the five vertices added on the bounding box in the first stage. Also, whenever we subdivide a horizontal edge with a new vertex in order to accommodate a new vertical edge, the newly added vertex has degree three, and so we will not need to extend another edge from the newly added vertex. Every vertical edge that is added to the target in the second stage is incident to at least one of the vertices of the original target drawing.

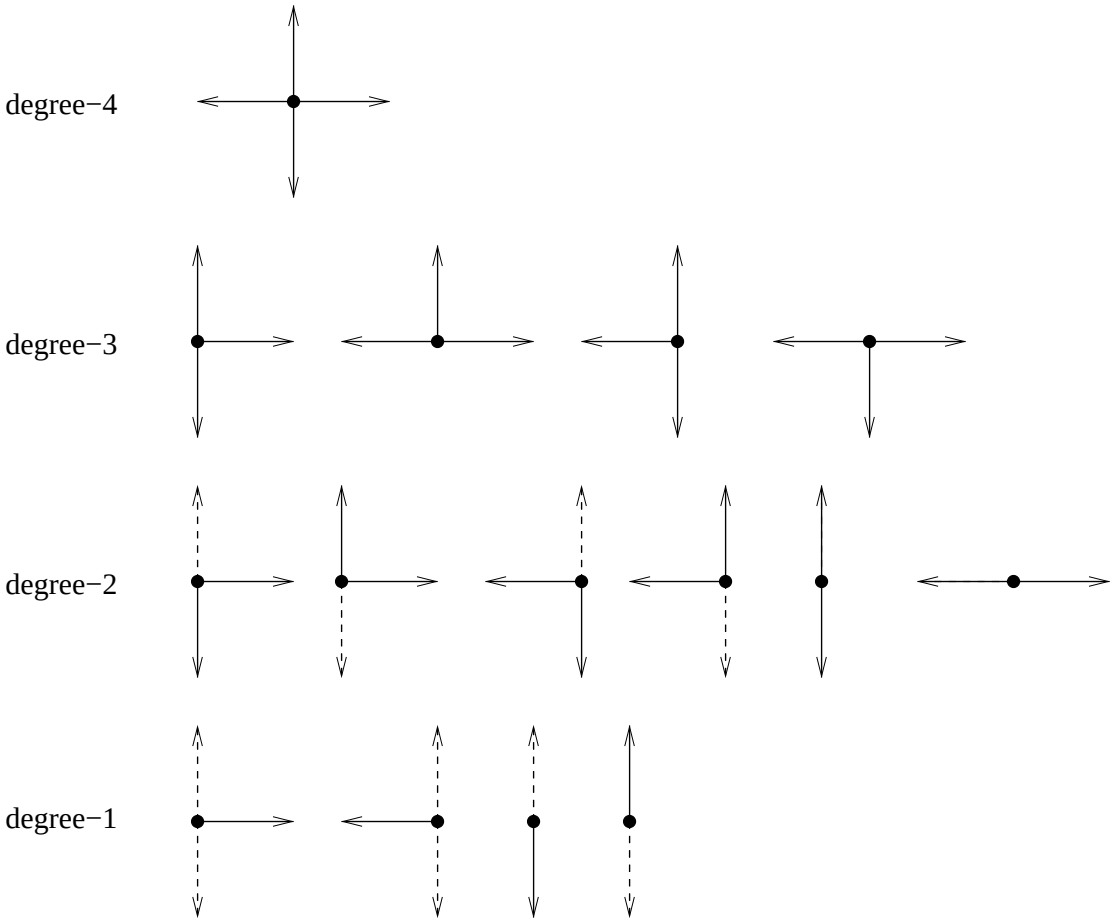

Figure 3.8: When adding vertical edges in the target drawing, we only need to extend edges from degree-1 and degree-2 vertices. Here, vertices are dots, edges of the target are solid lines, and each dashed line represents a new vertical edge extended from a vertex.

So, each edge added in the second stage lies either between two vertices of the original input graph, or between one original vertex and a newly added vertex. Let us bound the number of edges of each type. This bound will be important to us shortly, when we bound the number of slides required in the second stage. First, in Lemma 3.2.5 we prove a relationship between the vertices in the input graph and the number of edges added. Then, in Lemma 3.2.6 we use Lemma 3.2.5 to prove a relationship between the number of edges of each type and the number of vertices in the input.

Lemma 3.2.5 *Let G be a connected graph of maximum vertex degree four. Then,*

$$2n_1 + n_2 \leq \frac{4n+4}{3}$$

where n_i is the number of degree-i vertices in G, and $n = |\mathrm{V(G)}|$.

Proof: Observe that

$$\begin{aligned} n_1 + n_2 + n_3 + n_4 &= n \\ \implies \frac{8(n_1 + n_2 + n_3 + n_4)}{3} &= \frac{8n}{3}. \end{aligned} \tag{3.1}$$

Next, observe that

$$\begin{aligned} n_1 + 2n_2 + 3n_3 + 4n_4 &\geq 2(n-1) \\ \implies \frac{-2(n_1 + 2n_2 + 3n_3 + 4n_4)}{3} &\leq \frac{-2(2n-2)}{3}. \end{aligned} \tag{3.2}$$

By summing (3.1) and (3.2), we get that

$$2n_1 + \frac{4n_2}{3} + \frac{2n_3}{3} \leq \frac{4n+4}{3}.$$

In conclusion,

$$2n_1 + n_2 \leq 2n_1 + \frac{4n_2}{3} + \frac{2n_3}{3} \leq \frac{4n+4}{3}.$$

□

Lemma 3.2.6 *Let n be the number of vertices in the connected input graph G. Following the second stage, we have that*

$$k_1 + 2k_2 \leq \frac{4n+4}{3}$$

where k_1 is the number of edges added during the second stage with a single vertex in G, and k_2 is the number of edges added with both vertices in G.

Proof: Let n_1 and n_2 denote the number of degree-1 and degree-2 vertices in G, respectively. We have already seen that each degree-1 vertex may be connected to at most two edges in the second stage, while each degree-2 vertex may be connected to at most one. Trivially,

$$k_1 + 2k_2 \leq 2n_1 + n_2.$$

Hence, by Lemma 3.2.5,

$$k_1 + 2k_2 \leq \frac{4n+4}{3}.$$

□

As each edge is added to the target, we generate a corresponding straightening path in the source drawing. We then morph the source using this straightening path, and add the new edge to the resulting source drawing. By Lemma 3.2.6, $O(n)$ edges are added to the

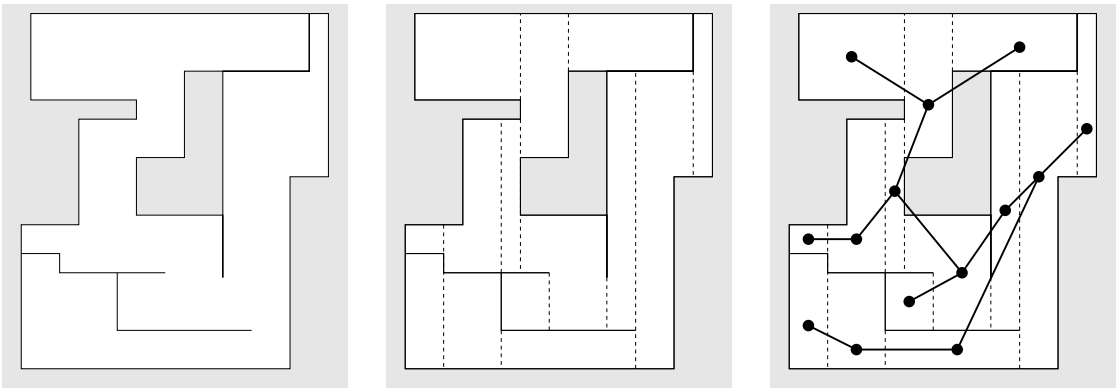

Figure 3.9: *Left:* a simple orthogonal drawing in which all vertices and edges are incident to a single interior face (the shaded region is exterior to the face). *Center:* partition the face into rectangular faces by vertical edges. *Right:* the dual of the partition.

target in the second stage. It is not difficult to show that in the worst case, $\Omega(n)$ edges are added to the target. If we are not careful, the number of vertices in all the straightening paths might not be $O(n)$, in which case, the total number of slides used to morph the source drawing in the second stage will not be $O(n)$.

With some care, it is possible to ensure that each straightening path has constant complexity. The following simple lemma is key.

Lemma 3.2.7 *Given a simple (connected) orthogonal drawing such that some interior face f is not rectangular, we can augment the drawing with a vertical edge between two vertices—possibly following the subdivision of a horizontal edge—such that the edge partitions f into two faces, one of which is rectangular.*

Proof: Strip away all vertices and edges from the drawing of the graph that are not incident on f, as illustrated in Figure 3.9 (left). For every vertex that remains, extend a vertical edge upward and/or downward through f—subdividing horizontal edges with vertices as needed—such that each newly added vertical edge lies in the interior of f and maintains the simplicity of the drawing; see Figure 3.9 (center). These newly added edges partition f into a number of rectangular faces.

Let H be the graph based on the partition of f defined as follows: associate each interior rectangular face with a vertex; connect two vertices of H by an edge if the corresponding faces are adjacent at one of the newly added edges; see Figure 3.9 (right). If there exists a cycle in H then the graph underlying the original target drawing is disconnected. However, since the original graph is connected, H is a tree. The edge that is incident with any leaf

of H corresponds to a single newly added edge of the target. This newly added edge partitions f into two faces, one of which is rectangular. □

For each iteration of the second stage, we want to add a vertical edge to the target that partitions an interior face, such that one of the two faces induced by the partition is rectangular. By Lemma 3.2.7 such an edge always exists. To compute the corresponding straightening path of the edge in the source, we can follow the edges in the source that correspond to the rectangular face of the target. In this way we can always find such a straightening path having $O(1)$ vertices. We explain this in more detail in the proof of Lemma 3.2.8. Now, let us consider how we might efficiently compute the sequence in which edges should be added to the target.

At the beginning of the second stage, we compute the set of all vertical edges that will be added to the target during this stage to make it rectangular. This is just the rectangular case of trapezoidation, and thus can be computed in deterministic $O(n)$ time with the algorithm of Chazelle [15], or in expected $O(n)$ time with the simpler randomized algorithm of Amato et al. [2].

From this decomposition, compute the dual graph H as described in Lemma 3.2.7. In general, H will be a forest, with one tree per interior face of the drawing. It is not difficult to see that the H can be computed from the trapezoidal decomposition in an additional $O(n)$ time.

Initialize a queue containing all leaves of H in $O(n)$ time. For each iteration of the second stage, to decide which edge to add next in the target, remove a leaf l from the queue, and delete this leaf from H, updating the queue if any new leaf results from this deletion. The next vertical edge to be added to the target is the one that corresponds to the edge formerly connecting l to H. For each iteration of the second stage, the process of computing the next edge to add to the target can be performed in $O(1)$ time.

Lemma 3.2.8 *The second stage of the algorithm uses at most $4n + 4$ slides. The morph can be computed in $O(n^2)$ time and $O(n)$ space.*

Proof: From the above discussion, the preprocessing step takes $O(n)$ time. Following the preprocessing step, each edge that is added to the target can be determined in $O(1)$ time. Moreover, when an edge $\{u, v\}$ is added to the target, one of the two faces created in the target is rectangular. Let us now consider the problem of computing a straightening path in the source drawing between u and v, such that the terminal edges of this straightening path are vertical.

Without loss of generality, assume that in the target drawing $\{u, v\}$ lies on the right side of the rectangular face induced by the partition. Figure 3.10 (top) illustrates all possible

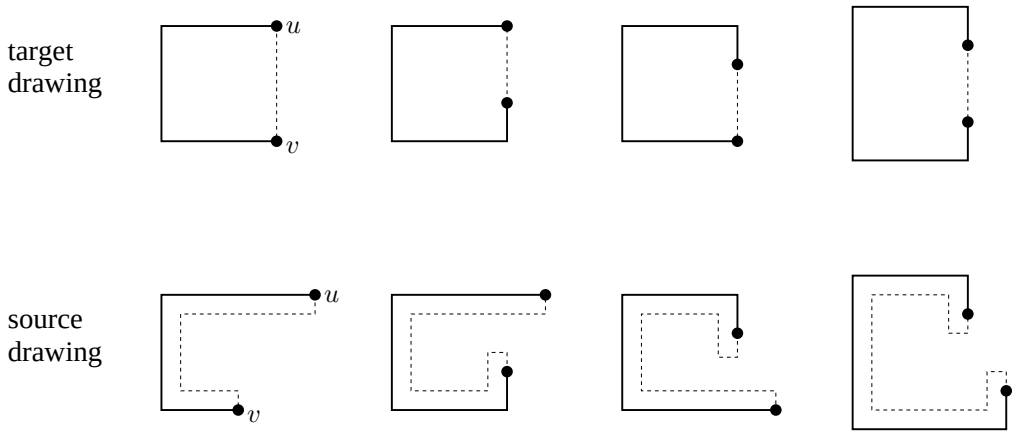

Figure 3.10: The possible configurations of the newly added edge $\{u, v\}$ and corresponding straightening paths in the source drawing.

configurations of $\{u, v\}$ with respect to the edges on the boundary of the rectangular face; the dots represent the vertices u and v, the dashed line represents $\{u, v\}$, and the solid curve represents the edges that bound the rectangular face lying to the left of $\{u, v\}$. Each segment of the solid curve may be composed of a number of vertices and edges in the drawing.

In the source drawing, we cannot add $\{u, v\}$ if either u and v do not have equal horizontal coordinates in the source, or other parts of the drawing are in the way. Let δ denote the minimum feature size of the current source drawing. Generate a straightening path in the source drawing between u and v that follows the ϵ-shell of the face that contains it, where $\epsilon = \delta/3$; see Figure 3.10 (bottom).

The straightening path for the leftmost configuration illustrated in Figure 3.10 has six vertices (including u and v), the middle two have eight, and the rightmost has ten. In any of these cases, the straightening path can be computed in $O(1)$ time.

By Lemma 3.2.2, in the rightmost case, $\{u, v\}$ can be added to the source drawing following a parallel morph consisting of at most $(10 - 2)/2 = 4$ slides. In the other cases, $\{u, v\}$ can be added after at most $(8 - 2)/2 = 3$ slides. Computing this morph takes $O(n)$ time and $O(n)$ space.

By Lemma 3.2.6, $O(n)$ edges are added to the target in total. Hence, to compute the second stage of the morph takes $O(n^2)$ time and $O(n)$ space in total, in the online model.

Let us bound the number of slides performed in the second stage. In the rightmost configuration of Figure 3.10, we observe that the end-vertices of the new edge added to the target must both belong to the original graph G. This is not true of the other configura-

tions, which require only that at least one of these vertices belongs to G.

Let k_2 be the number of straightening paths between two vertices of G—such as in the rightmost configuration—and let k_1 denote the number of straightening paths that contain only a single vertex of G. Let T denote the total number of slides used in the second stage. From our discussion above, it follows that,

$$T \leq 3k_1 + 4k_2. \tag{3.3}$$

From Lemma 3.2.6 we know that

$$k_1 + 2k_2 \leq \frac{4n+4}{3}.$$

Therefore,

$$k_1 \leq \frac{4n+4}{3} - 2k_2.$$

Plugging this into (3.3) we get that

$$T \leq 3\left(\frac{4n+4}{3} - 2k_2\right) + 4k_2 = 4n + 4 - 2k_2 \leq 4n + 4.$$

□

Putting everything together

By Lemmas 3.2.4 and 3.2.8, the first and second stages of the morph can be computed in $O(n^2)$ time and $O(n)$ space. To represent the third stage of the morph, we need only output the target drawing, which takes an additional $O(n)$ time and space.

Summing up the total number of linear morphs used in the three stages, we get that this number is

$$\left(\frac{4n}{3} + O(1)\right) + (4n+4) + 1 = \frac{16n}{3} + O(1).$$

This completes our proof of Theorem 3.2.1.

In the next section, we describe another algorithm for generating parallel morphs between orthogonal drawings in the plane, with an even better bound on the number of linear morphs required.

3.3 The Second Algorithm: Reducing the Number of Linear Morphs

In this section, we describe another algorithm for parallel morphing between orthogonal drawings in the plane. Comparing this algorithm with the one of Section 3.2, this algorithm

admits a smaller bound on the number of linear morphs used in a parallel morph. However, this improvement comes at the cost of an increased runtime.

Theorem 3.3.1 *In the plane, every pair of parallel simple orthogonal drawings of a connected graph admits a parallel morph comprising a sequence of at most* $2n + O(1)$ *linear morphs, where* n *is the number of vertices in the graph. The morph can be computed in* $O(n^4)$ *time and* $O(n^2)$ *space.*

As with the algorithm of Section 3.2, this algorithm has three stages. The first and third stages of the two algorithms are identical. They differ only in their second stages.

Recall the second stage of the previous algorithm: $O(n)$ vertical edges are added one-by-one to the target drawing to turn it into a rectangular drawing. As each vertical edge is added, a straightening path of size $O(1)$ is generated in the source, and the source is morphed to remove bends from the straightening path.

In contrast, the new algorithm generates $O(n)$ straightening paths all at once at the beginning of the second stage. Each of these straightening paths is of complexity $O(n)$. Hence, all straightening paths can be stored in $O(n^2)$ space. While there exists a non-trivial straightening path, choose at least one zigzag from each non-trivial straightening path, and compute the drawing that would result if the slides determined by these zigzags were performed on the source drawing; in general, the drawing that results will depend on the order of the zigzags, but we can use any ordering. Perform a linear morph from the current source to this new drawing. Two coincident vertices are then removed from each zigzag.

In effect, each linear morph compresses a sequence of slides into a single motion. We call such a linear morph a *compressed slide*. It is crucial to prove that a compressed slide is a parallel morph. We do this in Section 3.3.1.

3.3.1 Compressed slides

Let G be a connected graph, let P be a simple orthogonal drawing of a G, and let

$$\phi^1, \phi^2, \ldots, \phi^k$$

be k non-trivial straightening paths of P such that every two straightening paths have at most a single vertex in common (such a vertex must be a terminal vertex of the straightening paths); and, such that

$$R = P \cup \phi^1 \cup \phi^2 \cup \ldots \cup \phi^k$$

is a simple drawing. We say that R is simple orthogonal drawing that has been *augmented* with straightening paths.

By Observation 3.2.1 (page 63), every non-trivial straightening path contains at least one zigzag. Two zigzags (v_a, v_b, v_c, v_d) and (v_d, v_c, v_b, v_a) in the graph underlying a straightening path ϕ^i are considered to be identical, i.e. reversing the order of vertices in a zigzag does not produce a new zigzag.

Definition 3.3.1 *Let R be a simple orthogonal drawing augmented with straightening paths. A sequence $Z = (z_1, \ldots, z_l)$ of zigzags of the straightening paths of R is called* diffuse *if all zigzags in Z are distinct, and there exists no $i, j \in \{1, \ldots, l\}$ such that $z_i = (v_a, v_b, v_c, v_d)$ and $z_j = (v_b, v_c, v_d, v_e)$.*

A diffuse sequence of zigzags may contain multiple zigzags belonging to the same straightening path. Recall that every zigzag of a straightening path in a drawing determines a *slide* of the drawing (Definition 3.2.4).

Definition 3.3.2 *Let R be a simple orthogonal drawing augmented with straightening paths. Let $Z = (z_1, \ldots, z_l)$ be a diffuse sequence of zigzags of the straightening paths of R. The* drawing sequence *of R that is* generated *by Z is the sequence $(R^0, \ldots, R^l)$, such that*

1. *$R^0 = R$, and*
2. *R^i is the drawing that results when the slide determined by z_i is performed on R^{i-1}, where $i \in \{1, \ldots, l\}$.*

Definition 3.3.3 *Let R be a simple orthogonal drawing augmented with straightening paths and let Z be a diffuse sequence of zigzags of the straightening paths. The* morph *of R* generated *by Z is the morph composed of the sequence of linear morphs from R^j to R^{j+1} for all $j \in \{0, \ldots, l-1\}$, where $(R^0, \ldots, R^l)$ is the drawing sequence of R generated by Z.*

To see why a drawing sequence (and thus the associated morph) always exists, recall that when the slide determined by a zigzag is performed on a simple orthogonal drawing, the only pair of elements to intersect during the slide are the two middle vertices of the zigzag. The middle edge of the zigzag shrinks to zero length, and every other edge remains parallel with its initial representation (Lemma 3.2.1).

During a slide performed on an orthogonal drawing in which some pairs of vertices coincide (and so the edges between such pairs have zero length) the two middle vertices

of the zigzag that determines the slide become coincident. Every pair of vertices that coincides at the beginning of the slide remains coincident throughout the slide, and all other elements remain pairwise non-intersecting. Further, every edge that is of non-zero length at the end of the slide is parallel with its realization in the initial drawing.

Suppose that a zigzag sequence contains the zigzags

$$z_1 = (v_{\mathrm{a}}, v_{\mathrm{b}}, v_{\mathrm{c}}, v_{\mathrm{d}}) \text{ and } \mathrm{z}_2 = (\mathrm{v}_{\mathrm{b}}, \mathrm{v}_{\mathrm{c}}, \mathrm{v}_{\mathrm{d}}, \mathrm{v}_{\mathrm{e}})$$

and that z_1 precedes z_2 in this sequence. When the slide determined by z_1 is performed, edge $\{v_{\mathrm{b}}, v_{\mathrm{c}}\}$ goes to zero length. Edge $\{v_{\mathrm{b}}, v_{\mathrm{c}}\}$ is also an edge of z_2. However, slides are not defined for zigzags with zero-length edges. Thus, there is no drawing sequence determined by a zigzag sequence in which z_2 follows z_1. We avoid this problem by requiring zigzag sequences to be diffuse.

The above discussion is summarized by the following observation.

Observation 3.3.1 *Let R be a simple orthogonal drawing in the plane augmented with straightening paths, and let $Z = (z_1, \ldots, z_l)$ be a diffuse sequence of zigzags of the straightening paths of R. Let $(R^0, \ldots, R^l)$ denote the drawing sequence of R generated by Z. For all $i \in \{1, \ldots, l\}$,*

1. *the two middle vertices of each zigzag $z_1, \ldots, z_i$ coincide in R^i,*
2. *no other pair of elements (vertices or edges) intersect in R^i; and,*
3. *every edge that has non-zero length in R^i is parallel with its representation in R.*

What happens if we reorder the zigzags in the zigzag sequence that generates a drawing sequence? Does this alter the final drawing of the drawing sequence? In general, the order of zigzags does matter. Figure 3.11 illustrates the motions of a vertex and two zigzags as the slides determined by these zigzags are performed.

The upper sequence of drawings in this figure illustrate the morph that results when the slide determined by the leftmost zigzag is performed first (time is increasing from left to right in the figure). In the lower sequence, the slide determined by the rightmost zigzag is performed first. The resulting drawing depends on the order in which the slides are performed. In this example, the two final drawings have the same vertex ordering. We leave it as an open problem to determine whether this is always the case.

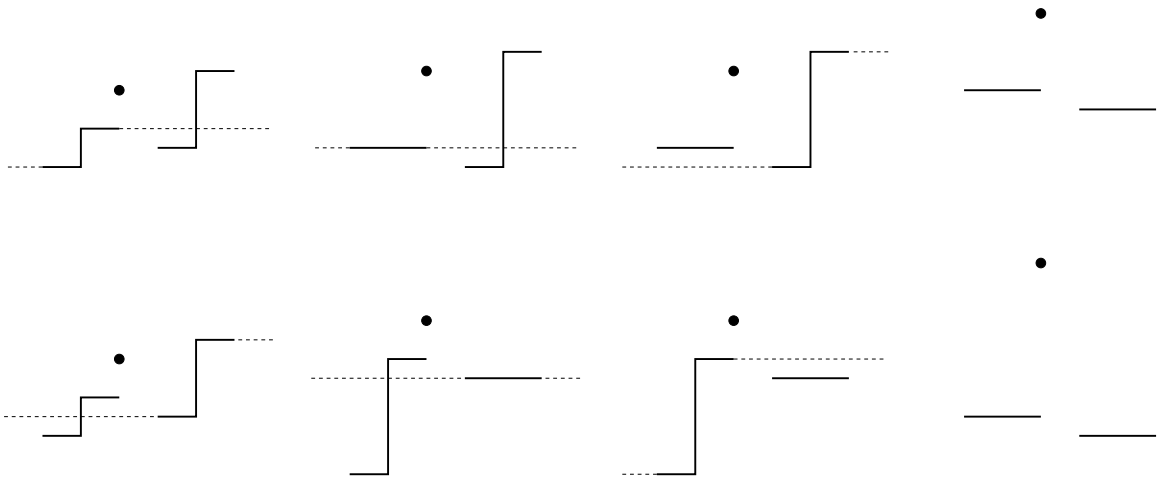

Figure 3.11: The drawing we obtain depends on the order in which we straighten the zigzags.

Definition 3.3.4 *Let R be a simple orthogonal drawing that has been augmented with straightening paths, and let $(R^0, \ldots, R^l)$ be the drawing sequence of R generated by a diffuse sequence of zigzags Z of the straightening paths of R. The* compressed slide *of R determined by Z is the linear morph from R^0 to R^l.*

We are now ready to state the main result of this section, that a compressed slide of a simple orthogonal drawing augmented with straightening paths provides a parallel morph of the original graph (i.e. the drawing modulo straightening paths).

Theorem 3.3.2 *Let P be a simple orthogonal drawing of a graph G. Let R be P augmented with straightening paths $\phi^1, \ldots, \phi^k$. Let M denote the compressed slide of R determined by a diffuse sequence Z of zigzags of the straightening paths of R. For all $t \in [0, 1]$:*

1. *every edge of G is parallel in $M(t)$ with its realization in P, and*
2. *no two elements of G intersect in $M(t)$.*

Proof: Let $(R^0, \ldots, R^l)$ denote the drawing sequence of R generated by Z. By Observation 3.3.1, each edge of G has non-zero length in every drawing R^i and is represented by parallel segments in P and R^i, where $i \in \{0, \ldots, l\}$. By Lemma 2.1.1 (page 27) the segment that represents an edge in each intermediate drawing of the linear morph from R^0 to R^l is parallel with the segment that represents the edge in P. This proves the first point of the theorem.

To prove the second point, Lemma 3.3.1 establishes that if the linear morph between a pair of parallel simple orthogonal drawings does not preserve simplicity, then there exists a pair of vertices $u, v \in \mathrm{V}(G)$ such that in every parallel morph between the terminal drawings of the linear morph,

1. there exists an intermediate drawing in which u and v have equal x-coordinates; and,

2. there exists an intermediate drawing in which u and v have equal y-coordinates.

Lemma 3.3.3 establishes that in the parallel morph of R generated by Z, no two vertices of G have equal x-coordinates in one intermediate drawing, and equal y-coordinates in another intermediate drawing.

From Lemma 3.3.1 and Lemma 3.3.3, it follows that the set of edges and vertices of G is pairwise non-intersecting in every intermediate drawing of a compressed slide, which establishes our second point. With that, the theorem is proven. □

Lemma 3.3.1 *Let M be a parallel morph between a pair of simple orthogonal drawings of a graph G in the plane. If the linear morph from $M(0)$ to $M(1)$ does not preserve simplicity, then there exist $u, v \in \mathrm{V}(G)$ and $t_1, t_2 \in [0, 1]$ such that*

$$M_{\mathrm{x}}(t_1; u) = M_{\mathrm{x}}(t_1; v) \quad \text{and} \quad \mathrm{M_y(t_2; u) = M_y(t_2; v)}.$$

Proof: Let M' denote the linear morph from $M(0)$ to $M(1)$. Suppose that M' does not preserve simplicity. By Lemma 1.3.2 (page 17), there exists some $t \in [0, 1]$ such that $M'(t)$ exhibits either vertex-vertex or vertex-edge intersection. Let $u \in \mathrm{V}(G)$ be a vertex that intersects either a vertex or an edge of G in $M'(t)$.

If u intersects another vertex $v \in \mathrm{V}(G)$ in an intermediate drawing of M', then the ordering of u and v differs between $M(0)$ and $M(1)$ with respect to both the x-axis and y-axis. Since M is continuous, the lemma follows.

Suppose that u intersects an edge $\{v_1, v_2\} \in \mathrm{E}(G)$ in an intermediate drawing of M'. Without loss of generality, let us assume that $M(t; v_1, v_2)$ is parallel with the y-axis for all $t \in [0, 1]$, and that:

$$M_{\mathrm{y}}(t; v_1) > M_{\mathrm{y}}(t; v_2).$$

Since u and $\{v_1, v_2\}$ intersect during M', the ordering of u and $\{v_1, v_2\}$ differs between $M(0)$ and $M(1)$ with respect to the x-axis. Since M is continuous, there must exist some intermediate drawing $M(t_1)$ in which

$$M_{\mathrm{x}}(t_1; u) = M_{\mathrm{x}}(t_1; v_1) = M_{\mathrm{x}}(t_1; v_2).$$

Next, consider the y-coordinates. If for all $t \in [0,1]$,

$$M_{\mathrm{y}}(t;u) > M_{\mathrm{y}}(t;v_1) \tag{3.4}$$

then u has a higher y-coordinate than any point on the closed segment between v_1 and v_2 in every intermediate drawing of M'. That is, for all $t' \in [0,1]$,

$$\begin{aligned} M'_{\mathrm{y}}(t';u) &= (1-t')M_{\mathrm{y}}(0;u) + t'M_{\mathrm{y}}(1;u) \\ &> (1-t')M_{\mathrm{y}}(0;v_1) + t'M_{\mathrm{y}}(1;v_1) \\ &= M'_{\mathrm{y}}(t';v_1). \end{aligned}$$

Thus, if (3.4) holds, then u cannot intersect $\{v_1, v_2\}$ in an intermediate drawing of M'.

Similarly, if for all $t \in [0,1]$,

$$M_{\mathrm{y}}(t;u) < M_{\mathrm{y}}(t;v_2)$$

then u and $\{v_1, v_2\}$ cannot intersect in an intermediate drawing of M'.

Therefore, there must exist some $t_0 \in [0,1]$ such that

$$M_{\mathrm{y}}(t_0;v_2) \leq M_{\mathrm{y}}(t_0;u) \leq M_{\mathrm{y}}(t_0;v_1).$$

However, the y-coordinate of u cannot be between the y-coordinates of v_1 and v_2 in all intermediate drawings of M, otherwise $M(t_1)$ will not be simple (but M is a parallel morph and hence, maintains simplicity). Therefore, there exists some $t_2 \in [0,1]$ such that either

$$M_{\mathrm{y}}(t_2;u) = M_{\mathrm{y}}(t_2;v_1) \quad \text{or} \quad \mathrm{M_y(t_2;u) = M_y(t_2;v_2)}.$$

With that, the lemma is proven. □

Our goal now is to prove (in Lemma 3.3.3) that in the morph generated by a diffuse sequence of zigzags, no two vertices have equal x-coordinates in one intermediate drawing and equal y-coordinates in another. The general approach is that of a proof-by-contradiction. We assume that two vertices u and v have equal x-coordinates in one intermediate drawing of the morph, and equal y-coordinates in another. It cannot be that u and v have equal x-coordinates in every intermediate drawing of the morph, otherwise they surely will intersect. Likewise, they cannot have equal y-coordinates in every intermediate drawing.

Therefore, during some slide of the morph, the relative x-coordinates of u and v change, and are equal in some intermediate drawing of the slide. During another slide, the relative y-coordinates of u and v change, and are equal in some intermediate drawing of the slide. We shall demonstrate that if these slides occur one right after the other then straightening paths of the original drawing are not simple. If not, then we can reorder the zigzag sequence in such a way that two such slides are adjacent.

Definition 3.3.5 *In an orthogonal drawing augmented with straightening paths, a zigzag of a straightening path is an* x-zigzag *if its middle edge is parallel with the x-axis. The zigzag is a* y-zigzag *if its middle edge is parallel with the y-axis.*

Observe that the slide determined by an x-zigzag changes only x-coordinates of vertices, and the slide determined by a y-zigzag changes only y-coordinates.

The following lemma states that we can exchange the order of an x-zigzag and an adjacent y-zigzag in a zigzag sequence without affecting the coordinates of the vertices in the final drawing of the morph generated by the zigzag sequence.

Lemma 3.3.2 *Let R be a drawing that has been augmented with straightening paths. Let (z_1, z_2) be a diffuse sequence of zigzags of the straightening paths, such that z_1 is an x-zigzag and z_2 is a y-zigzag. Let (R, R^1, R^{12}) and (R, R^2, R^{21}) denote the drawing sequences of R generated by (z_1, z_2) and (z_2, z_1), respectively. For all vertices v of R,*

$$R^1_{\mathrm{x}}(v) = R^{12}_{\mathrm{x}}(v) = R^{21}_{\mathrm{x}}(v).$$

Proof: Since z_2 is a y-zigzag, the x-coordinates of the vertices will not change during the slide from R^1 to R^{12}. Therefore,

$$R^1_{\mathrm{x}}(v) = R^{12}_{\mathrm{x}}(v).$$

Consider the motions of v with respect to the x-axis during the slide from R to R^1. Since z_1 is an x-zigzag, only x-coordinates of vertices will change during the slide from R to R^1. All vertices will move by a distance equal to half the length of the middle edge of z_1 in R. Depending on the position of v with respect to z_1, v will move either positively or negatively with respect to the x-axis (recall Definition 3.2.4).

Next, consider the motions of v when we first apply the slide determined by z_2 followed by the slide determined by z_1. Since z_2 is a y-zigzag, only the y-coordinates of the vertices will change during the slide from R to R^2. Therefore,

$$R_{\mathrm{x}}(v) = R^2_{\mathrm{x}}(v). \tag{3.5}$$

The length of the middle edge of z_1 is the same in R^2 as it is in R. Therefore, in the slide from R^2 to R^{21}, all vertices move by the same amount with respect to the x-axis as during the slide from R to R^1, i.e. by a distance equal to half the length of the middle edge of z_1 in R. It remains to prove that v moves in the same direction during the slide from R^2 to R^{21} as it does during the slide from R to R^1.

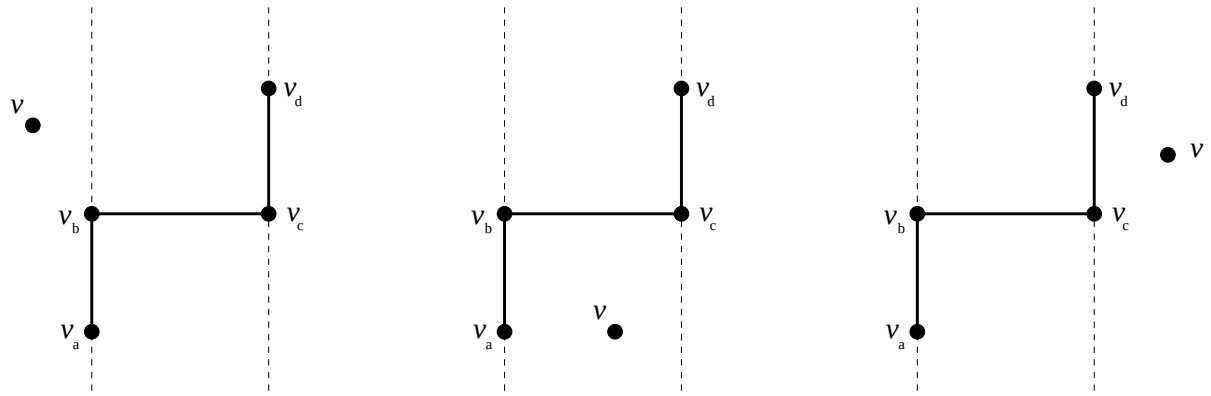

Figure 3.12: Three possible locations for v with respect to v_{b} and v_{c}.

If v is a vertex of z_1, then obviously v moves in the same direction during the slide from R^2 to R^{21} as it does during the slide from R to R^1. So assume that v is not a vertex of z_1.

Let $z_1 = (v_{\mathrm{a}}, v_{\mathrm{b}}, v_{\mathrm{c}}, v_{\mathrm{d}})$. Without loss of generality, assume that

$$R_{\mathrm{x}}(v_{\mathrm{b}}) < R_{\mathrm{x}}(v_{\mathrm{c}}). \tag{3.6}$$

By (3.5) and (3.6) either:

1. $R_{\mathrm{x}}(v) = R^2_{\mathrm{x}}(v) < R_{\mathrm{x}}(v_{\mathrm{b}}) = R^2_{\mathrm{x}}(v_{\mathrm{b}})$,
2. $R_{\mathrm{x}}(v_{\mathrm{b}}) = R^2_{\mathrm{x}}(v_{\mathrm{b}}) \leq R_{\mathrm{x}}(v) = R^2_{\mathrm{x}}(v) \leq R_{\mathrm{x}}(v_{\mathrm{c}}) = R^2_{\mathrm{x}}(v_{\mathrm{c}})$, or
3. $R_{\mathrm{x}}(v_{\mathrm{c}}) = R^2_{\mathrm{x}}(v_{\mathrm{c}}) < R_{\mathrm{x}}(v) = R^2_{\mathrm{x}}(v)$.

These are illustrated in Figure 3.12; ignore the y-coordinates of the vertices in this figure.

In the first and third cases, by the definition of a slide v moves in the same direction during the slide from R^2 to R^{21}, i.e. away from the zigzag. In the second case, v will move in opposite directions only if the relative y-coordinates of v and $\{v_{\mathrm{b}}, v_{\mathrm{c}}\}$ differ between the two drawings. That is, v will move in opposite directions in the two slides only if:

$$R_{\mathrm{y}}(v_{\mathrm{b}}) = R_{\mathrm{y}}(v_{\mathrm{c}}) < R_{\mathrm{y}}(v)$$

and

$$R^2_{\mathrm{y}}(v_{\mathrm{b}}) = R^2_{\mathrm{y}}(v_{\mathrm{c}}) > R^2_{\mathrm{y}}(v),$$

or vice versa.

However, the slide from R to R^2 changes only the y-coordinates of the vertices. Therefore, v intersects either one of the vertices v_{b} or v_{c}, or the edge $\{v_{\mathrm{b}}, v_{\mathrm{c}}\}$ during this slide.

This contradicts Observation 3.3.1. It follows that v moves in the same direction during the slide from R^2 to R^{21} as it does in the slide from R to R^1. □

For convenience, we have given Lemma 3.3.2 in the context of the x-coordinates of vertices. However, there is nothing special about the x-axis. The lemma can be restated with the labels on the coordinate axes swapped.

Lemma 3.3.3 *Let R be a simple orthogonal drawing of a graph G augmented with straightening paths $\phi^1, \ldots, \phi^k$. Let Z be a diffuse sequence of zigzags of the straightening paths, and let M denote the compressed slide of R generated by Z. For all $u, v \in \mathrm{V}(G)$, either:*

1. *$M_{\mathrm{x}}(t; u) \neq M_{\mathrm{x}}(t; v)$, for all $t \in [0, 1]$, or*
2. *$M_{\mathrm{y}}(t; u) \neq M_{\mathrm{y}}(t; v)$, for all $t \in [0, 1]$.*

Proof: To the contrary, assume that there exist $u, v \in \mathrm{V}(G)$ and $t_1, t_2 \in [0, 1]$ such that

$$M_{\mathrm{x}}(t_1; u) = M_{\mathrm{x}}(t_1; v) \quad \text{and} \quad M_{\mathrm{y}}(t_2; u) = M_{\mathrm{y}}(t_2; v).$$

Let $Z = (z_1, \ldots, z_l)$, and let $(R^0, \ldots, R^l)$ denote the drawing sequence of R generated by Z. For some $i \in \{1, \ldots, l\}$, the x-coordinates of u and v are equal in some intermediate drawing of the slide of R^{i-1}, determined by x-zigzag

$$z_i = (u_{\mathrm{a}}, u_{\mathrm{b}}, u_{\mathrm{c}}, u_{\mathrm{d}}).$$

Similarly, for some $j \in \{1, \ldots, l\}$ the y-coordinates of u and v are equal in some intermediate drawing of the slide of R^{j-1}, determined by y-zigzag

$$z_j = (v_{\mathrm{a}}, v_{\mathrm{b}}, v_{\mathrm{c}}, v_{\mathrm{d}}).$$

For the moment, suppose that z_i and z_j are adjacent in Z. It is shown below that this leads to a contradiction. Claim 3.3.1 establishes that if z_i and z_j are not adjacent, then there exists an ordering Z' of the zigzags of Z in which z_i and z_j are adjacent, such that the compressed slide determined by Z is identical to the compressed slide determined by Z'.

