Graph Morphing via Orthogonal Box Drawings

by

Jack Spalding-Jamieson

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2023

© Jack Spalding-Jamieson 2023

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

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

Statement of Contributions

I would like to acknowledge the names of my co-authors who contributed to the research described in this dissertation. These include:

- my supervisors, Prof. Therese Biedl and Prof. Anna Lubiw

Abstract

A graph is a set of vertices, with some pairwise connections given by a set of edges. A graph drawing, such as a node-link diagram, visualizes a graph with geometric features. One of the most common forms of a graph drawings are straight-line point drawings, which represent each vertex with a point and each edge with a line segment connecting its relevant points, and poly-line point drawings, which more generally allow edges to be represented by poly-lines. Of particular interest to this work are planar straight-line drawings and planar poly-line drawings, in which no two vertices share a location, and no two edges cross (except at shared endpoints).

We study the morphing problem for planar drawings: Given two planar drawings of the same graph, can we output a continuous transformation (a "morph") from one to the other, such that each intermediate drawing is also a planar drawing? It is quite easy to test if a morph exists, but the test is non-constructive. We are interested in the problem of constructing morphs with simple representations. Specifically, we study sequences of linear morphs, which represent the overall morph with a sequence of drawings, so that each pair of adjacent drawings in the sequence can be linearly interpolated. Each drawing in the sequence is called an "explicit" intermediate drawing, since it given explicitly in the output.

Previous work has shown that a pair of straight-line drawings of an n-vertex graph can be morphed using $O(n)$ linear morphs, so that every explicit intermediate drawing is a straight-line drawing. We show that an additional constraint can be added, at the cost of a small tradeoff: We further restrict the explicit intermediate drawings to lie on an $O(n) \times O(n)$ grid, while allowing them to be poly-line drawings with $O(1)$ bends per edge. Additionally, we give an algorithm that computes this sequence in $O(n^2)$ time, which is known to be tight. Our methods involve morphing another class of drawings—orthogonal box drawings—which represent each vertex with an axis-aligned rectangle, and each edge with an orthogonal poly-line. Our methods for morphing orthogonal box drawings make use of methods known for morphing orthogonal point drawings, which are poly-line drawings that restrict each poly-line to use only axis-aligned line segments.

Acknowledgements

I would like to thank my supervisors Prof. Anna Lubiw and Prof. Therese Biedl for their patience and guidance, particularly during the writing of my thesis.

I thank Prof. William Evans, Prof. Bruce Shepherd and Prof. Alan Hu for aiding me to get involved with Theoretical Computer Science research during my undergraduate studies.

I would also like to thank my friends David Zheng, Tam Nguyen, Muggy Li, Eugene Shen, Carolyn Shen, and Colin Chen for their support during my studies.

Table of Contents

List of Figures

Chapter 1

Introduction

Graphs are an extensively studied mathematical model for all sorts of real life objects, such as train networks, social links among groups of people, molecules, and integrated circuits. A graph consists of a set of vertices (e.g., train stations, people, atoms, circuit components) and a set of edges which connect pairs of vertices (e.g., routes between train stations, pairs of friends, chemical bonds, wires between components). It is also often assumed that the graph is a ***simple graph***, meaning that each edge connects a pair of distinct vertices, and no two edges connect the same pair of vertices.

Graphs can be visualized with all sorts of diagrams, and in particular ***graph drawing*** is the field which studies these diagrams. In the field of graph drawing, one of the primary research topics is the topic of ***planar graph drawings***. These are a special form of two-dimensional node-link diagrams. A node-link diagram is a geometric representation of a graph with explicit shapes (e.g., points, circles, rectangles) representing vertices, and simple curves representing edges joining pairs of these points. It is a planar graph drawing if no two vertex shapes intersect, and no two edge curves intersect except at a common endpoint.

A ***morph*** between a pair of planar graph drawings of the same graph (with the correspondence of vertices) is a continuous transformation between them. Of particular interest are ***planarity-preserving morphs*** (also called morphs that ***preserve planarity***), where the continuous transformation preserves planarity throughout. It should be noted that morphs that are not planarity-preserving are not interesting, since such morphs can be typically be achieved by simply interpolating between the coordinates of the shapes/curves in each of the two drawings (assuming some kind of interpolation is possible). The field of graph morphing studies constructions for planarity-preserving morphs, and it has applica-

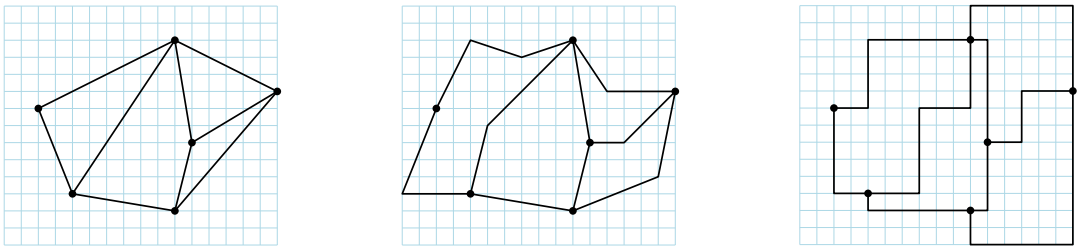
tions in animation and computer graphics [26].

A ***face*** of a planar graph drawing D is a maximal connected open region of $\mathbb{R}^2 \setminus D$ and is identified by the clockwise cycle of edges in D forming its boundary in $\mathbb{R}^2$. The ***outer face*** of a planar graph drawing is the unique face with unbounded area. A necessary condition for a planarity-preserving morph to exist between two planar graph drawings is that the drawings are ***compatible***, i.e., they have the same set of faces and the same outer face. Such pairs of drawings are also sometimes called "topologically equivalent" or "isomorphic" (if the underlying graph is connected). In fact, all pairs of compatible planar graph drawings have a planarity-preserving morph between them (for related discussion, see Diestel's graph theory textbook [19, Ch 4.3]). Furthermore, for a connected graph, it can be tested in linear time whether a pair of drawings are compatible by checking that the faces are made up of the same sets of vertices and edges, and that the outer faces in particular match. However, this test does not give a morph on its own. Explicitly constructing a morph which can be rendered and animated requires more work, which is the topic of this thesis.

1.1 Background and Related Work

Three well-studied types of planar graph drawings are planar straight-line drawings, planar poly-line drawings, and planar orthogonal point drawings. All of these represent vertices with points, and each has a restriction on the types of curves used to represent edges: A ***straight-line drawing*** uses a single line segment for each edge, a ***poly-line drawing*** uses a poly-line for each edge (a non-self-intersecting path made up of line segments, whose non-terminal endpoints are called ***bends***), and an ***orthogonal point drawing*** (sometimes called an "orthogonal drawing") uses orthogonal poly-lines (only horizontal and vertical line segments in each edge path). See Figure 1.1 for some examples of these types of drawings.

Straight-line drawings, poly-line drawings, and orthogonal point drawings all represent vertices with points. For this reason, such drawings are sometimes called ***point drawings***, but other types of drawings where vertices are represented by other classes of shapes exist too. For example, planar ***flat orthogonal drawings***, used by Biedl [6], represent vertices with horizontal line segments, and represent edges with orthogonal poly-lines that intersect exactly two vertex representations—one at each endpoint. Another important example are planar ***orthogonal box drawings***, which represent vertices with axis-aligned rectangles with non-empty interior, and edges with orthogonal poly-lines which again intersect vertex representations at exactly their endpoints. In an orthogonal box drawing, it is additionally



(a) A planar straight-line drawing (b) A planar poly-line drawing (c) A planar orthogonal point drawing

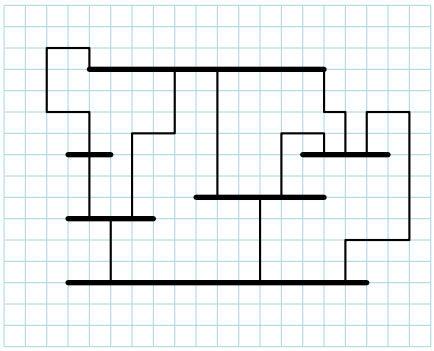
Figure 1.1: Different types of planar graph drawings of the same graph on a 15×17 integer grid.

required that nothing is placed in the interior of any vertex rectangle, and that no two edge poly-lines share an endpoint along a vertex rectangle. The requirement that the interiors of the axis-aligned rectangles are non-empty is not standard in the literature, but it will help simplify the discussion of some results in this work. These types of drawings both use the same class of edges as orthogonal point drawings, but use different vertex representations. See Figure 1.2 for some examples of these types of drawings. We will occasionally use the term ***orthogonal drawing*** when it is clear from context whether the vertices are represented by points or boxes.

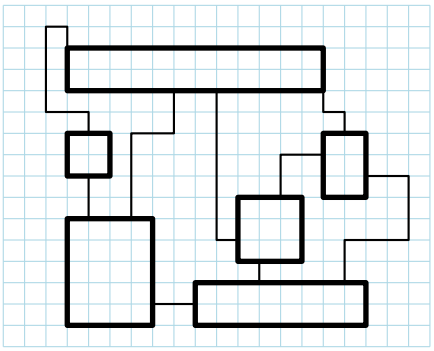
It is known that all planar graphs admit a planar straight-line drawing [49, 22, 39]. Algorithms for computing such drawings are also known, such as Tutte's force-directed drawing algorithm [45]. Moreover, it is known that all planar graphs with n vertices admit a planar straight-line drawing whose vertices are all placed on an $O(n) \times O(n)$ grid [16, 36]. Furthermore, this drawing can be computed in $O(n)$ time [36]. Various similar results exist for the other types of drawings. See [41] for many results from the field of graph drawing.

In contrast to the problem of simply drawing planar graphs, it is an open and ongoing topic of study to determine whether there exist planarity-preserving morphs that maintain the same desirable properties (i.e., straight-line edges, vertices on a grid) for certain intermediate drawings during the morph, while also maintaining certain desirable properties of morphs. Progress made on this question is the topic for the remainder of this background section. Note that from now on, except when otherwise specified, the word 'morph' alone will always be used to refer to planarity-preserving morphs, and constructions of morphs will be accompanied by proofs that they are planarity-preserving when it is appropriate.

There are two broad classes of representations for morphs: Those which only allow the computation of "snapshots" of the drawing at a particular time, and those which can



(a) A planar flat orthogonal drawing



(b) A planar orthogonal box drawing

Figure 1.2: Different types of planar graph drawings of the same graph which are not point drawings, each drawn on a 17×21 grid.

be broken into a number of distinct steps with explicit trajectories for each vertex (see Figure 1.3). The former kind of morph can be thought of as an "implicit" morph, since the full trajectories of the vertices are not known, while the latter can complementarily be thought of as an "explicit" morph. We now clarify the variety of objectives that may be desirable for computing planarity-preserving morphs.

First are the properties of the intermediate drawings at all times throughout the morph. There are many interesting properties for graph drawings, and it is possible to maintain some of them continuously. For example, one can strive to require the simplest possible edge representations throughout the morph (**Objective EdgeSimplicity**), so that ideally the drawing remains a straight-line drawing throughout the morph. If the morph involves distinct steps, then additional properties can be required after each distinct step. In particular, it may be possible to require that after each distinct step, the vertices all lie on a small integer grid (**Objective Grid**).