It is convenient to assume that z_i precedes z_j, and thus $i + 1 = j$. If not, relabel the coordinate axes. If the x-coordinates of u and v remain equal throughout the slide from R^{i-1} to R^i, then u and v coincide during the slide from R^{j-1} to R^j. However, the morph of R generated by Z keeps the vertices of G pairwise non-intersecting. It follows that the

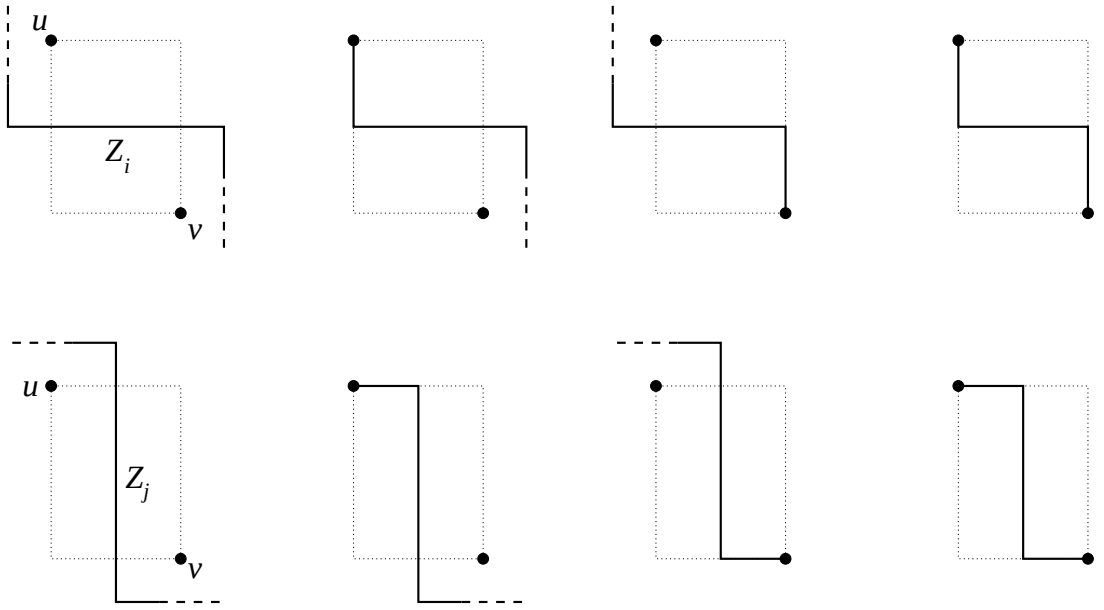

Figure 3.13: *Top:* possible drawings of z_i in R^{i-1}. *Bottom:* possible drawings of z_j in R^{i-1}.

relative x-coordinates of u and v must change during the slide from R^{i-1} to R^i. By the same argument, the relative y-coordinates of u and v change during the slide from R^{j-1} to R^j.

Since the relative x-coordinates of u and v change during the slide from R^{i-1} to R^i and are equal at some point, it must be that in R^{i-1}, the middle edge $\{u_\mathrm{b}, u_\mathrm{c}\}$ of x-zigzag z_i partitions the closed interior of the smallest axis-aligned bounding box containing $R^{i-1}(u)$ and $R^{i-1}(v)$; see Figure 3.13 (top).

The y-coordinates of the vertices in R^{i-1} are the same as those in $R^i = R^{j-1}$. By Lemma 3.3.2, if we apply the slide determined by z_j to R^{i-1}, we obtain a drawing whose y-coordinates are identical to those in R^j. Therefore, the relative y-coordinates of u and v are equal at some point of this slide. Using an argument similar to the one above, we can show that the relative y-coordinates of u and v must change during this slide.

It follows that in R^{i-1}, the middle edge $\{v_\mathrm{b}, v_\mathrm{c}\}$ of z_j partitions the closed interior of the smallest axis-aligned bounding box containing $R^{i-1}(u)$ and $R^{i-1}(v)$; see Figure 3.13 (bottom). However, the middle edges of the zigzags z_i and z_j intersect in R^{i-1}, which contradicts Observation 3.3.1. Therefore, the lemma holds under the assumption that z_i and z_j are adjacent in Z.

The following claim establishes that if z_i and z_j are not adjacent, then there exists a reordering of the zigzags of Z that can be used to obtain a contradiction.

Claim 3.3.1 *If z_i and z_j are not adjacent in Z, then there exists a reordering Z' of the zigzags of Z such that z_i is adjacent to z_j, the compressed slide determined by Z' is identical to M, and in the morph of R generated by Z',*

1. *there exists an intermediate drawing of the slide determined by z_i in which u and v have equal x-coordinates, and*

2. *there exists an intermediate drawing of the slide determined by z_j in which u and v have equal y-coordinates.*

Proof: Let $(z_1^{\mathrm{x}}, \ldots, z_{l_{\mathrm{x}}}^{\mathrm{x}})$ denote the subsequence of x-zigzags of Z, and let $(z_1^{\mathrm{y}}, \ldots, z_{l_{\mathrm{y}}}^{\mathrm{y}})$ denote the subsequence of y-zigzags. Choose $i' \in \{1, \ldots, l_{\mathrm{x}}\}$ and $j' \in \{1, \ldots, l_{\mathrm{y}}\}$ such that $z_i = z_{i'}^{\mathrm{x}}$ and $z_j = z_{j'}^{\mathrm{y}}$. Then, fix Z' such that

$$Z' = (z_1^{\mathrm{y}}, \ldots, z_{j'-1}^{\mathrm{y}}, z_1^{\mathrm{x}}, \ldots, z_{i'}^{\mathrm{x}}, z_{j'}^{\mathrm{y}}, \ldots, z_{l_{\mathrm{y}}}^{\mathrm{y}}, z_{i'+1}^{\mathrm{x}}, \ldots, z_{l_{\mathrm{x}}}^{\mathrm{x}}).$$

Observe that Z' is a reordering of the zigzags of Z. In this reordering, z_i and z_j are adjacent. The subsequence of x-zigzags in Z' is identical to the subsequence of x-zigzags in Z. Likewise for the y-zigzags. It is clear that Z' can be obtained from Z via a finite number of swaps of adjacent pairs of zigzags, where each pair consists of one x-zigzag and one y-zigzag. Hence, by Lemma 3.3.2, the final drawing in the drawing sequence generated by Z' is identical to the final drawing in the drawing sequence generated by Z. Therefore, the compressed slides determined by Z and Z' are identical.

All x-zigzags that precede z_i in Z must precede $z_i = z_{i'}^{\mathrm{x}}$ in Z'. From Lemma 3.3.2 it follows that

Consider the drawing sequence of R generated by zigzag sequence

$$(z_1^{\mathrm{y}}, \ldots, z_{j'-1}^{\mathrm{y}}, z_1^{\mathrm{x}}, \ldots, z_{i'-1}^{\mathrm{x}}).$$

Let R' denote the final drawing of this drawing sequence. By Lemma 3.3.2, the x-coordinates of the vertices in R' are identical to those in R^{i-1}, and the y-coordinates are identical to those in R^{j-1}. When the slide determined by $z_{i'}^{\mathrm{x}} = z_i$ is applied to R', the x-coordinates of the vertices change from their values in R^{i-1} to their values in R^i. Let R'' denote the resulting drawing. Thus, at some point of this slide the x-coordinates of u and v must be equal.

When the slide determined by $z_{j'}^{\mathrm{y}} = z_j$ is applied to to R'', the y-coordinates of the vertices change from their values in R^{j-1} to their values in R^j. Thus, at some point of this slide the y-coordinates of u and v are equal. □

By this claim, it may be assumed that z_i and z_j are adjacent zigzags in Z. With that, the proof of Lemma 3.3.3 is complete. □

3.3.2 The algorithm

We now describe our algorithm in detail and complete the proof of Theorem 3.3.1.

As we have already stated, there are three stages to the morphing algorithm. The first and third stages are identical to those of the algorithm of Section 3.2. Let G denote the graph that underlies the input drawings. During the first stage, the source is subjected to a parallel morph, and both the source and the target are augmented with a bounding box and a vertical edge $\{u_0, v_0\}$, such that u_0 subdivides the upper horizontal edge of the bounding box, and $v_0 \in \mathrm{V}(G)$. By Lemma 3.2.4, the number of linear morphs used in the first stage is bounded by

$$\frac{4n}{3} + O(1)$$

where $n = |\mathrm{V(G)}|$. The third stage consists of a single linear morph.

In the second stage, vertical edges are added to the target to make it rectangular. We use the same procedure as was used in the second stage of the algorithm of Section 3.2. Horizontal edges are subdivided as necessary. By Lemma 3.2.6, $O(n)$ vertical edges are added in total.

Next, we subdivide horizontal edges of the source drawing to mirror the subdivisions performed in the target. We want to perform a parallel morph from the source to a new drawing that permits the addition of vertical edges, thereby making the source and target drawings parallel. To do this, we generate a set of $O(n)$ straightening paths in the source drawing all at once. This is unlike the previous algorithm, where straightening paths were processed one at a time.

We repeatedly perform compressed slides on the union of the source drawing and the straightening paths until all straightening paths are trivial. Each compressed slide is determined by a sequence of zigzags containing at least one zigzag from each non-trivial straightening path. Thus, we are able to bound the number of compressed slides used in the morph by the number of vertices in the straightening path of highest complexity.

Generating the straightening paths

The following lemma bounds both the size of the straighting paths added in the second stage, and also the computational resources required to compute them.

Lemma 3.3.4 *In the second stage, we can generate $O(n)$ straightening paths $\phi^1, \ldots, \phi^k$ of the source drawing, where $n = |\mathrm{V(G)}|$, such that:*

1. *for each edge $\{u, v\}$ added to the target in the second stage, there is a straightening path ϕ^i in the source between u and v, with at most $\frac{4n}{3} + O(1)$ vertices; and,*

2. *the non-terminal vertices of the straightening paths are pairwise distinct, and the union of the source drawing and $\phi^1 \cup \ldots \cup \phi^k$ is a simple drawing.*

The set of straightening paths can be generated in time $O(n^2)$ and space $O(n^2)$.

Proof: To begin, we describe how to add a single straightening path to the source drawing by tracing around an ϵ-shell of the source either clockwise or counterclockwise, such that the end-vertices of this straightening path are the vertices of an edge that is added to the target in the second stage. Multiple straightening paths are added all at once (not sequentially as in the previous algorithm). We must show that we can route these straightening paths so as not to produce crossings. To this end, we employ multiple ϵ-shells with varying values of ϵ so that multiple straightening paths can traverse the same portion of a face boundary without intersecting.

Let $\{u, v\}$ be a vertical edge that is added to the target in the second stage, such that u lies below v. We trace out a straightening path in the source drawing as follows: starting at u, move some small distance ϵ upward, such that ϵ is less than half the minimum feature size of the source drawing. This puts us on the ϵ-shell of a face of the source. Follow the ϵ-shell either clockwise or counterclockwise until we arrive at a point on the ϵ-shell that is distance ϵ directly below v. Extend the path upward distance ϵ, terminating at v.

Claim 3.3.2 *The above procedure generates a straightening path between u and v.*

Proof: It is clear that the above procedure will generate a simple drawing of a path with terminal vertices u to v, such that the union of this path drawing with the source drawing is a simple drawing. It remains only to prove that the path is balanced.

To count the excess number of right/left turns in the path in the source drawing, it suffices to count the excess number of turns we would encounter if we were to use the same procedure to generate a path in in the target drawing. So assume that ϵ is less than half the minimum feature size of the target drawing. Trace out the following drawing of a path in the target: starting at u, move upward distance ϵ and follow the ϵ-shell either clockwise or counterclockwise before finally reaching the point on the ϵ-shell directly below v. Finally, move upward distance ϵ to v; see Figure 3.14.

In this figure, the path is drawn clockwise from u to v. We discuss only the clockwise case, as the counterclockwise case is symmetric. In a clockwise traversal of a (simple) orthogonal polygon, we will encounter an excess of four right turns. In the figure, the shaded region enclosed by the drawing of the path and the edge $\{u, v\}$ forms the interior of an orthogonal polygon. The path drawn from u to v in the target has the same excess

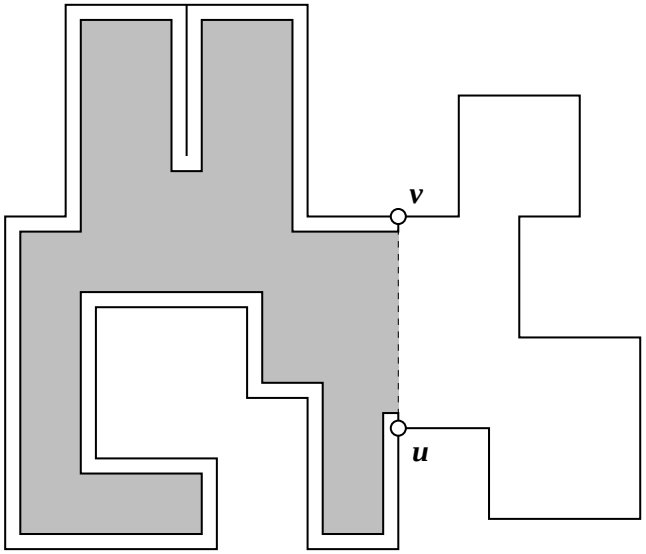

Figure 3.14: A balanced path between u and v inside a face of the target drawing.

of turns as a clockwise traversal of an orthogonal polygon, but with two additional right turns and two fewer left turns. Therefore, the path is balanced. □

In general, we will need to add multiple straightening paths within a single face of the source drawing. To simplify our discussion, let us restrict our attention to a single face of the source—straightening paths belonging to different faces will not intersect.

Suppose that we want to add a straightening path to the source corresponding to an edge $\{u, v\}$, such that u lies below v in the target. In the face of the source drawing destined to contain the straightening path, we call the point on the ϵ-shell that lies distance ϵ directly above u a *portal*, as it is at this point that the straightening path from u to v first enters the ϵ-shell. Another portal appears at the point distance ϵ immediately below v. These two portals are called the *terminal portals* of the straightening path between u and v. In a clockwise or counterclockwise traversal of the ϵ-shell, we will encounter a number of portals as well as a number of *bends*, which are the vertices of the ϵ-shell. No portal is a terminal portal of two distinct straightening paths.

For each edge $\{u, v\}$ added in the second stage there are two possible straightening paths along the ϵ-shell from u to v, one going clockwise and the other going counterclockwise. To decide which straightening path to use, we use the following greedy algorithm: choose the straightening path that contains the fewest bends. If both straightening paths have the same number of bends, choose either one arbitrarily.

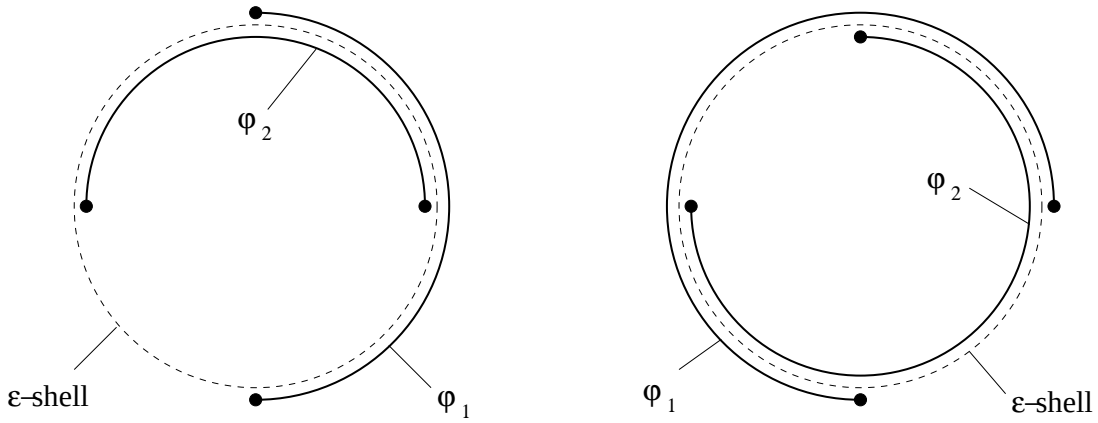

Figure 3.15: Two configurations of straightening paths violate the claim. The dots represent terminal portals. *Left:* each of ϕ_1 and ϕ_2 intersects exactly one terminal vertex of the other. *Right:* both ϕ_1 and ϕ_2 intersect all four terminal portals.

Using this greedy approach, each straightening path will have at most half the number of vertices in the ϵ-shell of the face that contains the straightening path, plus a small constant. By Claim 3.2.4, the number of vertices in such a straightening path is at most

$$\frac{1}{2} \cdot \frac{8n-4}{3} + O(1) = \frac{4n}{3} + O(1).$$

In general, having multiple straightening paths use a single ϵ-shell will lead to collisions. We will deal with this issue shortly by allowing different straightening paths to use ϵ-shells with different values for ϵ. First, we will establish a key property of the set of straightening paths generated using the greedy algorithm.

Claim 3.3.3 *The greedy approach generates a set of straightening paths such that for any pair ϕ_1, ϕ_2 of distinct straightening paths in the set, if ϕ_1 intersects a terminal portal of ϕ_2, then it intersects all portals intersected by ϕ_2, while ϕ_2 does not intersect the terminal portals of ϕ_1.*

Proof: Each straightening path intersects the ϵ-shell in a closed interval, with terminal portals on the boundary. Suppose that two straightening paths ϕ_1 and ϕ_2 in the set generated by the greedy algorithm do not satisfy the claim. There are two possibilities: (1) ϕ_1 intersects exactly one of the terminal portals of ϕ_2, and ϕ_2 intersects exactly one of the terminal portals of ϕ_1; or (2) both straightening paths intersect all four terminal portals of these two straightening paths. These are illustrated in Figure 3.15. We will prove that neither of these cases can occur, starting with the first case.

Topologically, the ϵ-shell is a circle. Portals of the ϵ-shell are points on this circle, ordered such that there exists a set of pairwise non-intersecting chords where for each straightening path there is a chord in this set, joining its terminal portals. If ϕ_1 intersects exactly one of the terminal portals of ϕ_2 and vice versa, then it is impossible that the terminal portals of ϕ_1 and the terminal portals of ϕ_2 can be connected by non-intersecting chords. Therefore, the first case cannot occur.

To see why the above topological property holds, observe that there exists a homeomorphism from the ϵ-shell of each face of the source to the ϵ-shell of the corresponding face of the target, such that:

1. the terminal portals (in the source) of a single straightening path map to a pair of points p_1, p_2 on the ϵ-shell of the target; and,

2. each such pair p_1, p_2 in the target intersect a single vertical segment that subdivides the interior of the ϵ-shell, i.e. one of the vertical edges added in the second stage.

These vertical segments are pairwise non-intersecting.

Next, we will demonstrate that the second case is impossible also. For each straightening path ϕ in the set, let ϕ' denote the straightening path with the same terminal portals as ϕ, but going the opposite way around the ϵ-shell. Let $b(\phi)$ denote the number of bends in ϕ. It must be that

$$b(\phi_1) \leq b(\phi_1')$$

and

$$b(\phi_2) \leq b(\phi_2')$$

otherwise, a different set of straightening paths would have been chosen.

Since each of ϕ_1 and ϕ_2 cross terminal portals of the other (and they are not identical) it must be that the bends contained in ϕ_1' are also contained in ϕ_2, and the bends contained in ϕ_2' are also contained in ϕ_1. Therefore,

$$b(\phi_1') \leq b(\phi_2)$$

and

$$b(\phi_2') \leq b(\phi_1).$$

Since

$$b(\phi_2') \leq b(\phi_1) \leq b(\phi_1') \leq b(\phi_2) \leq b(\phi_2'),$$

it follows that

$$b(\phi_2') = b(\phi_1) = b(\phi_1') = b(\phi_2).$$

There are at least four bends in an ϵ-shell. We have shown above that the terminal vertices of two greedily chosen straightening paths must have non-intersecting chords. It is clear that there can be at most one straightening path ϕ such that $b(\phi) = b(\phi')$. Therefore, the second case cannot happen. □

The set of straightening paths generated using the greedy algorithm described above will all reside on a single ϵ-shell. In general, they will be pairwise intersecting. To make them non-intersecting, we will alter the value of ϵ to put different straightening paths onto different ϵ-shells, while keeping the route the same as determined by the greedy algorithm.

Fix a positive integer $\alpha \in \mathbb{R}$ such that

$$0 < k\alpha < \delta/2$$

where δ is the minimum feature size of the source drawing (recall that k denotes the number of straightening paths to be added in the second stage). For each straightening path generated by the greedy algorithm, move it to lie on the ϵ-shell where $\epsilon = \alpha i$, such that the $2i$ is the number of portals crossed by the straightening path, including the two terminal portals of the straightening path.

Note that by Claim 3.3.3, if two straightening paths in the set intersect, then one contains all the portals of the other. It follows that each straightening path crosses an even number of portals. Thus the value i defined above is an integer.

We have set α to be sufficiently small that for all $i \in \{1, \ldots, k\}$, an ϵ-shell exists in each face of the source where $\epsilon = i\alpha$ (recall Observation 3.2.2). For all $i, j \in \{1, \ldots, k\}$ such that $i < j$, the ϵ-shell of any face of the source for which $\epsilon = j\alpha$ is contained in the open interior of the ϵ-shell of the same face for which $\epsilon = i\alpha$. Thus, by varying the value of ϵ in this manner, we obtain k nested ϵ-shells.

Figure 3.16 (top) depicts a drawing with three ϵ-shells in the single interior face, shown as polygons with dashed edges. The white dots represent end-vertices of the proposed straightening paths. The heavy dashed segments show the connections to be made by the straightening paths. Figure 3.16 (bottom) illustrates how multiple straightening paths are routed by our algorithm.

Claim 3.3.4 *After adjustment, all straightening paths are pairwise non-intersecting.*

Proof: Suppose that two straightening paths intersect following the adjustment. Either they both follow the same ϵ-shell, or they follow different ϵ-shells. If they both follow the same ϵ-shell, then one of the straightening paths must intersect a terminal portal of the other. By Claim 3.3.3, this straightening path intersects all portals intersected by the

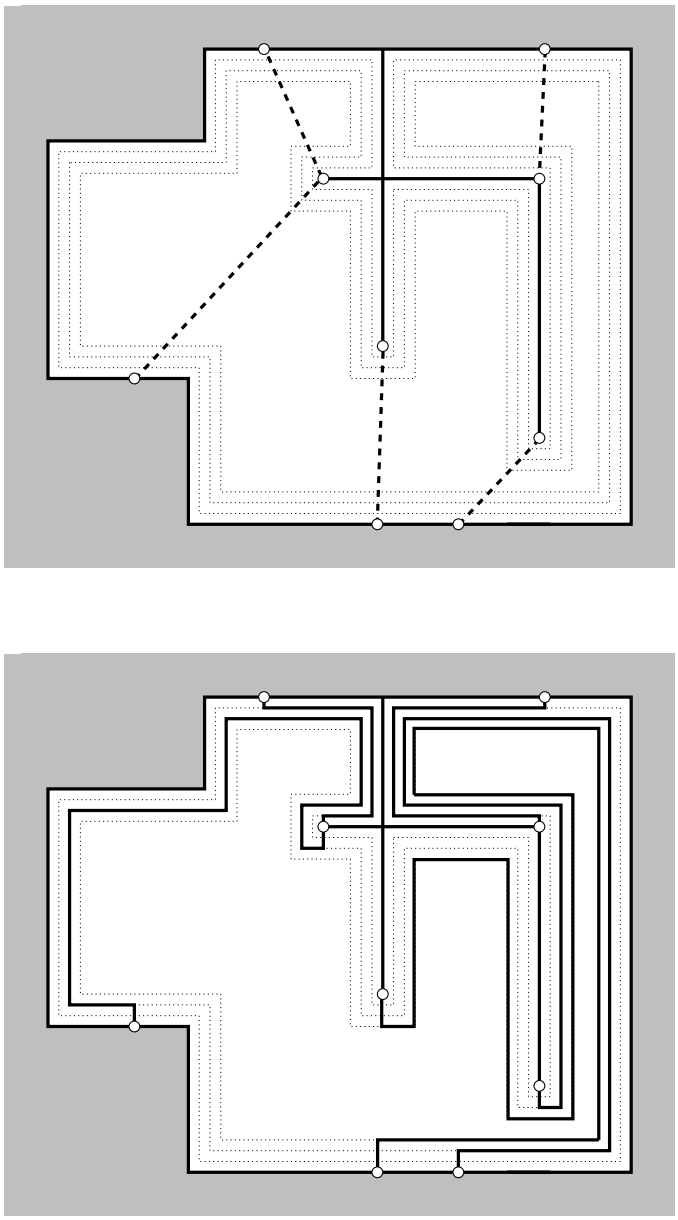

Figure 3.16: *Top:* an interior face with three ϵ-shells; the shaded region is exterior to the face. *Bottom:* five straightening paths traced on the three ϵ-shells; the white dots represent the terminal vertices of the straightening paths.

other, while the converse is not true. Since they intersect different numbers of portals, they cannot follow the same ϵ-shell.

Now, let us consider the case that two intersecting straightening paths follow different ϵ-shells. Observe that two distinct straightening paths lying on different ϵ-shells will intersect

only at the terminal portals of the straightening path that follows the inner ϵ-shell (the points of intersection will be on the outer ϵ-shell).

Let ϕ_1 and ϕ_2 denote intersecting straightening paths on two different ϵ-shells such that ϕ_1 follows the inner ϵ-shell, and ϕ_2 follows the outer ϵ-shell. If they intersect, then ϕ_2 must intersect a terminal portal of ϕ_1. Again, we can use Claim 3.3.3 to show that ϕ_2 contains strictly more portals than ϕ_1. Therefore, ϕ_2 must follow an ϵ-shell that lies inside the ϵ-shell followed by ϕ_1. This contradicts our assumption that ϕ_1 follows the inner shell. We conclude that the two straightening paths do not intersect. □

It is straightforward to generate each straightening path in $O(n)$ time. In total, $O(n)$ straightening paths are generated. Therefore, in total $O(n^2)$ time and space suffices to generate the set of straightening paths. □

Morphing the source drawing

Once the set of straightening paths has been generated in the source, we iterate as follows: while there exists a non-trivial straightening path, choose a *maximal* diffuse sequence Z of zigzags of the straightening paths, where Z is maximal if adding another zigzag to Z would make it non-diffuse. Perform the compressed slide determined by Z on the union of the source drawing and the straightening paths. This compressed slide provides a parallel morph of the source drawing (by Theorem 3.3.2).

Following each compressed slide, the union of the source drawing and the straightening paths is an orthogonal drawing. This drawing would be simple, except that the middle pair of vertices in each zigzag coincide (by Observation 3.3.1). For each zigzag $(v_{\mathrm{a}}, v_{\mathrm{b}}, v_{\mathrm{c}}, v_{\mathrm{d}})$ in the zigzag sequence, remove v_{b} and v_{c} (and incident edges) and add edge $\{v_{\mathrm{a}}, v_{\mathrm{d}}\}$. We obtain a new source drawing in which each non-trivial straightening path is reduced by at least two vertices.

When all straightening paths are trivial, the source and target are rectangular and parallel. The linear morph between them is a parallel morph. This is the third stage of the algorithm.

Lemma 3.3.5 *The number of linear morphs performed on the source drawing in the second stage is bounded by* $\frac{2n}{3} + O(1)$.

Proof: Every non-trivial straightening path contains at least one zigzag (Observation 3.2.1). Two zigzags that are contained within distinct straightening paths of the source drawing may share at most one end-vertex in common. For every maximal diffuse

sequence Z of zigzags, each non-trivial straightening path contains at least one zigzag of Z. Therefore, every compressed slide that is performed in the second stage will decrease the number of vertices in each non-trivial straightening path by two.

By Lemma 3.3.4, the longest straightening path in the source has $\frac{4n}{3} + O(1)$ vertices, initially. Since every compressed slide reduces the number of vertices in each non-trivial straightening path by two, we can bound the total number of compressed slides by

$$\frac{\left(\frac{4n}{3} + O(1)\right) - 2}{2} = \frac{2n}{3} + O(1).$$

□

The total number of linear morphs in a parallel morph generated by the algorithm of this section (in all three stages) is at most

$$\left(\frac{4n}{3} + O(1)\right) + \left(\frac{2n}{3} + O(1)\right) + 1 = 2n + O(1).$$

This is the bound we set out to prove on the number of linear morphs (Theorem 3.3.1).

Time and space requirements

Let us consider the time and space required by the algorithm. The first and third stages are identical to the first and third stages of the algorithm of Section 3.2. Therefore, these stages can be implemented to run in $O(n^2)$ time and $O(n)$ space in the online setting, where n is the number of vertices in the graph underlying the input drawings.

In the second stage, the target is augmented to make it rectangular using the same procedure as the algorithm of Section 3.2. Thus, this step can be implemented to run in $O(n^2)$ time and $O(n)$ space. Then in the source, $O(n)$ straightening paths are generated using $O(n^2)$ time and $O(n^2)$ space, by Lemma 3.3.4.

Next, we morph the source as determined by the straightening paths, using a sequence of compressed slides. For each compressed slide, we generate a maximal diffuse sequence of zigzags of the straightening paths. It is straightforward to do this in $O(n^2)$ time by scanning each straightening path linearly. Thus, over $O(n)$ compressed slides, $O(n^3)$ time is spent computing zigzag sequences.

The drawing that is the union of the source drawing and the straightening paths has complexity $O(n^2)$. We can compute the result of a compressed slide determined by a sequence of k zigzags on a drawing of complexity $O(n^2)$ by simply performing all k slides on the drawing. Thus, the result of the compressed slide can be computed in $O(kn^2)$ time.

Over all iterations, $O(n^2)$ zigzags are straightened in compressed slides. Thus, the time to compute all compressed slides is $O(n^4)$. Clearly, the space used by the algorithm is $O(n^2)$ in the online setting. With that, Theorem 3.3.1 has been proved.

It seems likely that the time bound on computing the result of a compressed slide can be improved. We leave this as an open problem.

3.4 Morphing with Disconnected Graphs

Until now, we have assumed that the graph underlying the input drawings is connected. In Sections 3.2 and 3.3 we showed that every pair of parallel simple orthogonal drawings of a connected graph admits a parallel morph. It is not difficult to construct a pair of parallel simple orthogonal drawings of a disconnected graph that does not morph: consider a drawing of a box and a lone vertex that resides inside the box. Consider a second drawing that is parallel with the first in which the lone vertex lies outside the box. There does not exist any simplicity-preserving morph between these two drawings, let alone a parallel morph.

In this section, we describe an easy test that determines whether a pair of parallel simple orthogonal drawings of a disconnected graph admits a parallel morph. If a parallel morph exists, then it can be efficiently generated.

Let P be a simple drawing of a graph G, such that G consists of connected components $G_1, \ldots, G_k$. The drawing of each component partitions the plane into a number of faces. For every pair of components G_i and G_j the drawing of G_i resides entirely in some face of G_j (possibly the exterior face) and vice versa.

Let Q be a simple drawing of G that is parallel with P. If, for all $i, j \in \{1, \ldots, k\}$ such that $i \neq j$, the drawing of G_i resides in the same face of G_j in both P and Q, then the two drawings have the same *face containments*.

Below, we prove that a pair of parallel simple orthogonal drawings of a graph admits a parallel morph if and only if the drawings have the same face containments. It can be efficiently determined whether parallel drawings P and Q have the same face containments as follows. Add a non-intersecting bounding box around each drawing and triangulate the drawings. Triangulation can be performed in $O(n \log n)$ time [51]. Color the original edges of each drawing black, and the newly added edges of the triangulations red.

Each closed face of P can be represented as the union of the closed interiors of a maximally connected set of triangles, where two triangles are connected if they share a red edge in common. Likewise for Q. Deciding whether P and Q have the same face containments

is equivalent to deciding whether, for each maximally connected set of triangles in P, there exists a maximally connected set of triangles in Q such that both sets are incident on the same set of black edges. Moreover, if a face of P corresponds to a face of Q, then for each black edge e incident on both faces, there must be a triangle in the triangulation of P on the same side of e as a triangle in the triangulation of Q. It is relatively straightforward to determine whether P and Q have the same face containments from the triangulated drawings in $O(n)$ time.

Theorem 3.4.1 *A pair of parallel simple orthogonal drawings of a disconnected graph admits a parallel morph if and only if the drawings have the same face containments.*

Proof: If a pair of drawings admits a parallel morph then the drawings must have the same face containments, otherwise during a morph the vertices of one component moves between two faces of some other component. This is not possible for a morph that preserves simplicity. Conversely, let P and Q be a pair of parallel orthogonal drawings of a graph G that exhibit the same face containments. It is proved by induction on the number of connected components that there exists a parallel morph from P to Q.

Augment each of P and Q by adding a non-intersecting bounding box to each drawing. This increases the number of components in the underlying graph by one. Following augmentation, both drawings have the same face containments. Clearly, if P and Q admit a parallel morph then the drawings produced by adding bounding boxes also admit a parallel morph. Adding bounding boxes in this way means that there exists a vertical line that intersects at least two components of the target drawing, hence two components can be connected with the addition of a vertical edge between them.

If G consists of two components (including the bounding box) then it follows from Theorem 3.2.1 that the drawings admit a parallel morph. Assume that for every pair of parallel simple orthogonal drawings whose underlying graph consists of at most $k-1$ connected components, if the drawings have the same face containments then they admit a parallel morph. Suppose that G consists of k components. Below, it is shown that Q can be augmented while P can be morphed and augmented so that the resulting drawings are parallel and have the same face containments, and the underlying graph consists of $k-1$ connected components. By the inductive assumption, the resulting drawings admit a parallel morph. Therefore, by Lemma 3.1.2 (page 57) the original drawings P and Q also admit a parallel morph.

Let $G_1, \ldots, G_k$ denote the $k \geq 2$ connected components of G. Add a vertical edge $\{u, v\}$ in target drawing Q between two components. To keep the source and target parallel, $\{u, v\}$

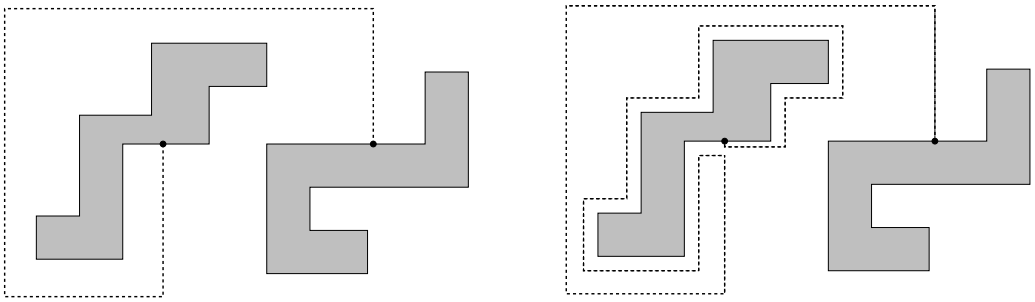

Figure 3.17: A drawing of a path between two components that is not balanced (left) may be made into a straightening path (right) by wrapping around the boundary of one of the components.

must be added to the source drawing P also. Add vertices u and v to the source if they are not already present. It is shown below that a straightening path exists in the source between u and v, such that the terminal edges of this straightening path are vertical. Once a straightening path is generated, Lemma 3.2.2 is employed to establish that the source drawing admits a parallel morph to a new drawing in which edge $\{u, v\}$ can be added as a vertical edge.

Let δ denote the minimum feature size of the source drawing. Add vertical edges $\{u, u'\}$ and $\{v, v'\}$ to the source where u' and v' are new vertices, such that u' lies distance $\delta/4$ below u, and v' lies distance $\delta/4$ above v. The source drawing remains simple following the addition of these edges. Since P and Q exhibit the same face containments, vertices u' and v' lie on the boundary of the same face of the source drawing. Therefore, there exists an orthogonal drawing of a path graph with terminal vertices u' and v' that can be added to the source without destroying simplicity. Add this drawing to the source.

The newly added path from u to v is not necessarily balanced: it may be the case that as the path is traversed from u to v an excess of left or right turns is encountered. This excess must be some multiple of four. Let G_i and G_j denote the components to be connected, such that $u \in \mathrm{V}(G_i)$ and that $v \in \mathrm{V}(G_j)$. Either one of these components is contained within an interior face of the other, or else both components reside in each others' exterior faces.

In the former case, the path must have no excess turns. It is therefore a straightening path. In the latter case, the path can be made to balance by wrapping it around G_i some integral number of times, staying just within the $\delta/3$-shell of the face of G_i in which the path is drawn, before connecting the path to u'; see Figure 3.17. By Lemma 3.2.2, there exists a parallel morph of the original source P (i.e. the source drawing prior to the

addition of this path) such that edge $\{u, v\}$ can be added to the drawing that results from this morph as a vertical edge, maintaining simplicity.

After morphing P and adding $\{u, v\}$, the source and target are parallel simple orthogonal drawings. Moreover, they have the same face containments since we have not created any new faces. The underlying graph consists of $k - 1$ components. By our inductive assumption, there exists a parallel morph from the current source to the current target. Therefore, by Lemma 3.1.2 there exists a parallel morph from P to Q. □

The proof of Theorem 3.4.1 outlines an algorithm for performing a parallel morph between a pair of parallel simple orthogonal drawings of a disconnected graph in cases where such a morph is possible. The approach is to augment the target with edges until it is connected. As each edge is added in the target, morph and augment in the source so that a corresponding edge can be added, keeping source and target parallel and with the same face containments. Once the graph underlying the drawings is connected, use the algorithm of either Section 3.2 or Section 3.3 to generate the rest of the morph.

To join k components, $k+O(1)$ edges and $2k+O(1)$ vertices are added to the drawings. Thus, once the graph has been connected, the number of linear morphs required to generate the rest of the morph is either

$$\frac{16n + 32k}{3} + O(1)$$

or

$$2n + 4k + O(1),$$

depending on the algorithm used, where n is the number of vertices in the input graph (see Theorem 3.2.1 and Theorem 3.3.1).

Let us bound the number of linear morphs that are used in making the underlying graph connected. The number of slides performed in the source for every edge added in the target depends on the number of vertices in the straightening path generated. The algorithm of Das and Narasimhan [19] can be used to generate a straightening path. Their algorithm generates an orthogonal minimum-link path between two points in the plane in the presence of of rectilinear obstacles, and runs in $O(n \log n)$ time where n is the number of vertices in the obstacles. The number of vertices in an orthogonal minimum-link path in the source is trivially bounded by $O(k)$ plus the number of vertices contained in the ϵ-shell of the face that contains the path; when the underlying graph is disconnected, the ϵ-shell of a face of a drawing may be composed of multiple polygons. However, the minimum-link path is not necessarily balanced.

To turn a minimum-link path between two components into a straightening path, the path can be wrapped around one of the components in the manner described in the above

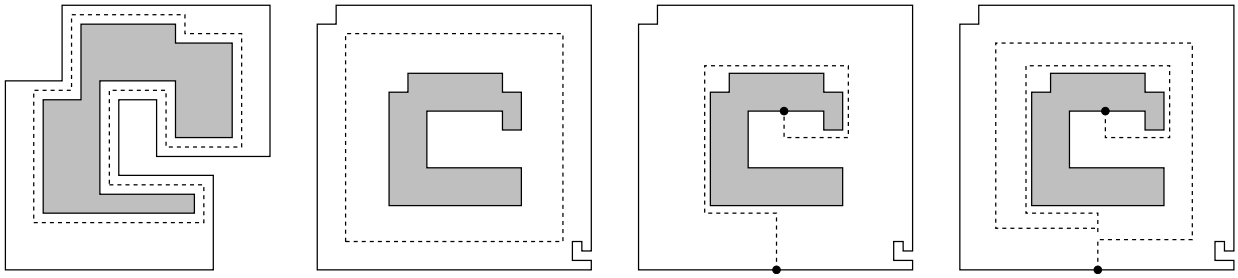

Figure 3.18: Building a straightening path in the source drawing.

proof. Unfortunately, this approach can lead to straightening paths with $\Omega(n^2)$ vertices, e.g. when the minimum-link path contains an excess of $\Omega(n)$ right or left turns, and the ϵ-shells of the components joined by this path each have $\Omega(n)$ vertices.

To ensure that the size of the straightening path is linear in the complexity of the graph, we take the following approach. Prior to generating the minimum-link path between vertices $u \in \mathrm{V}(G_i)$ and $v \in \mathrm{V}(G_j)$, choose some $\epsilon > 0$ that is less than half the minimum feature size of the source drawing. Augment the source drawing by adding the ϵ-shell of the drawing of G_i in the face in which we intend to draw the straightening path. Figure 3.18 (left) illustrates a drawing that consists of two polygonal components (shown with solid line segments) such that one component lies in the interior of the other component. The drawing is augmented with a ϵ-shell (shown with dashed line segments) around the component that lies in an interior face of the other component.

Morph the source to remove all zigzags from this ϵ-shell. For every zigzag involved, replace the edges and middle two vertices of the zigzag by an edge between the end-vertices of the zigzag, in the usual way. What remains of the ϵ-shell is a box containing the drawing of G_i in its interior; see Figure 3.18 (center-left). The region of the source drawing that lies between G_i and the box is empty.

Next, subdivide an edge of the box by a new vertex w. Generate a minimum-link path from u to w, and another minimum link path from v to w (using the trick we employed in the proof of Theorem 3.4.1, using vertices u' and v' to ensure that the path intersects u from below, and v from above). Remove all vertices and edges of the box except w from the source drawing. We are left with a drawing of a path from u to v; see Figure 3.18 (center-right).

This path drawing can turned into a balanced path by wrapping it around just inside the boundary of the face where the box formerly appeared; see Figure 3.18 (right). We will need to add only turns of one orientation (i.e. either left or right, with respect to one orientation of the path). Therefore, when converting the minimum-link path to a

straightening path, the number of vertices at most doubles. The result is a straightening path from u to v.

Let us study the properties of the algorithm just described.

Theorem 3.4.2 *Let P and Q be a pair of parallel simple orthogonal drawings of a graph G with k components, such that P exhibits the same face containments as Q. There exists a parallel morph from P to Q that uses $4n+O(k^2)$ linear morphs in addition to the number of linear morphs required for a parallel morph between a pair of parallel simple orthogonal drawings of a connected graph with $n+2k+O(1)$ vertices, where $n = |\mathrm{V(G)}|$. Generating this morph takes an additional $O(kn^2)$ time.*

Proof: The algorithm iterates $k-1$ times. In each iteration, at most two vertices are added. Let n_i denote the number of vertices in the underlying graph at the beginning of the ith iteration, i.e. $n_i = n+2i-2$. As discussed above, in each iteration at most a single linear morph is required in the target drawing. Let us bound the number required in the source drawing in an iteration.

Consider the ith iteration. Each ϵ-shell has at most $\frac{8n_i-4}{3}$ vertices (by Claim 3.2.4). An ϵ-shell is a collection of orthogonal polygons. We may think of the ϵ-shell in the external face of a given component as being a polygon composed of four straightening paths, connected at four vertices of the polygon. Therefore, the morph of the source drawing that turns an ϵ-shell into a box requires no more than

$$\frac{\frac{8n_i-4}{3}-4}{2} = \frac{4n_i-8}{3}$$

slides, by Lemma 3.2.2.

Once we have subdivided an edge of the box by w, the drawing of a path from u to w, and from w to v contains at most

$$\frac{8n_i-4}{3}+O(k) = \frac{8n_i}{3}+O(k)$$

vertices, as described above, i.e. the number of vertices in an ϵ-shell, plus $O(k)$. To turn this path drawing to a straightening path, in the worst case the number of vertices is doubled. Therefore, the number of vertices in the straightening path from u to v is at most

$$2\left(\frac{8n_i}{3}+O(k)\right) = \frac{16n_i}{3}+O(k).$$

By Lemma 3.2.2, the number of slides sufficient to perform the morph determined by this straightening path is at most

$$\frac{\frac{16n_i}{3}+O(k)-2}{2} = \frac{8n_i}{3}+O(k).$$

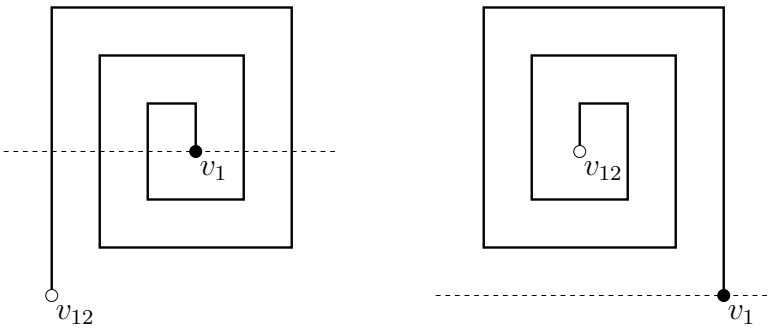

Figure 3.19: We use parallel orthogonal spirals to prove a lower bound on the number of linear morphs needed for a parallel morph.

Summing the slides performed in the source drawing over all $k-1$ iterations

$$\sum_{i=1}^{k-1} \frac{(4n_i - 8) + 8n_i}{3} + O(k) = O(k^2) + \sum_{i=1}^{k-1} 4(n + 2i - 2) = 4n + O(k^2).$$

It is straightforward that for each iteration, the straightening path in the source can be determined in $O(n^2)$ time. The number of slides in a single iteration is $O(n)$ and since the drawing that is the result of a single slide can be generated in $O(n)$ time, the total runtime of the algorithm is $O(kn^2)$ over all $k-1$ iterations. □

3.5 A Lower Bound on the Number of Linear Morphs

For each of the algorithms described in Sections 3.2 and 3.3, we established an upper bound on the number of linear morphs used in a parallel morph between two drawings. In this section, we prove a corresponding lower bound.

Consider the pair of parallel orthogonal spirals illustrated in Figure 3.19. The underlying path graph contains twelve vertices, and the terminal edges are vertical. Although these drawings are parallel, they appear to be reflections of each other through a vertical line. To highlight the correspondence between vertices in these drawings, the terminal vertices are colored white and black in the figure.

There exist parallel orthogonal spirals like this for any positive integer n. We note that it is only when n is a multiple of four that the spirals appear to be reflections of one another through a vertical line.

Theorem 3.5.1 *For any integer $n \geq 4$ that is a multiple of four, there exists a pair of parallel orthogonal spirals of n vertices, such that every parallel morph between them that is composed of a sequence of linear morphs uses at least $\frac{n-2}{2}$ linear morphs.*

Proof: To prove this theorem, we construct a pair of parallel orthogonal spirals with n vertices that appear to be reflections of one another. Intuitively, the only way to morph from one spiral to the other is to have one of the terminal vertices cycle around the other terminal vertex a number of times, proportional to the number of vertices. This requires that the terminal vertices swap vertical order the same number of times. Each swap requires a linear morph.

So, let us study the class of parallel morphs between two parallel orthogonal spirals of n vertices that appear to be reflections of each other through a vertical line, such that the terminal edges of each spiral are vertical. Label the vertices $v_1, \ldots, v_n$ such that in both spirals, as we follow the drawing of the path from v_1 to v_n we encounter only left turns (i.e. the spirals are counterclockwise with respect to this sequence of vertices). Let P denote the drawing in which v_n is to the left of v_1 (e.g. the drawing on the left side of Figure 3.19) and let Q denote the drawing in which v_n is to the right of v_1.

Suppose that we fix vertex v_1 at the origin in P and Q, and do not allow it to move during a parallel morph. For every parallel morph from P to Q, there exists a parallel morph that is equivalent following translations of its intermediate drawings, and such that v_1 resides at the origin throughout. Let l denote the horizontal line through the origin.

Claim 3.5.1 *Let $i \in \{2, 4, 6 \ldots, n-2\}$. In any parallel morph from P, if vertex v_i crosses l during the morph, then it does so for the first time strictly after v_{i+2} crosses l.*

Proof: (By induction on the vertex index i.) Observe that v_2 never crosses l in the course of a parallel morph from P. Therefore, the claim is trivially true in this case. Assume that v_{i-2} crosses l strictly after v_i crosses l for the first time. Below, we will prove that if v_i crosses l in a parallel morph, then it does so for the first time strictly after v_{i+2} crosses l.

Without loss of generality, assume that v_i resides above l in P, as illustrated in Figure 3.20. It is clear that v_{i-1} crosses l at the same time as v_{i-2}, and that v_i crosses l at the same time as v_{i+1}. If v_i crosses l before (or at the same time) as either v_{i+2} or v_{i-2}, then at least one of the edges $\{v_{i-1}, v_i\}$, $\{v_i, v_{i+1}\}$ or $\{v_{i+1}, v_{i+2}\}$ must intersect v_1 during the morph. Therefore, if v_i crosses l during the parallel morph, then either v_{i-2} or v_{i+2} will cross l strictly before this.

However, by our inductive assumption, v_{i-2} (and thus v_{i-1} also) may only cross l strictly after v_i has already crossed l. Thus, in a parallel morph from P, v_i does not cross l until strictly after v_{i+2} has crossed l. □

Let M denote a parallel morph from P to Q that is composed of a sequence of linear morphs. As above, assume that v_1 resides at the origin in every intermediate drawing of M.

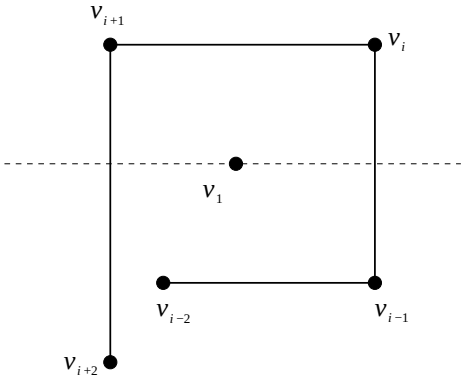

Figure 3.20: Vertex v_i cannot leave its quadrant in a parallel morph unless either v_{i-1} or v_{i+1} does so first.

Claim 3.5.2 *Let $i \in \{2, 4, 6 \ldots, n-2\}$. If v_i and v_{i+2} lie on opposite sides of l, then they cannot both move to the opposite side during a linear morph.*

Proof: Without loss of generality, assume that in P, v_i lies above l and v_{i+2} lies below l. In every drawing parallel with P, v_{i+2} has a vertical coordinate that is strictly smaller than the vertical coordinate of v_i. For a contradiction, assume that in M, both v_i and v_{i+2} cross l for the first time during a single linear morph. In the source drawing of this linear morph, v_i lies above l and v_{i+2} lies below l. In the target this situation is reversed: v_i lies above l and v_{i+2} lies below l (recall Lemma 2.3.1, page 40).

This contradicts the fact that v_{i+2} has a larger vertical coordinate than v_i in all drawings that are parallel with P. Therefore, it cannot be that both vertices cross l for the first time during a single linear morph of M. □

We now put everything together. Observe that in P, vertex v_4 lies below l while in Q, it lies above l. Therefore, some linear morph of M must take v_4 across l. By Claims 3.5.1 and 3.5.2, this linear morph is preceded by a linear morph that takes v_6 across l, which in turn is preceded by a linear morph that takes v_8 across l, etc. It follows that for each vertex $v_4, v_6, v_8, \ldots, v_n$ there is at least one corresponding linear morph in M. Thus, the total number of linear morphs is at least $\frac{n-2}{2}$ (since n is a multiple of four). □

One direction for future research is to try to narrow the gap between the upper and lower bounds on the number of linear morphs required for a parallel morph between parallel simple orthogonal drawings in the plane. What is the complexity of finding the optimum number of linear morphs required in a parallel morph? Does there exist an efficient algorithm for generating a parallel morph using a number of linear morphs that is within a

constant factor of optimum?

It is interesting to compare the drawings used in the proof of the lower bound with the number of linear morphs that are used by the algorithms of Sections 3.2 and 3.3. Recall that the first stages of these algorithms are identical. Suppose that the input drawings are spirals of n vertices, akin to the spirals in Figure 3.19. It is not difficult to see that in their first stages, both algorithms use $n + O(1)$ linear morphs.

Following the first stage, it is possible to complete a parallel morph between the spirals with a single linear morph. However, in trying to construct rectangular drawings the morphing algorithms might use more linear morphs than this, depending on the implementation. For example, in the second stage of the algorithm of Section 3.2, if straightening paths are generated by tracing around ϵ-shells, then for each edge added in the second stage two linear morphs are performed in the source drawing. However, if minimum-link paths are used, no additional linear morphs are required in the source. The situation is similar for the algorithm of Section 3.3.

3.6 Edge length concerns

Morphs that are composed of slides are *expansive*, in the sense that all edges are non-decreasing in length over the course of the slide (of course, we are ignoring edges of the straightening paths). A compressed slide takes us directly from our current drawing to the drawing that results when a sequence of slides is applied to the current drawing, by using a single linear morph. Morphs composed of compressed slides are also expansive in the above sense, since edges change length monotonically during a linear morph (Lemma 2.1.1, page 27).

A morph that is generated by the algorithm of Section 3.2 or Section 3.3 is composed of a sequence of slides, or (respectively) a sequence of compressed slides, followed by a single linear morph. We may consider such morphs as having two stages: an expansive stage, in which edges are non-decreasing in length, followed by a linear morph in which edges change length monotonically. If we were to insist that all edges be non-increasing in length during the second stage, we can include a scaling operation prior to the second stage that takes each edge to a length that is longer than its target length. We will explore issues of edge-length monotonicity in parallel morphs further in Chapter 4.

From the perspective of graph visualization, one nice feature of the morphs produced by the algorithms of Sections 3.2 and 3.3 is that edges do not change from being increasing-in-length to being decreasing-in-length more than once. The major drawback is that edges

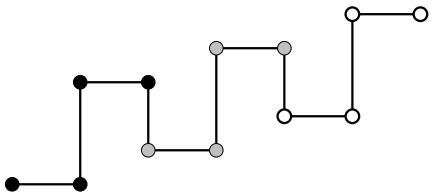

Figure 3.21: A straightening path that may lead to exponential increases in edge lengths.

can potentially grow by a factor that is exponential in the complexity of the graph during the expansive stage. To animate such a morph, it may be necessary to either display only a portion of the drawing at some times, or to scale down intermediate drawings so that the minimum feature size is less than the size of a pixel.

To see how an exponential increase in edge lengths may occur, consider a morph determined by the straightening path illustrated in Figure 3.21. Vertices of this straightening path are colored black, gray and white to highlight three of the zigzags of the straightening path. Suppose that we perform three slides: first, the slide determined by the black zigzag, followed by the gray zigzag, and finally the white zigzag.

Initially, the middle edges of all three zigzags are equal in lengths. When we perform the slide determined by the black zigzag, the middle edges of both gray and white zigzags double in length. When we perform the slide determined by the gray zigzag, the middle edge of the white zigzag doubles again. We can extend this example to include any given number k of zigzags, such that the middle edge of the rightmost zigzag will achieve a length that is 2^k times its original length.

To circumvent this exponential increase, we will show how to modify a parallel morph produced by one of our morphing algorithms, obtaining a new morphing algorithm thereby. The following observation is crucial. Recall the definition of *vertex ordering* (Definition 2.2.1, Chapter 2).

Observation 3.6.1 *Let P be an orthogonal drawing of a graph G in the plane. There exists a drawing R with the same vertex ordering as P, such that each vertex resides at integer coordinates in the range $\{1,\ldots,n\} \times \{1,\ldots,n\}$, where $n = |\mathrm{V(G)}|$.*

Proof: For each vertex $v \in \mathrm{V}(G)$, set $R_{\mathrm{x}}(v)$ equal to one plus the number of distinct x-coordinates of vertices in P that are smaller than $P_{\mathrm{x}}(v)$. Set y-coordinates similarly. □

Drawing R can be generated in $O(n \log n)$ time by first sorting the vertices of P with respect to each axis. By Observation 2.2.1 (page 29), P and R are parallel. By Observation 2.2.2 (page 30), R is simple if and only if P is simple.

The morphing algorithm based on slides (i.e. the first one) produces a sequence of drawings. Except for the final pair of drawings in this sequence, adjacent drawings of the sequence differ only in either their horizontal or vertical coordinates, not both. To modify the morph, we alter each intermediate drawing in the sequence so that vertices lie at integer coordinates in the range $\{1, \ldots, n\} \times \{1, \ldots, n\}$, leaving the vertex ordering unchanged.

Lemma 3.6.1 *Let P^1, P^2 be a pair of parallel simple orthogonal drawings of a graph G, such that:*

1. *for every vertex, its horizontal coordinates are equal in both drawings; and,*
2. *the linear morph from P^1 to P^2 is a parallel morph.*

Let R^1 and R^2 be a pair of drawings of G with the same vertex orderings as P^1 and P^2, respectively, and integer coordinates on all vertices. The linear morph from R^1 to R^2 is a parallel morph that maintains a minimum feature size of one.

Proof: The linear morph from R^1 to R^2 will keep all edges parallel (Lemma 2.1.1, page 27). Therefore, it suffices to prove that the linear morph from R^1 to R^2 maintains a unit minimum feature size.

For any two elements (vertices or edges) of G, either there exists a vertical line that intersects both elements in R^1 and a vertical line that intersects both elements in R^2; or, in both drawings it is impossible to draw a vertical line intersecting both elements. If this is not true, then either R^1 and R^2 do not have the same vertex orderings as P^1 and P^2, or the horizontal coordinates of vertices are not equal in P^1 and P^2.

If a vertical line intersects both elements in R^1 and a vertical line intersect both elements in R^2, then the vertical distance between these elements in each drawing is at least one, since the drawings are simple and vertices have integer coordinates. These elements must intersect the vertical line in the same order with respect to the vertical axis in both drawings, otherwise the linear morph from P^1 to P^2 does not preserve simplicity. The linear morph from R^1 to R^2 keeps these two elements at least unit distance apart with respect to the vertical axis.

Alternatively, if in both drawings there does not exist a vertical line that intersects both elements, then since all vertices lie on integer coordinates, the horizontal distance between the two elements is at least one with respect to the horizontal axis. The elements must appear in the same order with respect to the horizontal axis in both drawings. Therefore, during the linear morph from R^1 to R^2 the two elements remain at least unit distance

apart with respect to the horizontal axis. We conclude that the linear morph from R^1 to R^2 maintains a minimum feature size of at least one. □

The final linear morph of a morph produced by one of our morphing algorithms takes us between two drawings that have the same vertex ordering. Suppose that both of these drawings have integer coordinates. In the following lemma, we establish that such a linear morph maintains a unit minimum feature size.

Lemma 3.6.2 *The linear morph between two simple orthogonal drawings with the same vertex ordering and integer coordinates is a parallel morph that maintains a minimum feature size of one.*

Proof: Let R^1 and R^2 be a pair of parallel simple orthogonal drawings with integer coordinates. By Lemma 2.1.1 (page 27) the linear morph keeps all edges parallel. For every pair of elements, either there exists a vertical or horizontal line that intersects both of them in both drawings, or there does not exist such a line in either drawing.

If there exists a vertical or horizontal line between two non-incident elements in both drawings, then the distance between them with respect to this line is at least one in both drawings since all vertices lie at integer coordinates. Moreover, since the two drawings have equal vertex orderings, the relative order in which these elements intersect the line is the same in both drawings. Therefore, these elements remain at least unit distance apart throughout the linear morph. If there does not exist such a line, then the two elements are at least unit distance apart with respect to both axes. Again, the linear morph keeps the elements at least unit distance apart. □

Now, we present the main result of this section, that there exists a parallel morph between any given pair of parallel simple orthogonal drawings that avoids the exponential edge lengths in intermediate drawings. For convenience, we will assume that the source and target drawings have vertices at integer coordinates, and in the range $\{1, \ldots, n\} \times \{1, \ldots, n\}$. If this is not the case, we can apply a linear morph to the source to take us to such a drawing, with the same vertex ordering as the source. Likewise, we can apply a linear morph to the target, taking us to such a drawing.

Theorem 3.6.1 *Let P and Q be a pair of parallel simple orthogonal drawings of a connected graph G in the plane, such that vertices have integer coordinates in the range $\{1, \ldots, n\} \times \{1, \ldots, n\}$, where n is the number of vertices in G. There exists a parallel morph from P to Q in which each intermediate drawing has a minimum feature size of at least one, and can be contained within an $n \times n$ bounding box.*

- *We can generate such a morph, composed of* $\frac{16n}{3} + O(1)$ *linear morphs, in* $O(n^2)$ *time.*
- *We can generate such a morph, composed of* $\frac{8n}{3} + O(1)$ *linear morphs, in* $O(n^4)$ *time.*

Proof: We treat each of the two cases in turn. Let us begin with the first case. To start, generate a parallel morph from P to Q using the algorithm of Section 3.2 (recall that this is the morphing algorithm based on slides). By Theorem 3.2.1, the morph can be generated in $O(n^2)$ time. The algorithm generates a sequence

$$P = P^1, P^2, \ldots, P^k = Q$$

of parallel simple drawings, where

$$k \leq \frac{16n}{3} + O(1).$$

Modify each drawing in the sequence so that vertices have integer coordinates in the range $\{1, \ldots, n\} \times \{1, \ldots, n\}$, and the modified drawing has the same vertex ordering as the original. Let

$$P = R^1, R^2, \ldots, R^k = Q$$

denote the new sequence. Each drawing R^i can be generated from P^i in $O(n)$ time. Therefore, in total the new sequence is generated using $O(n^2)$ time.

Since all vertices lie in the range $\{1, \ldots, n\} \times \{1, \ldots, n\}$, all intermediate drawings in the morph that correspond to this new sequence can be contained within an $n \times n$ bounding box. We must prove that this morph is a parallel morph that maintains a unit minimum feature size.

For all $i \in \{1, \ldots, k-2\}$, drawing P^{i+1} is obtained from P^i using a slide operation. Therefore, P^i and P^{i+1} will have either equal horizontal coordinates, or equal vertical coordinates. By Lemma 3.6.1, the linear morph between R^i and R^{i+1} is a parallel morph that maintains a unit minimum feature size. The final two drawings of the sequence, R^{k-1} to R^k, have the same vertex ordering. By Lemma 3.6.2, the linear morph between these two drawings is a parallel morph that maintains a unit minimum feature size. This proves the first case.

To prove the second case, we would like to use the morphing algorithm described in Section 3.3 (i.e. the algorithm that uses compressed slides). By Lemma 3.3.1, this morph can be computed in $O(n^4)$ time. However, adjacent drawings in the sequence produced by this algorithm may differ in both their horizontal and vertical coordinates. We must first convert the morph produced by this algorithm to a form in which adjacent drawings

(except for the final pair) differ only with respect to one axis. We may then use the same technique as above, converting each drawing to a form with integer coordinates in the range $\{1, \ldots, n\} \times \{1, \ldots, n\}$.

Recall that morphs produced by our second morphing algorithm have three stages. These stages are identical to the first and third stages of our first algorithm. Thus, we need only concern ourselves with the second stage.

The second stage relies on compressed slides. At each stage, we have a drawing that is augmented with a number of disjoint straightening paths. To generate a compressed slide, we choose a diffuse sequence of zigzags of the straightening paths, and then compute the drawing that results when we perform the slides determined by this sequence of zigzags to the current drawing. The compressed slide is the linear morph from the current drawing to the computed drawing.

For each compressed slide, partition the diffuse zigzag sequence into two disjoint sequences: one containing all of the x-zigzags of the original zigzag sequence, and the other containing all of the y-zigzags. Since the original sequence is diffuse, so are each of these new sequences. We perform the compressed slides determined by each of these sequences, one after the other.

A compressed slide determined by a sequence of x-zigzags will change vertex coordinates with respect to only the (horizontal) x-axis, and a compressed slide determined by a sequence of y-zigzags will change vertex coordinates with respect to only the (vertical) y-axis.

By Lemma 3.2.4, at most $\frac{4n}{3} + O(1)$ slides are performed in the first stage. By Lemma 3.3.5, at most $\frac{2n}{3} + O(1)$ compressed slides are performed in the second stage. Since we generate at most two compressed slides in our new sequence for each compressed slide in the original sequence, the total number of compressed slides is bounded by

$$2\left(\frac{2n}{3} + O(1)\right) = \frac{4n}{3} + O(1).$$

Thus, over all three stages the number of linear morphs is at most

$$\left(\frac{4n}{3} + O(1)\right) + \left(\frac{4n}{3} + O(1)\right) + 1 = \frac{8n}{3} + O(1).$$

This proves the second case of the theorem. □

3.7 Conclusion

In this chapter we have studied parallel morphing in the context of orthogonal drawings in the plane. Our main result is that every pair of parallel simple orthogonal drawings of a graph admits a parallel morph. Moreover, a parallel morph can be efficiently generated for such drawings. We have presented two morphing algorithms. These algorithms offer a trade off between the bound on the number of linear morphs used in a parallel morph, and the asymptotic runtime.

It is not clear which of the two algorithms will generate morphs that are better for graph visualization. The first algorithm has the advantage that it makes use primarily of the slide operation. In effect, a slide partitions the plane into two regions with a boundary of low complexity. It sends vertices on one side of the partition in one direction, and the vertices on the other side in the opposite direction. Due to the uniformity of vertex motions during a slide, a parallel morph composed of a sequence of slides might better preserve a user's mental map than a parallel morph in which no two vertices move in the same direction.

In contrast, our second morphing algorithm uses compressed slides. A compressed slide is a linear morph between one drawing of a graph, and a second drawing that is itself the result of a sequence of slides performed on the first drawing. During a compressed slide, every vertex may be moving in a different direction. However, a morph based on compressed slides will in general require fewer linear morphs than a morph based on slides. It would be interesting to implement both algorithms, and to compare the morphs produced by each.

There are a number of heuristics that will improve the performance of our morphing algorithms in general. For example, in the first stage of our morphing algorithms we should add a straightening path that has the fewest bends possible. Another heuristic is, when in the second stage of the second algorithm, to perform a compressed slide based on as many zigzags as possible in each iteration, instead of limiting ourselves to one per straightening path. An avenue for future research is the exploration of such heuristics.

In any morph composed of a sequence of linear morphs, each vertex will follow a path that has a sharp turn wherever two linear morphs meet. It may be desirable that vertices move in smooth paths, i.e. paths of bounded curvature. One direction for future research is to produce parallel morphs of orthogonal drawings in the plane in which each vertex moves in a smooth path, possibly by modifying some given parallel morph that is composed of linear morphs.

Finally, can the algorithms we presented in this chapter be used as subroutines of

algorithms that generate simplicity-preserving morphs between non-parallel drawings? Already, our algorithms for morphing parallel orthogonal drawings have been applied to morphing between (non-parallel) orthogonal drawings of a graph in which each vertex is represented by a point, and each edge is represented by an orthogonal polyline [45].

Chapter 4

Morphing with Monotonically Changing Edge Lengths

4.1 Introduction

The previous chapter describes two algorithms for generating parallel morphs between parallel simple orthogonal drawings in the plane. Morphs generated by one of these algorithms can be thought of as having two stages: an expansive stage in which each edge is non-decreasing in length, followed by a linear morph in which edges change length monotonically.

For graph visualization, morphs generated by these algorithms may be undesirable since edge-lengths can grow exponentially with respect to the number of vertices in the underlying graph. When viewing such a morph using a device with fixed resolution, it might be impossible to display such a drawing in its entirety such that all elements are clearly distinguishable.

It was discussed how this exponential blowup can be avoided in Section 3.6. The idea is to take the sequence of drawings produced by the algorithm, and put the vertices onto a grid. When using this technique, the morphs no longer have the property that edge lengths change in two monotonic stages. Instead, each edge may alternately increase and decrease in length $\Omega(n)$ times, where n is the number of vertices in the underlying graph.

A morph may be more desirable if edges do not alternately grow and shrink several times. The strictest such parallel morph we might imagine is the following.

Definition 4.1.1 *A* monotone morph *is a parallel morph in which every edge is either non-increasing or non-decreasing in length over the course of the morph. We say that the*

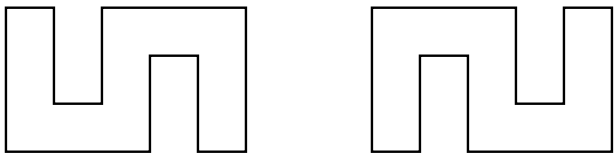

Figure 4.1: These polygons do not admit a monotone morph.

edges in such a morph change length monotonically.

In this chapter, we study monotone morphing. Our goal is to identify classes of drawing that will always admit a monotone morph. For those classes that do not always admit a morph, we would like to know whether there exists an efficient algorithm to decide whether a morph exists, and to compute a monotone morph if it exists.

Unfortunately, we have few positive results for the problem of monotone morphing. The following observation is implied by Lemma 2.1.1 (page 27).

Observation 4.1.1 *If a linear morph between parallel drawings maintains simplicity, then the linear morph is a monotone morph.*

One easy corollary of Theorem 2.3.1 (page 41) is the following.

Corollary 4.1.1 *For every simple star-shaped polygon, the linear morph from this polygon to any polygon that is parallel with it is a monotone morph.*

An orthogonal polygon is called *orthogonally convex* if every line, parallel with one of the coordinate axes, intersects the closed interior of the polygon in either a closed line segment, or an empty set. In Section 4.2 we prove that every pair of orthogonally convex polygons admits a monotone morph. This is surprisingly non-trivial to prove. For such polygons, a linear morph does not always suffice.

It is easy to see that parallel orthogonal polygons that are convex with respect to, say, the vertical axis (as opposed to being convex with respect to both vertical and horizontal axes, as is the case with orthogonally convex polygons) do not always admit a parallel morph; see Figure 4.1 for example. We discuss this problem briefly in Section 4.3, but have not been able to devise an efficient algorithm, nor an NP-hardness proof.

One might try to extend the positive result obtained for orthogonally convex polygons to the problem of monotone morphing for more general graphs in the plane in which all faces are orthogonally convex. However, parallel orthogonal drawings in the plane do not always admit a monotone morph, even when each interior face is orthogonally convex

and the boundary of the exterior face is an orthogonally convex polygon. This is proved in Section 4.4. The proof is constructive, and we use the unmorphable drawings devised in the proof to construct a pair of parallel orthogonally convex polyhedra that does not admit a monotone morph. In Chapter 5 we return to the questions of monotone morphing, proving it to be NP-hard to decide whether two orthogonal polygons admit a parallel morph.

In Section 4.5 we study another class of morph that is related to monotone morphs.

Definition 4.1.2 *A* bimonotone morph *is a parallel morph composed of two monotone morphs, performed one after the other.*

The morphs generated by the algorithms of Sections 3.2 and 3.3 are bimonotone. Thus every pair of parallel simple orthogonal drawings in the plane admits a bimonotone morph. In Section 4.5, we prove that for every pair of drawings in $\mathbb{R}^d$ that admits a parallel morph, there exists a bimonotone morph between the drawings. Lower bounds on the area requirements of monotone and bimonotone morphs are also established in this section.

Concluding remarks are made in Section 4.6.

4.2 Orthogonally-Convex Polygons

In this section, we prove the following.

Theorem 4.2.1 *Every pair of parallel simple orthogonally-convex polygons with n vertices admits a monotone morph consisting of at most $n-2$ steps, where in each step we change the lengths of exactly two edges.*

Throughout the proof, let P and Q denote the input source and target polygons, respectively (assumed to be orthogonally convex). Let G denote cycle graph underlying the polygons.

Definition 4.2.1 *An edge $e \in \mathrm{E(G)}$ is:*

- *a* +-edge *of P and a* −-edge of *Q, if e must increase in length during a monotone morph from P to Q;*
- *a* −*-edge of P and a* +*-edge of Q, if e must decrease in length during a monotone morph from P to Q; and,*
- *a* 0-edge *of both P and Q if e has the same length in both drawings, and thus remains static in length during a monotone morph.*

Definition 4.2.2 *Let $||P(e)||$ denote the L_2 (Euclidean) length of edge $e \in \mathrm{E(G)}$ in a drawing P of a graph G.*

Several times in the course of the proof, we use the technique of monotonically morphing P to an intermediate polygon P' and Q to an intermediate polygon Q', so that the changes in edge lengths match the labels. That is to say, if e is a +-edge of P then

$$||P(e)|| \leq ||P'(e)|| \leq ||Q'(e)|| \leq ||Q(e)||.$$

The problem is reduced to the problem of finding a monotone morph from P' to Q', because such a morph, preceded by the morph from P to P', and followed by the reverse of the morph from Q to Q', provides a monotone morph from P to Q.

Applying this idea for the first time, we morph to alter the horizontal edges of P and Q without changing the lengths of the vertical edges. Vertical edges are treated later. More precisely, we morph P to P' and Q to Q' so that corresponding horizontal edges of P' and Q' have the same length, corresponding vertical edges of P and P' have the same length, and corresponding vertical edges of Q and Q' have the same length.

The edges of an orthogonally convex polygon can be partitioned into four *chains* by breaking the cycle of edges at the leftmost and rightmost vertical edges, and the uppermost and lowermost horizontal edges, as shown in Figure 4.2. These chains are referred to as upper-left, upper-right, lower-left, and lower-right, in the obvious way. Each of the four extreme edges is considered to be part of two chains.

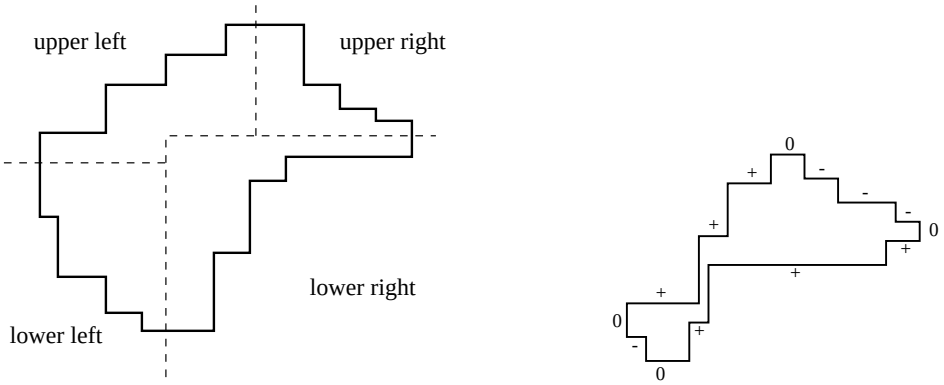

Figure 4.2: The four chains, and a polygon in alternating form.

We morph to realize certain properties of these chains. In particular, a chain is *positive* [*negative*] if all horizontal edges in it are +-edges or 0-edges [−-edges or 0-edges]; and a chain is *mixed* if it contains both +- and −-edges. A polygon is in *alternating form* if its

chains are alternately positive and negative. See Figure 4.2. Our monotone morph has the following steps: (1) morph to eliminate mixed chains; (2) morph to put each polygon in alternating form; (3) morph so that all horizontal edges are 0-edges. The morphs in steps (1) and (2) (see Figure 4.3) strictly increase the polygon (by containment) and thus preserve simplicity; the morphs in step (3) (see Figure 4.4) need a more careful justification to establish that they preserve simplicity.

Lemma 4.2.1 *A pair of parallel orthogonally convex polygons P and Q can be morphed monotonically to arrive at two intermediate polygons P' and Q' with no mixed chains.*

Proof: Without loss of generality consider the horizontal edges of the upper-left chain of P and Q. Suppose the chain has two edges e_1 and e_2, in clockwise order, with opposite sign. In one of P or Q, e_1 must be a $-$-edge and e_2 a $+$-edge. Then the morph shown in Figure 4.3 (left) is monotone (in particular, it preserves simplicity) and can be continued until either e_1 or e_2 becomes a 0-edge. Repeating this process yields the result. □

Lemma 4.2.2 *A pair of parallel orthogonally convex polygons P and Q with no mixed chains can be morphed monotonically to arrive at two intermediate polygons P' and Q' in alternating form.*

Proof: Because we can morph in P or in Q (which has opposite signs) it suffices to show how to eliminate consecutive chains with $+$-edges. If both top chains of P contain $+$-edges then, because they are not mixed chains, the width of P must be less than the width of Q. In this case a lower chain of P must also have a $+$-edge. Thus we reduce to the case where, say, the two left chains of P contain $+$-edges. Then the morph shown in Figure 4.3 (right) is monotone (in particular, it preserves simplicity) and can be continued until one edge becomes a 0-edge. Repeating this process yields the result. □

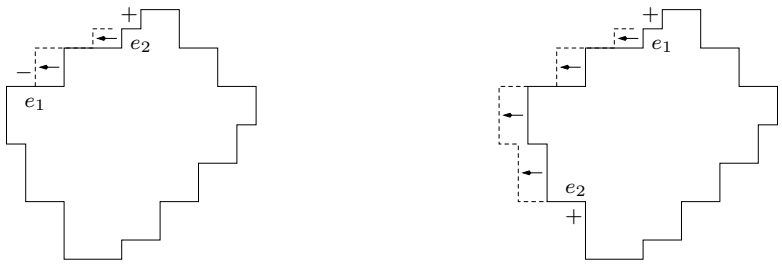

Figure 4.3: (Left) morphing to eliminate a mixed chain, and (right) morphing to eliminate two positive left chains.

Lemma 4.2.3 *A pair of parallel orthogonally convex polygons P and Q in alternating form can be monotonically morphed to arrive at two intermediate polygons P' and Q' whose horizontal edges are all 0-edges.*

Proof: Assume (by renaming if necessary) that the upper left chain of P has a +-edge. Thus P's lower left chain is a $-$-chain; Q's upper left chain is $-$, Q's lower left chain is +, etc. The idea is to trade off a +-edge in P's upper left chain against either a $-$-edge in P's upper right chain, or a +-edge in P's lower right chain; see Figure 4.4. However, we must guard against the possibility of P's upper left chain colliding with the lower right chain, and find it necessary to morph Q rather than P in some situations.

Let $l^+(P)$ be the horizontal length of P's left +-chain not counting the uppermost edge. Define $l^-(P)$, $l^+(Q)$, and $l^-(Q)$ for the other left chains analogously. Note that $l^+(P) \leq l^-(Q)$ and $l^+(Q) \leq l^-(P)$, since these are corresponding chains that must be morphed into each other.