Third, there are the desirable properties of the morph itself, which may involve the simplicity/explicitness of the trajectories of the vertices (**Objective Trajectories**), or the number of distinct steps if applicable (**Objective StepCount**). In the case of distinct steps, the paths the vertices can take between morph steps should ideally be as simple to represent as possible (also Objective Trajectories), ideally linearly interpolated paths, and the total number of distinct steps should be minimized (Objective StepCount). The computational model and complexity have little effect on the quality of the morph as a visualization, but the properties of the intermediate drawings and the properties of the morph itself can have a drastic effect. If the intermediate drawings are also good drawings, and the paths of the vertices are simple, then it will be easier to follow the morph. Moreover,

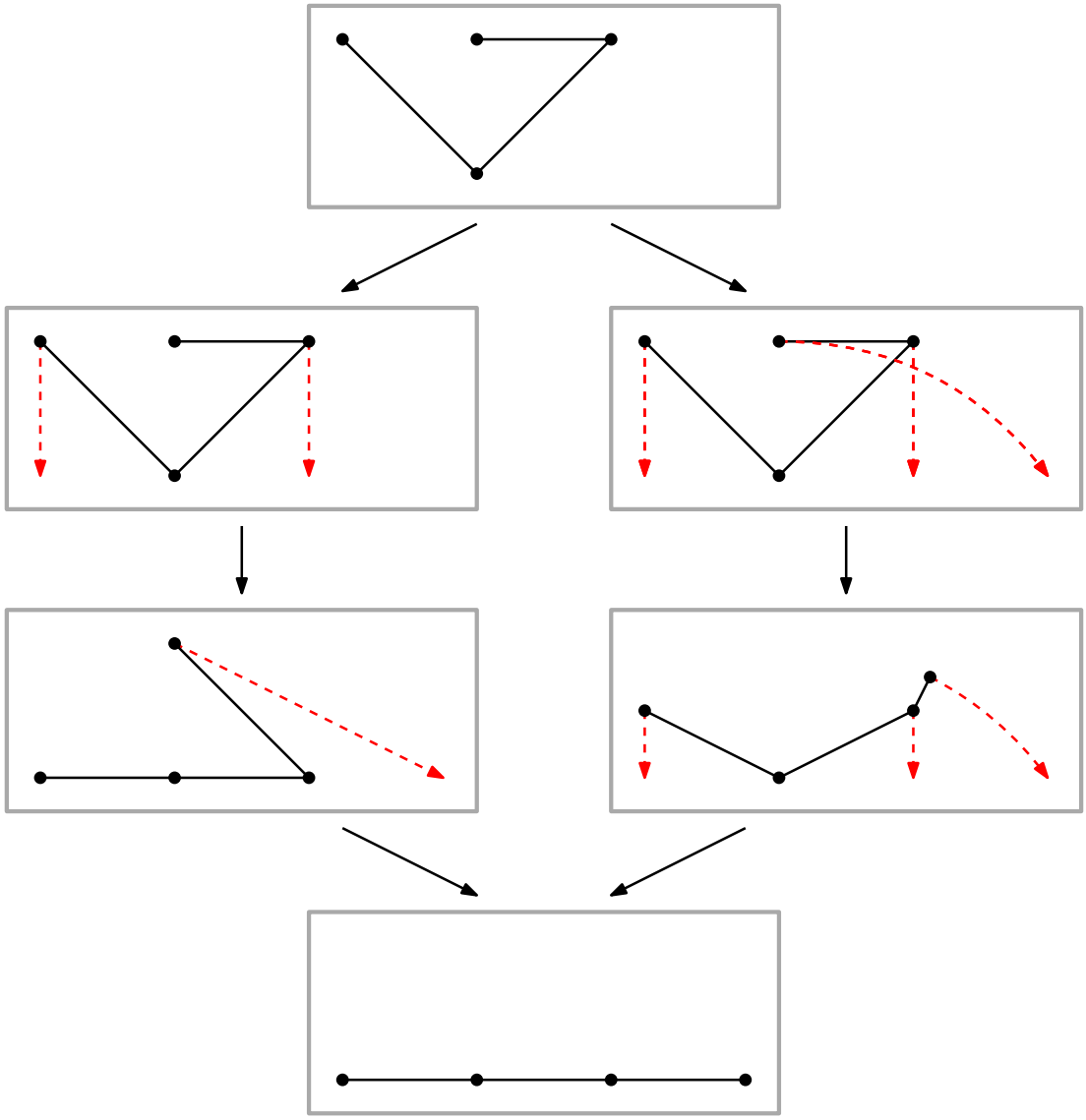


Figure 1.3: Two different types of morphs. The left path divides the morph into distinct simple steps. The right path has the vertices follow complex paths for which snapshots can be computed, where in general it might not be possible to compute the paths ahead of time.

if the morph can be broken into distinct steps, and the number of these steps is small, then an animation of the morph will be shorter.

Fourth, it is of course desirable for a morph technique to work for all planar graphs (**Objective Universality**), but it is sometimes easier or even potentially more interesting to work with a particular strict subclass.

Fifth is the computational model chosen for the computation (**Objective Model**): The types of values which need to be represented (real numbers, rational numbers, integers) and the operations which can be performed on them (e.g., whether square roots are allowed). Two of the most common types of computational models are real RAM and word RAM, which allow the storage of real numbers and integers of size at most $O(\log n)$ (for a graph of size n) respectively. Real RAM may also be endowed with operations beyond standard arithmetic operations (addition/multiplication/subtraction/division), such as the ability to find square roots. Lastly is the computational complexity of a chosen algorithm for computing morphs (**Objective Time**). If the morph involves snapshots, then we consider the algorithm that computes a snapshot. If the morph involves distinct steps of some kind, then we consider the algorithm for computing all steps. In either case, the complexity for the algorithm should be minimized. In the former case, the algorithm for computing a snapshot should ideally take $O(n)$ time. In the latter case, the algorithm for computing all steps should ideally take $O(n \cdot \#\text{steps})$ time. Both of these ideal bounds are simply the respective sizes needed to represent the output itself.