Claim 4.2.1 *For either P or Q, it must be that $l^- > l^+$.*

Proof: To the contrary, assume that $l^-(P) \leq l^+(P)$ and $l^-(Q) \leq l^+(Q)$. Then,

$$l^-(P) \leq l^+(P) \leq l^-(Q) \leq l^+(Q) \leq l^-(P)$$

so all left chains are equal. However, since $l^+(P) = l^-(Q) = l^-(P) = l^+(Q)$ and no chains are mixed, it must be that all horizontal edges in the left chains of P and Q are 0-edges. It follows that all horizontal edges in both P and Q are 0-edges. This contradicts the earlier assumption that the upper left chain of P has a +-edge. □

Finally, observe that if $l^-(P) > l^+(P)$ then the morphs shown in Figure 4.4 preserve simplicity, where a +-edge in the upper-left chain of P is traded off against either a $-$-edge in the upper-right chain of P, or a +-edge in the lower right chain of P. (If instead $l^-(Q) > l^+(Q)$, then similar morphs in Q preserve simplicity.)

During this morph, the lower left chain cannot collide with the upper right chain since the edges in the upper right chain move strictly rightward, while those in the lower left chain remain stationary. In P, all edges in the upper left chain lie strictly to the left of the edges in the lower right chain. Therefore, the +-edge in the upper left chain can be extended without collision until either it becomes a 0-edge, or $l^-(P) = l^+(P)$. However, if $l^-(P) = l^+(P)$ then as shown in Claim 4.2.1, all horizontal edges must be 0-edges. □

This completes the description of the monotone morph between two parallel orthogonally convex polygons. To complete the proof, it remains to analyze the number of steps involved in the morph.

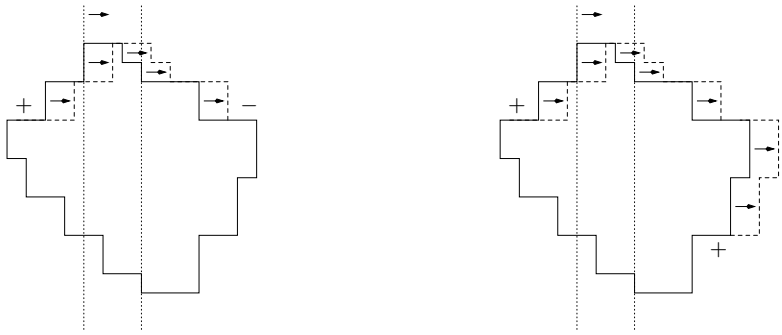

Figure 4.4: Morphing to trade-off a +-edge in the upper left chain against an edge of the right chain.

The number of steps needed to morph horizontal edges is at most the number of horizontal edges minus one (i.e., $n/2-1$), since with every step we convert at least one horizontal edge labeled + or − into an edge labeled 0 (this happens at most once per edge). In the final step, exactly two horizontal edges are labeled 0. Repeating the same argument for the vertical edges, we see that we can morph any orthogonally convex polygon monotonically with at most $n-2$ steps.

4.3 Vertically Convex polygons

In this section, we discuss the possibility of computing monotone morphs for another class of orthogonal polygons.

Definition 4.3.1 *A polygon in the plane is said to be* vertically convex[1] *if every vertical line intersects the closed interior of the polygon in either an empty set, or a closed line segment.*

Every orthogonally convex polygon is also vertically convex. We define horizontally convex polygons analogously.

It is not difficult to construct pairs of vertically convex polygons that do not admit a monotone morph; recall Figure 4.1 for example. Throughout a monotone morph between these two polygons, each vertical edge must remain static in length. So clearly, no monotone morph is possible.

For some pairs of parallel vertically convex polygons, the order in which we change edge lengths is critical to maintaining edge monotonicity. Figure 4.5 illustrates a monotone

[1]Vertically convex polygons are normally called *x-monotone*.

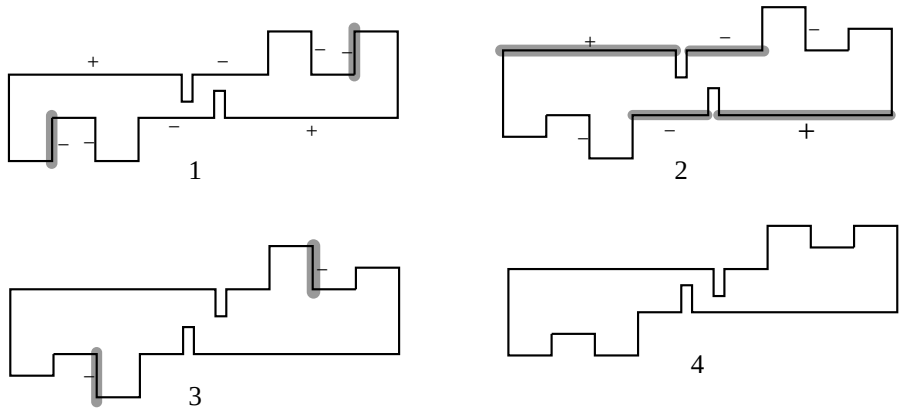

Figure 4.5: Intermediate drawings of a monotone morph between two vertically convex polygons.

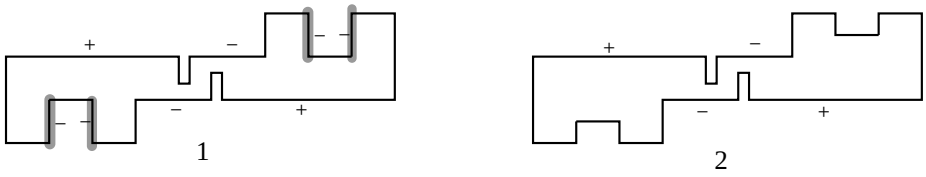

Figure 4.6: A monotone morph that takes each vertical edge to its target length—there is no way to morph to the target drawing using a monotone morph.

morph between two vertically convex polygons. This monotone morph is composed of a sequence of three linear morphs. The labels + and − in each drawing of this figure indicate how edges must change as we go from the source (1) to the target (4). The 0-edges are unlabeled. For each drawing, the highlighted edges are the ones that change as we go to the next drawing in the sequence.

In the first linear morph, two vertical edges shorten to their target lengths. In the second linear morph, two horizontal edges lengthen and two shorten to their target lengths. In the final linear morph to the target, two vertical edges shorten.

Now, let us consider a different morph between the same source and target drawings. Without altering horizontal coordinates of the vertices, morph the source drawing so that all vertices have their target vertical coordinates; see Figure 4.6. Following this morph, all vertical edges are at their target lengths. However, from this drawing no parallel morph will take us the rest of the way to the target without altering lengths of the vertical edges.

This example illustrates why computing a monotone morph between a pair of vertically convex polygons is more difficult than for orthogonally convex polygons. If we choose to change edge lengths in the wrong order, we can get stuck. We leave it as an open problem:

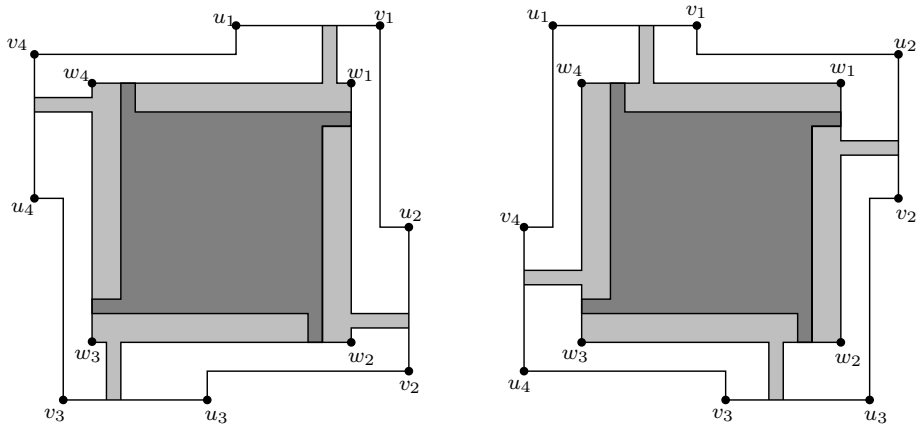

Figure 4.7: Parallel orthogonal drawings of a graph that do not admit a monotone morph; each interior face is an orthogonally convex polygon.

does there exist an efficient algorithm to decide whether a given pair of vertically convex polygons admits a monotone morph?

4.4 Drawings with Orthogonally Convex Faces

In Section 4.2 we proved that every pair of parallel orthogonally convex polygons admits a monotone morph. A natural generalization of this is to drawings of graphs in which all faces are bounded by orthogonally convex polygons. We prove below that not all such drawings will admit a parallel morph. The pair of unmorphable drawings we describe can be used to prove that not all pairs of parallel orthogonally convex polyhedra will admit a parallel morph.

Theorem 4.4.1 *There exists a pair of parallel drawings that does not admit a monotone morph, such that, in both drawings:*

1. *the boundary of each interior face is an orthogonally convex polygon; and,*
2. *the boundary of the union of the interior faces is an orthogonally convex polygon.*

Proof: Consider the pair of parallel drawings shown in Figure 4.7. The shading on the faces is to aid the reader in distinguishing between the faces. It is easy to verify that these drawings are parallel and have the two necessary orthogonal convexity properties. It is established below that this pair of drawings does not admit a monotone morph.

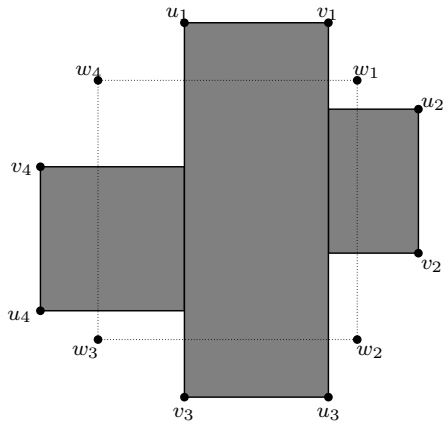

Figure 4.8: Region R cannot contain all of $w_1, \ldots, w_4$.

Let P be the left drawing of Figure 4.7 and let Q be the right drawing. Notice that every edge of the darkly shaded central face has the same length in both P and Q. Assume—for a proof by contradiction—that there exists a monotone morph M from P to Q.

In both P and Q the vertices $w_1, \ldots, w_4$ form the corners of a square—the edges of this square do not appear in the drawings. The side-length of the square is the same in both P and Q. Let α denote this common side-length. For all $i, j \in \{1, \ldots, 4\}$,

$$||P(u_i), P(v_i)||_2 = ||P(u_j), P(v_j)||_2 = ||Q(u_i), Q(v_i)||_2 = ||Q(u_j), Q(v_j)||_2.$$

Let β denote this distance. From Figure 4.7, it is clear that $\alpha > \beta$.

If an edge has equal lengths in P and Q, then this edge must remain of constant length throughout M. For every pair u_i, v_i there exists a path of four collinear vertices between u_i and v_i, such that every edge in the path has equal lengths in both P and Q. Therefore, the relative coordinates of each pair u_i, v_i remains constant throughout M. By similar reasoning, the relative coordinates of each pair w_i, w_j remains constant throughout M.

Since u_1 lies to the right of v_3 in P and to the left of v_3 in Q, there exists some $t_1 \in [0, 1]$ such that u_1 and v_3 have equal horizontal coordinates in $M(t_1)$. Also, v_1 and u_3 have equal horizontal coordinates in $M(t_1)$.

Let R denote the set of all points of the plane that lie outside of the open exterior face of $M(t_1)$. Region R can be described as the union of three rectangles (see Figure 4.8): a central rectangle of width β, and two rectangles of height β attached as shown on either side of the central rectangle. By definition, R must contain the four vertices $w_1, \ldots, w_4$ as located in $M(t_1)$. However, since $\alpha > \beta$, it is clear that R cannot contain all of these vertices at once. Contradiction. □

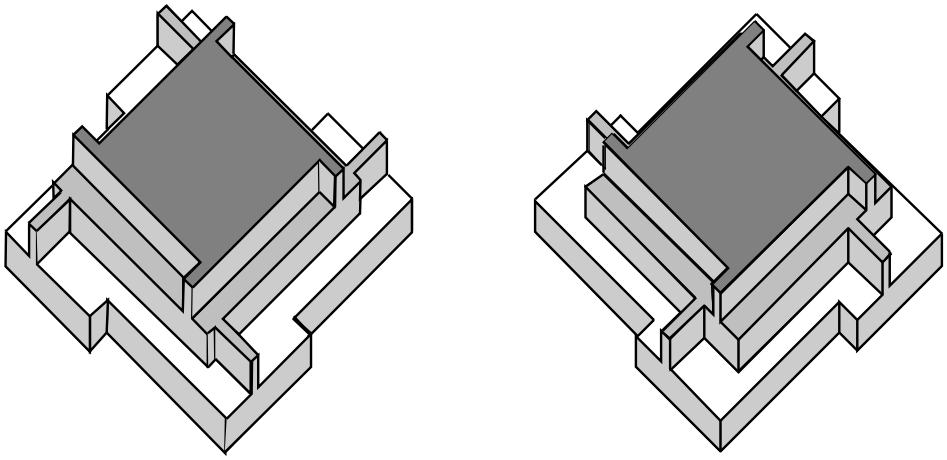

Figure 4.9: A pair of orthogonally convex polyhedra that does not admit a parallel morph.

Corollary 4.4.2 *There exist pairs of parallel orthogonally convex polyhedra that do not admit a parallel morph.*

Proof: We use the drawings from Figure 4.7 (assume that these lie in the x-y plane) to construct a pair of parallel, orthogonally convex polyhedra in x-y-z space that do not admit a monotone morph.

Generate a polyhedron from each drawing. Each face of one of the two drawings in the x-y plane becomes a face of a polyhedron, such that this face lies in a plane perpendicular to the z-axis. The x-y coordinates of each vertex of a polyhedron are equal to the x-y coordinates of some vertex of the corresponding drawing; see Figure 4.9.

In each polyhedron, the darkly shaded central face in the generating drawing has the highest z-coordinate. The lightly shaded faces are the second highest, and the unshaded faces are lowest. The polyhedra are orthogonally convex, and every face is bounded by an orthogonally convex polygon.

It is clear that if these polyhedra admit a monotone morph, then so do the drawings of Figure 4.7. Hence, no monotone morph exists between the two polyhedra. □

Although the parallel polyhedra described in Corollary 4.4.1 do not admit a monotone morph, it is not difficult to see that they will admit a *parallel morph*. What is more surprising is that not all pairs of parallel orthogonally convex polyhedra admit a parallel morph. We establish this in Chapter 6.

4.5 Bimonotone Morphing

As we have seen so far in this chapter, restricting parallel morphs to have monotonically changing edge lengths severely limits the class of morphable drawings. If we weaken this restriction to allow parallel morphs that are composed of a sequence of two monotone morphs, then we can morph between every pair of parallel simple orthogonal drawings in the plane (Theorem 3.2.1, page 58) and between every pair of polygons that are parallel with a third, star-shaped polygon (Corollary 2.3.2, page 48). In the introduction to the present chapter, we defined such a morph as *bimonotone*.

In this section, we prove that every pair of drawings that admits a parallel morph will admit a morph that is bimonotone. We then discuss lower bounds on the area requirements of bimonotone morphs.

Let us identify special classes of monotone and bimonotone morphs that will be of interest to us in this section. Within these special classes, we are able to prove lower bounds on area requirements of a bimonotone morph, while we have been unable to do so for bimonotone morphs in general.

Definition 4.5.1 *A $(+)$-morph is a monotone morph in which every edge is non-decreasing in length throughout the morph. A $(-)$-morph is defined analogously.*

Definition 4.5.2 *A $(+,-)$-morph is a parallel morph composed of a $(+)$-morph followed by a $(-)$-morph. A $(-,+)$-morph is defined analogously.*

In Section 4.5.1 we prove that every pair of drawings that admits a parallel morph will also admit both a $(+,-)$-morph and a $(-,+)$-morph. In Section 4.5.2 we establish exponential worst-case lower bounds on the area of intermediate drawings of $(+,-)$-morphs.

4.5.1 On the existence of bimonotone morphs

Theorem 4.5.1 *If there exists a parallel morph from a drawing P in $\mathbb{R}^d$ to a parallel drawing Q, then there exists both a $(+,-)$-morph and a $(-,+)$-morph from P to Q.*

Proof: Let M denote a parallel morph from P to Q, and let G denote the graph underlying both drawings. We prove only the existence of a $(+,-)$-morph from P to Q. The existence of a $(-,+)$-morph can be established similarly.

We generate a $(+,-)$-morph M' from M by scaling up each intermediate drawing of M by an amount sufficiently large that every edge is at least as long in $M'(t)$ as it is in

$M(t)$, for all $t \in [0,1]$. This is a $(+)$-morph. To complete the $(+,-)$-morph to Q, compose M' with a scaling from $M'(1)$ to Q.

To generate M', define a function f such that for all $t \in [0,1]$,

$$f(t) = \max_{\{u,v\} \in \mathrm{E}(G)} \left(\frac{\sup_{t_0 \in [0,t]} ||M(t_0; u, v)||}{||M(t; u, v)||} \right).$$

Next, define M' as the morph of G such that, for all $t \in [0,1]$ and $v \in V$,

$$M'(t; v) = f(t) \cdot M(t; v).$$

We must prove that M' is a $(+)$-morph, such that $M'(1)$ is a scaling up of Q. Since vertices follow continuous paths in M, each edge of G changes length continuously over time. It follows that f is continuous. Thus, each vertex follows a continuous path in M'. So, M' is a morph of G.

For all $t \in [0,1]$, $M'(t)$ is a scaling of $M(t)$, so it is simple and parallel with P and Q. Therefore, M' is a parallel morph.

Observe that, for all $t_1 < t_2$ where $t_1, t_2 \in [0,1]$,

$$\begin{aligned} ||M'(t_2; u, v)|| &\geq ||M(t_2; u, v)|| \left(\frac{\sup_{t_0 \in [0,t_2]} ||M(t_0; u, v)||}{||M(t_2; u, v)||} \right) \\ &\geq ||M(t_1; u, v)|| \left(\frac{\sup_{t_0 \in [0,t_1]} ||M(t_0; u, v)||}{||M(t_1; u, v)||} \right) \\ &= ||M'(t_1; u, v)||. \end{aligned}$$

Therefore, M' is a $(+)$-morph. □

4.5.2 Area requirements of a bimonotone morph

The area of a drawing in the plane is defined as the area contained in the smallest axis-aligned bounding box that contains the drawing (Definition 1.3.10, page 17).

Below, we prove that, for any given n, there exists a pair of parallel orthogonal drawings—described in terms of their vertex orderings instead of vertex coordinates—each with n vertices, such that there exists a $(+)$-morph from the source to the target if and only if the area of the target is exponentially larger in n than than the area of the source.

Since a $(+)$-morph in reverse is a $(-)$-morph, the above result implies that there exists a pair of drawings such that a $(-)$-morph exists between them if and only if the area of the target is exponentially smaller in n than the area of the source.

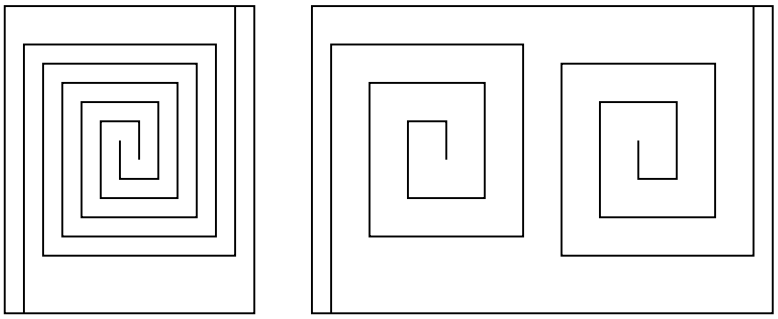

Figure 4.10: *Left:* boxed spirals with spirals entwined. *Right:* boxed spirals with spirals untwined.

We use these results to prove exponential lower bounds on the areas of intermediate drawings of various bimonotone morphs, include $(+,-)$-morphs.

Define a *boxed spiral* as an orthogonal drawing in the plane that consists of a box (rectangle) inside of which is a spiral that is affixed to the box at one of the end-vertices of the spiral. Multiple spirals can be contained in the same box. Figure 4.10 illustrates a pair of parallel orthogonal drawings, each of which is a pair of boxed spirals sharing a common box: in Figure 4.10 (left) the spirals are *entwined* and in Figure 4.10 (right) the spirals *untwined*.

Each of the spirals shown in Figure 4.10 has a total of twelve vertices (this includes the vertices connecting the spirals to the bounding rectangle). It is clear that for any positive integer k, we can generate a pair of boxed spirals in where each spiral contains k vertices. The total number of vertices in such a drawing is $n = 2k + 4$.

Lemma 4.5.1 *Let P be a drawing of a pair of entwined boxed spirals of k vertices, each of which has integer coordinates, and with area (k^2). Let Q be a drawing parallel with P, but with the spirals untwined. If there exists a $(+)$-morph from P to Q, then the area of Q is $2^{\Omega(k)}$.*

Proof: Let M be a $(+)$-morph from P to Q. Let G denote the graph underlying P and Q, where $u_0, \ldots, u_k \in \mathrm{V}(G)$ and $v_0, \ldots, v_k \in \mathrm{V}(G)$ denote the vertices of each of the two spirals, listed in order starting with the vertices u_0, v_0 that are incident with the box.

Each spiral induces a hierarchy of nested axis-aligned boxes: for every $i \in \{4, \ldots, k\}$ and $t \in [0, 1]$, let $B_i^v(t)$ denote the axis-aligned box with corners at $M(t; v_{i-1}), M(t; v_{i-2}), M(t; v_{i-3})$, and the point at which the line containing $M(t; v_i)$ and $M(t; v_{i-1})$ intersects the line containing $M(t; v_{i-3})$ and $M(t; v_{i-4})$; see Figure 4.11. Define $B_i^u(t)$ similarly. We will some-

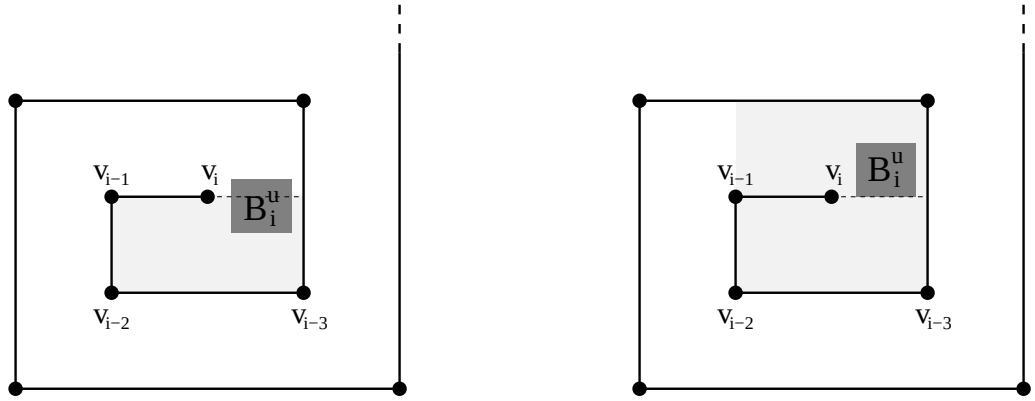

Figure 4.11: *Left:* B_i^u and B_i^v intersect, and B_{i-1}^v contains B_i^u. *Right:* the interiors of B_i^u and B_i^v are disjoint, but B_{i-1}^v still contains B_i^u.

times use the notation B_i^v without the parameter t to refer to a box over all times during the morph.

The nested boxes of each spiral in P remain nested in all drawings that are parallel with P—i.e. at all times $t \in [0,1]$, $B_i^u \subset B_{i-1}^u$ and $B_i^v \subset B_{i-1}^v$. This is because the spirals are anchored on the outer box.

When two spirals are entwined (as in P) each nested box of one spiral intersects at least one nested box of the other spiral. When the spirals are untwined (as in Q) two boxes belonging to different spirals are pairwise disjoint. That is to say, for all pairs $i, j \in \{4, \ldots, k\}$, $B_i^u(0)$ intersects $B_j^v(0)$, while $B_i^u(1)$ is disjoint from $B_j^v(1)$.

Since the nested boxes of each spiral retain their nested structure throughout M, if for some $t \in [0,1]$ two boxes $B_i^u(t)$ and $B_i^v(t)$ intersect, then the enclosing boxes (if they exist) $B_{i-1}^u(t)$ and $B_{i-1}^v(t)$ must also intersect. So, fix a sequence $0 \leq t_n \leq t_{n-1} \leq \ldots \leq t_4 \leq 1$ such for each i, B_i^u becomes disjoint from B_i^v at time t_i.

Observe that for every $i \in \{5, \ldots, k\}$, at time t_i boxes B_i^u and B_i^v are disjoint, but B_{i-1}^u contains B_i^v, and B_{i-1}^v contains B_i^u; see Fig. 4.11 (left and right). Therefore, in a $(+)$-morph from P,

$$\text{Area}(B_{i-1}^u(t_{i-1})) \geq \text{Area}(B_{i-1}^u(t_i)) \geq \text{Area}(B_i^u(t_i)) + \text{Area}(B_i^v(t_i))$$

and

$$\text{Area}(B_{i-1}^v(t_{i-1})) \geq \text{Area}(B_{i-1}^v(t_i)) \geq \text{Area}(B_i^u(t_i)) + \text{Area}(B_i^v(t_i))$$

where $\text{Area}(B)$ denotes the area of a box B. Since P is embedded on a unit grid, $\text{Area}(B_k^u(0)) \geq 1$ and $\text{Area}(B_k^u(0)) \geq 1$. Therefore, the area of each of $B_4^u(t_4)$ and $B_4^v(t_4)$ is $2^{\Omega(k)}$. □

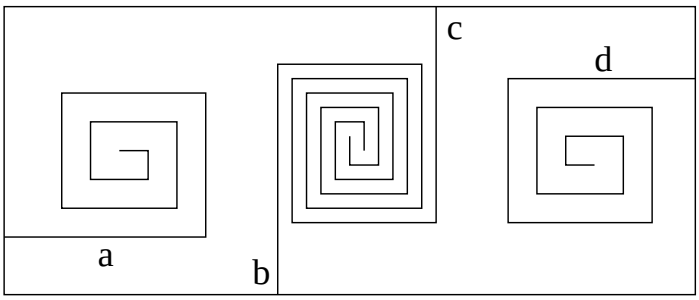

Figure 4.12: Any $(+,-)$-morph that untwined b and c, and entwined a and d, must have an edge of exponential length.

A $(+)$-morph run in reverse is a $(-)$-morph. If we exchange the source and target drawings devised in above proof, instead of untwining a pair of boxed spirals we must entwine a pair of untwined spirals. The following lemma is obtained thereby.

Lemma 4.5.2 *Let Q be a drawing of a pair of entwined boxed spirals of k vertices, each of which lies at integer coordinates, and with area that is at most $O(k^2)$. Let P be a drawing parallel with Q, but with the spirals untwined. If there exists a $(-)$-morph from P to Q, then P has area that is $2^{\Omega(k)}$.*

Theorem 4.5.2 *For any integer $n \geq 4$, there exists a pair of simple parallel orthogonal drawings in the plane, such that both drawings have n^2 area, and every parallel morph from source to target that is composed of a $(+)$-morph followed by a single scaling operation has an intermediate drawing with area $2^{\Omega(n)}$.*

Proof: For the source drawing, we use a pair of entwined boxed spirals; and for the target, we have the spirals untwined. Since the drawings are orthogonal and have n vertices, in each drawing every vertex can be located at integer coordinates such that the drawing has area n^2 (recall our discussion in Section 3.6).

Each spiral has $\lfloor (n-4)/2 \rfloor = \Omega(n)$ vertices. By Lemma 4.5.1, any $(+)$-morph that untwines the spirals must terminate with a drawing whose area is $2^{\Omega(n)}$. Therefore, in any parallel morph that is a $(+)$-morph followed by a scaling, there must be an intermediate drawing of the morph that has this area. □

We can obtain a similar exponential decrease in area for parallel morphs composed of a $(-)$-morph followed by a scaling. Let us consider worst-case area bounds for intermediate drawings of $(+,-)$-morphs.

Theorem 4.5.3 *For any positive integer n there exists a pair of parallel orthogonal drawings in the plane with n vertices, such that all vertices lie at integer coordinates and both drawings have n^2 area, and such that in any $(+,-)$-morph between the drawings, there exists an intermediate drawing with area that is $2^{\Omega(n)}$.*

Proof: Examine the orthogonal drawing consisting of four boxed spirals a, b, c and d shown in Fig. 4.12. Spirals b and c are entwined and spirals a and d are untwined. Call this drawing P. We assume that P is drawn on a unit grid with n^2 area, where n is the number of vertices. The number of vertices contained in each spiral is $\lfloor (n-4)/4 \rfloor = \Omega(n)$.

Now, imagine a drawing Q that is parallel with P, except that spirals a and d are entwined while b and c are untwined. Assume that Q is also on the unit grid with area n^2.

In any parallel morph from P to Q we cannot begin to entwine a and d until b and c are untwined. More precisely, in every simple drawing that is parallel with P and Q the nested *boxes* (as defined in the proof of Lemma 4.5.1) induced by spirals a and d must remain disjoint unless all nested boxes induced by b and c become disjoint.

Let M be a $(+,-)$-morph from P to Q. Let $t_0 \in [0,1]$ denote the time at which in M, nested boxes of a and d change from being pairwise disjoint, to having intersection. Let $t_1 \in [0,1]$ be the time at which M switches from $(+)$-morph to $(-)$-morph.

If $t_0 \leq t_1$, then the spirals b and c have become untwined using a $(+)$-morph. By Lemma 4.5.1 there exists an intermediate drawing of M with area that is $2^{\Omega(n)}$. On the other hand, if $t_0 > t_1$, then spirals a and d must be entwined using a $(-)$-morph. By Lemma 4.5.2 some intermediate drawing of the $(+,-)$-morph has area that is $2^{\Omega(n)}$. □

We have been unable to produce a construction that leads to an exponential change in the area of intermediate drawings of $(-,+)$-morphs. We note that this is not implied by Theorem 4.5.3 since a $(+,-)$-morph in reverse is again a $(+,-)$-morph. We leave it as an open problem to determine whether $(-,+)$-morphs exhibit exponential growth or shrinkage.

4.6 Conclusion

In this chapter, we have considered parallel morphs with the added constraint that edges are limited in the number of times in which they may grow and then shrink.

We first introduced monotone morphs, in which each edge may be either non-decreasing or non-increasing in length throughout the morph. When we restrict ourselves to monotone morphs, the class of drawings that can be morphed is severely limited. We are able to prove

that orthogonally convex polygons always morph in this context, but that even for vertically convex polygons, the problem of deciding whether or not a monotone morph exists does not seem to be straightforward. In the next chapter, we prove that deciding monotone morphability for orthogonal polygons is NP-hard.

On the other hand, every pair of drawings that admits a parallel morph will admit a bimonotone morph. Bimonotone morphs that consist of a (+)-morph followed by a (−)-morph will, in the worst case, have intermediate drawings that are exponentially larger in area than source and target, even when we limit our attention to orthogonal drawings in the plane. To animate such a morph, it is likely that we will be forced to either display only a portion of the entire drawing at one time or else scale intermediate drawings down, and thereby lose details of the drawing due to the fixed resolution of the display device.

From the point of view of graph visualization, is it desirable that edges change length monotonically? Do there exist efficient algorithms that generate parallel morphs of orthogonal drawings in the plane and achieve some sort of balance between edge-length monotonicity and area requirements? We leave these as problems for future research.

Chapter 5

NP-Hardness of Morphing in the Plane

5.1 Introduction

In this chapter, we establish NP-hardness for three problems related to parallel morphing in the plane. First, in Section 5.2 we prove it to be NP-hard to decide whether there exists a parallel morph between two orthogonal polygons when we are given a subset of edges that have equal lengths in both the source and target polygons, and require that during the morph these edges remain static in length. Besides being an interesting result on its own, this result is useful to subsequent NP-hardness proofs in this chapter.

In Section 5.3 we prove that monotone morphing is NP-hard by modifying the hardness proof for the problem of morphing with static edges. Finally, in Section 5.4 we prove that to decide whether there exists a parallel morph between a pair of parallel simple (non-orthogonal) drawings in the plane is NP-hard. This is established with a reduction from the problem of deciding parallel morphability with static edges.

5.2 Orthogonal Polygons with Static Edges

Let P and Q be parallel orthogonal polygons. For each edge of the underlying graph, each edge is labeled as being either 'elastic' or 'static'. A parallel morph from P to Q *respects the static edges* if each static edge remains unchanged in length throughout the morph. We are not restricting to monotone morphs here, so the elastic edges may change arbitrarily in length, though they may not shrink to zero length. Clearly, every static edge must have the same length in both P and Q for there to exist a morph.

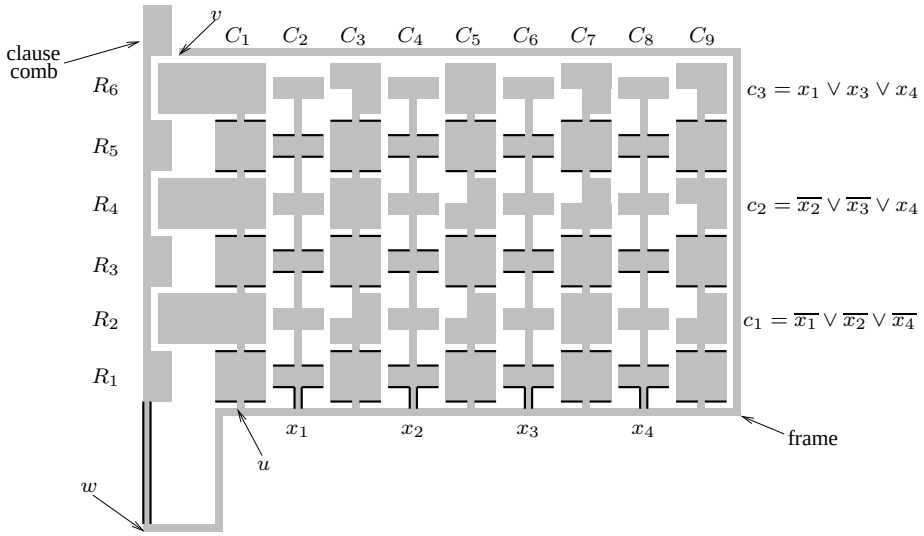

Figure 5.1: The source polygon, P. Elastic edges are drawn with heavy lines.

Theorem 5.2.1 *Given parallel, orthogonal polygons P and Q and a labeling of each edge as either elastic or static, it is NP-hard to decide whether P and Q admit a parallel morph from P to Q that respects the static edges.*

We reduce from Boolean Satisfiability [29]: given a CNF-boolean expression ϕ with m clauses $c_1, \ldots, c_m$, over n variables $x_1, \ldots, x_n$, can the variables be assigned a value (TRUE or FALSE) in a way that satisfies ϕ? Without loss of generality, assume that not both a variable and its negation appear in a clause of ϕ. We construct parallel orthogonal polygons P, Q with an edge labeling such that a parallel morph with static edges exists if and only if ϕ is satisfiable.

5.2.1 The construction.

We illustrate our construction with an example. Suppose that ϕ has three clauses: $c_1 = \overline{x_1} \vee \overline{x_2} \vee \overline{x_4}$, $c_2 = \overline{x_2} \vee \overline{x_3} \vee x_4$ and $c_3 = x_1 \vee x_3 \vee x_4$. For this input, construct the parallel polygons P and Q shown in Figures 5.1 and 5.2. The interiors of the polygons are shaded. Elastic edges are drawn with heavy lines. All other edges (on the boundary of each polygon) are static.

In this construction, both P and Q consist of an array of *blocks* surrounded on three sides by a rigid *frame*. The comb-like portion of P and Q attached to the frame by vertical elastic edges is called the *clause comb*. The two polygons differ only in the lengths of the elastic edges connecting the clause comb to the frame: to morph from P to Q these edges

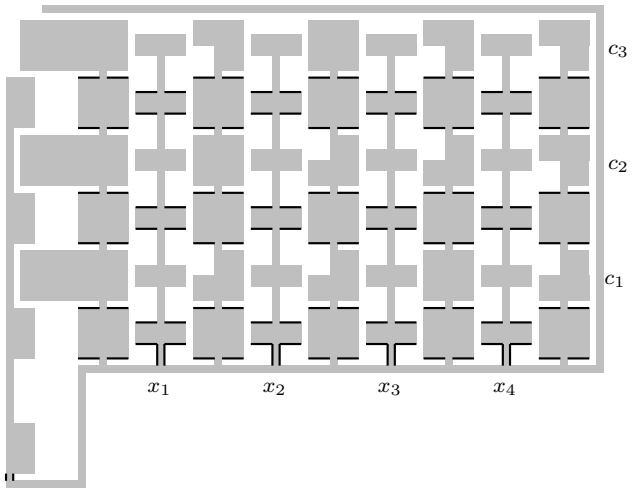

Figure 5.2: The target polygon, Q, in which the clause comb has moved downward.

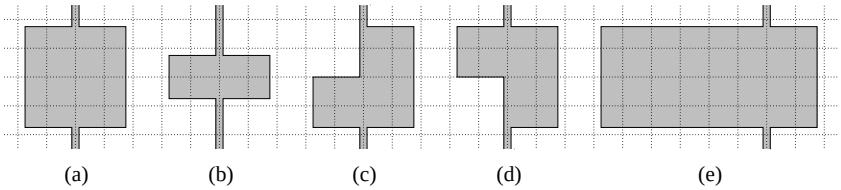

Figure 5.3: The basic block types used in the construction.

must contract (from length $8 + \epsilon$ to length ϵ), thereby moving the clause comb downward with respect to the frame.

In P, vertical movement of the clause comb is hampered by blocks protruding into indentations in the clause comb. In any morph from P to Q every one of these blocks must be in a retracted state (i.e. moved rightward, out of its indentation) simultaneously. Only then, the clause comb is free to move downward. After moving the clause comb downward, the blocks must return to their original positions to arrive at Q.

Consider the blocks used in the construction; see Figure 5.3. Fix some small positive value $\epsilon = (4n + 100)^{-1}$. Ignoring the vertical edges that connect a block to blocks above or below it, the dimensions of blocks of type (a), (c) and (d) are $(4 - \epsilon) \times (4 - \epsilon)$. The dimensions of (b) are $(4 - \epsilon) \times (2 - \epsilon)$. Type (e) has dimensions $(8 - \epsilon) \times (4 - \epsilon)$.

There are $2n + 1$ columns of blocks, and $2m$ rows. Beginning at the leftmost column, denote the columns $C_1, \ldots, C_{2n+1}$; beginning with the lowest row, denote the rows $R_1, \ldots, R_{2m}$. Each block is connected by two vertical edges—spaced ϵ apart—to the block below it, except for the blocks in R_1 which are connected to the frame. The horizontal gap between each block and its neighbor to the right is of size ϵ. A horizontal gap of size ϵ

appears between each block in the rightmost column C_{2n+1} and the inner wall of the frame. The vertical gap between the inner wall of the frame and every block in the highest row R_{2m} and in odd-numbered columns is also of size ϵ.

Even-numbered columns (i.e. $C_2, C_4, C_6, \ldots$) correspond to the variables that appear in ϕ. These columns are attached to the frame by vertical elastic edges of length $1 + \epsilon$, allowing blocks in a column to move vertically with respect to the frame. Blocks in odd-numbered columns (i.e. $C_1, C_3, C_5, \ldots$) are attached to the frame by static edges of length ϵ, and are allowed only very limited vertical movement (using elastic edges of neighboring columns).

Even-numbered rows (i.e. $R_2, R_4, R_6, \ldots$) correspond to the clauses of ϕ. Notice that only even-numbered rows contain blocks that protrude into the clause comb. All blocks in odd-numbered rows (i.e. $R_1, R_3, R_5, \ldots$) have horizontal elastic edges that allow the blocks in even-numbered rows to move horizontally with respect to the frame.

Blocks in the odd-numbered columns serve to mate the variable-columns to the clause-rows as follows: let x and c be a variable and a clause of ϕ, respectively. Let C_i be the (even-numbered) column that corresponds to x and let R_j be the (even-numbered) row that corresponds to c. If x does not appear in c then, in position (C_{i+1}, R_j) of the block array, place a block of type (a) (recall Figure 5.3). Otherwise, if the literal x appears in c place a block of type (c) at this location, and if literal $\overline{x}$ appears in c use a block of type (d). The blocks to appear at other locations of the block array follow the pattern illustrated in Figures 5.1 and 5.2.

We construct the clause comb such that in P and Q there is a $1 - \epsilon$ overlap—with respect to the horizontal axis—between the clause comb and the blocks that protrude into the clause comb.

5.2.2 When the expression is satisfiable.

Suppose that ϕ is satisfiable. Fix a particular satisfying assignment for ϕ. To establish that there exists a parallel morph from P to Q that respects the static edges, we begin morphing in P by moving some of the even-numbered columns upward and moving others downward. The direction moved by each of these columns is determined by the values of the variables in the satisfying assignment of ϕ.

For a variable x that corresponds to column C_i, if x is TRUE in the satisfying assignment, move the column downward, shrinking the elastic edges that connect C_i to the frame to length ϵ. If x is FALSE, then move the column upward by expanding these elastic edges to length $2 + \epsilon$. Figure 5.4 shows a drawing in which the columns in P corresponding to x_2

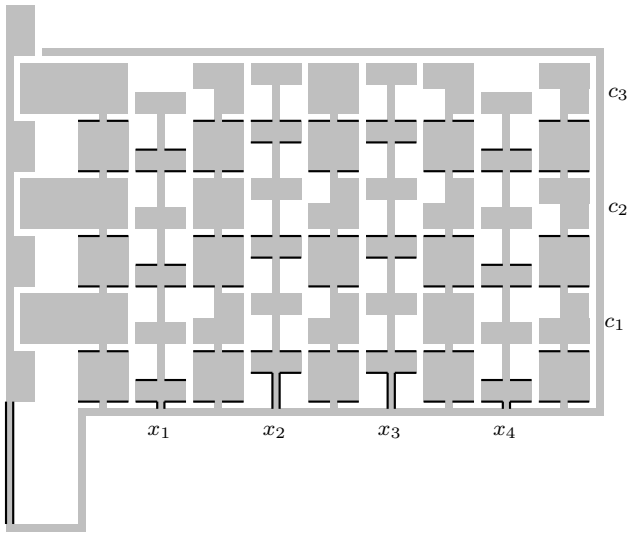

Figure 5.4: The columns of x_2 and x_3 have moved up, and those of x_1 and x_4 have moved down, corresponding to x_2 and x_3 FALSE, and, x_1 and x_4 TRUE.

and x_3 have moved upward, i.e. the variables are FALSE. The columns that correspond to x_1 and x_4 have moved downward, i.e. the variables are TRUE.

Let column C_i correspond to variable x and row R_j correspond to clause c. Suppose that C_i is in the down (TRUE) position and x is in c, then the block in position (C_i, R_j) can slide rightward part-way into the block in position (C_{i+1}, R_j). Likewise, if column C_i is in the up (FALSE) position and $\overline{x}$ is in c, the block in position (C_i, R_j) can slide rightward part-way into the block in position (C_{i+1}, R_j). In either case, it is possible for the block at position (C_i, R_j) to move rightward by at least unit distance. The blocks in row R_j to the left of C_i now have room to move rightward by this same amount. This is illustrated in Figure 5.5.

Recall that there is a unit overlap—with respect to the horizontal axis—between the clause comb and the blocks that protrude into the clause comb. Clearly, if ϕ is satisfiable then the variable columns of P can be morphed to correspond to a satisfying assignment. This satisfying assignment allows the clause comb to move downward. The movements of the blocks can then be reversed, returning the blocks to their original positions, resulting in Q (recall Figure 5.2).

5.2.3 When P and Q admit a morph.

Let M be a parallel morph from P to Q that respects the static edges, such that $M(0) = P$ and $M(1) = Q$. We prove that for some intermediate polygon of M the variable columns

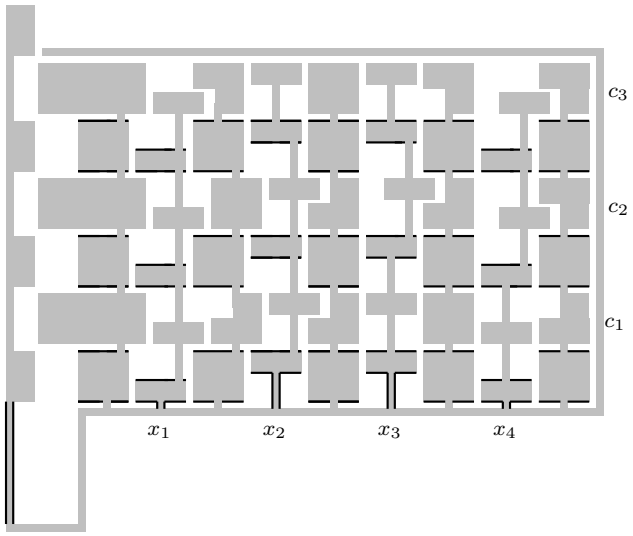

Figure 5.5: The assignment is satisfying. In each even row there is a space to push a block into a neighboring block, allowing all blocks to be withdrawn from the clause comb, which is now free to move.

correspond to a satisfying assignment of ϕ.

Lemma 5.2.1 *Let v_1, v_2 be two vertices of P such that no horizontal (respectively, vertical) elastic edges are encountered in at least one of the two possible traversals of the boundary of P from v_1 to v_2, i.e. going either clockwise or counterclockwise. Morph M must keep v_1 and v_2 a constant horizontal (respectively, vertical) distance apart.*

Proof: Trivial. □

The outer wall of the frame—that is, edges encountered in a clockwise traversal from v to w in Figure 5.1—contains neither horizontal nor vertical elastic edges and is therefore completely rigid. We find it convenient to fix the coordinates of v. In the remainder of this section, we describe the motions of components of our construction relative to the fixed coordinates of v.

For a given column of the block array, call the two vertices that connect the column to the frame the *anchor vertices*, e.g. in Figure 5.1, u is an anchor vertex of C_1. For each column there is both a left anchor vertex and a right anchor vertex. Likewise, the clause comb has left and right anchor vertices.

Define the *height* of a column as difference between the vertical coordinate of the highest vertex in the column, and the lowest vertical coordinate of a vertex of the drawing. Define the height of the clause comb likewise.

In every column, when traversing the boundary of P clockwise from the left anchor vertex to the right anchor vertex, no vertical elastic edge is encountered. Thus, by Lemma 5.2.1, as the height of a column changes the vertical coordinate of each vertex in the column changes by the same amount. Observe that a block in an even-numbered row can shrink with respect to the horizontal axis when the gap between the vertical edges connecting it to blocks above and below it collapses. Since every pair of vertices in a column remain a constant vertical distance from each other, no block in an even-numbered row may shrink with respect to the horizontal axis by more than ϵ.

To limit the movements of blocks in the block array we make the frame very thin: construct the frame so that every point on the inner wall of the frame is within distance ϵ of a point on the outer wall. In regions of (unit) horizontal overlap between a block and the clause comb, build the clause comb so that there is a vertical gap of size ϵ between that block and the clause comb. Finally, locate v and w (Figure 5.1) so that the horizontal coordinate of v is $1 + 3\epsilon$ to the right of the horizontal coordinate of w.

Lemma 5.2.2 *Morph M must have some intermediate drawing $M(t)$, $t \in [0, 1]$ in which all blocks are simultaneously in a retracted state.*

Proof: (By contradiction.) Assume that throughout M at least one block of C_1 protrudes into the clause comb at all times. To morph from $M(0)$ to $M(1)$ the height of the clause comb must decrease by 8 units. However, while a block of C_1 protrudes into the clause comb, the height of the clause comb cannot change by more than ϵ relative to the height of C_1—recall the ϵ-sized vertical gap between the clause comb and blocks protruding into the clause comb.

Let u be the left anchor vertex of C_1; see Figure 5.1. There are no horizontal elastic edges between u and v in a counterclockwise traversal from u to v. Thus, by Lemma 5.2.1, u cannot move horizontally. Notice, u cannot move downward more than distance ϵ without intersecting the edge that bounds the frame below. Therefore, the height of C_1 does not decrease by more than ϵ. Thus, while some block of C_1 protrudes into the clause comb, the height of the clause comb does not decrease by more than $2\epsilon < 8$. Contradiction. □

Let $t_0 \in [0, 1]$ be the instant at which, in morph M, all blocks are simultaneously retracted from the clause comb for the first time. Below, we prove that the heights of the columns in $M(t_0)$ correspond to a satisfying assignment of ϕ.

Lemma 5.2.3 *For all $t \in [0, t_0]$, each odd-numbered column in $M(t)$ is within ϵ of its original height and each even-numbered column is within $1 + 2\epsilon$ of its original height.*

Proof: Above, we proved that throughout M the height of C_1 does not decrease by more than ϵ. The remaining columns have anchor vertices that reside between the anchor vertices of C_1 and the right wall of the frame. Clearly, these anchor vertices also cannot move downward by more than ϵ. Hence, the heights of the odd-numbered columns cannot decrease by more than ϵ. For even-numbered columns, the vertical elastic edges allow the height of the columns to be reduced by up to an additional $1 + \epsilon$. Therefore, no even-numbered column can decrease height by more than $1 + 2\epsilon$.

It remains to prove that columns are bounded from above. In P, the height of each column is less than the vertical coordinate of v. We will prove that this must hold also in M up to time t_0. Observe, the only way that a column may rise above v is through the gap between v and the portion of the clause comb to the left of v. However, until the clause comb descends the size of this gap does not exceed $1+3\epsilon$, i.e. the horizontal distance between u and w. Throughout M blocks in the even-numbered rows (including the highest row R_{2m}) are at least of dimensions $(4 - 2\epsilon) \times (4 - \epsilon)$, and so will not fit through the gap. We conclude that for all $M(t)$, $t \in [0, t_0]$ no column is higher than v; upward movement of odd-numbered columns is bounded by ϵ, and $1 + \epsilon$ for even-numbered columns. □

Consider the two blocks that reside in the odd-numbered columns C_{i-1} and C_{i+1} and an even-numbered row R_j. These two blocks are of type (a), (c) or (d); recall Figure 5.3. In every intermediate drawing of M up to time t_0, each vertex of these blocks has a vertical coordinate that is within ϵ of its vertical coordinate in P. From the shape of the blocks, it is not difficult to see that the block at (C_{i+1}, R_j) lies strictly to the right of the block at (C_{i-1}, R_j) until time t_0. Between these two, the block at (C_i, R_j)—of type (b)—lies strictly to the right of (C_{i-1}, R_j) until time t_0. However, it may have horizontal overlap with the block at (C_{i+1}, R_j).

Let x be the literal that corresponds to C_i and let c be the clause that corresponds to R_j. If the block at (C_{i+1}, R_j) is of type (a) then there can be no horizontal overlap with the block at (C_i, R_j). Suppose there is a horizontal overlap between these two blocks. If the block at (C_{i+1}, R_j) is of type (b) then the height of C_i must have increased with respect to its position in P—setting x to FALSE—and if it is of type (d) the height of C_i must have decreased—setting x to TRUE.

Let $R_{j_1}, \ldots, R_{j_k}$ denote the even-numbered rows in which a block of C_i overlaps with a block of C_{i+1} in $M(t_0)$. There must be a single block type—either (c) or (d)—in all positions (C_{i+1}, R_j), where $j = j_1, \ldots, j_k$. Hence by our construction, $M(t_0)$ cannot have overlaps between blocks of C_i—corresponding to x—and blocks in both: (1) rows that correspond to clauses containing x, and (2) rows that correspond to clauses containing $\overline{x}$.

Lemma 5.2.4 *In $M(t_0)$ each even-numbered row of the block array must contain at least two blocks that overlap with respect to the horizontal axis.*

Proof: (By contradiction.) Assume that in $M(t_0)$ no two blocks of R_j overlap with respect to the horizontal axis (where j is even). To morph from P to $M(t_0)$ the leftmost vertical edge e of the block at (C_1, R_j) must move rightward by at least unit distance. Recall, no block in an even-numbered row can shrink with respect to the horizontal axis by more than ϵ. The horizontal gaps between adjacent blocks in R_j—and between the frame and the rightmost vertical edge of the rightmost block in R_j—may collapse, allowing e to move rightward, but only by an additional ϵ per gap. After summing, the rightward movement of e is bounded by $(4n+2)\epsilon = (4n+2)(4n+100)^{-1} < 1$. Contradiction. □

Suppose that in $M(t_0)$ the block at (C_i, R_j) overlaps (horizontally) with the block at (C_{i+1}, R_j), where i and j are even. This situation corresponds with an assignment to the literal x—that corresponds to C_i—in such a way that the clause c—that corresponds with R_j—is satisfied. By Lemma 5.2.4 every even-numbered row contains some such overlap, hence all clauses are satisfied. We have already shown that the assignment of literals in $M(t_0)$ is self-consistent. Therefore, $M(t_0)$ encodes a satisfying assignment of ϕ.

We conclude that ϕ is satisfiable if and only if there exists a morph from P to Q that respects the static edges. To complete the proof of Theorem 5.2.1 it suffices to show that P and Q can be computed in time that is polynomial with respect to the complexity of ϕ. Each of P and Q consists of $O(n)$ vertices and can be contained within a bounding square of size length $O(n)$. The coordinates of each vertex lie on a grid of squares of side length $\epsilon = (4n+100)^{-1}$. Therefore, any reasonable representation of the polygons requires only $O(n \log n)$ space. Clearly, the polygons can be generated in a straightforward manner from ϕ in time that is polynomial in n.

5.3 Monotone morphs of orthogonal polygons

In this section, we continue the discussion of monotone morphing that we began in Chapter 4. We prove the following theorem.

Theorem 5.3.1 *Given a pair of parallel simple orthogonal polygons in the plane, it is NP-hard to decide whether they will admit a monotone morph.*

To prove this theorem, we again perform a reduction from Boolean satisfiability. While it might be possible to instead use a reduction from the problem of morphing with static

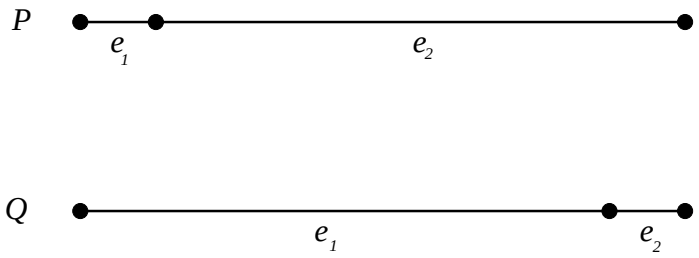

Figure 5.6: Using two edges to simulate an elastic edge.

edges, we find it easier to prove by making a minor modification to the construction devised in the proof of Theorem 5.2.1. The polygons in this construction have the useful property that, if they admit a parallel morph that respects the static edges, then they admit such a morph in which each elastic edge changes length in two monotone stages, either growing and then shrinking, or vice versa. This behavior is simulated in a monotone morph with a pair of collinear edges that are adjacent upon a common vertex.

Let P and Q be the parallel orthogonal polygons of graph G constructed in Section 5.2, and let $E_{\mathrm{e}} \subset \mathrm{E}(G)$ denote the set of elastic edges whose length is the same in both P and Q, i.e. those elastic edges $e \in \mathrm{E}(G)$ for which $||P(e)|| = ||Q(e)||$; recall, from Section 4.2, the notation $||P(e)||$ which denotes the length of edge e in drawing P (Definition 4.2.2, page 119).

The set E_{e} includes all elastic edges except for the two that connect the clause comb to the frame. For each edge $e \in E_{\mathrm{e}}$, partition e by a vertex into two parallel incident edges e_1, e_2, such that

$$||P(e_1)|| = ||Q(e_2)|| = \epsilon/2$$

and

$$||P(e_2)|| = ||Q(e_1)|| = ||P(e)|| - \epsilon/2.$$

See Figure 5.6. Note that this is the same ϵ as defined in Section 5.2. Call the resulting parallel orthogonal polygons P' and Q' and let G' denote the underlying graph of these drawings.

Observe,

$$||P(e)|| = ||Q(e)|| = ||P'(e_1)|| + ||P'(e_2)|| = ||Q'(e_1)|| + ||Q'(e_2)||.$$

Our aim is to prove that CNF expression ϕ is satisfiable if and only if P' and Q' admit a monotone morph. First, let us assume that ϕ is satisfiable.

Lemma 5.3.1 *If there exists a parallel morph between P and Q that respects static edges, then there exists a monotone morph from P' to Q'.*

Proof: In Section 5.2.2, we proved that if ϕ is satisfiable, then there exists a parallel morph from P to Q that respects the static edges. Furthermore, there exists such a morph with the following special form.

First, morph P so that odd-numbered columns have heights corresponding to a satisfying assignment of ϕ. In doing so, the vertical edges of length $1+\epsilon$ that connect columns to the frame either grow to $2+\epsilon$ or shrink to ϵ. Next, some subset of the blocks slide rightward distance $1+\epsilon$, thereby retracting all blocks from the clause comb. All horizontal elastic edges are initially length $2-\epsilon$; the horizontal motion can be done by shrinking some horizontal elastic edges by $1+\epsilon$ and increasing others by the same length. Finally, the clause comb moves down and the motion applied to the blocks is reversed. Call this morph M.

Clearly, there exists a parallel morph from P' to Q' in which all vertices common to G and G' follow the same paths as in M. Let M' be such a morph. All edges common to G and G' will change length monotonically in M', i.e. the elastic edges connected to the clause comb change length monotonically (by assumption) and the static edges do not change length at all, and are thus (trivially) monotonic. We need only to consider the edges of E_{e}.

Each elastic edge $e \in E_{\mathrm{e}}$ is simulated in P' and Q' by two edges e_1 and e_2. For every $t \in [0,1]$, $||M'(t;e_1)|| + ||M'(t;e_2)|| = ||M(t;e)||$. From above, we have that for all $e \in E_{\mathrm{e}}$, in M edge e will either: (1) decrease to some length $\geq \epsilon$ and then increase to length $||Q(e)||$, or else (2) increase to some length $\leq 2||P(e)|| - \epsilon$ and then decrease to length $||Q(e)||$. Either of these two behaviors can be simulated with e_1 and e_2 in M', even when we insist that e_1 be non-decreasing in length and e_2 be non-increasing, i.e. e_1 and e_2 change monotonically. Hence, there exists a monotone morph from P' to Q'. □

Now assume that ϕ is not satisfiable; hence, there is no parallel morph from P to Q that respects the static edges. We must prove that there is no monotone morph from P' to Q'. It suffices to show that if there exists a monotone morph between P' and Q' then there exists a parallel morph between P and Q that respects static edges.

Lemma 5.3.2 *If there exists a monotone morph from P' to Q' then there exists a parallel morph from P to Q that respects the static edges.*

Proof: Given a monotone morph M' from P' to Q' we can build a parallel morph M from P to Q in which every vertex that appears in both P' (and Q') and P (and Q) follows

the same path in M as it does in M'. By our construction, every static edge e of P (and Q) also appears in P' (and Q'), and $||P'(e)|| = ||Q'(e)||$. Since M' is a monotone morph, it must keep e constant in length. Hence M respects the static edges. □

We conclude that ϕ is satisfiable if and only if there exists a monotone morph between P' and Q'. By our construction, P' and Q' can be constructed in time that is polynomial in the complexity of ϕ. With that, we have proven Theorem 5.3.1.

5.4 Non-orthogonal drawings

Now let us consider general (i.e. not necessarily orthogonal) drawings in the plane. In this section, our aim is to prove the following theorem.

Theorem 5.4.1 *Given a pair of parallel simple drawings of a connected graph in the plane, it is NP-hard to decide whether the drawings admit a parallel morph.*

We reduce from the problem of deciding of morphing with static edges. Let P and Q be parallel simple orthogonal polygons. Let G denote the underlying cycle graph, such that each edge of G is labeled as either *static* or *elastic*. Let $E_{\mathrm{s}} \subseteq \mathrm{E}(G)$ denote the set of static edges. Hence, for each $e \in E_{\mathrm{s}}$, $||P(e)|| = ||Q(e)||$.

By Theorem 5.2.1, it is NP-hard to decide whether there exists a parallel morph from P to Q in which each edge of E_{s} remains of constant length.

Without loss of generality, assume that all vertices of P and Q have integer coordinates. Below, we describe a polynomial time procedure to generate two drawings: P' and Q' of a graph G'. We will subsequently prove that there exists a parallel morph from P to Q that respects the static edges if and only if there exists a parallel morph from P' to Q'.

Fix a small value $\epsilon = \frac{1}{8n}$, where $n = |\mathrm{V}(G)|$. For each $v \in \mathrm{V}(G)$, add to P' a $2\epsilon \times 2\epsilon$ square with a diagonal between its lower-left and upper-right corners, such that the four vertices of this square are located at the four points $P(v) + (\pm\epsilon, \pm\epsilon)$. We call this the *diagonalized square* of v. Observe that a diagonalized square permits only translation and scaling during a parallel morph.

For each edge $\{u, v\} \in \mathrm{E}(G)$, connect the diagonalized squares corresponding to u and v in P'; see Figure 5.7 (right). So, each elastic edge of P results in a pair of parallel axis-aligned edges in P'. A static edge results in these two parallel edges, plus a diagonal; see Figure 5.7 (left).

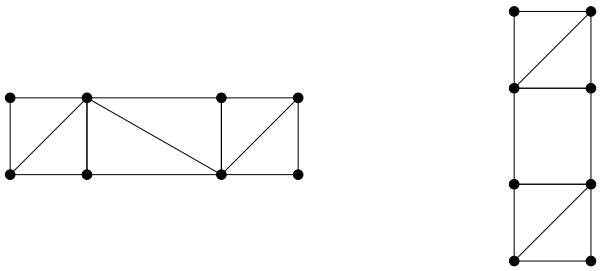

Figure 5.7: *Left:* representation of a (horizontal) static edge. *Right:* representation of a (vertical) elastic edge.

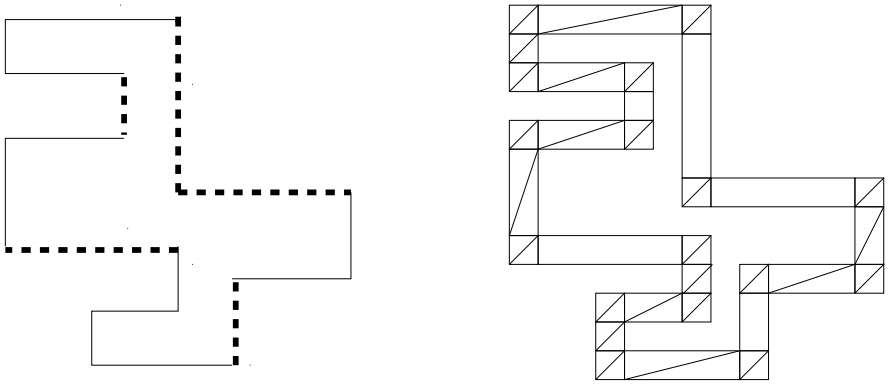

Figure 5.8: *Left:* an orthogonal polygon P; dashed edges are elastic and solid edges are static. *Right:* the drawing P' that corresponds with P.

This completes drawing P'. An example of our construction is illustrated in Figure 5.8. We have chosen ϵ sufficiently small that P' is simple. A parallel drawing Q' is constructed from Q using an analogous procedure.

The number of vertices in $\mathrm{V}(G')$ is linear in n. It is clear that P' and Q' can be computed in time that is polynomial in the complexity of P and Q.

Lemma 5.4.1 *If there exists a parallel morph from P' to Q', then there exists a parallel morph from P to Q that respects the static edges.*

Proof: Let M' denote a parallel morph from P' to Q'. Observe that the representation of a a static edge of G (including the diagonalized squares) permits only translation and scaling in M'. The representation of an elastic edge permits the two corresponding parallel edges to change in length. Notice that scaling a single diagonalized square requires that all others are scaled by the same factor. Therefore, we may assume—by scaling if necessary—that throughout M', each diagonalized square has side lengths equal to 2ϵ.

Let M be the morph from P to Q in which, for every $t \in [0,1]$, each $v \in \mathrm{V}(G)$ resides in the center of the corresponding diagonalized square of $M'(t)$. Morph M is a parallel morph from P to Q that keeps each edge in E_{s} constant in length. □

Lemma 5.4.2 *If there exists a parallel morph from P to Q that respects the static edges, then there exists a parallel morph from P' to Q'.*

Proof: In Lemma 5.4.4 (below) we prove that if P and Q—which have integer coordinates—admit a parallel morph that respects the static edges, then there exists a parallel morph between these drawings that respects the static edges and maintains a minimum feature size of at least $\frac{1}{\sqrt{2}n}$. Let M denote such a parallel morph.

Let us construct a morph M', from P' to Q'. For all $v \in \mathrm{V}(G)$ and $t \in [0,1]$, let the four vertices of G' that belong to the diagonalized square corresponding $v \in \mathrm{V}(V)$ have their coordinates at the four points $M(t) + (\pm\epsilon, \pm\epsilon)$. Clearly, M' keeps edges parallel. We must also prove that M' maintains simplicity.

Suppose that M' does not maintain simplicity. Thus, there exists some $t \in [0,1]$ such that the minimum feature size of $M'(t)$ is zero. For every element (i.e. vertex or edge) of $M'(t)$ there exists an element of $M(t)$ that is within at most L_2 (rectilinear) distance of $\sqrt{2}\epsilon$. Hence, if the minimum feature size of $M'(t)$ is zero, then the minimum feature size of $M(t)$ is no more than $2(\sqrt{2}\epsilon) = \frac{\sqrt{2}}{4n}$. This cannot be, since we have shown above that the minimum feature size of M is no less than $\frac{1}{\sqrt{2}n} = \frac{2\sqrt{2}}{4n}$. Therefore, M' maintains simplicity and is thus a parallel morph. □

5.4.1 Additional lemmas

It remains to bound the minimum feature size that can be maintained in a parallel morph with static edges. Such a bound is given in Lemma 5.4.4. First, we introduce a lemma to be used in the proof of Lemma 5.4.4. Although this lemma is required only in the plane, we present the lemma in the context of $\mathbb{R}^d$ since the result itself seems sufficiently interesting and increases the complexity of the proof only slightly.

Lemma 5.4.3 *Let P be a simple orthogonal drawing in $\mathbb{R}^d$ of a graph G, and let $E_s \subseteq E(G)$, such that every edge in $e \in E_s$ has integer length. There exists a simple orthogonal drawing Q of G, such that:*

1. *P and Q are parallel, and have the same vertex ordering,*

2. for all $e \in E_s$, $||P(e)|| = ||Q(e)||$, and,

3. Q maps each vertex to a uniform grid in $\mathbb{R}^d$ with side length $1/n$.

Proof: For each dimension $i \in \{1, \ldots, d\}$ and vertex $v \in \mathrm{V}(G)$, let $\sigma_i(v)$ denote the fractional part of $P_i(v)$; i.e.,

$$\sigma_i(v) = P_i(v) - \lfloor P_i(v) \rfloor. \tag{5.1}$$

Let f_i be a function that maps each value in the set $\{\sigma_i(v) : v \in \mathrm{V}(G)\}$ to an integer in the set $\{0, \ldots, n-1\}$ such that if $\sigma_i(u) < \sigma_i(v)$ then $f_i(\sigma_i(u)) < f_i(\sigma_i(v))$. For each dimension $i \in \{1, \ldots, d\}$ there are at most n unique values of $\sigma_i(v)$. Therefore function f_i exists.

Construct drawing Q such that for all $v \in \mathrm{V}(G)$,

$$Q(v) = \left(\left(\lfloor P_1(v) \rfloor + \frac{f_1(\sigma_1(v))}{n} \right), \ldots, \left(\lfloor P_d(v) \rfloor + \frac{f_d(\sigma_d(v))}{n} \right) \right). \tag{5.2}$$

All vertices in Q are aligned with a d-dimensional grid composed of hypercubes of side length $1/n$. For each dimension $i \in \{1, \ldots, d\}$ the relative ordering of the vertices with respect to dimension i is the same in Q and P. Therefore, by Observations 2.2.1 and 2.2.2 (pages 29 and 30) the two drawings are parallel and simple. It remains to prove that all edges in E_s have the same length in both drawings.

Let $\{u, v\} \in E_\mathrm{s}$ be an arbitrary static edge, and let $i \in \{1, \ldots, d\}$ be the unique dimension for which $|P_i(u) - P_i(v)| \neq 0$. The length of $\{u, v\}$ in P is

$$z = |P_i(u) - P_i(v)|$$

where z is a positive integer. Since $\sigma_i(u) = \sigma_i(v)$,

$$f_i(\sigma_i(u)) = f_i(\sigma_i(v)).$$

Therefore,

$$|Q_i(u) - Q_i(v)| = \left| \left(\lfloor P_i(u) \rfloor + \frac{f_i(\sigma_i(u))}{n} \right) - \left(\lfloor P_i(v) \rfloor + \frac{f_i(\sigma_i(v))}{n} \right) \right| = z, \tag{5.3}$$

i.e., all edges in E_s have the same length in both P and Q. □

Lemma 5.4.4 *Let P and Q be parallel simple orthogonal drawings of a graph G in the plane, such that all vertices have integer coordinates. Let $E_s \subseteq E$ where P and Q admit a parallel morph M that keeps edges in E_s static. Then, P and Q admit such a morph that maintains a minimum feature size of at least $\frac{1}{n\sqrt{2}}$.*

Proof: Without loss of generality, assume that M exhibits a finite sequence of vertex orderings. This may be assumed since the number of possible distinct vertex orderings is finite, and subsequences that begin and end with the same ordering can be replaced by a single ordering; recall Claim 2.2.2 of Theorem 2.2.1 (page 36). It is safe to replace subsequences in the case of static edges since by Lemma 2.1.1 (page 27) linear morphs between parallel drawings change edge lengths monotonically.

Extract from M the finite sequence of times

$$0 = t_1 < t_2 \ldots < t_k = 1$$

such that for all $i \in \{1, \ldots, k-1\}$,

1. $M(t_i)$ and $M(t_{i+1})$ have consonant vertex orderings, and
2. $M(t_i)$ and $M(t_i + \epsilon)$ have the same vertex ordering, where $0 \leq \epsilon < 1$.

By Lemma 5.4.3, for every $i \in \{1, \ldots, k-1\}$ there exists a drawing $M'(t_i)$ with the same vertex ordering as $M(t_i)$, such that every vertex of $M'(t_i)$ has coordinates that are multiples of $1/n$. Hence, $M'(t_i)$ is simple and parallel with $M(t_i)$, and the minimum feature size of $M'(t_i)$ is $1/n$. Drawings $M'(t_i)$ and $M'(t_{i+1})$ are parallel simple orthogonal drawings with consonant vertex orderings. By Lemma 2.2.3 (page 32), the linear morph from $M'(t_i)$ to $M'(t_{i+1})$ is a parallel morph that maintains a minimum feature size of $\frac{1}{n\sqrt{2}}$. A composition of such linear morphs results in the desired parallel morph from P to Q. □

5.5 Conclusion

In this chapter, it was shown to be NP-hard to decide whether a given pair of parallel simple orthogonal polygons admits a parallel morph such that a given subset of edges remain constant in length throughout the morph. Following this, it was shown to be NP-hard to decide whether a pair of parallel simple orthogonal polygons admit a monotone morph. The construction is a slight modification of the construction used to establish the NP-hardness the first problem. Finally, we proved that parallel morphing in the plane is NP-hard for non-orthogonal drawings, using a reduction from the first problem.

If disconnected graphs are allowed, then parallel morphing between orthogonal drawings with static edges can easily be shown to be PSPACE-hard via a reduction from the *warehouseman's problem* [40]: given two configurations of a finite set of pairwise-disjoint axis-aligned rectangles in the plane such that all rectangles are contained within one larger

rectangle, can we get from one configuration to the other by translating rectangles in a continuous manner while keeping the rectangles pairwise disjoint? Clearly, monotone morphing is also PSPACE-hard in the case of disconnected graphs. The only surprise is that parallel morphing with static edges and monotone morphing are NP-hard when the input drawings are orthogonal polygons. We leave it open to determine whether the problems discussed in this chapter are PSPACE-hard. Also open is whether any of these problems are in NP or PSPACE.

The results of this chapter can be strengthen somewhat by further restricting the class of graphs for which NP-hardness is preserved. It is not difficult to show that both morphing with static edges and monotone morphing remain NP-hard even for orthogonal drawings of a path: notice that the proofs carry through when the topmost horizontal edge of the clause comb is removed in the constructions. The implication is that deciding parallel morphability for non-orthogonal drawings is NP-hard even when the input drawings are outerplanar.

Deciding parallel morphability for non-orthogonal drawings is NP-hard even for graphs of degree six, as is evident from our proof. It would be interesting to try to prove NP-hardness for this problem for graphs of smaller degree.

Chapter 6

Trees, Cycles, and Polyhedra

6.1 Introduction

In this chapter, we turn our attention to parallel morphs of orthogonal and non-orthogonal drawings in higher dimensional spaces, as well as orthogonal polyhedra in $\mathbb{R}^3$. The techniques developed in Chapter 3 for morphing orthogonal drawings in the plane do not apply generally in higher dimensions. For instance, the $\mathbb{R}^3$ analogue of the path-straightening operation is an is an operation that flattens an orthogonal surface. It turns out that not every such surface can be flattened, as we will see in Section 6.4. On the positive side, every pair of parallel simple (but not necessarily orthogonal) drawings of a tree graph in $\mathbb{R}^d$ admits a parallel morph, where d is any positive integer greater than or equal to two; see Section 6.2.3.

When the underlying graph is a cycle, parallel morphing in $\mathbb{R}^3$ seems much more difficult than when the underlying graph is a tree. The problem of deciding whether a parallel morph exists in this case bears a strong resemblance to the problem of deciding the equivalence of two knots—a problem whose computational complexity has not yet been fully resolved. If parallel simple drawings of a cycle correspond to different knots then there can be no simplicity-preserving morph between them, and hence no parallel morph. However, simply because parallel drawings of a cycle correspond to the same knot does not mean that a parallel morph exists. In Section 6.3 we examine a pair of parallel simple drawings in $\mathbb{R}^3$ of a cycle graph that does not admit a parallel morph, even though both drawings are topologically equivalent to the *trivial knot*, i.e. the knot that can be deformed to lie in a plane without self-intersection.

It is interesting to consider parallel morphing in the context of orthogonal polyhedra in $\mathbb{R}^3$. Surprisingly, even when restricting attention to those polyhedra that are orthogonal

and topologically equivalent to a sphere, not every parallel pair admits a parallel morph. Unmorphable such pairs are examined in Section 6.4.

6.2 Paths and Trees

In this section, we examine the problem of generating a parallel morph when the underlying graph is a tree. In Section 6.2.1 it is established that every pair of parallel simple orthogonal drawings in $\mathbb{R}^d$ admits a parallel morph when the underlying graph is a path, for any integer $d \geq 2$. In Section 6.2.2 this is extended to non-orthogonal drawings of paths. Section 6.2.3 generalizes even further, establishing that every pair of parallel simple drawings in $\mathbb{R}^2$ admits a parallel morph when the underlying graph is a tree.

6.2.1 Orthogonal drawings of paths

Throughout this section we employ the technique of defining a *canonical form* for each drawing in the class under consideration. If two drawings are parallel, then their canonical forms are parallel drawings that are identical up to scaling and translation. To prove that there exists a parallel morph between parallel drawings P and Q, it suffices to prove that there exists a parallel morph from each of P and Q to its canonical form: a parallel morph from P to Q may be accomplished by first morphing P to its canonical form, scaling and translating the result with a linear morph, and then performing the reverse of the morph from Q to its canonical form.

Theorem 6.2.1 *Every pair of parallel simple orthogonal drawings of a path graph in $\mathbb{R}^d$ admits a parallel morph.*

Proof: Let G be a path graph such that

$$\mathrm{V}(G) = \{v_1, \ldots, v_n\}$$

and

$$\mathrm{E}(G) = \{\{v_1, v_2\}, \ldots, \{v_{n-1}, v_n\}\}.$$

Let P be a simple orthogonal drawing of G in $\mathbb{R}^d$ with minimum feature size δ_P.

Define the *canonical form* of P as a drawing that is parallel with P, such that each edge $\{v_i, v_{i+1}\} \in \mathrm{E}(G)$ has length

$$\delta_P 2^{i-n-1}.$$

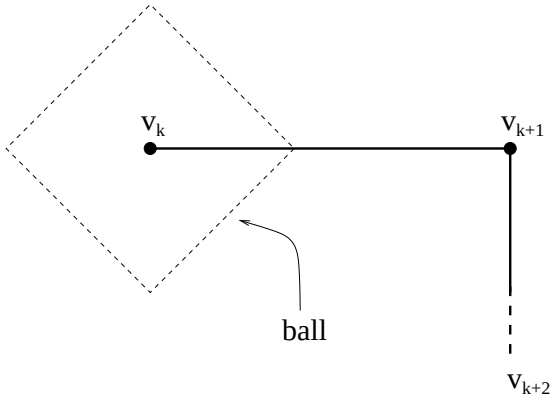

Figure 6.1: The d-dimensional ball, after shrinking edges $\{v_1, v_2\}, \ldots, \{v_{k-1}, v_k\}$.

Any pair of parallel drawings P and Q have identical canonical forms up to scaling and translation, where the magnitude of the scaling depends upon the ratio between δ_P and δ_Q.

To morph P to its canonical form, every edge must shrink by some amount. Morph P to its canonical form as follows: for all $i = 1$ to $n-1$, perform a linear morph that shrinks edge $\{v_i, v_{i+1}\}$ to its canonical length, while leaving all other edges at their original lengths. To shrink $\{v_i, v_{i+1}\}$, move $v_1, \ldots, v_i$ uniformly, while keeping $v_{i+1}, \ldots, v_n$ stationary. Vertices $v_1, \ldots, v_i$ move by a distance equal to the length of $\{v_i, v_{i+1}\}$ in P, minus the canonical length of the edge. The direction of movement is the same as the direction of the vector from $P(v_i)$ to $P(v_{i+1})$.

This procedure results in a drawing in the canonical form of P. Every intermediate drawing of the morph is parallel with P. It remains to prove that the morph maintains simplicity.

Claim 6.2.1 *Every intermediate drawing of the morph from P to its canonical form is simple.*

Proof: (By induction on the number of edges.) Trivially, while edge $\{v_1, v_2\}$ shrinks, the drawing remains simple. Assume that the morph maintains simplicity while edges $\{v_1, v_2\}, \ldots, \{v_{k-1}, v_k\}$ shrink.