1.1.1 Morphs that Maintain Straight-line Drawings

One of the most well-studied classes of morphs are those that maintain a straight-line drawing at all intermediate times throughout the morph. In other words, satisfying Objective EdgeSimplicity with straight-lines is considered a requirement, Objective Grid is ignored, and Objective StepCount is minimized. A long series of work [8, 9, 44, 24, 27, 40, 3, 2, 4, 1, 30, 31, 21] has derived increasingly more efficient algorithms to compute increasingly shorter and simpler planarity-preserving morphs between compatible planar straight-line drawings. This series of work also simultaneously considers a variety of combinations of objectives Model, Time, Trajectories, and Universality. We briefly review these results here.

In 1944, Cairns [8, 9] gave an algorithm for computing morphs of compatible triangulated (maximal) planar graphs. The main method involves distinct steps which shrink edges to become so small that they are effectively contracted, thereby reducing the number of vertices. This method produces poor visualizations, since most of the detail of the graph

is effectively lost. Furthermore, the number of distinct steps in the resulting morph is exponential. In 1983, Thomassen [44] extended the same arguments to work for compatible straight-line drawings of all planar graphs by finding compatible triangulations.

In 1999, Floater and Gotsman [24] devised a method for performing morphs continuously between compatible straight-line drawings of triangulated (maximal) planar graphs, based on Tutte's drawing algorithm [45]. They were the first to use a snapshot method. Specifically, they produce an algorithm which can, given some time value $t \in [0, 1]$, produce a snapshot of the current drawing in the morph. The algorithm for producing this snapshot takes $O(n^{\omega/2})$ time, where ω is the matrix multiplication exponent, using a 2013 algorithm of Alon and Yuster [3] for solving certain linear systems. However, the computational model required is quite powerful. Square roots are required to compute even the initial weights. Furthermore, snapshot-based methods do not have any sort of guarantees on the quality of the visualizations, since it is not clear which snapshots should be computed, or how many. In 2001, Gotsman and Surazhsky [27, 40] extended this approach to compatible straight-line drawings of all planar graphs using compatible triangulations with Steiner points. In 2021, Di Battista and Frati [18] analyzed the "resolution" of the resulting morphs (essentially the minimum distance between any two features of the drawing throughout the morph), and found that it was exponentially small, even for morphs between drawings each on a small (polynomial-sized) grid.

Recent work often uses smaller morph steps where the vertices follow simple trajectories (i.e., treating the optimization of Objective Trajectories as a hard constraint). Specifically, these morphs consist of distinct steps called ***linear morph***s between adjacent pairs of explicit drawings in the sequence. The continuously changing drawing during a linear morph is defined by a time value $t \in [0, 1]$, so that the position of each vertex is the linear interpolation of its initial and final positions with t as a parameter. Sometimes they are further restricted to ***unidirectional linear morph***s, where all directions of movement (or equivalently, line segments joining initial and final positions) are also assumed to all be parallel. See Figures 1.4 and 1.5 for examples.

One breakthrough paper using linear morphs is a 2017 paper by Alamdari et al. [1] with 13 authors. They prove that between any two compatible planar straight-line drawings of the same graph with n vertices, there is a planarity-preserving morph that uses a sequence of $O(n)$ unidirectional morphs. Moreover, the algorithm runs in $O(n^3)$ time. In 2019, Kleist, Klemz, Lubiw, Schlipf, Staals, and Strash [30] improved this runtime to $O(n^{1+\omega/2} + n^2 \log n)$, where ω is the matrix-multiplication exponent. In 2021, Klemz [31] further improved this runtime to $O(n^2 \log n)$, and $O(n^2)$ if the underlying graph is 2-connected. The algorithm is based on the edge-shrinking techniques of Cairns, so it unfortunately produces poor visualizations for the same reason. Furthermore, the computational model

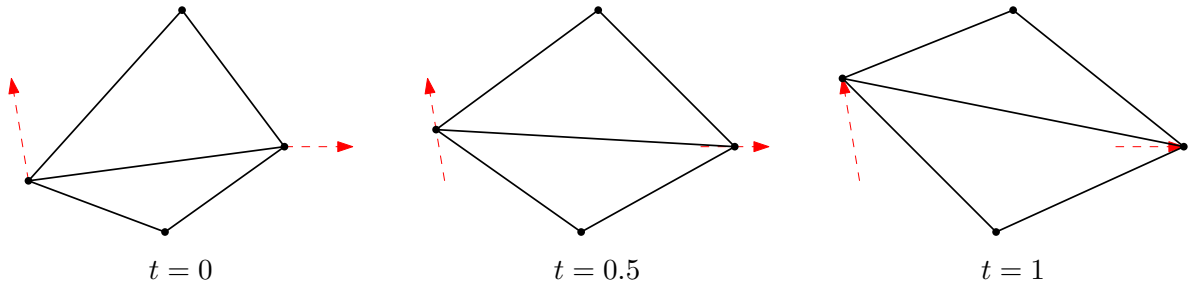


Figure 1.4: An example of a linear morph, at specified times in $t \in [0, 1]$. The vertex trajectories of each vertex are explicitly drawn as red dashed arrows for visualization (they are not part of the drawing itself). The vertices without drawn trajectories do not move.