After shrinking edge $\{v_{k-1}, v_k\}$, all vertices $v_1, \ldots, v_k$ lie entirely within a d-dimensional ball (see Figure 6.1) centered at v_k, whose L_1 (rectilinear) radius is at most the sum of the lengths of the edges $\{v_1, v_2\}$ to $\{v_{k-1}, v_k\}$, i.e.

$$\delta_P \sum_{i=1}^{k-1} 2^{i-n-1} = \delta_P 2^{-n-1}(2^k - 2) < \delta_P 2^{k-n-1} < \delta_P.$$

When edge $\{v_k, v_{k+1}\}$ shrinks, the vertices and edges inside the ball remain static with respect to each other. Likewise, the vertices and edges outside of the ball remain static with respect to each other. It suffices to establish that the vertices and edges inside the ball do not intersect those outside the ball.

The canonical length of $\{v_k, v_{k+1}\}$ is 2^{k-n-1}. Since this is greater than the radius of the ball, shrinking $\{v_k, v_{k+1}\}$ leaves the ball disjoint from v_{k+1} and from the edge $\{v_{k+1}, v_{k+2}\}$ (if it exists). No other vertex or edge outside the ball can enter the ball since they all have an L_1 distance of at least δ_P from $\{v_k, v_{k+1}\}$, which is greater than the radius of the ball. □

By the above claim, Theorem 6.2.1 is proved. □

The number of linear morphs used to get a drawing to its canonical form is $n-1$, where n is the number of vertices. Thus, in a parallel morph between two drawings of a path, a total of $2n-2$ linear morphs are used. We improve upon this bound when we discuss morphing between parallel drawings of a tree in Section 6.2.3.

6.2.2 Nonorthogonal drawings of paths

To extend Theorem 6.2.1 to drawings that are not necessarily orthogonal, we generalize the definition of the canonical form. As the reader might expect, the angles of the drawings to be morphed are crucial in this definition.

Theorem 6.2.2 *Every pair of parallel simple drawings of a path graph in $\mathbb{R}^d$ admits a parallel morph.*

Proof: Let P denote a (possibly non-orthogonal) drawing of a path graph G in $\mathbb{R}^d$. As in the proof of Theorem 6.2.1, let

$$\mathrm{V}(G) = \{v_1, \ldots, v_n\}$$

and

$$\mathrm{E}(G) = \{\{v_1, v_2\}, \ldots, \{v_{n-1}, v_n\}\}.$$

Let δ_P denote the minimum feature size of P, and let θ_P denote the size of the smallest angle between two incident edges of P. Define Γ as the value such that

$$\Gamma = \frac{2}{\sin(\min\{\pi/2, \theta_P\})}.$$

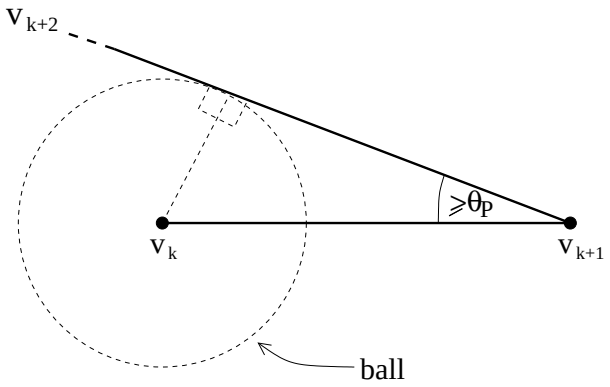

Figure 6.2: The d-dimensional ball after morphing the first $k-1$ edges.

Define the canonical form of P as a drawing that is parallel with P in which each edge $\{v_i, v_{i+1}\}$ has length

$$\delta_P \Gamma^{i-n-1}.$$

If a drawing Q is parallel with P, then these drawings share the same minimum angle, i.e. $\theta_P = \theta_Q$. Therefore, the canonical drawings of P and Q are identical up to scaling and translation. As in the orthogonal case, to morph P to its canonical form, shrink each edge in turn via a linear morph to its canonical length, beginning with $\{v_1, v_2\}$. It is clear that every intermediate drawing of the morph is parallel with P. To prove the theorem, it suffices to prove that the morph preserves simplicity.

Claim 6.2.2 *Every intermediate drawing of the morph from P to its canonical form is simple.*

Proof: (By induction on the number of edges.) Trivially, simplicity is preserved when $\{v_1, v_2\}$ shrinks. Assume that the morph preserves simplicity while shrinking edges $\{v_1, v_2\}$ to $\{v_{k-1}, v_k\}$. As illustrated in Figure 6.2, these edges reside within a d-dimensional ball centered at v_k whose L_2 (Euclidean) radius is at most

$$\delta_P \sum_{i=1}^{k-1} \Gamma^{i-n-1} = \delta_P \Gamma^{-n-1} \left(\sum_{i=1}^{k-1} \Gamma^i \right) < \delta_P \Gamma^{-n-1} \left(\frac{\Gamma^k - 1}{\Gamma - 1} \right). \tag{6.1}$$

Vertices $v_1, \ldots, v_k$ remain static with respect to one another while $\{v_k, v_{k+1}\}$ shrinks, as do vertices $v_{k+1}, \ldots, v_n$. The only possibility for self-intersection while $\{v_k, v_{k+1}\}$ shrinks is that some element inside the ball intersects some element outside the ball. Since the radius of the ball is smaller than δ_P, the ball cannot intersect any element outside of the

ball that is not incident upon v_{k+1}. It must be shown that the ball does not intersect either vertex v_{k+1} or edge $\{v_{k+1}, v_{k+2}\}$ (assuming that this edge exists).

Clearly, v_k is closest to v_{k+1} when $\{v_k, v_{k+1}\}$ is at its canonical length. However, the radius of the ball centered at v_k given in (6.1) is strictly smaller than the canonical length of $\{v_k, v_{k+1}\}$, which equals $\delta_P \Gamma^{k-n-1}$. Therefore, the ball centered at v_k does not intersect v_{k+1}. It remains only to prove that the ball centered at v_k does not intersect $\{v_{k+1}, v_{k+2}\}$.

Vertex v_k is closest to edge $\{v_{k+1}, v_{k+2}\}$ when $\{v_k, v_{k+1}\}$ is at its canonical length. If $\theta_P \geq \pi/2$, then when $\{v_k, v_{k+1}\}$ is at its canonical length, v_k is closer to v_{k+1} than to any point of $\{v_{k+1}, v_{k+2}\}$. It has already been shown that the ball centered at v_k does not intersect v_{k+1}. So, without loss of generality assume that $\theta_P < \pi/2$.

When $\{v_k, v_{k+1}\}$ is at its canonical length, the distance from v_k to $\{v_{k+1}, v_{k+2}\}$ is

$$\geq \sin(\theta_P)\delta_P \Gamma^{k-n-1}.$$

See Figure 6.2. To prove that the radius of the ball is strictly smaller than the minimum distance between v_k and $\{v_{k+1}, v_{k+2}\}$, it suffices to prove that

$$\sin(\theta_P)\delta_P \Gamma^{k-n-1} > \delta_P \Gamma^{-n-1} \left(\frac{\Gamma^k - 1}{\Gamma - 1} \right). \tag{6.2}$$

For a proof-by-contradiction, assume that

$$\begin{aligned} \sin(\theta_P)\delta_P \Gamma^{k-n-1} &\leq \delta_P \Gamma^{-n-1} \left(\frac{\Gamma^k - 1}{\Gamma - 1} \right) \\ \implies \sin(\theta_P)\Gamma^k(\Gamma - 1) &\leq \Gamma^k - 1 \\ \implies \sin(\theta_P) \left(\frac{2}{\sin(\theta_P)} - 1 \right) &< 1 \\ \implies 1 &< \sin(\theta_P). \end{aligned}$$

Therefore, by contradiction (6.2) holds. □

With that, Theorem 6.2.2 is proved. □

As in the case of orthogonal drawings of paths, the morphs as described above are composed of $2n - 2$ linear morphs.

6.2.3 Trees

Let us apply what we have learned about morphing drawings of a path to develop an algorithm for morphing drawings of a tree. The number of linear morphs in a morph generated by this algorithm depends on the *diameter* of the tree, i.e. the number of edges in the longest path through the tree.

Theorem 6.2.3 *Every pair of parallel simple drawings of a tree in $\mathbb{R}^d$ admits a parallel morph comprising $D-1$ linear morphs, where D is the diameter of the tree. The morph can be computed in $O(nD)$ time and $O(n)$ space, for fixed dimension d, where n is the number of vertices.*

Proof: Let G be a tree graph of diameter D. Let P be a drawing of G with minimum feature size δ_P and minimum angle θ_P. Designate some vertex to be the root of G, such that the longest path between the root and any other vertex of G contains at most $\lceil D/2 \rceil$ edges.

Claim 6.2.3 *A suitable root can be determined in $O(n)$ time.*

Proof: First, compute a path π of maximum length through G. If π contains an even number of edges, set the root to be the vertex of π midway between the end-vertices of π. Otherwise, choose one of the two central two vertices π to be the root. In either case, let v_{r} denote this choice of root. The distance between v_{r} and every other vertex of π is at most $\lceil D/2 \rceil$, where the distance between two vertices is the number of edges in the shortest path between the vertices. Suppose that the distance between v_{r} and some other vertex v_0 not of π is strictly greater than $\lceil D/2 \rceil$. Then, it cannot be that π is a longest path through G, since a longer path exists containing v_0. It follows that v_r is a suitable root vertex, and can be determined in $O(n)$ time from a maximum-length path of G.

To compute a maximum-length path in $O(n)$ time, choose some vertex $v_1 \in \mathrm{V(G)}$ arbitrarily. The choice of v_1 determines a parent-child relationship for every pair of adjacent vertices in V(G). For each $v \in \mathrm{V(G)}$, compute the length of the longest path $l(v)$ from v to a leaf of the subtree rooted at v. This can be done in $O(n)$ time with a depth-first search.

Observe that for every maximum-length path π in G, there exists a unique vertex v such that π contains v, and π is contained within the subtree rooted at v. For each vertex v with children $v_1, \ldots, v_k$, the length of the longest path that contains v and is contained within the subtree rooted at v is equal to

$$\max_{i,j \in \{1,\ldots,k\}, i \neq j} l(v_i) + l(v_j) + 2$$

where v has at least two children; there are similar expressions for vertices with either one or zero children. The length of a maximum-length path can be computed in $O(n)$ time by evaluating the above expression over all vertices in V(G). Once a vertex is determined for which this expression is maximized, a maximum-length path can be reconstructed in an additional $O(n)$ time. □

For all $v \in \mathrm{V}(G)$, define $d(v)$ as the *distance* from v to the root, i.e. the number of edges in the path from v to the root. For the root itself, d is zero. Define Δ as

$$\Delta = \frac{2}{\sin(\theta_P/2)}.$$

Define the canonical form of P as the drawing that is parallel with P in which each edge $\{u, v\} \in \mathrm{E}(G)$ has length

$$\delta_P \Delta^{-d(u)}$$

where $d(u) > d(v)$. Let $h = \lceil D/2 \rceil$ denote the *height* of the rooted tree, i.e. the maximum distance between the root and a leaf.

Morph P to its canonical form as follows: for $i = 1$ to $h-1$, perform the linear morph in which all vertices of distance $h-i$ or less from the root remain stationary, while all vertices of distance at least $h-i+1$ move linearly: for a vertex w that resides in a subtree rooted at u such that $d(u) = h-i+1$, w moves uniformly in the direction of the vector from $P(u)$ to $P(v)$, where v is the parent of u. The distance moved by w equals the difference between the length of $P(u, v)$ and the canonical length of $\{u, v\}$. This linear morph simultaneously shrinks all edges $\{u, v\}$ for which $d(u) = h - i + 1$ and $d(v) = h - i$ to canonical length.

All intermediate drawings of the morph are parallel. Following this sequence of linear morphs, each edge has its canonical length.

Claim 6.2.4 *Every intermediate drawing of the morph from P to its canonical form is simple.*

Proof: (By induction on the number of linear morphs.) It is trivial that simplicity is preserved during the first iteration. Assume that for all iterations $1, \ldots, k-1$ simplicity is preserved. It remains to prove that simplicity is also preserved during the k-th iteration.

Throughout the linear morph of the k-th iteration, all vertices whose distance from the root is strictly greater than k are contained in d-dimensional balls centered at one of the vertices of distance k from the root, and of L_2 (Euclidean) radius

$$\delta_P \sum_{i=1}^{k-1} \Delta^{i-h-1} = \delta_P \Delta^{-h-1} \sum_{i=1}^{k-1} \Delta^i < \delta_P \Delta^{-h-1} \left(\frac{\Delta^k - 1}{\Delta - 1} \right).$$

The radius of each ball is strictly smaller than the canonical length of an edge between vertices of distance k and $k+1$ from the root, $\delta_P \Delta^{k-h-1}$. Applying the same arguments used in the proof of Theorem 6.2.2, it can be established that for every intermediate drawing of the k-th linear morph none of the balls intersects any vertex u or edge $\{u, v\}$ for which $d(u) \leq h - k$. It remains only to prove that no two balls intersect.

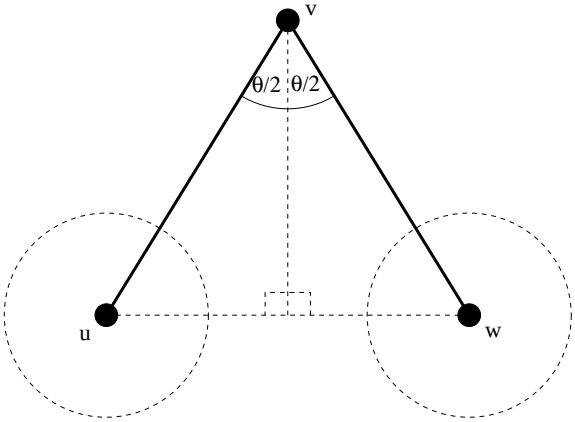

Figure 6.3: Illustrating the minimum distance between u, w when both are connected to a common vertex, v. The edges $\{u, v\}$ and $\{v, w\}$ have their canonical length of Δ^{k-h-1} in this figure.

Let $u, w \in \mathrm{V}(G)$ where $d(u) = d(w) = h - k + 1$. If there exists a vertex $v \in \mathrm{V}(G)$ such that $\{u, v\}, \{v, w\} \in \mathrm{E}(G)$ then $d(v) = h - k$. As $\{u, v\}$ and $\{v, w\}$ shrink during the k-th linear morph to a length of $\delta_P \Delta^{k-h-1}$, u and w remain separated by a Euclidean distance of at least

$$2\delta_P \sin\left(\frac{\theta_P}{2}\right) \Delta^{k-h-1} < \delta_P$$

See Figure 6.3. If v does not exist, u and w remain at least distance δ_P apart.

To prove that the d-dimensional balls around u and w remain non-intersecting throughout the morph, it suffices to prove that

$$2\delta_P \Delta^{-h-1} \left(\frac{\Delta^k - 1}{\Delta - 1}\right) < 2\delta_P \sin\left(\frac{\theta_P}{2}\right) \Delta^{k-h-1}. \tag{6.3}$$

To prove (6.3) by contradiction, assume that

$$\begin{aligned} 2\delta_P \Delta^{-h-1} \left(\frac{\Delta^k - 1}{\Delta - 1}\right) &\geq 2\delta_P \sin\left(\frac{\theta_P}{2}\right) \Delta^{k-h-1} \\ \implies \left(\frac{\Delta^k - 1}{\Delta - 1}\right) &\geq \sin\left(\frac{\theta_P}{2}\right) \Delta^k \\ \implies 1 &> \sin\left(\frac{\theta_P}{2}\right) (\Delta - 1) \\ \implies 1 &> 2 - \sin\left(\frac{\theta_P}{2}\right). \end{aligned}$$

By contradiction, (6.3) holds. □

Using the approach described, a parallel morph from a drawing to its canonical form requires no more than $h = \lceil D/2 \rceil$ linear morphs. Therefore, a parallel morph between two drawings can be done using

$$2h = 2\lceil D/2 \rceil \leq D - 1$$

linear morphs. In the online model, the terminal drawings of these linear morphs can be generated using $O(nD)$ time and $O(n)$ space. □

Theorem 6.2.3 implies that $n - 2$ linear morphs suffice for a parallel morph between parallel drawings of a path.

Morphs produced by the algorithms described in this section all suffer from having intermediate drawings in which the longest edge is exponentially longer than the shortest edge. This can be remedied in the case of orthogonal drawings of trees, using a technique similar to the one introduced in the proof of Corollary 3.6.1 of Chapter 3 (page 112), i.e. defining a sequence of linear morphs such that all vertices in terminal drawings of the linear morphs lie on a grid.

6.3 Cycles

Every pair of parallel simple drawings of a cycle in the plane (i.e. parallel polygons) will admit a parallel morph [34, 33]. However, as soon as we move out of the plane and into three-dimensional space we must contend with issues of "knottedness".

A *knot* is defined as an embedding of a circle into $\mathbb{R}^3$, i.e. a non-self-intersecting curve in $\mathbb{R}^3$. Two knots are equivalent if one can be continuously deformed to the other. A knot is *trivial* if it can be continuously deformed to lie in a plane without self-intersection. See e.g. [50] for an introduction to knot theory.

A simple drawing of a cycle in $\mathbb{R}^3$ represents a knot. If two drawings of a cycle represent two different knots, then there does not exist a simplicity-preserving morph (parallel or otherwise) between the drawings. The computational complexity of deciding whether two drawings of cycles in $\mathbb{R}^3$ correspond to the same knot has not been completely determined [36].

However, simply because two parallel drawings correspond to the same knot does not imply that they will admit a parallel morph.

Theorem 6.3.1 *There exist pairs of parallel drawings of a cycle that correspond to the trivial knot, yet do not admit a parallel morph.*

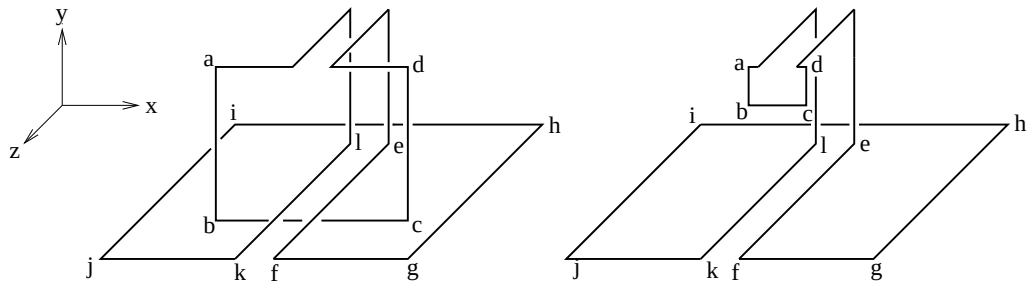

Figure 6.4: Parallel orthogonal drawings of a cycle that do not admit a parallel morph.

This theorem is established by the following lemma. This lemma will be useful to us later in this chapter, when we prove the existence of unmorphable pairs of parallel orthogonal polyhedra.

Lemma 6.3.1 *The parallel drawings of a cycle illustrated in Figure 6.4 do not admit a parallel morph.*

Proof: Let P denote the drawing illustrated on the left of Figure 6.4, and let Q denote the drawing on the right.

Notice that vertices labeled $e, \ldots, l$ must lie in a common x-z plane in any drawing that is parallel with P and Q. Throughout this proof, we use notation of the form $P_{\mathrm{y}}(e, \ldots, l)$, for example, to denote the common y-coordinate of vertices $e, \ldots, l$ in P. We use this notation to emphasize that all of these vertices have the same y-coordinate in any drawing that is parallel with P.

For a contradiction, assume that there exists a parallel morph M from P to Q. Extend the notation defined above such that, e.g.

$$M_{\mathrm{y}}(t; e, \ldots, l)$$

refers to the y-coordinate of the vertices $e, \ldots, l$ in drawing $M(t)$.

Observe that the following inequalities will hold for all $t \in [0, 1]$:

$$\max\{M_{\mathrm{z}}(t; e), M_{\mathrm{z}}(t; l)\} < M_{\mathrm{z}}(t; a, b, c, d) \tag{6.4}$$

and

$$M_{\mathrm{y}}(t; e, \ldots, l) < \min\{M_{\mathrm{y}}(t; a), M_{\mathrm{y}}(t; d)\} \tag{6.5}$$

Since $P_{\mathrm{y}}(b, c) < P_{\mathrm{y}}(e, \ldots, l)$ and $Q_{\mathrm{y}}(b, c) > Q_{\mathrm{y}}(e, \ldots, l)$, there must exist some $t \in [0, 1]$ such that

$$M_{\mathrm{y}}(t; b, c) = M_{\mathrm{y}}(t; e, \ldots, l). \tag{6.6}$$

Let $t_0 \in [0,1]$ denote the smallest value t for which (6.6) holds.

At time t_0, vertices b, c and $e, \ldots, l$ all lie in a common x-z plane, i.e. a plane that is perpendicular to the y-axis. Strictly before these vertices can be made coplanar, edge $\{b,c\}$ must be moved to a position that does not overlap with edges $\{e,f\}$ and $\{k,l\}$ with respect to the x- and z-axes. We will prove that this cannot happen in drawing $M(t)$, for any $t < t_0$. We may then conclude that there exists no parallel morph from P to Q.

Let us restrict our attention to drawings $M(t)$ where $t < t_0$. By definition, when $t < t_0$

$$M_{\mathrm{y}}(t; b, c) < M_{\mathrm{y}}(t; e, \ldots, l).$$

Hence, by (6.5), for all $t < t_0$,

$$M_{\mathrm{y}}(t; b, c) < \min\{M_{\mathrm{y}}(t; a), M_{\mathrm{y}}(t; d)\}.$$

Therefore, for all $t < t_0$, edge $\{a,b\}$ in drawing $M(t)$ intersects the x-z plane through $M_{\mathrm{y}}(t; e, \ldots, l)$ at some point $\alpha^t = (\alpha^t_{\mathrm{x}}, \alpha^t_{\mathrm{y}}, \alpha^t_{\mathrm{z}})$. Observe that $\alpha^t_{\mathrm{z}} = M_{\mathrm{z}}(t; a, b, c, d)$. Thus, by (6.5), α^t_{z} must be larger than $M_{\mathrm{z}}(t; l)$. Since the path of α must be continuous and remains in the same x-z plane as $e, \ldots, l$ for $t < t_0$,

$$M_{\mathrm{z}}(t; l) < \alpha^t_{\mathrm{z}} < M_{\mathrm{z}}(t; j, k) \tag{6.7}$$

and, it is implied by (6.7) and edges $\{i,j\}$, $\{j,k\}$ and $\{k,l\}$ that

$$M_{\mathrm{x}}(t; j, i) < \alpha^t_{\mathrm{x}} < M_{\mathrm{x}}(t; k, l). \tag{6.8}$$

Symmetrically, let $\beta^t = (\beta^t_{\mathrm{x}}, \beta^t_{\mathrm{y}}, \beta^t_{\mathrm{z}})$ where $t < t_0$ denote the point of intersection in $M(t)$ between edge $\{c,d\}$ and the x-z plane through $M_{\mathrm{y}}(t; e, \ldots, l)$. Then,

$$M_{\mathrm{z}}(t; e) < \beta^t_{\mathrm{z}} < M_{\mathrm{z}}(t; f, g) \tag{6.9}$$

and

$$M_{\mathrm{x}}(t; e, f) < \beta^t_{\mathrm{x}} < M_{\mathrm{x}}(t; g, h). \tag{6.10}$$

Notice that $\alpha^t_{\mathrm{z}} = \beta^t_{\mathrm{z}} = M_{\mathrm{z}}(t; a, b, c, d)$, where $t < t_0$. Putting this together with (6.5), (6.7) and (6.9) we have that

$$\begin{aligned} \max(M_{\mathrm{z}}(t; e), M_{\mathrm{z}}(t; l)) &< M_{\mathrm{z}}(t; a, b, c, d) \\ &< \min(M_{\mathrm{z}}(t; j, k), M_{\mathrm{z}}(t; f, g)). \end{aligned} \tag{6.11}$$

We claim that that for all $t < t_0$,

$$M_{\mathrm{x}}(t; k, l) < M_{\mathrm{x}}(t; e, f). \tag{6.12}$$

If this is not true, then there must exist some $t < t_0$ such that either $M_z(t; j, k) < M_z(t; e)$ or $M_z(t; f, g) < M_z(t; l)$. However, by (6.11) neither of these can hold. Hence, by contradiction (6.12) holds for all $t < t_0$.

Now, for $t < t_0$, $\alpha_x^t = M_x(t; a, b)$ and $\beta_x^t = M_x(t; c, d)$. Putting these facts together with (6.8), (6.10) and (6.12), we have that for all $t < t_0$,

$$M_x(t; a, b) < M_x(t; k, l) < M_x(t; e, f) < M_x(t; c, d). \tag{6.13}$$

By (6.11) and (6.13), we conclude that for all $t < t_0$ in $M(t)$ edge $\{b, c\}$ will intersect both $\{k, l\}$ and $\{e, f\}$ with respect to the x- and z- coordinates, contradicting the assumption that M is a parallel morph.

Hence, it is not possible that in $M(t_0)$ vertices $b, c, e, \ldots, l$ lie in the same x-z plane. Therefore, we may conclude that P and Q do not admit a parallel morph. □

6.4 Orthogonal Polyhedra

In this section we prove that parallel pairs of orthogonal polyhedra do not always admit a parallel morph. In Section 6.4.1 we describe parallel orthogonal drawings on the surface of a cube (we call these *cuboid* drawings) that do not admit a parallel morph. A cuboid drawing is a type of orthogonal polyhedron consisting of multiple coplanar faces. We show also that our unmorphable cuboid drawing can be turned into a pair of parallel orthogonally convex polyhedra without coplanar faces that does not admit a parallel morph.

Our unmorphable cuboid drawings have faces that are not rectangular. In contrast, if we restrict our attention to cuboid drawings in which all faces are bounded by rectangles, then a linear morph between parallel cuboid drawings will maintain simplicity, and thus is a parallel morph.

In Section 6.4.2, we will describe a pair of parallel orthogonal polyhedra that do not admit a parallel morph, yet every face is rectangular. Both polyhedra of this example have multiple coplanar faces. In Chapter 7, we will adapt the example of Section 6.4.2 to aid us in proving that it is PSPACE-hard to decide whether a pair of parallel orthogonal polyhedra will admit a parallel morph.

6.4.1 Unmorphable drawings on a cuboid

We say that an orthogonal drawing is a *cuboid drawing* if it is composed of the edges of a box in $\mathbb{R}^3$, augmented with the addition of orthogonal drawings on each face of the box; we allow that edges of the box are subdivided by vertices.

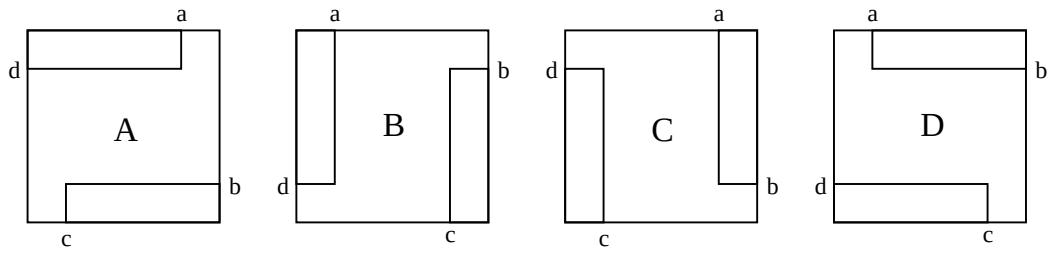

Figure 6.5: Orthogonal drawings in the x-y plane.

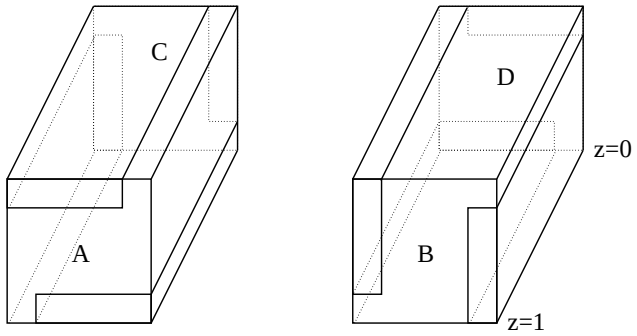

Figure 6.6: Parallel unmorphable cuboid drawings.

Theorem 6.4.1 *Not every pair of parallel cuboid drawings admits a parallel morph.*

Proof: Consider the orthogonal drawings in Figure 6.5. Observe that A is parallel with B, and C is parallel with D. In any parallel morph from A to B, the y-order—i.e. the vertical order of vertices in the figure—of b and d can change only if prior to that, the x-order of a and c changes. Similarly, in any parallel morph from C to D, the x-order of a and c can change only if prior to that, the y-order of b and d has changed.

We use the drawings of Figure 6.5 to construct parallel unmorphable cuboid drawings. For the first drawing, locate C such that all vertices have z-coordinate 0. Locate A such that all vertices have z-coordinate 1.

Connect each vertex with z-coordinate 0 by an edge parallel with the z-axis to the corresponding vertex with z-coordinate 1; see Figure 6.6. Construct the second drawing similarly, except that D has z-coordinate 0 and B has z-coordinate 1.

It is easy to verify that we have constructed a parallel pair of cuboid drawings. If there existed a parallel morph between these drawings, then A would morph to B, and simultaneously C would morph to D. During the morph from A to B, the y-order of b and d changes. Therefore the x-order of a and c must have changed even earlier. However, this is the same a and c as in the drawings of C and D. Hence this necessitates that the

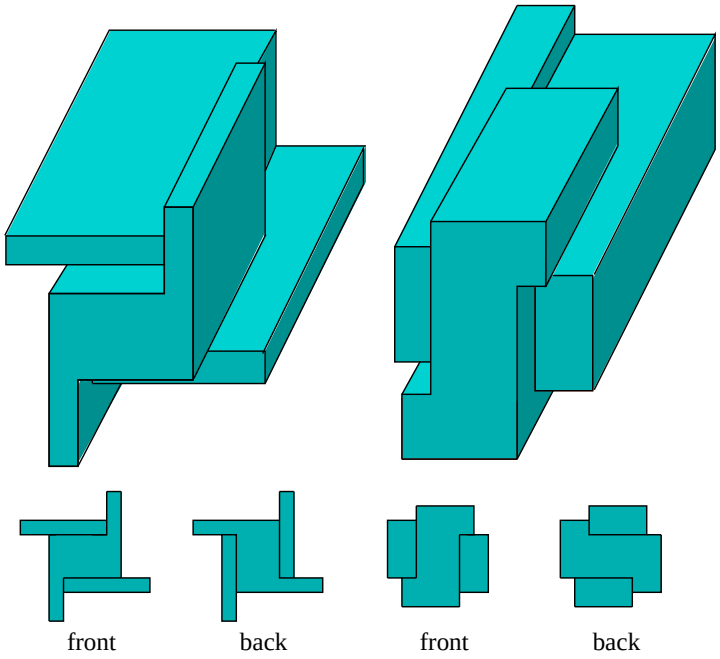

Figure 6.7: Parallel polyhedra that do not admit a parallel morph; no two faces are coplanar.

y-order of b and d has changed even earlier—a contradiction. Therefore, there does not exist a parallel morph. □

In the drawings in Figure 6.6 every face is a simple polygon, and so the drawings are orthogonal polyhedra. From this counterexample, we may conclude that not all pairs of parallel orthogonal polyhedra admit a parallel morph.

If we are to restrict ourselves to polyhedra without co-planar faces, it is still possible to build parallel orthogonal polyhedra that do not admit a parallel morph; see Figure 6.7. These polyhedra have essentially the same structure as the polyhedra of the previous example, and therefore do not morph. Note that these polyhedra are orthogonally convex.

What if we further restrict the shapes that faces may take? For parallel *rectangular* drawings, a linear morph provides a parallel morph; recall Lemma 3.1.1 of Chapter 3 (page 56). It follows from this lemma that a linear morph provides a parallel morph for all pairs of parallel cuboid drawings in which all faces are rectangular. However, restricting faces to be rectangles does not alone suffice to make parallel orthogonal polyhedra morphable, as we establish in the next section.

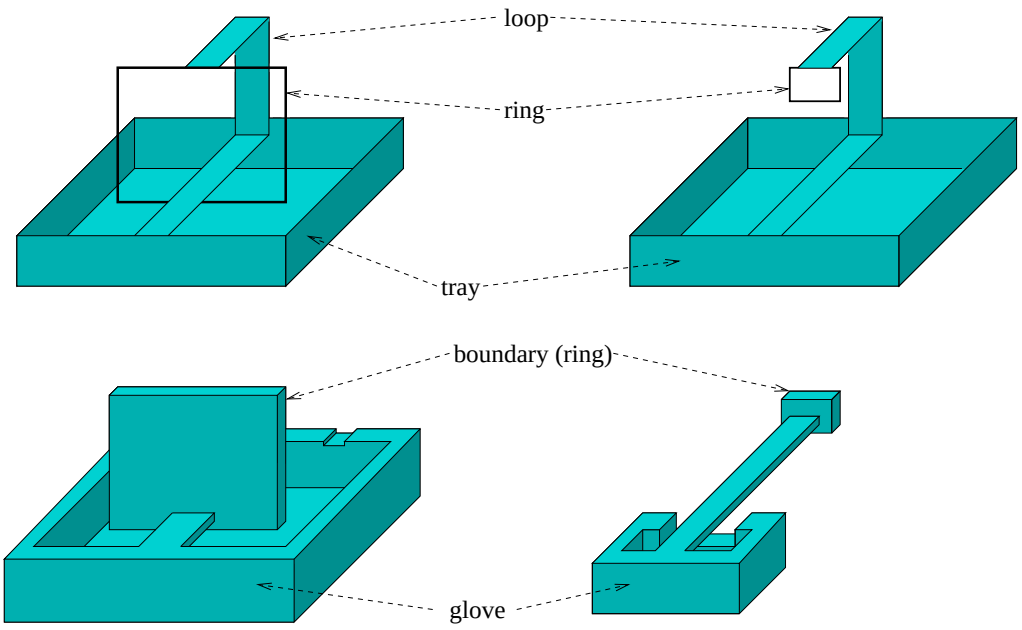

Figure 6.8: Building parallel orthodisks P^* (left) and Q^* (right).

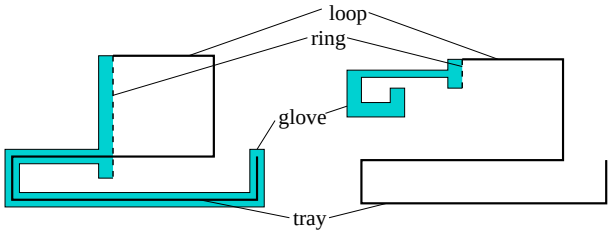

Figure 6.9: Cross sections of orthodisks P^* and Q^*.

6.4.2 Unmorphable rectangular-faced orthogonal polyhedra

We define an *orthodisk* as a region of an orthogonal polyhedron bounded by an orthogonal drawing of a cycle. This region of the polyhedron is topologically equivalent to a closed disk. To prove the existence of parallel pairs of unmorphable polyhedra, we find it convenient to first prove this result for orthodisks.

Theorem 6.4.2 *There exist parallel orthodisks that do not admit a parallel morph, even when all faces are rectangles.*

Proof: We construct parallel orthodisks P^* and Q^* whose boundaries are the unmorphable drawings P and Q of cycles illustrated in Figure 6.4, respectively. Since, by Lemma 6.3.1, P and Q do not admit a parallel morph, orthodisks P^* and Q^* also do not admit a parallel morph.

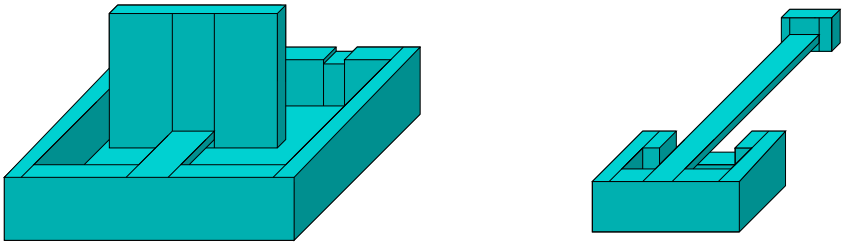

Figure 6.10: Add edges to the glove to make all faces rectangles.

We begin our construction of P^* and Q^* by adding new vertices, edges and faces to each of P and Q, as illustrated by the topmost two drawings in Figure 6.8. In particular, to both P and Q we add a lower structure that looks like a box without a top, called the *tray*. Attached to the tray is the *loop*, which consists of three rectangles. The loop connects the tray to a *ring*, which is simply the boundary of a rectangle. It should be clear that these parallel structures do not admit a parallel morph. However, due to the presence of the ring, the structures are not orthodisks.

To convert these unmorphable structures to orthodisks, we incorporate new parallel orthodisks called *gloves*. The lower-most drawings in Figure 6.8 depict the glove for each of P^* and Q^*. The boundary of each glove is a rectangle. In both P^* and Q^* the boundary of the glove is arranged to coincide with the ring. With the addition of the glove, the construction is complete.

For the sake of visualizing our construction, imagine that in Q^* the glove is a rubber surface stretched over the ring. To get to P^* from Q^*, the tray passes through the ring, extending the rubber surface around the tray. Hence, in P^* the glove encloses the tray, while in Q^* the tray is not enclosed by the glove (see Figure 6.9). To complete the construction the gloves of both P^* and Q^* must be transformed from rubber surfaces to parallel orthodisks.

The glove has five faces that are not rectangles (three of these are visible in Figure 6.8). However, in each of them the x-order of the vertices is the same in both polyhedra. Therefore, we can subdivide these faces into rectangles by adding, in both orthodisks, lines perpendicular to the x-axis from reflexive vertices; the other endpoint of each of these lines hits the same edge in both orthodisks since the x-order is the same, and hence the orthodisks stay parallel. See Figure 6.10. □

Corollary 6.4.3 *There exist pairs of parallel orthogonal genus-0 polyhedra that do not admit a parallel morph, even when faces are restricted to be rectangles.*

Proof: The construction above produced parallel polyhedral surfaces, which we now turn into polyhedra. The simplest approach is, for each surface, to glue together two

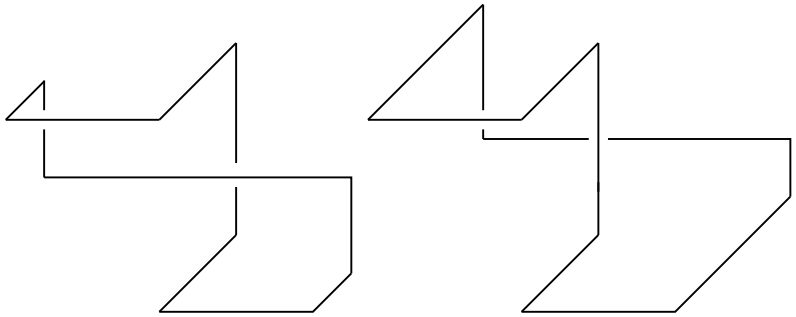

Figure 6.11: An unmorphable pair of parallel orthogonal drawings of a cycle.

identical copies of the surface, producing a degenerate polyhedron of zero volume. To avoid degeneracy we can "shrink" one copy of the surface away from the other keeping it combinatorially the same. It is not difficult to construct parallel polyhedra such that, in each, the band connecting the two copies is composed of rectangular faces. □

6.5 Conclusion

In this chapter we proved that when the underlying graph is a tree, every pair of parallel of drawings admit a parallel morph. However, this is not the case when the underlying graph is a cycle. We have given a pair of parallel orthogonal drawings of a cycle in $\mathbb{R}^3$ that does not admit a parallel morph, even though both drawings correspond to the trivial knot. Likewise, there exist pairs of parallel orthogonal genus-0 polyhedra that do not admit a parallel morph.