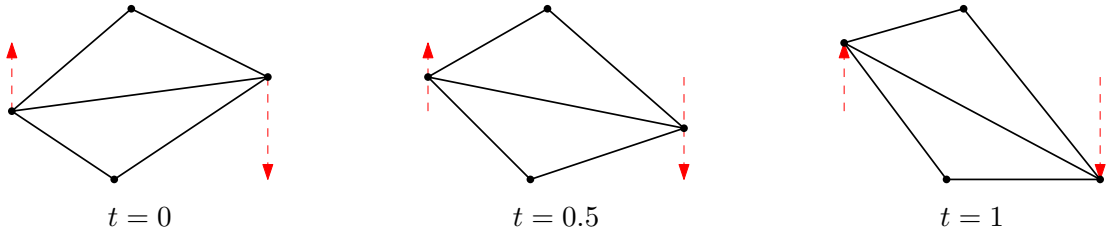


Figure 1.5: An example of a unidirectional linear morph, at specified times in $t \in [0, 1]$. Observe that all vertex trajectories are parallel.

required is the strongest of all results discussed in this section: It requires computing both square and cube roots.

In 2023, Erickson and Lin [21] derived an algorithm which finds sequences of $O(n)$ unidirectional linear morphs between compatible drawings of 3-connected planar graphs with identical convex outer-polygons. It runs in $O(n^{1+\omega/2})$ time, where ω is the matrix multiplication exponent. Their algorithm is quite simple and easy to implement, and is adapted from the framework used by Floater and Gotsman for continuous morphs, rather than the framework used by Cairns. They claim that their morphs are more visually appealing than Alamdari et al.'s morphs, since they do not involve edge contractions. Unfortunately, the resolution of the resulting morphs is also exponentially small, so there is no polynomial-sized grid onto which one could place all of the vertices of the intermediate drawings in between distinct linear morph steps (i.e., no progress for Objective Grid).

Many of the morphs discussed can be separated into distinct linear morph steps, and hence can be represented by giving a sequence of drawings, where a linear morph can be performed for each adjacent pair. The drawings in the sequence are called the ***ex-***

plicit intermediate drawings, since they are explicitly represented, as opposed to the intermediate drawings during each linear morph which have interpolated coordinates. The presence of these drawings is what makes this type of morph "explicit" (as discussed in the previous subsection). Explicit intermediate drawings are analogous to keyframes in more traditional forms of animation. None of the morph techniques discussed so far make any guarantees about whether their explicit intermediate drawings are on a small grid. It is an open question whether there is a sequence of $O(n)$ linear morphs between any two compatible planar straight-line drawings on an $O(n) \times O(n)$ grid, so that all explicit intermediate drawings are also straight-line drawings on an $O(n) \times O(n)$ grid. It is also a (weaker) open question whether such morphs exist with polynomial length and grid-size. However, it is known that there are pairs of drawings on $O(n) \times O(n)$ grids which require a sequence of $\Omega(n)$ linear morphs [1], so a result of $O(n)$ linear morphs on an $O(n) \times O(n)$ grid would be tight. Hence, such a construction would essentially optimize/satisfy objectives EdgeSimplicity, Grid, Trajectories, and StepCount.

1.1.2 Morphs for Alternative Classes of Drawings

One way to weaken the requirements for construction of morphs on a grid is by significantly restricting the class of drawings (relaxing Objective Universality). For a strict subclass of planar straight-line drawings known as weighted Schnyder drawings, Barrera-Cruz, Haxell and Lubiw [5] were able to find sequences of $O(n^2)$ planarity-preserving linear morphs on an $O(n) \times O(n)$ sized integer grid.

Another way to weaken the requirements for construction of morphs on a grid is by allowing the intermediate drawings to be poly-line drawings (thereby relaxing Objective EdgeSimplicity), while also requiring that the newly-added bends of the poly-lines be placed on the same fixed integer grid. For a pair of compatible planar straight-line drawings of a graph with n vertices, in 2011 Lubiw and Petrick [33] derived a construction of $O(n^6)$ linear morphs between poly-line drawings whose edges each have at most $O(n^5)$ bends, and all of whose vertices and bends lie on an $O(n^3) \times O(n^3)$ grid. A standard word RAM model can be used for the algorithm. This particular result is the primary motivation for this thesis, and all of these bounds will be significantly improved.

There are several other classes of drawings that have been studied in the context of morphing (i.e., alternative interesting choices for Objective Universality). One such class is the class of planar orthogonal point drawings. In 2013, Biedl, Lubiw, Petrick and Spriggs [7] showed that between any two compatible planar orthogonal point drawings, there exists a sequence of $O(n^2)$ planarity-preserving linear morphs for which each

intermediate drawing is also an orthogonal point drawing, and furthermore each explicit intermediate drawing is on an $O(n) \times O(n)$ grid. In 2022, Van Goethem, Speckmann, and Verbeek [47] improved their result to a sequence of only $O(n)$ linear morphs, while maintaining at most $O(1)$ bends per edge, although they do not discuss the grid size of their explicit intermediate drawings. They also show that their algorithm uses at most one more than the minimum number of linear morphs. Both algorithms can be implemented on a standard word RAM model.

Another class studied in the context of morphing is the class of "rectangular duals", a contact-representation of a graph with disjoint rectangles whose union must also be a rectangle. Recently, Chaplick, Kindermann, Klawitter, Rutter, and Wolff [11] showed that between any two compatible rectangular duals, there is a sequence of $O(n^2)$ linear morphs which can be found in $O(n^3)$ time, but only by allowing for non-rectangular polygons as intermediate representations during the individual morphs. Rectangular duals are a type of "contact representation", on which there is also a larger body of morphing results mostly concerning the existence of morphs rather than their construction [34]. Interestingly, this body of results additionally relates to the complexity class $\exists\mathbb{R}$, for which many of the resulting morph problems (that is, checking if certain kinds of morphs exist) are hard.

The plane is not the only manifold that admits morphs. Erickson and Lin [21] explored morphs of some straight-line drawings on a flat torus. Since linear interpolation between start and end-points is not well-defined in a torus (at least without specifying a homotopy class), linear morphs as defined cannot be used on such a surface. Instead, Erickson and Lin devise a more specialized method for representing and computing morphs on the torus.

1.2 Goal

For the remainder of this thesis, we will deal with the problem of morphing (simple) planar graphs. It should be noted that an n-vertex planar graph has $O(n)$ edges, which will be important for making many of the bounds we achieve possible.

In this work, it will be shown that between a pair of compatible planar straight-line drawings of a connected graph with n vertices, each drawn on an $O(n) \times O(n)$ grid, there is a planarity-preserving morph consisting of a sequence of $O(n)$ linear morphs, for which every explicit intermediate drawing is a poly-line drawing on a $O(n) \times O(n)$ grid with at most $O(1)$ bends per edge. This morph can be computed with only a word RAM model. In particular, no special operators (such as square roots or cube roots) will be required. Moreover, this construction will be computed in $O(n^2)$ time. This essentially

optimizes every named objective except Objective EdgeSimplicity, which is slightly relaxed (further optimizing this objective is an open problem, as we will briefly discuss in Chapter 9).

This can be seen as a significant improvement over the results of Lubiw and Petrick [33], who obtained $O(n^6)$ linear morphs, an $O(n^3) \times O(n^3)$ grid, and $O(n^5)$ bends per edge, while using a word RAM model. Alternatively, it can be seen as an improvement over the results of Alamdari et al. [1] or Erickson and Lin [21], who obtained $O(n)$ linear morphs on an exponentially large grid, and used a real RAM model, except that our algorithm adds a constant number of bends to each edge during the morph.

Performing sequences of linear morphs with bends sometimes involves adding bends to previously-straight edges. Previous work implicitly added bends as a part of linear morph steps. For additional clarity, in this work, linear morphs will be separated from steps of the morph that introduce bends. Bends which are coincident with other bends, or coincident with vertices, will be permitted. Bends whose two incident edge segments form a 180° angle will also be permitted. These kinds of "invisible" bends are called ***degenerate bends***. A pair of drawings is called ***equivalent*** if the drawings are identical after removing all degenerate bends. In this work, a ***linear morph sequence*** is a sequence of discrete morph steps of two types. The first type is a linear morph between two drawings with the same number of bends along each edge. The second type is the replacement of a drawing with an equivalent drawing that has a different number of bends along some edges. This definition for a linear morph sequence is consistent with previous work, but more explicit. A ***planarity-preserving linear morph sequence*** is simply one where each linear morph step is planarity-preserving.

The following main theorem, with all the desired guarantees, will be proven:

Theorem 1.2.1 (Main)**.** *Let G be a connected planar graph with n vertices. For a compatible pair of planar straight-line drawings P and Q of G, whose vertices lie on an $O(n) \times O(n)$ grid, there exists a planarity-preserving linear morph sequence from P to Q of length $O(n)$, where each explicit intermediate drawing lies on an $O(n) \times O(n)$ grid and has $O(1)$ bends per edge. Moreover, this sequence can be found in $O(n^2)$ time.*

To achieve this result, three phases are used: First, the problem of morphing straight-line drawings is reduced to morphing orthogonal box drawings. Next, the resulting orthogonal box drawings are morphed to become "parallel", in a sense that is somewhat similar to the definition of parallel lines. Finally, these "parallel" drawings are morphed directly. The middle step is the most complex, and involves adapting techniques used for morphing planar orthogonal point drawings by Biedl et al. [7] and Van Goethem et al. [47]. A more

detailed overview can be found in Section 2.5, after some discussion of preliminary results and definitions earlier in Chapter 2.

Chapter 2

Preliminaries

In this chapter, we discuss some key relevant results in graph morphing (Section 2.1, Section 2.2, Section 2.3), discuss how some of these results extend to orthogonal box drawings (Section 2.4), and give a high-level overview of our methods (Section 2.5).

2.1 Drawings, Linear Morphs, and Planar Poly-line Drawings

Let G be a planar graph, and let v be some vertex of G. For any drawing D, let $D(v)$ denote the geometric representation of v in D. In particular, if D is a straight-line drawing, $D(v)$ is a point, while if D is instead an orthogonal box drawing, $D(v)$ is an orthogonal rectangle. Then, given two planar straight-line drawings P and Q of G, let $\boldsymbol{M_{P,Q}}$ denote the linear morph from P to Q (as defined in Chapter 1). In particular, let $M^t_{P,Q}$ denote the (straight-line) drawing at a time $t \in [0, 1]$. Recall that a ***linear morph*** linearly interpolates between the two drawings. That is, $M^0_{P,Q}(v)$ is $P(v)$, $M^1_{P,Q}(v)$ is $Q(v)$, and for arbitrary $t \in [0, 1]$, $M^t_{P,Q}(v)$ is $(1 - t)P(v) + tQ(v)$. Edges in $M^t_{P,Q}$ are straight-line segments between the locations of their endpoints at time t. Recall also that we do not assume a-priori that edges do not cross or go through non-incident endpoints—this must be proven as an additional property (discussed more in the next section).

If P and Q are planar poly-line drawings of a graph G with the same number of bends along each edge, then the linear morph $M_{P,Q}$ also linearly interpolates the position of each bend. The correspondence between the bends in each drawing is given by the ordering of

the bends along each edge. If multiple bends are coincident along an edge, any compatible ordering suffices.