The parallel drawings of a cycle that we introduce in Lemma 6.3.1 seem to require a complicated proof to establish their unmorphability. There do exist other examples for which it is easier to establish unmorphability. In particular, Biedl et al. [9] proved that the parallel drawings illustrated in Figure 6.11 will not admit a parallel morph. The reason that we present the drawings of Figure 6.4 as our example, is in part because we use these drawings to build parallel unmorphable orthogonal polyhedra. It does not seem possible to incorporate the drawings of Figure 6.11 into parallel polyhedra.

In knot theory [50], a knot is often represented by a *knot diagram*, which is a curve in the plane with a finite number of *crossings*, where the curve self-intersects. At each crossing, a knot diagram must specify which section of the knot crosses "over" and which section crosses "under".

Imagine that we put an orientation on a knot. It then becomes possible to classify

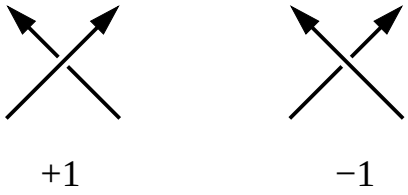

Figure 6.12: Positive (+1) and negative (-1) crossings.

the crossings as either positive or negative, as shown in Figure 6.12. The *writhe* of a knot diagram is defined as the number of positive crossings minus the number of negative crossings.

We might define writhe for drawings of a cycle: given a drawing of a cycle and a plane, take the orthogonal projection of the point set determined by the drawing onto the plane. If the curve formed in this projection crosses itself only at a finite number of points, then we can compute the writhe for the drawing and plane in a manner similar to computing the writhe for a knot diagram.

We conjecture that writhe defined in this way is invariant over all drawings that result from a parallel morph of a drawing of a cycle, with respect to a given plane of projection. If true, then this provides a quick way to prove that two drawings will not morph; we need only to prove that the writhe of the two drawings is different for some particular plane. Notice that the projections of the drawings shown in Figure 6.11 have different writhes. However, those in Figure 6.4 are equal. Hence, having equal writhes does not imply morphability.

It is well known that all knots are equivalent in $\mathbb{R}^4$ (or higher dimensions). Similarly, it is always possible to morph a drawing of a cycle in $\mathbb{R}^4$ maintaining simplicity and keeping edge lengths constant [17]. The same does not hold for parallel morphing: an unmorphable pair of parallel drawings in $\mathbb{R}^2$ remains unmorphable in higher dimensions. However, it would be interesting to study drawings in higher dimensions that will always morph.

Finally, we mention an open problem pertaining to morphing orthogonal polyhedra. For every pair of parallel cuboid drawings in which all faces are rectangular, the linear morph is a parallel morph. Certainly a linear morph does not suffice for a parallel morph for all orthogonally-convex polyhedra, even if all faces are restricted be rectangular. However, we have not been able to prove the existence of such a pair that will not morph. So, we leave it open: do all parallel pairs of orthogonally-convex polyhedra in which all faces are rectangular admit a parallel morph?

Chapter 7

PSPACE-Hardness Of Morphing

7.1 Introduction

In this chapter, we show three related problems to be PSPACE-hard. In Section 7.2, we prove that it is PSPACE-hard to decide whether two drawings in the plane admit a parallel morph with *static* vertices: given a pair P, Q of drawings of a connected graph G in the plane, and a distinguished subset $V_{\mathrm{S}} \subset \mathrm{V}(G)$ of static vertices such that for all $u \in V_{\mathrm{S}}$, $P(u) = Q(u)$, decide whether there exists a parallel morph M from P to Q such that, for all $v \in V_{\mathrm{S}}$ and $t \in [0, 1]$,

$$P(v) = M(t; v) = Q(v).$$

To prove this to be PSPACE-hard, we reduce from the problem of deciding whether there exists a sequence of moves taking us between two given configurations of a *nondeterministic constraint logic* (NCL) machine [37]. Background on NCL machines and the reconfiguration problem appears in Section 7.1.1.

While parallel morphing with static vertices seems to be a natural problem that is worthy of study for its own sake, the main reason that we study this problem here is that the problem reduces nicely to the other problems that we prove PSPACE-hard in this chapter. In Section 7.3 we prove that it is PSPACE-hard to decide whether a pair of parallel orthogonal drawings in $\mathbb{R}^3$ admits a parallel morph, and in Section 7.4 we prove it to be PSPACE-hard to decide parallel morphability for pairs of parallel orthogonal polyhedra, topologically equivalent to the sphere. It remains open to determine whether the PSPACE-hard problems discussed in this chapter are in PSPACE.

7.1.1 Nondeterministic constraint logic model of computation

A *nondeterministic constraint logic* (NCL) machine is defined by Hearn and Demaine [37] as an undirected graph (possibly with multi-edges and self-loops) with nonnegative integer weights on edges and vertices. The weights on the vertices are called *minimum inflow constraints*.

A *configuration* of an NCL machine is an assignment of orientations to each of the edges. A configuration is called *feasible* if for every vertex, the sum of weights on incoming edges is at least the minimum inflow constraint of the vertex.

Starting with an NCL in some feasible configuration, a *move* is the reversal of the orientation of a single edge, such that the resulting configuration is feasible. It is PSPACE-complete to decide each of the following, where an NCL machine is given as part of the input:

1. Given two feasible configurations A, B of the machine, does there exist a sequence of moves that takes us from A to B?

2. Given a feasible configuration A and an edge e of the graph, does there exist a sequence of moves that reverses the orientation of e?

3. Given orientations for two edges $e_{\mathrm{a}}, e_{\mathrm{b}}$ of the graph, does there exist a pair A, B of feasible configurations such that e_{a} has the specified orientation in A; e_{b} has the specified orientation in B; and, there exists a sequence of moves that takes us from A to B?

A vertex of an NCL machine is called an AND *vertex* if its minimum inflow constraint is 2, and it is incident upon three edges: two weight-1 edges, and one weight-2 edge. An AND vertex behaves like a logical AND in the sense that in a feasible configuration, when the weight-2 edge is oriented outward, both weight-1 edges must be oriented inward.

A vertex is an OR *vertex* if its minimum inflow constraint is 2, and it is incident upon three weight-2 edges. An OR vertex behaves as a logical OR in the sense that an incident edge may be oriented outward only if at least one of the other two edges are oriented inward.

An AND/OR *constraint graph* is a NCL machine in which every vertex is either an AND vertex or an OR vertex. All three decision problems given above remain PSPACE-complete even when the NCL machine is restricted to be a planar AND/OR constraint graph with no multi-edges or self-loops [37].

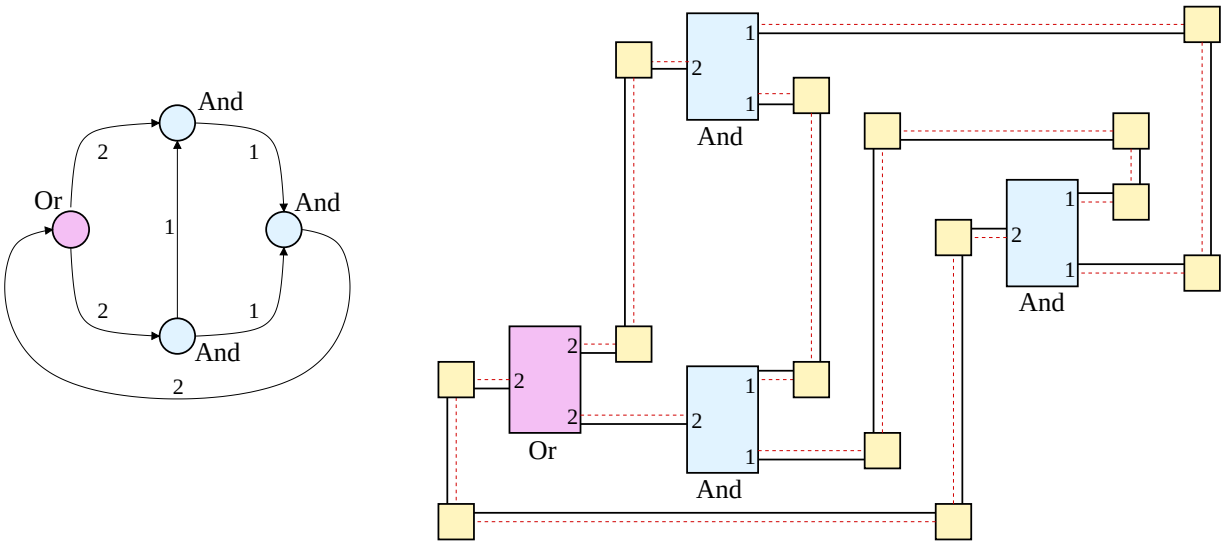

Figure 7.1: *Left:* a directed graph that describes a configuration of an AND/OR constraint graph. *Right:* outline of a drawing based on this configuration.

7.2 Orthogonal Drawings with Static Vertices

In this section, we prove the following.

Theorem 7.2.1 *Parallel morphing with static vertices is PSPACE-hard.*

The proof is via a reduction from the following PSPACE-hard problem [37]: given a simple planar AND/OR constraint graph H, and two feasible configurations C_P, C_Q of H, decide whether there exists a sequence of moves taking us from one feasible configuration to the other.

7.2.1 The reduction in overview

Given an AND/OR constraint graph H and two feasible configurations C_P, C_Q of H, we generate parallel orthogonal drawings P, Q in the plane of a connected graph G, and a subset $V_S \subset V(G)$ such that $P(v) = Q(v)$, for all $v \in V_S$. It is established that these drawings admit a parallel morph that keeps all vertices of V_S static if and only if H can be reconfigured from C_P to C_Q via some sequence of moves. The construction is illustrated in overview in Figure 7.1.

The boxes in Figure 7.1 represent the various *gadgets* used in the construction. The solid line segments, including those on the boundaries of the boxes, represent edges of the drawing. The dashed segments are not edges of the drawing, but instead are guidelines between pairs of static vertices that are horizontally or vertically aligned. The positions of edges in the drawings relative to these guidelines encode edge orientations of configurations of H.

Three types of gadget are used in the construction:

1. AND gadgets, corresponding to AND vertices,
2. OR gadgets, corresponding to OR vertices; and,
3. *turn gadgets*, which enable us to connect the AND and OR gadgets.

Each AND and OR gadget has a 2×3 box on its boundary, and a turn gadget has a 1×1 box on its boundary.

An AND [OR] vertex of H is represented in P and Q by an AND [OR] gadget. A turn gadget may be thought of as corresponding to a degree-2 vertex of an NCL machine—call this a *turn vertex*—in which the two incident edges have equal weight, and the minimum inflow constraint on the vertex is equal to the weight of each edge. In every feasible configuration at least one of the edges incident at a turn vertex must be directed toward the turn vertex.

Three edges are incident on each AND and OR gadget: a horizontal edge on the left, and two horizontal edges on the right. In an AND gadget, the single edge on the left corresponds to the weight-2 edge of an AND vertex, and the two edges on the right correspond to the two weight-1 edges of an AND vertex.

Exactly one vertical edge and one horizontal edge are incident on each turn gadget. Zero or more turn gadgets are strung together to connect AND/OR gadgets in the same manner as edges in H.

To construct P and Q, first compute a simple drawing of H in which each vertex is represented by a point and each edge is represented by an *orthogonal polyline*, i.e. an orthogonal drawing of a path graph. The linear-time algorithm of Kant [43] can be used for this purpose. This algorithm generates an orthogonal drawing of a given connected graph of maximum vertex degree 3. The number of bends in a drawing generated by the algorithm is at most $\lceil n/2 \rceil + 1$, where n is the number of vertices in the graph.

Once an orthogonal drawing of H has been created, generate drawings P and Q from this drawing, such that the AND and OR vertices are replaced by the appropriate gadgets. This may require adding bends to the edges to route them so that they meet the AND and

OR gadgets in the prescribed manner. Finally, the bends are replaced by turn gadgets. Each gadget has $O(1)$ complexity. It is not difficult to see that the construction can be generated in time that is polynomial in the complexity of H.

Turn vertices

To simplify the proof, we want that there is a one-to-one correspondence between vertices of H and gadgets appearing in P and Q. However, by assumption all vertices of H are either AND or OR vertices; there are no turn vertices in H. It is shown below that turn vertices can be added to H without affecting whether there exists a sequence of moves between C_P and C_Q.

Consider the following modification of H: choose a positive integer k, and and edge $\{u, v\}$ of H. Subdivide $\{u, v\}$ by k vertices, i.e. remove $\{u, v\}$ from H, and add vertices $w_1, \ldots, w_k$ and edges $\{u, w_1\}, \ldots, \{w_k, v\}$ to H. Assign a weight equal to the weight of $\{u, v\}$ to each of the newly added edges and vertices. Clearly, each of $w_1, \ldots, w_k$ is a turn vertex.

The feasible configurations C_P and C_Q are modified as follows. If $\{u, v\}$ has orientation (u, v) in one feasible configuration—i.e. the edge $\{u, v\}$ is directed from u to v—modify this configuration such that the newly added edges are oriented: $(u, w_1), \ldots, (w_k, v)$. It is not difficult to see that the modified configurations correspond to a feasible configuration of the new NCL machine.

Repeat the above process for any number of times. Let H' and C'_P, C'_Q denote the modified graph and feasible configurations that result.

Lemma 7.2.1 *There exists a sequence of moves from C_P to C_Q if and only if there exists a sequence of moves from C'_P to C'_Q.*

Proof: Assume that there exists a sequence of moves on H from C_P to C_Q. Consider the first move in the sequence. In this move, a single edge $\{u, v\} \in \mathrm{E(H)}$ reverses orientation from (u, v) to (v, u). In H', $\{u, v\}$ has been replaced by some number of edges $\{u, w_1\}, \ldots, \{w_k, v\} \in \mathrm{E(H')}$. By definition, in C'_P these edges are oriented $(u, w_1), \ldots, (w_k, v)$.

Simulate the reversal of (u, v) in machine H' by first reversing (w_k, v), followed by (w_{k-1}, w_k), etc. until finally (u, w_1) is reversed. It is easy to verify that this procedure keeps the inflow constraint of each vertex satisfied throughout. Following this sequence of moves, the configuration of H' corresponds to the configuration of H following the first

move in the sequence from C_P to C_Q. By applying the above procedure repeatedly, a sequence of moves is obtained, from C'_P to C'_Q.

Conversely, assume that there exists a sequence of moves in H' from C'_P to C'_Q. For every set of edges $\{u, w_1\}, \ldots, \{w_k, v\} \in \mathrm{E(H')}$ that corresponds to a single edge $\{u, v\}$ of H, notice that a feasible configuration of H' may have either (w_1, u) or (w_k, v), but never both; if a configuration has both (w_1, u) and (w_k, v), then for some w_i where $i \in \{1, \ldots, k\}$, both incident edges are oriented away from w_i.

Simulate the sequence in H as follows. Whenever (u, w_1) reverses to (w_1, u) in the sequence of H', if $\{u, v\}$ is oriented (u, v) in the current configuration of H, reverse this edge. Similarly, whenever (v, w_k) reverses to (w_k, v) in the sequence of H', if $\{v, u\}$ is oriented (v, u) in the current configuration of H, reverse this edge. This sequence of H keeps all inflow constraints satisfied. Therefore, there exists a sequence of moves from C_P to C_Q. □

Thus, by Lemma 7.2.1, turn vertices can be added to H without affecting the decision problem from which we are reducing. For the remainder of the proof of Theorem 7.2.1, assume that H is a planar, simple AND/OR constraint graph with turn vertices, such that P, Q and V_S can be constructed from C_P, C_Q, and H such that gadget of our construction corresponds to a vertex of H.

Encoding edge orientations

There exists a one-to-one correspondence between vertices of H and gadgets of the construction. If $\{u, v\} \in \mathrm{E(H)}$ then there exists either a horizontal or a vertical edge connecting the two corresponding gadgets in both drawings of the construction. The position of this horizontal or vertical edge relative to the position of the dashed segment between the two gadgets is used to encode the orientation of $\{u, v\}$ in a configuration of H. Recall that each dashed segment connects two static vertices of V_S.

Suppose that the gadgets corresponding to vertices u and v are connected by a horizontal edge in one of the drawings such that the gadget associated with u lies to the left of the gadget associated with v. If the horizontal edge between the gadgets lies strictly below the dashed segment, the drawing encodes left-to-right orientation, from u to v. If the edge lies strictly below the dashed segment, the drawing encodes a right-to-left orientation, from v to u; recall Figure 7.1.

Similarly, suppose that the two gadgets corresponding with u and v are connected by a vertical edge and that u lies below v. If the edge lies strictly to the left of the dashed segment, the drawing encodes bottom-to-top orientation, from u to v. If the edge

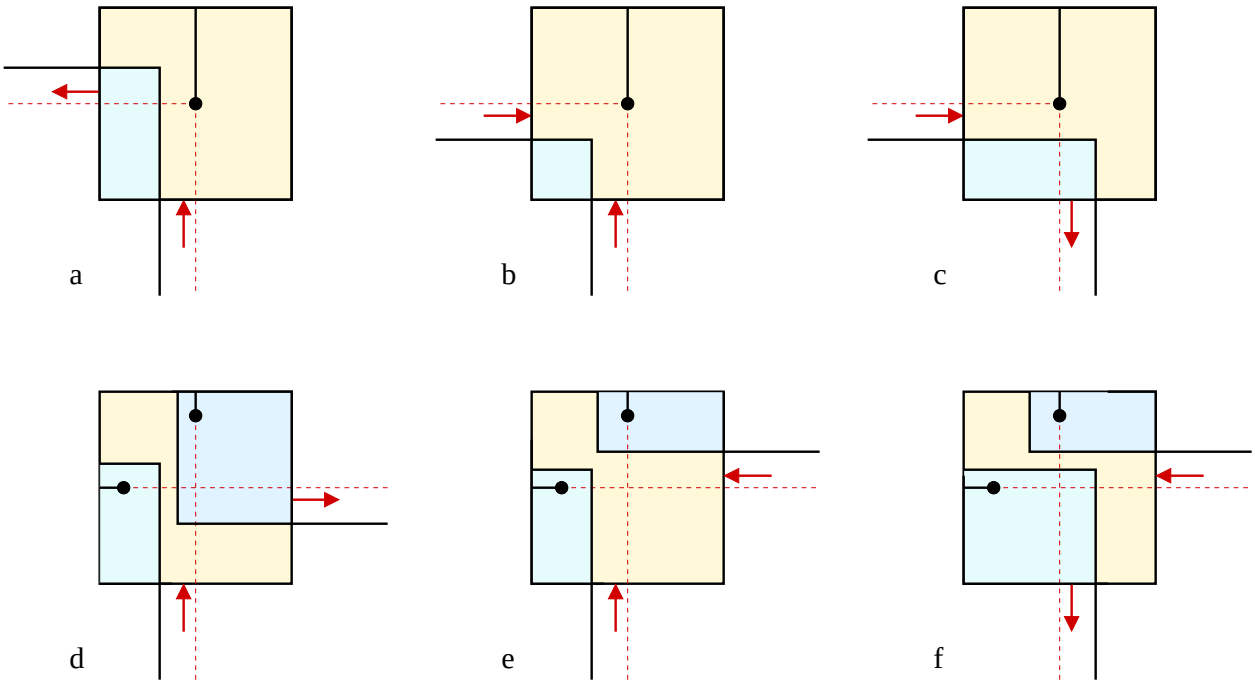

Figure 7.2: Turn gadgets.

lies strictly to the right of the dashed segment, the drawing encodes a top-to-bottom orientation, from v to u.

When an edge of the drawing coincides with its dashed segment, the drawing can be thought to encode a state of H in which the associated edge of H does not contribute to the inflow of either of its vertices. Note that such a state is not realizable in an NCL machine.

7.2.2 Gadgets

Having described the construction in overview, it remains to introduce the gadgets, and to prove that each gadget behaves in a manner equivalent to the corresponding vertex of an NCL machine.

Turn gadgets

A turn gadget is connected by two edges (one horizontal and one vertical) to two other gadgets. Consider the gadget illustrated in Figure 7.2 (a,b,c). In this figure the solid line segments represent edges of the drawing, and the dots represent static vertices of V_S. Dashed line segments serve to connect vertices of V_S, and are not part of the drawing. The

three drawings representing this turn gadget are parallel and share a common set of static vertices.

Whether to use (a), (b) or (c) to represent a turn vertex in some configuration of H depends on the orientations of the edges incident with the turn vertex in the configuration. Drawing (a) encodes a configuration locally at the turn vertex in which the edge on the left is directed away from the turn vertex, while the edge below is directed toward the turn vertex. Drawing (c) encodes the reverse of the edge orientations of (a), while drawing (b) encodes both edges directed toward the turn vertex.

The gadget illustrated in Figure 7.2 (a,b,c) alone does not suffice. Also required is a turn gadget that connects to a gadget below and a gadget to the right. If we were simply to rotate/reflect the drawings already described, the edge orientation encoding scheme would be destroyed. Instead, we introduce another turn gadget, illustrated in Figure 7.2 (d,e,f).

To complete the collection, turn gadgets are required that connect a turn gadget to another gadget above it and another gadget either to the left or right. These gadgets are obtained by rotating each of the drawings of Figure 7.2 by π. Observe that a rotation by π leaves the edge orientation encoding scheme intact.

Lemma 7.2.2 *In a parallel morph that keeps vertices in V_S static, a turn gadget behaves locally in a manner equivalent to a turn vertex in an NCL machine.*

Proof: Drawings (a,b,c) of Figure 7.2. correspond to the three feasible configurations of edge orientations at a turn vertex in an NCL machine, as do the drawings (d,e,f). Observe that both (a) and (c) admit a parallel morph to (b) that leaves the single vertex of V_S static. Both (d) and (f) admit a parallel morph to (e) that leaves the two vertices of V_S static. These parallel morphs are sufficient to simulate any move of an NCL machine.

In the other direction, consider a parallel morph between two drawings of a turn gadget that keeps vertices of V_S static. Crucially, it is impossible to morph a drawing of a turn gadget to a drawing corresponding to a configuration that violates the minimum inflow constraint of the associated turn vertex. To morph between drawings (a) and (c), it is necessary that an intermediate drawing of this morph encodes both edges directed inward. Likewise, to morph between drawings (d) to (f) it is necessary to go via an intermediate drawing that encodes both edges directed inward. Thus, for any parallel morph between two drawings of a turn gadget that keeps vertices of V_S static, there exists a corresponding sequence of moves about a turn vertex in an NCL machine. □

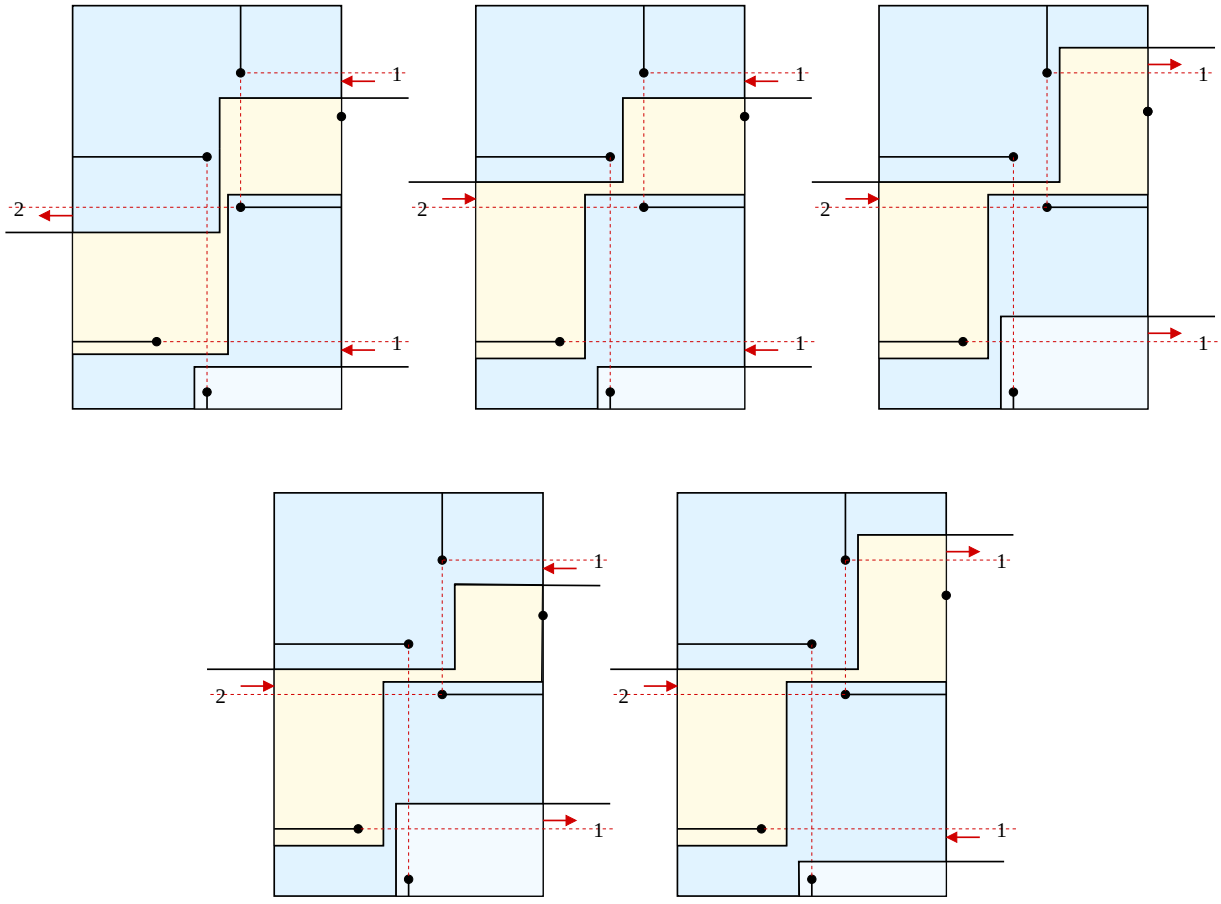

Figure 7.3: The AND gadget.

The AND gadget

The AND gadget is illustrated in Figure 7.3. Recall that an AND vertex of H has two incident weight-1 edges and one incident weight-2 edge, and the minimum inflow constraint is 2. In the AND gadget, the horizontal edge on the left corresponds to the weight-2 edge, and the two horizontal edges on the right correspond to the two weight-1 edges. All drawings in the figure are parallel, with a common set of static vertices.

The five drawings of Figure 7.3 together encode all feasible configurations of edges incident upon an AND vertex of H. The leftmost drawing in the upper row encodes the configuration in which the weight-2 edge is directed away from the AND vertex, and the two weight-1 edges are directed toward it. The remaining gadgets encode the four possible configurations in which the weight-2 edge is directed toward the AND vertex, and the weight-1 edges have arbitrary orientations.

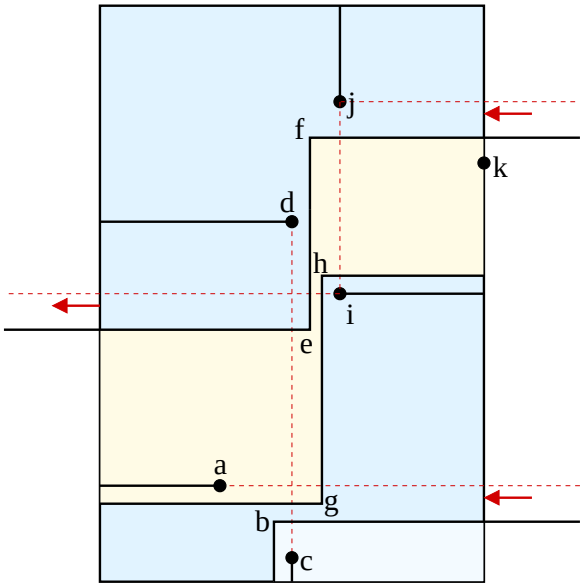

Figure 7.4: Proving that the AND gadget behaves like an AND vertex.

Lemma 7.2.3 *During a parallel morph that keeps all vertices in V_S static, an* AND *gadget behaves locally in a manner equivalent to an* AND *vertex in an NCL machine.*

Proof: Consider a move of an NCL machine H that reverses the orientation of an edge incident to an AND vertex. For the configurations of H immediately before and immediately after the move, the orientations of the edges incident to the AND vertex correspond to one of the drawings of Figure 7.3. For any pair of drawings of the AND gadget whose corresponding configurations differ locally in the orientation of a single edge, it is easy to verify that this pair admits a parallel morph that keeps the vertices in V_S static. Therefore, any sequence of moves of H involving the edges incident to an AND vertex can be simulated by a parallel morph between drawings of an AND gadget.

Conversely, consider a parallel morph from one drawing of the AND gadget to another that keeps the vertices in V_S static. The following claim establishes the impossibility of morphing to a drawing encoding an infeasible configuration of H.

Claim 7.2.1 *Every drawing that is parallel with the drawings of the* AND *gadget corresponds locally to a feasible configuration of an* AND *vertex of an NCL machine.*

Proof: Consider the drawing of the AND gadget illustrated in Figure 7.4. This drawing is identical to the leftmost drawing in the top row of Figure 7.3. For the vertex labeled a, let a_x and a_y denote the x- and y-coordinates of a, respectively; likewise for vertices

$b, \ldots, j$. To establish the claim, it is sufficient to prove that in the drawing resulting from a parallel morph of the AND gadget, if $e_y < i_y$ then both $f_y < j_y$ and $b_y < a_y$.

Assume that $e_y < i_y$. In any drawing parallel with an AND gadget that does not alter the static vertices, $i_y < d_y < k_y < f_y$. It must be that

$$e_y < i_y < d_y < f_y.$$

Observe that if either $e_x = f_x \geq i_x$ or $d_x \geq e_x = f_x$, then the drawing self-intersects. In particular, the edge between e and f intersects either the edge connected to d or i. Therefore,

$$c_x = d_x < e_x = f_x < i_x = j_x. \tag{7.1}$$

Notice, if both $f_y \geq j_y$ and $j_x \geq f_x$ then the drawing self-intersects. However, since $f_x < j_x$ it follows that $f_y < j_y$.

Next, we prove that $b_y < a_y$. In any drawing that is parallel with an AND gadget and with static vertices in the same positions as in the AND gadget,

$$g_y < a_y < i_y < h_y.$$

Also, $a_y < e_y$. Since $e_y < i_y$ (by assumption),

$$g_y < e_y < h_y.$$

If $e_x \geq g_x = h_x$ then the vertical edge connecting g and h intersects the horizontal edge connecting e to the left boundary. Therefore, $e_x < g_x = h_x$; so by (7.1), $c_x < e_x < g_x$. Finally, since $g_y < a_y$ and $c_x < g_x$, it follows that $b_y < a_y$. □

By the preceding claim, every intermediate drawing of a parallel morph between two drawings of an AND gadget encodes a feasible configuration of an NCL machine locally at the AND vertex, assuming that static vertices are not moved during the morph. □

The OR gadget

Recall that an OR vertex of H has three incident weight-2 edges, and a minimum inflow constraint of 2. Thus, in every configuration of H, at least one of the edges incident upon an OR vertex must be directed toward the vertex.

Figure 7.5 illustrates three OR gadgets that are parallel drawings with an identical set of static vertices. Each of the gadgets shown encodes a feasible configuration of the edges incident at an OR vertex in which one of the edges is directed toward the vertex, while the

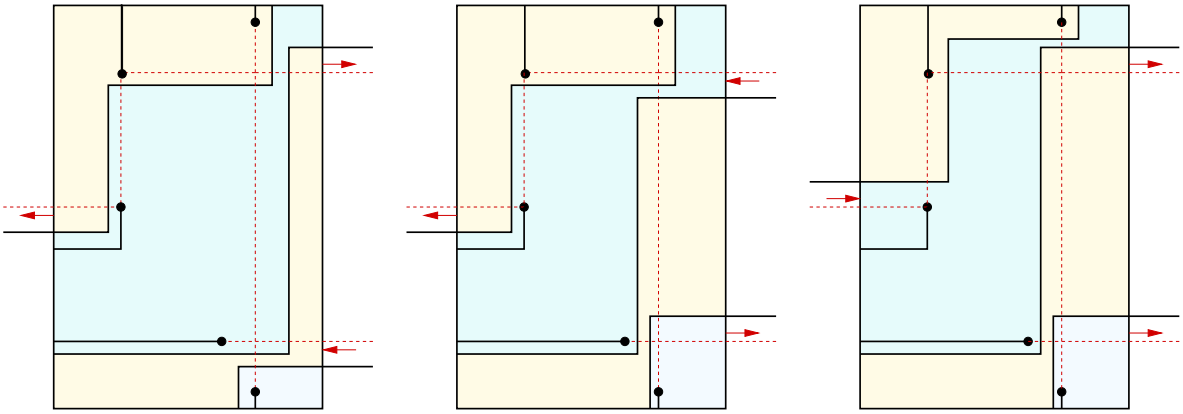

Figure 7.5: Three of the seven possible drawings of the OR gadget.

other two are directed away. From these gadgets, it is not difficult to construct parallel OR gadgets upon the same set of static vertices, encoding all feasible configurations in which at least one edge is directed toward an OR vertex. In total, $2^3 - 1 = 7$ OR gadgets are required, including the three illustrated in Figure 7.5.

Lemma 7.2.4 *During a parallel morph that keeps all vertices in V_S static, an* OR *gadget behaves locally in a manner equivalent to an* OR *vertex in an NCL machine.*

Proof: In an NCL machine H, every feasible configuration of the edges incident with an OR vertex corresponds to the encoding determined by one of the seven possible OR gadgets. So, for any move of H, the configuration of the edges incident with an OR vertex immediately before the move will correspond to the encoding determined by an OR gadget. Likewise, for the configuration immediately following the move.

It is easy to verify that for each possible move, there exists a parallel morph between the corresponding gadgets, that keeps the vertices of V_S static. Thus, for every sequence of moves of H, the sequence of configurations of edges incident with an OR vertex can be simulated by a parallel morph that starts with one of the OR gadgets, and keeps all vertices of V_S static.

Conversely, to see that every parallel morph from an OR gadget that keeps vertices of V_S static corresponds, locally, to a sequence of moves between feasible configurations of H, observe that we can never morph an OR gadget in such way that in the resulting drawing the horizontal edge on the left is on or below its dashed segment, while both edges on the right are on or above their respective dashed segments—such a drawing encodes all edges directed away from the corresponding OR vertex. □

With that, we have completed our proof of Theorem 7.2.1.

7.2.3 Strengthening the theorem

In this section, we strengthen Theorem 7.2.1 to allow all vertices at least some limited freedom of movement, while maintaining the PSPACE-hardness of the decision problem. Then, we prove that the problem remains PSPACE-hard even when we restrict the input so that no two static vertices share an x- or y-coordinate. This strengthened version of the problem will reduce more easily than the original problem when we prove PSPACE-hardness for other problems later in this chapter.

Order preserving morphs

We say that a morph M from a drawings P of a graph G in the x-y plane *preserves the order* of a subset $V_{\mathrm{S}} \subseteq \mathrm{V(G)}$ if, in every intermediate drawing of the morph, the vertices of V_{S} exhibit the same vertex ordering as in P, i.e. either:

1. $P_i(u) < P_i(v)$ and $M_i(t;u) < M_i(t;v)$,
2. $P_i(u) = P_i(v)$ and $M_i(t;u) = M_i(t;v)$; or,
3. $P_i(u) > P_i(v)$ and $M_i(t;u) > M_i(t;v)$,

for all $i \in \{\mathrm{x}, \mathrm{y}\}$, all $u, v \in V_S$, and all $t \in [0, 1]$.

Theorem 7.2.2 *Given a pair of parallel orthogonal drawings of a graph in the plane and a distinguished subset of the vertices of the graph, it is PSPACE-hard to decide whether there exists a parallel morph between the drawings that preserves the ordering of the vertices in the subset.*

Proof: This follows immediately from Theorem 7.2.1 and Lemma 7.2.5 (below). □

Besides using the following lemma in the proof of Theorem 7.2.2, we refer to this lemma in Sections 7.3 and 7.4 when we prove that parallel morphing is PSPACE-hard for orthogonal drawings in $\mathbb{R}^3$, and for orthogonal polyhedra.

Lemma 7.2.5 *Given a pair P and Q of parallel orthogonal drawings in the plane of a connected graph G, and a subset $V_{\mathrm{S}} \subset \mathrm{V(G)}$ where for all $v \in V_{\mathrm{S}}$, $P(v) = Q(v)$; there exists a parallel morph from P to Q that keeps all vertices of V_{S} static if and only if there exists a parallel morph from P to Q that preserves the order of V_{S}.*

Proof: Trivially, if there exists a parallel morph from P to Q that keeps vertices of V_{S} static, then this morph preserves the order of V_{S}. So assume that a parallel morph M exists from P to Q that preserves the order of V_{S}. We will demonstrate the existence of a parallel morph M' from P to Q that keeps vertices of V_{S} static.

For all $t \in [0,1]$, let $x_1(t), \ldots, x_k(t)$ denote the values of the x-coordinates that are occupied by at least one vertex of V_{S} in $M(t)$, such that for all $i \in \{1, \ldots, k-1\}$,

$$x_i(0) < x_{i+1}(0).$$

Since M preserves the order of V_{S}, the x_i values will change continuously with t, but will never collide. Hence, for all $t \in [0,1]$ and $i \in \{1, \ldots, k-1\}$,

$$x_i(t) < x_{i+1}(t).$$

Define, for all $t \in [0,1]$, $x_0(t)$ and $x_{k+1}(t)$ to be values that are strictly smaller and larger (respectively) than all x-coordinates of vertices in $M(t)$, such that the functions x_0 and x_{k+1} change in a continuous manner within the range $[0,1]$.

Now, define morph M' as follows: for every $v \in \mathrm{V(G)}$ and $t \in [0,1]$ such that

$$x_i(t) \leq M_{\mathrm{x}}(t, v) < x_{i+1}(t),$$

let

$$M'_{\mathrm{x}}(t; v) = \left(\frac{M_{\mathrm{x}}(t; v) - x_i(t)}{x_{i+1}\ (t) - x_i(t)} \right) (x_{i+1}(0) - x_i(0)) + x_i(0). \tag{7.2}$$

Use an analogous approach to determine the vertex positions in M' with respect to the y-axis.

Since $x_i(t) < x_{i+1}(t)$ for all $t \in [0,1]$, the denominator on the right side of (7.2) never goes to zero. Hence, $M'(t; v)$ is defined for all $t \in [0,1]$ and $v \in \mathrm{V(G)}$. It is easy to verify that the *vertex ordering* (Definition 2.2.1, page 29) of drawing $M'(t)$ is the same as the vertex ordering of $M(t)$. Hence, by Lemmas 2.2.1 and 2.2.2 (pages 29 and 30, respectively) each drawing $M'(t)$ is simple and parallel with P and Q. Also, the vertices follow continuous paths in morph M'. It follows that M' is a parallel morph from P to Q.