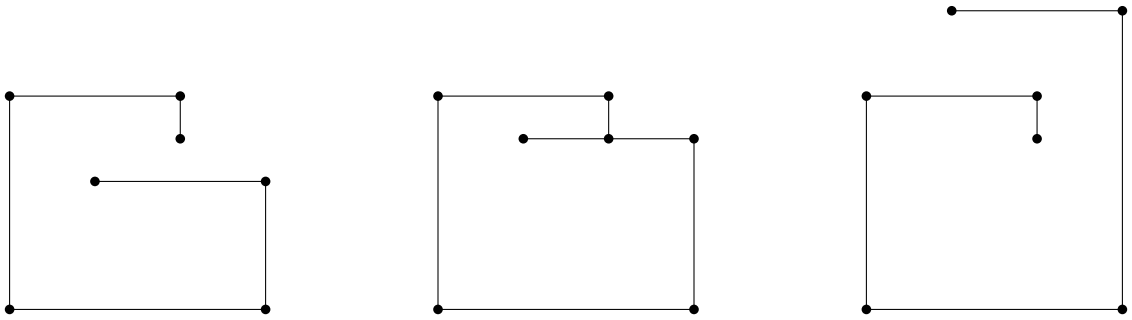
Linear morphs between planar poly-line drawings require that the drawings have the same number of bends along each edge. In order to perform morphs between planar poly-line drawings with different numbers of bends along some edges, we must use additional morph steps to unify the number of bends along each edge. We can accomplish this by substituting equivalent drawings, as discussed in Section 1.2. Importantly, in this work, we will allow our planar poly-line drawings to contain ***degenerate bends***, which either coincide with other bends or vertices along the same edge, or result in an angle of 180° (or both). Given a pair of planar poly-line drawings P, Q of the same graph G, bends can be added (e.g., coincident with vertices) until the resulting respective equivalent drawings P' and Q' have the same number of bends along each edge. In this work, we will take care to describe where and when bends are added.

2.2 Planarity-Preserving Linear Morphs and Sequences of Linear Morphs

Linear morphs are extremely simple to store as a data structure: they store exactly the start and end locations of each vertex and bend. However, not all linear morphs preserve planarity. For a simple example, see Figure 2.1. It is known that certain broad classes of linear morphs always preserve planarity. In particular, simultaneous translations and scales of all vertex/bend coordinates are always planarity-preserving.

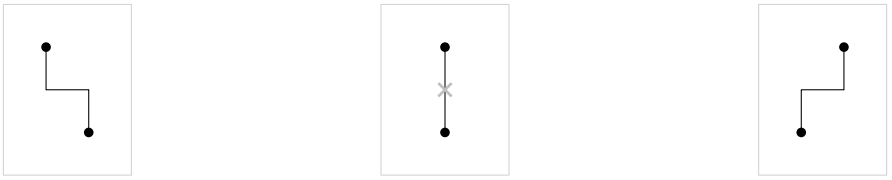
Recall from Chapter 1 that a planarity-preserving linear morph sequence can exist only if the two drawings P and Q are compatible (i.e., same set of faces and same outer face), so we are only concerned with compatible pairs of drawings.

Some care must be taken to define planarity-preserving linear morphs of planar poly-line drawings that permit degenerate bends. Recall that a linear morph $M_{P,Q}$ is considered planarity-preserving if every implicit intermediate drawing $M^t_{P,Q}$ (for $t \in [0, 1]$) is itself planar. When P and Q are planar poly-line drawings that permit degenerate bends, we add one more requirement: For all intermediate times $t \in (0, 1)$, a bend may only coincide with another bend or a vertex if it does so in both P and Q. For example, the linear morphs in Figure 2.2 are not planarity-preserving since a violation of this requirement occurs at time $t = 0.5$ in each. Note that we separately still require that the path through $\mathbb{R}^2$ for any edge contains no self-intersections (i.e., is simple), but that path can be thought

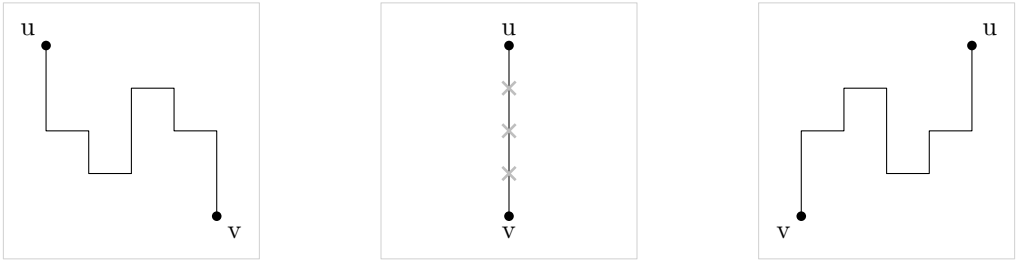


(a) $M^0_{P,Q} = P$ (b) $M^{\frac{1}{4}}_{P,Q}$ (c) $M^1_{P,Q} = Q$

Figure 2.1: An example of a linear morph between two (compatible) planar straight-line drawings P and Q that is not planarity-preserving. The first intersection occurs at $t = \frac{1}{4}$ between a vertex and an edge.



(a) P_1 (b) $M^{0.5}_{P_1,Q_1}$ with coinciding bends marked by a cross. (c) Q_1



(d) P_2 (e) $M^{0.5}_{P_2,Q_2}$ with coinciding bends marked by a cross. (f) Q_2

Figure 2.2: Two pairs of compatible planar poly-line drawings P_1, Q_1 and P_2, Q_2, none of which have degenerate coinciding bends. The linear morphs from P_1 to Q_1 and from P_2 to Q_2 are both not planarity-preserving because at time $t = 0.5$ bends will coincide.

of as ignoring degenerate bends (i.e., the planarity check operates on the drawing with degenerate bends omitted).

We can make a useful (albeit simple) observation about linear morphs that are not planarity-preserving:

Observation 2.2.1. *Let P, Q be planar poly-line drawings of the same graph G with the same number of bends along each edge, where G has no isolated vertices. Assume the linear morph $M_{P,Q}$ from P to Q is* not *planarity-preserving. Let $t^* \in [0,1]$ be the infimum of all times t at which $M^t_{P,Q}$ is not planar. Then t^* is also a minimum, and $t^* \in (0,1)$. Moreover, there exists an edge segment e, and a vertex or bend x not incident to e, such that x and e intersect in $M^{t^*}_{P,Q}$.*

Proof. Since the linear morph from P to Q is not planarity-preserving, there is some time $t \in (0,1)$ such that a pair of distinct edge segments intersect in $M^t_{P,Q}$ at some location that is not a shared endpoint. Note that if there is an intersection between a bend/vertex and an edge in $M^t_{P,Q}$, then there is also an intersection between two edges, since there are no isolated vertices.