Observe that each vertex $v \in V_{\mathrm{S}}$ has the same coordinates over all intermediate drawings of M'. Therefore, M' keeps all vertices in V_{S} static. □

Enforcing distinct coordinates on static vertices

The gadgets we devised in our proof of Theorem 7.2.1 require that we allow multiple static vertices to share x- and y-coordinates. When we prove that deciding parallel morphability

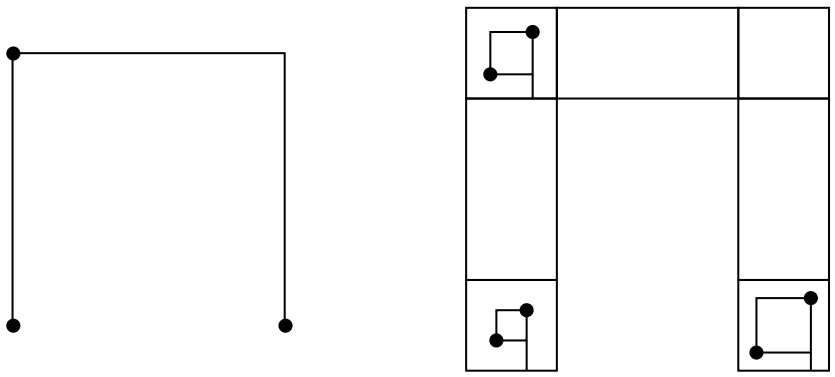

Figure 7.6: Proving the PSPACE-hardness of parallel morphing with static coordinates whose x- and y-coordinates are distinct. *Left:* drawing P. *Right:* drawing P'. The dots represent the vertices of V_{S} and V'_{S}, respectively.

for orthogonal polyhedra is PSPACE-hard (Section 7.4) we find it convenient to reduce from the problem of deciding parallel morphability with static vertices, but under the assumption that all static vertices have distinct x- and y-coordinates.

Theorem 7.2.3 *Parallel morphing with static vertices is PSPACE-hard, even when we require that all static vertices have distinct x- and y-coordinates.*

Proof: Let P and Q be parallel drawings in the plane of a graph G, and let

$$V_{\mathrm{S}} = \{v_1, \ldots, v_k\} \subset \mathrm{V(G)}$$

be a subset such that $P(v) = Q(v)$ for all $v \in V_{\mathrm{S}}$. Without loss of generality, assume that both P and Q map vertices of V_{S} to integer coordinates. By Theorem 7.2.1 it is PSPACE-hard to decide whether P and Q will admit a parallel morph that keeps all vertices of V_{S} static.

We will show how, in polynomial time, to generate a pair of parallel orthogonal drawings P' and Q' of a connected graph G' in the plane, and a subset $V'_{\mathrm{S}} \subset \mathrm{V(G')}$ such that all vertices of V'_{S} have distinct x- and y-coordinates in each of P' and Q'; and, P' admits a parallel morph to Q' that keeps vertices of V'_{S} static if and only if P admits a parallel morph to Q that keeps vertices of V_{S} static. Below, we describe the construction of P' and V'_{S} (G' is implicit in this construction). Drawing Q' is generated similarly.

For every vertex $u \in \mathrm{V(G)}$ locate a square of side length $1/3$ centered at $P(u)$. For every edge $\{v, w\} \in \mathrm{E(G)}$, connect the squares corresponding to v and w by two parallel edges; see Figure 7.6. For each $v_i \in V_{\mathrm{S}}$, construct a square of side length $\frac{i}{4k}$ centered at

$P(v_i)$, i.e. this square resides inside the already existing square that corresponds to v. So that the resulting drawing is of a connected graph, join the two squares by a vertical or horizontal edge, subdividing edges of the squares by vertices as necessary. Add the vertices of the lower-left and upper-right of the inner square to V'_{S}.

This completes the construction of P' and V'_{S}. Drawing Q' is generated similarly from Q and V_{S}. It is easy to verify that:

1. P' and Q' are parallel drawings of a common graph,
2. P' and Q' are simple; and,
3. the x- and y-coordinates of vertices in V_{S} are distinct in these drawings.

If P' admits a parallel morph to Q' that keeps all vertices of V'_{S} static, then P admits a parallel morph to Q that keeps all vertices of V_{S} static—to obtain a morph from P to Q, the coordinates of each vertex are determined by a point in the center of the associated square of G'.

Conversely, assume that there exists a parallel morph M from P to Q that keeps all vertices of V_{S} static. We must show that P' admits a parallel morph to Q' that keeps vertices of V'_{S} static. By Lemma 7.2.5 it suffices to prove the existence of a parallel morph M' from P' to Q', such that this morph preserves the ordering of vertices in V'_{S}.

The idea is to move the subset of $\mathrm{V(G')}$ that corresponds to a single vertex $v \in \mathrm{V(G)}$ in unison, following the motions of v in M. To ensure that simplicity is maintained, we must first perform a parallel morph of P' so that each subset of $\mathrm{V(G')}$ that corresponds to a vertex $v \in \mathrm{V(G)}$ lie sufficiently close to $P(v)$. This process must be reversed at the end of the morph.

Let $\alpha \in (0, 1]$ be a value that is strictly smaller than the minimum feature size of $M(t)$, for all $t \in [0, 1]$. To construct M', we will first perform a linear morph from P' to a drawing R, such that for every $v' \in \mathrm{V(G')}$ whose corresponding vertex in G is v

$$R(v') = \alpha(P'(v') - P(v)).$$

Observe that in R, all vertices lie at coordinates that are within an L_1-distance of $\frac{\alpha}{6}$ of the coordinates of the corresponding vertex of P. The linear morph from P' to R is a parallel morph that preserves the ordering of the vertices of V'_{S}.

Next, we move each vertex $v' \in \mathrm{V(G')}$ that corresponds to a single vertex $v \in \mathrm{V(G)}$ in unison, so that v' follows the same motions in M' as v does in M. If two elements intersect during this stage, then M does not maintain simplicity. Thus, our approach maintains simplicity and is clearly a parallel morph. Finally, we perform a linear morph to Q'. □

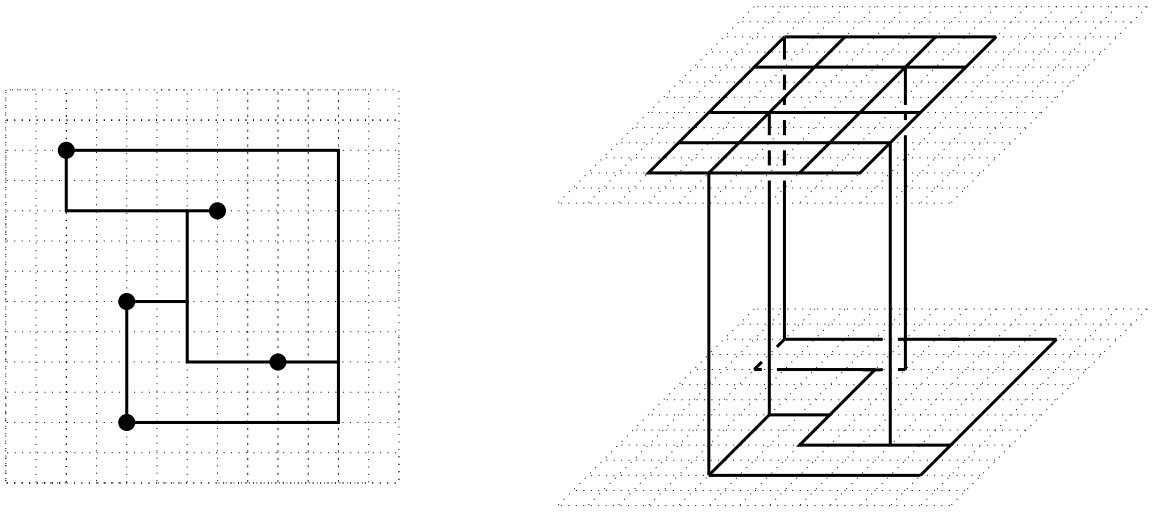

Figure 7.7: Proving PSPACE-hardness for orthogonal drawings in $\mathbb{R}^3$. *Left:* an example of drawing P in the x-y plane; the dots are vertices of V_S. *Right:* the drawing P' in x-y-z space.

7.3 Orthogonal Drawings in Space

In this section, we prove the following theorem.

Theorem 7.3.1 *It is PSPACE-hard to decide whether two parallel orthogonal drawings of a connected graph in $\mathbb{R}^3$ will admit a parallel morph.*

Proof: We reduce from the problem of parallel morphing with static vertices (Theorem 7.2.1). So, suppose that we are given a pair of parallel orthogonal drawings P and Q of a connected graph G in the x-y plane, and a subset $V_S \in \mathrm{V}(G)$ such that for all $v \in V_S$, $P(v) = Q(v)$. We want to decide whether there exists a parallel morph from P to Q that keeps vertices in V_S static. Without loss of generality, assume integer coordinates on all vertices of the input drawings.

Generate a pair of drawings P' and Q' in x-y-z space as follows (see Figure 7.7):

1. Construct P' from P and construct Q' from Q, by setting the z-coordinate of each vertex in $\mathrm{V}(G)$ equal to 0.

2. Augment both Q' and P', by adding a (non-uniform) grid in an x-y plane at some positive z-coordinate. For each x- (and y-) coordinate used by a vertex of V_S, include a grid-line with this x- (or y-) coordinate. Vertices are added at the intersection of grid lines.

3. For each vertex of $v \in V_S$, add an edge parallel to the z-axis between v and the vertex of the grid with the same x-y coordinates.

The orthogonal drawings P' and Q' can be computed in time that is polynomial in the complexity of P and Q. It remains to prove that there exists a parallel morph from P to Q that keeps vertices of V_S static if and only if there exists a parallel morph from P' to Q'.

So, let us assume that there exists a parallel morph from P to Q that keeps vertices of V_S static. Clearly, there exists a parallel morph from P' to Q' that keeps the z-coordinate of each vertex static, and keeps every vertex that belongs to the grid static.

Conversely, let us assume that there exists a parallel morph M' from P' to Q'. For every $t \in [0, 1]$ the drawing of G determined by $M'(t)$ is parallel with P and Q and lies in a single plane, perpendicular to the z-axis. In $M'(t)$, all vertices belonging to the grid lie in another plane perpendicular to the z-axis. The grid keeps the ordering of the vertices of V_S from changing with respect to either the x- or y-axis during M'.

From M' we can obtain a parallel morph from P to Q that preserves the ordering of the vertices of V_S. Hence, by Lemma 7.2.5, there exists a parallel morph from P to Q that keeps the vertices of V_S static. □

7.4 Orthogonal Polyhedra

We now turn to the problem of morphing orthogonal polyhedra. It is not difficult to prove it PSPACE-hard to decide whether orthogonal polyhedra will admit a parallel morph, by reducing from the problem of deciding parallel morphability for orthogonal drawings in $\mathbb{R}^3$ (Theorem 7.3.1). For example, we might devise a reduction in which edges and vertices of a pair of orthogonal parallel drawings in $\mathbb{R}^3$ are "thickened" to boxes, in a manner reminiscent of the procedure used in the proof of Theorem 6.4.2 to convert unmorphable orthodisks to orthogonal polyhedra.

Such a reduction is somewhat unsatisfying, since it creates polyhedra of arbitrarily high genus. It is more interesting that deciding parallel morphability for orthogonal polyhedra remains PSPACE-hard even for polyhedra of genus-0, i.e. topological spheres.

Theorem 7.4.1 *It is PSPACE-hard to decide whether a given pair of parallel orthogonal genus-0 polyhedra will admit a parallel morph.*

Proof: We reduce from the problem of deciding parallel morphability for orthogonal drawings in the plane with static vertices, such that all static vertices have distinct x- and

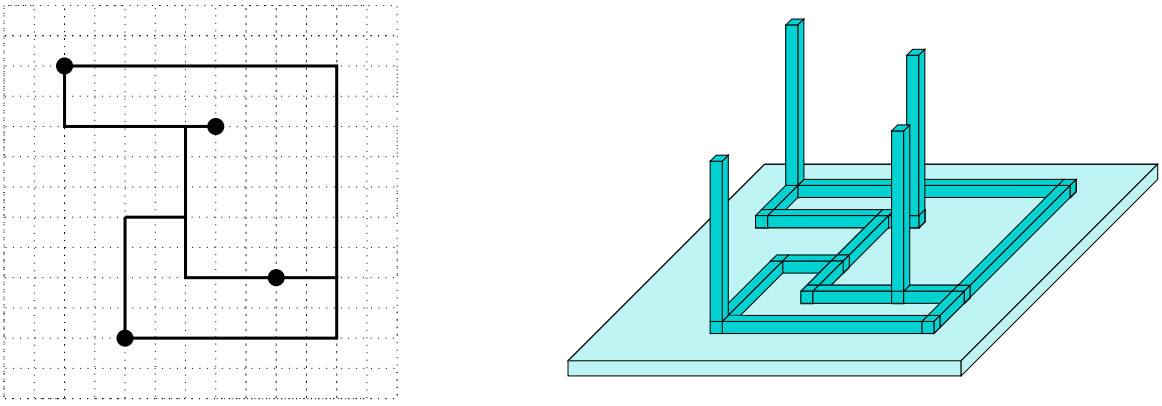

Figure 7.8: The construction of P^* (right) from P (left). The dots in P represent the static vertices.

y-coordinates (Theorem 7.2.3). Let us assume we are given a pair P and Q of parallel orthogonal drawings of a connected graph G in the plane, and a subset $V_{\mathrm{S}} \subset \mathrm{V}(G)$, such that no two vertices of V_{S} share a common x- or y-coordinate. We generate parallel orthogonal polyhedra P^* and Q^* that admit a parallel morph if and only if P and Q admit a parallel morph that keeps the vertices of V_{S} static.

Below, we describe only the generation of P^*. Q^* is generated using a similar procedure. Without loss of generality, assume that all vertices of P and Q lie at integer coordinates in the range $\{1, \ldots, m\}$.

It is convenient to describe P^* in terms of a set of solid closed axis-aligned boxes. We begin our construction with the solid axis-aligned box that is defined by the set

$$\{(x, y, z) \mid x, y \in [0, m+1], z \in [-\epsilon, 0]\},$$

where ϵ is some small rational value such that $0 < \epsilon < 1/100$. We call this box the *base*.

For every vertex $v \in \mathrm{V}(G)$, we stack an $\epsilon \times \epsilon \times \epsilon$ box onto the base, defined by the set

$$\{(x, y, z) \mid x - P_{\mathrm{x}}(v) \in [-\epsilon/2, \epsilon/2],\ y - P_{\mathrm{y}}(v) \in [-\epsilon/2, \epsilon/2],\ z \in [0, \epsilon]\}.$$

For every vertex $v \in V_{\mathrm{S}}$, we include in the set of boxes an $\epsilon \times \epsilon \times 1$ box defined by

$$\{(x, y, z) \mid x - P_{\mathrm{x}}(v) \in [-\epsilon/2, \epsilon/2],\ y - P_{\mathrm{y}}(v) \in [-\epsilon/2, \epsilon/2],\ z \in [\epsilon, 1]\}.$$

For every $\{u, v\} \in \mathrm{E}(G)$, we stack a box on the base with an $\epsilon \times \epsilon$ cross section, whose square faces lie in planes perpendicular to either the x- or y-axis, such that this box serves to connect the pair of $\epsilon \times \epsilon \times \epsilon$ boxes corresponding to u and v; see Figure 7.8. Note that not all edges are to scale in this figure.

We take P^* to be the boundary of the union of the solid boxes defined above, including all edges and vertices of the boxes that lie on the boundary. Coincident edges and vertices that belong to multiple solid boxes of the set become single edges and vertices in P^*.

There is a slight problem with this construction: we have defined polyhedra as objects with a connected edge graph, but one face of our construction has a hole in it. The ill-behaved face is the topmost face of the base that corresponds to the exterior face of P. This can be remedied by cutting away from the (solid) base all points except those with x-y coordinates that belong to closed interior faces of P. After constructing P^* from this modified set of boxes, the graph underlying P^* is connected.

Call the portion of P^* formed by the boxes that correspond with static vertices of V_S, the *vertex towers*. By our construction, there is a well defined ordering to these vertex towers with respect to the x- and y-axes. If a parallel morph of P^* contains an intermediate drawing in which two vertex towers are intersected by a plane that is perpendicular to either the x- or y-axis, then the ordering is no longer well defined. We say that such a morph changes the *ordering* of the vertex towers.

Claim 7.4.1 *There exists a parallel morph from P to Q that keeps the vertices of V_S static if and only if there exists a parallel morph from P^* to Q^* in which the ordering of the vertex towers does not change.*

Proof: Assume that there exists a parallel morph from P^* to Q^* in which the x- and y-orderings of the vertex towers do not change. In this case, there exists a parallel morph from P to Q in which the x-y coordinates of each vertex is determined by the center of the corresponding box. In this morph, the ordering of the vertices of V_{S} is preserved. By Lemma 7.2.5, there exists a parallel morph from P to Q that keeps the vertices of V_{S} static.

Conversely, assume that there exists a parallel morph from P to Q that keeps the vertices of V_{S} static. We can use a technique similar to the one used in the proof of Corollary 7.2.3 to show that there exists a parallel morph from P^* to Q^* in which the ordering of the vertex towers does not change. □

As we have defined P^* and Q^*, there is nothing to prevent the ordering of the vertex towers from changing. Below, we describe how to augment P^* and Q^* with the addition of a structure between each pair of towers. This structure prevents the ordering of the two vertex towers from changing. In particular, the structure that we describe ensures that one tower will always have smaller x-coordinates and larger y-coordinates than the other throughout a parallel morph. A structure that keeps one vertex tower at smaller x-coordinates and smaller y-coordinates than the other can be constructed similarly.

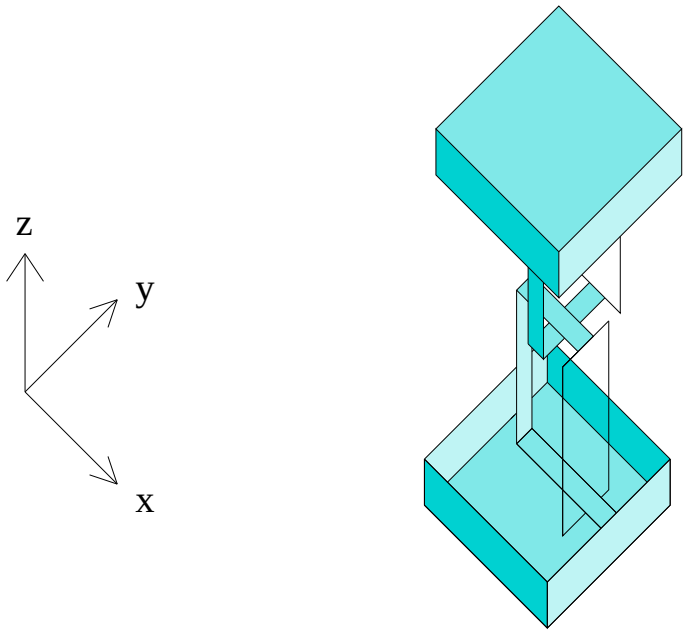

Figure 7.9: Linked structures that remain linked throughout every parallel morph.

The structures to be added to P^* and Q^* are based on the unmorphable orthodisks described in the proof of Theorem 6.4.2 (page 166). The structure shown in Figure 6.8 (upper-left) on page 166 is nearly an orthodisk, except that there are extra edges and vertices that are not incident on a face (i.e. the *ring*). Let us make a copy of this structure. Rotate and translate this copy to obtain the "linked" pair of structures shown in Figure 7.9.

The linked structure illustrated in Figure 7.9 can be turned into a pair of orthodisks in the manner described in the proof of Theorem 6.4.2 with the addition of *gloves*. Observe that in any parallel morph, the orthodisks remain *linked* in the sense that in every intermediate drawing of a parallel morph there exists a line parallel with the z-axis that intersects the *loops* of both orthodisks.

Let us consider how to incorporate these orthodisks into P^* and Q^*, starting with the lower orthodisk of Figure 7.9. Figure 7.10 illustrates this orthodisk (modulo the glove) when converted to a polyhedron that is connected to a vertex tower.

A similar structure is generated from the upper orthodisk. We obtain a polyhedron similar to the one illustrated in Figure 7.11. (Again, the portion of the polyhedron generated from the gloves is not shown.) Following any parallel morph of this polyhedron, there exists a line parallel with the z-axis that intersects both rings. From this fact, it is clear that every parallel morph preserves the ordering of the two towers shown.

Returning to our polyhedra P^* and Q^*, in both polyhedra, we add the linked structures

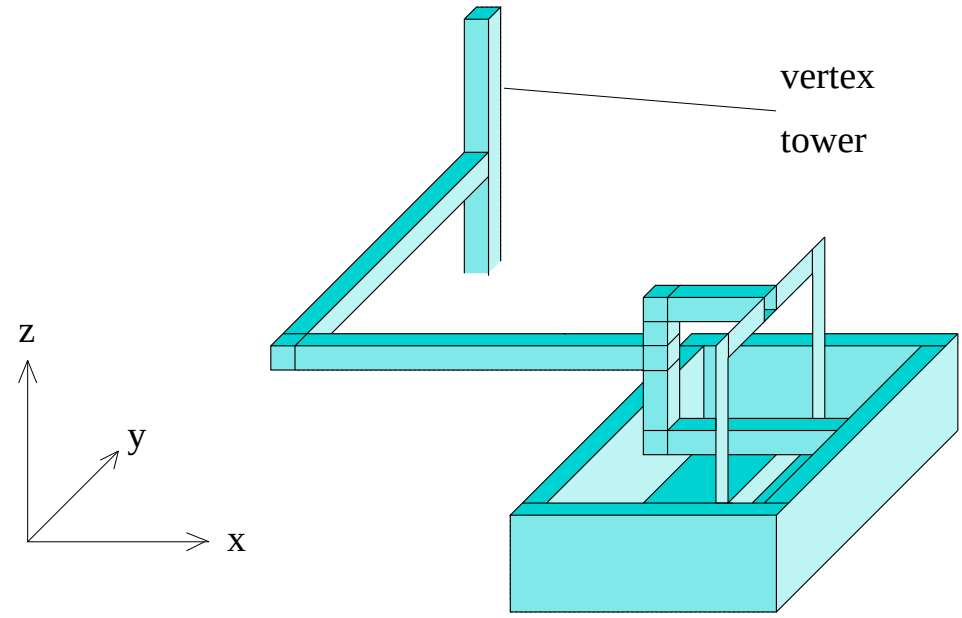

Figure 7.10: The lower orthodisk, converted to a polyhedron and connected to a vertex tower.

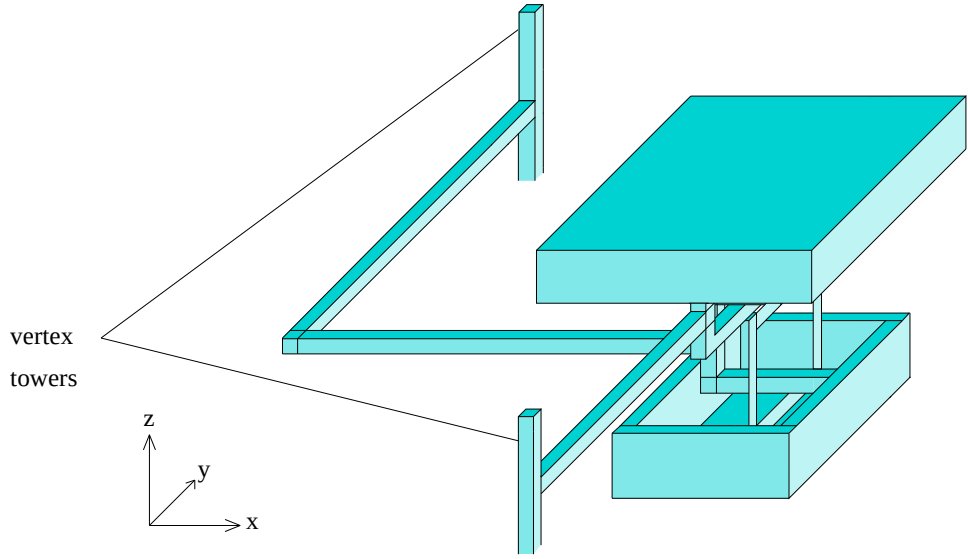

Figure 7.11: The linked structure between two vertex towers.

described above between every pair of vertex towers. We can keep these structures disjoint by varying the location of these with respect to the z-axis. Then, in every parallel morph from P^* to Q^* preserves the ordering of the vertex towers. The complexity of generating the polyhedra is polynomial in the complexity of P and Q (and V_S). Therefore, by Claim 7.4.1

it is PSPACE-hard to decide whether parallel orthogonal polyhedra will admit a parallel morph. □

7.5 Conclusion

In this chapter we proved PSPACE-hardness for problems involving parallel morphing. The first problem we proved to be PSPACE-hard is the problem of deciding whether a pair of parallel orthogonal drawings in the plane admit a parallel morph under the constraint that some given subset of vertices is required to remain static throughout the morph. We performed reductions from this problem to the problem of deciding whether a pair of parallel orthogonal drawings in $\mathbb{R}^3$ admit a parallel morph; and, also to the problem of deciding whether a pair of parallel orthogonal genus-0 polyhedra admit a parallel morph.

It remains an open problem to determine the computational complexity of parallel morphing where the input consists of a pair of parallel (orthogonal) drawings of a cycle in $\mathbb{R}^3$. If the problem turns out to be PSPACE-hard, we expect that the results and techniques described in this chapter to be of some use in devising a proof.

It also remains open to determine whether the PSPACE-hard problems described in this chapter are in PSPACE.

Chapter 8

Conclusion

In this thesis, we have explored a number of problems related to parallel morphing:

- In Chapter 2, we discussed properties of linear morphs. We gave an (inefficient) algorithm to decide the existence of a parallel morph between a pair of parallel simple orthogonal drawings in $\mathbb{R}^d$. We proved that a linear morph is a parallel morph between a pair of parallel polygons in the case that one of the polygons is star-shaped. We can decide whether two polygons will admit a parallel morph via an intermediate star-shaped polygon by solving a system of linear equalities and inequalities.

- In Chapter 3, we studied parallel morphing between parallel orthogonal drawings in the plane. We established that every pair of such drawings of a connected graph will admit a parallel morph, and derived upper and lower bounds on the number of linear morphs required by a parallel morph. We showed how to deal with drawings of disconnected graphs. As well, we discussed area requirements of such a morph.

- In Chapter 4, we studied problems related to monotone and bimonotone morphing. We proved that parallel orthogonally convex polygons will admit a monotone morph, and that every pair of drawings that admits a parallel morph will admit a bimonotone morph.

- In Chapter 5, we proved three problems related to parallel morphing in the plane to be NP-hard, including parallel morphing for non-orthogonal drawings, and monotone morphing.

- In Chapter 6, we showed that parallel simple drawings of a tree graph in $\mathbb{R}^d$ will always admit a parallel morph, but that this is not true of parallel drawings of a

cycle graph in $\mathbb{R}^3$. We gave pairs of parallel orthogonal polyhedra that do not admit a parallel morph.

- In Chapter 7, we proved the problem of deciding parallel morphability to be PSPACE-hard in three settings, including in the cases of orthogonal drawings in $\mathbb{R}^3$, and of orthogonal polyhedra.

Parallel drawings of a graph will exhibit an inherent visual similarity. Since a parallel morph will keep all intermediate drawings parallel and simple, it is plausible that a parallel morph will effectively communicate to a user the correspondence between the elements of its source and target drawings. For this reason, we believe parallel morphs may find application in graph visualization, especially in interactive graph drawing systems.

From the point of view of graph visualization, our most useful result seems to be that every pair of parallel simple orthogonal drawings of a connected graph in the plane will admit a parallel morph. That such morphs require an area of moderate size makes this result particularly useful for visualization. On the negative side, it is NP-hard to decide whether there exists a parallel morph between a pair of parallel non-orthogonal drawings in the plane, and PSPACE-hard to decide for orthogonal drawings in $\mathbb{R}^3$.

One avenue for future research is to determine whether one of our algorithms for morphing parallel orthogonal drawings in the plane can be used as a subroutine in an algorithm that generates a non-parallel simplicity-preserving morph between a given pair of drawings belonging to some class, such that all intermediate drawings of the morph reside within the class.

In this vein, Lubiw, Petrick and Spriggs [45] have devised an algorithm that generates a simplicity-preserving morph between a given pair of simple drawings of a graph, where vertices are represented by points and edges are represented by orthogonal polylines. The approach is to add bends and perform zigzag elimination to get the drawings into a form in which the polylines representing each edge are parallel in the two drawings. One of our parallel morphing algorithms is then applied.

It would be interesting to determine whether one of our morphing algorithms could be used similarly in an algorithm for morphing between non-orthogonal drawings of a graph. It is likely not difficult to devise such an algorithm for the case that edges are represented by polylines, and we are allowed to add bends to these polylines over the course of the morph. An investigation into the properties of such a morph (e.g. number of linear morphs, number of segments created/destroyed, complexity of intermediate drawings) may prove interesting. What about the the case in which edges are represented by line segments?

On the complexity side, it remains open to determine whether it is NP- or PSPACE-hard to decide whether a pair of parallel simple orthogonal drawings of a cycle graph will admit a parallel morph.

Bibliography

[1] H. Alt and L. Guibas. Discrete geometric shapes: matching, interpolation, and approximation. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 121–153. Elsevier Science Publishers B.V. North-Holland, Amsterdam, 1999.

[2] N. M. Amato, M. T. Goodrich, and E. A. Ramos. A randomized algorithm for triangulating a simple polygon in linear time. *Discrete and Computational Geometry*, 26(2):233–244, 2001.

[3] B. Aronov, R. Seidel, and D. Souvaine. On compatible triangulations of simple polygons. *Computational Geometry: Theory and Applications*, 3:27–35, 1992.

[4] L. Auslander and S. V. Parter. On imbedding graphs in the plane. *J. Math. and Mech.*, 10:517–523, 1961.

[5] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Upper Saddle River, New Jersey: Prentice Hall, 1999.

[6] G. Di Battista and R. Tamassia. Algorithms for plane representations of acyclic graphs. *Theoretical Computer Science*, 61:175–198, 1988.

[7] G. Di Battista, R. Tamassia, and I. G. Tollis. Constrained visibility representations of graphs. *Information Processing Letters*, 41:1–7, 1992.

[8] T. Biedl, E. Demaine, M. Demaine, S. Lazard, A. Lubiw, J. O'Rourke, S. Robbins, I. Streinu, G. Toussaint, and S. Whitesides. On reconfiguring tree linkages: Trees can lock. *Discrete Applied Mathematics*, 117:293–297, 2002.

[9] T. Biedl, A. Lubiw, and M. J. Spriggs. Parallel morphing of trees and cycles. In *Proc. 15th Annual Canadian Conference on Computational Geometry (CCCG)*, pages 29–34, 2003. *www.cccg.ca*.

[10] K. Booth and G. Lueker. Testing for the consecutive ones property interval graphs and graph planarity using pq-tree algorithms. *J. Comput. Syst. Sci.*, 13:335–379, 1976.

[11] U. Brandes, M. Güdemann, and D. Wagner. Fully dynamic orthogonal graph layout for interactive systems. Technical report, Universität Konstanz, 2000.

[12] S. Bridgeman and R. Tamassia. Difference metrics for interative orthogonal graph drawing. *Journal of Graph Algorithms and Applications*, 4(3):47–74, 2000.

[13] S. S. Cairns. Deformations of plane rectilinear complexes. *American Math. Monthly*, 51:247–252, 1944.

[14] J. H. Cantarella, E. D. Demaine, H. N. Iben, and J. F. O'Brien. An energy-driven approach to linkage unfolding. In *Proceedings of the 20th ACM Symposium on Computational Geometry (SoCG)*, pages 134–143, 2004.

[15] B. Chazelle. Triangulating a simple polygon in linear time. *Discrete and Computational Geometry*, 6(5):485–524, 1991.

[16] M. Closson, S. Gartshore, J. Johansen, and S. K. Wismath. Fully dynamic 3-dimensional orthogonal graph drawing. *Journal of Graph Algorithms and Applications*, 5(2):1–34, 2000.

[17] R. Cocan and J. O'Rourke. Polygonal chains cannot lock in 4d. *Discrete & Computational Geometry*, 20(3):105–129, 2001.

[18] R. Connelly, E. D. Demaine, and G. Rote. Straightening polygonal arcs and convexifying polygonal cycles. *Discrete and Computational Geometry*, 30(2):205–239, 2003.

[19] G. Das and G. Narasimhan. Geometric searching and link distance (extended abstract). In Frank K. H. A. Dehne, Jörg-Rüdiger Sack, and Nicola Santoro, editors, *WADS*, volume 519 of *Lecture Notes in Computer Science*, pages 261–272. Springer, 1991.

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

[21] H. de Fraysseix and P. Rosenstiehl. A depth-first-search characterization of planarity. *Ann. Discrete Math*, 13:75–80, 1982.

[22] R. Diestel. *Graph Theory*. Heidelberg: Springer-Verlag, 2000.

[23] A. Efrat, S. Har-Peled, L. J. Guibas, and T. M. Murali. Morphing between polylines. In *Proceedings of the 12th ACM-SIAM Symposium on Discrete Algorithms*, pages 680–689, 2001.

[24] S. Even and R. E. Tarjan. Computing an st-numbering. *Theoret. Comp. Sci.*, 2:339–344, 1976.

[25] I. Fary. On straight lines representation of planar graphs. *Acta Sci. Math. Szeged*, 11:229–233, 1948.

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

[27] U. Fößmeier and M. Kaufmann. *Drawing high degree graphs with low bend numbers*, volume 1027 of *Lecture Notes in Computer Science: Graph Drawing (Proc. GD'95)/edited by F. J. Brandenburg*, pages 254–266. Springer-Verlag, 1996.

[28] C. Friedrich and P. Eades. Graph drawing in motion. *Journal of Graph Algorithms and Applications*, 6(3):353–370, 2002.

[29] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[30] A. Garg and R. Tamassia. *On the computational complexity of upward and rectilinear planarity testing*, volume 894 of *Lecture Notes in Computer Science: Graph Drawing (Proc. GD'94)/edited by R. Tamassia and I. G. Tollis*, pages 286–297. Springer-Verlag, 1995.

[31] J. Gomes, L. Darsa, B. Costa, and L. Velho. *Warping and Morphing of Graphical Objects*. Morgan Kaufmann, 1999.

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

[33] U. Grenander, Y. Chow, and D. M. Keenan. *Hands: A Pattern Theoretic Study of Biological Shapes (Appendix D)*. Springer-Verlag, 1991.

[34] L. Guibas, J. Hershberger, and S. Suri. Morphing simple polygons. *Discrete and Computational Geometry*, 24(1):1–34, 2000.

[35] L. J. Guibas and J. Hershberger. Morphing simple polygons. In *Proc. 10th Annual Symposium on Computational Geometry*, pages 267–276, 1994.

[36] J. Hass, J. C. Lagarias, and N. Pippenger. The computational complexity of knot and link problems. *Journal of the ACM*, 46(2):185–211, 1999.

[37] R. A. Hearn and E. D. Demaine. PSPACE-completeness of sliding block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theoretical Computer Science*, 343(1-2):72–96, 2005.

[38] J. Hershberger and S. Suri. Morphing binary trees. In *Proc. 6th ACM-SIAM Symposium on Discrete Algorithms*, pages 396–404, 1995.

[39] J. Hopcroft and R. E. Tarjan. Efficient planarity testing. *Journal of the ACM*, 21(4):549–568, 1974.

[40] J. E. Hopcroft, J. T. Schwartz, and M. Sharir. On the complexity of motion planning for multiple independent objects: PSPACE-hardness of the "warehouseman's problem". *International Journal of Robotics Research*, 3(4):76–88, 1984.

[41] H. N. Iben, J. F. O'Brien, and E. D. Demaine. Refolding planar polygons. In *Proceedings of the 22th ACM Symposium on Computational Geometry (SoCG)*, pages 71–79, 2006.

[42] G. Kant. A more compact visibility representation. In *Proceedings 19th Int'l Workshop Graph-Theoretic Concepts Comput. Sci.*, 1993.

[43] G. Kant. Drawing planar graphs using the canonical ordering. *Algorithmica*, 16(1):4–32, 1996.

[44] N. Karmarkar. A new polynomial time algorithm for linear programming. *Combinatorica*, 4(4):373–395, 1984.

[45] A. Lubiw, M. Petrick, and M. J. Spriggs. Morphing orthogonal planar graph drawings. In *Proc. 17th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 222–230, 2006.

[46] A. Marzal and E. Vidal. Computation of normalized edit distance and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9):926–932, 1993.

[47] K. Misue, P. Eades, W. Lai, and K. Sugiyama. Layout adjustment and the mental map. *Journal of Visual Languages & Computing*, 6(2):183–210, 1995.

[48] J. O'Rourke. *Computational Geometry in C (2nd edition)*. Cambridge University Press, 1998.

[49] A. Papakostas and I. G. Tollis. Interactive orthogonal graph drawing. *IEEE Transactions on Computers*, 47(11):1297–1309, 1998.

[50] V. V. Prasolov and A. B. Sossinsky. *Knots, Links, Braids and 3-manifolds: an Introduction to the New Invariants in Low-Dimensional Topology*. American Mathematical Society, 2000.

[51] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction (Monographs in Computer Science)*. Springer, 1985.

[52] P. Rosenstiehl and R. E. Tarjan. Rectilinear planar layouts and bipolar orientations of planar graphs. *Discrete Comput. Geom.*, 1(4):343–353, 1986.

[53] W. Schnyder. Embedding planar graphs on the grid. In *Proceedings of the 1st ACM-SIAM Symposium on Discrete Algorithms*, pages 138–148, 1990.

[54] B. Servatius and W. Whiteley. Constructing plane configurations in computer-aided design: combinatorics of directions and lengths. *SIAM J. Discrete Math*, 12, No. 1:136–153, 1999.

[55] V. Shurazhsky and C. Gotsman. Controllable morphing of compatible planar triangulations. *ACM Transactions on Graphics*, 20(4):203–231, 2001.

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

[57] I. Streinu. A combinatorial approach to planar non-colliding robot arm motion planning. In *41st ACM Annual Symposium on Foundations of Computer Science (FOCS)*, pages 443–453, 2000.

[58] I. Streinu. Parallel-redrawing mechanisms, pseudo-triangulations and kinetic planar graphs. In *Proc. 13th Int'l Conf. on Graph Drawing, LNCS 3843*, pages 421–433, 2005.

[59] R. Tamassia. New layout techniques for entity relationship diagrams. In *Proceedings of the 4th Int'l Conf. on Entity Relationship Approach*, pages 304–311, 1985.

[60] R. Tamassia. On embedding a graph in the grid with the minimum number of bends. *SIAM Journal of Computing*, 16:421–444, 1987.

[61] R. Tamassia, G. Di Battista, and C. Batini. Automatic graph drawing and readability of diagrams. *IEEE Trans. Syst. Man Cybern.*, 1:61–79, 1988.

[62] R. Tamassia and G. Liotta. *Graph Drawing*, chapter 52, pages 1163–1185. Handbook of Discrete and Computational Geometry/edited by J.E Goodman and J. O'Rourke. CRC Press, USA, second edition, 2004.

[63] R. Tamassia and I. G. Tollis. A unified approach to visibility representations of planar graphs. *Discrete Comput. Geom.*, 1(4):321–341, 1986.

[64] R. Tamassia and I. G. Tollis. Tessellation representations of planar graphs. In *Proceedings 27th Allerton Conf. Commun. Control Comput.*, pages 48–57, 1989.

[65] R. Tamassia, I. G. Tollis, and J. S. Vitter. Lower bounds and parallel algorithms for planar orthogonal grid drawings. In *Proceedings IEEE Symposium on Parallel and Distributed Processing*, pages 386–393, 1991.

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

[67] W. T. Tutte. Convex representations of graphs. In *Proceedings London Mathematical Society*, volume 10(3), pages 304–320, 1960.

[68] W. T. Tutte. How to draw a graph. In *Proceedings London Mathematical Society*, volume 13(3), pages 743–768, 1963.

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

[70] W.Whiteley. Parallel redrawing of configurations in 3-space. *preprint, Dept. of Mathematics and Statistics, York University, Ontario*, 1986.