Let $K \subset E(G) \times E(G)$ denote the set of distinct edge segment pairs (e, e') that intersect at some point during the linear morph $M_{P,Q}$. For all $(e, e') \in K$, let $t^*(e, e')$ denote the infimum of all times t where e, e' intersect in $M^t_{P,Q}$. Let $t^* = \min_{(e,e') \in K} t^*(e, e')$, and let (e_1, e_2) be a corresponding argument of the minimum, breaking ties arbitrarily (i.e., $t^* = t^*(e_1, e_2)$). Thus, for the remainder of the proof, it suffices to show that e_1 and e_2 intersect at time t^*, and that their intersection at time t^* involves at least one endpoint.

First, we consider the cases where e_1 and e_2 share an endpoint (a shared vertex or bend). In this case, e_1 and e_2 may intersect only when they are collinear, which can only occur at exactly one point in time, which is therefore t^*. At such a time, the shorter of the two edge segments (without loss of generality, e_1) has its unshared endpoint overlapping with the other edge segment (e_2), so the desired intersection exists in this case.

We now consider the case where e_1 and e_2 do not share an endpoint. Let $d : [0,1] \to \mathbb{R}_{\geq 0}$ be the minimum distance function between e_1 and e_2 throughout the morph (that is, $d(t)$ is the minimum distance between e_1 and e_2 at time t). This is a continuous function, since the edges move continuously. Therefore, the inverse image $d^{-1}(0)$ (i.e., the set of times at which e_1 and e_2 intersect) is a closed set, which thus also has a minimum value t^*. This proves the first claim.

If there is more than one intersection point between e_1 and e_2 at time t^*, then e_1 and e_2 must be collinear at time t^*. Since they intersect, and are collinear, there must be at least

two endpoints involved in intersections, so we are done in this case, since we can choose x to be either of the two endpoints involved in an intersection. For the remainder of this proof, we may now assume that e_1 and e_2 intersect at exactly one point in $M_{P,Q}^{t^*}$.

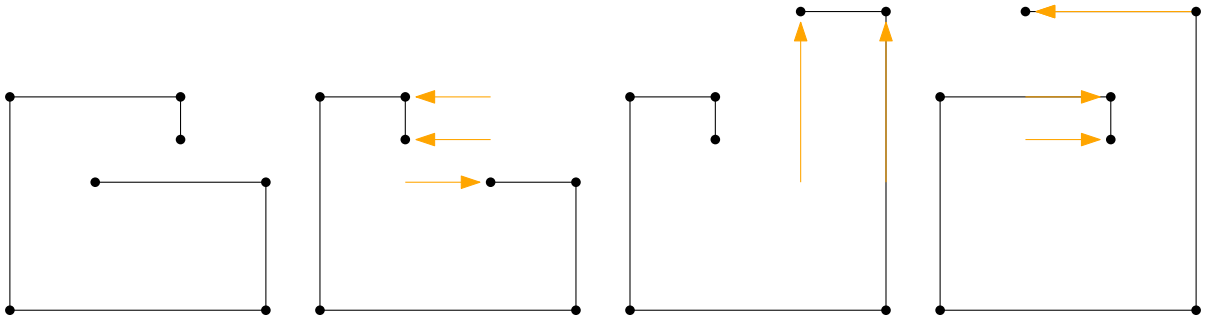
Suppose for contradiction that at t^*, the unique intersection point between e_1 and e_2 is in the strict interiors of each edge (i.e., it does not involve any endpoints). For $t \in [0, t^*]$, let $d'(t)$ denote the infimum distance from an intersection point of e_1 and e_2 to the closest of their endpoints, or 0 if e_1 and e_2 do not intersect (or intersect at more than one point, i.e., if they are parallel and intersecting). Then d' is a continuous function, and $d'(0) = 0$, while $d'(t^*) > 0$. By intermediate value theorem, there exists some $0 < t_0 < t^*$ such that $d'(t_0) = d'(t^*)/2$, contradicting the minimality of t^*. Therefore, at least one endpoint (i.e., bend or vertex) of e_1 or e_2 must be part of the intersection at time t^*, so we can choose x to be one of these endpoints, completing the proof. □

Not all pairs of drawings permit planarity-preserving linear morphs, so work involving linear morphs (including ours) focuses on breaking morphs into sequences of planarity-preserving linear morph steps. Given two planar poly-line drawings P and Q of the same graph, recall that the linear morph from P to Q is only well-defined if the two drawings have the same number of bends along each edge. In this work, unlike in previous works, bends will be explicitly added or removed as a separate step. Individual morph steps must therefore be one of two types: A step that adds or removes bends, or a step which performs a linear morph. As discussed in the previous section, adding or removing bends can be accomplished by replacing a drawing with an equivalent drawing.

As an example, there may be a sequence of drawings $P = D_1, D_2, D_3, D_4 = Q$, where the graphs D_1 and D_2 are equivalent drawings with different numbers of bends along each edge, and the linear morphs from D_2 to D_3 and D_3 to D_4 are planarity-preserving. In this case, D_2, D_3, and D_4 have the same number of bends along each edge. The exact definition of these kinds of sequences is as follows:

Definition 2.2.2. *Let P and Q be planar poly-line drawings of the same graph G. A **linear morph sequence** of length k from P to Q is a sequence of planar poly-line drawings $P = D_1, D_2, \ldots, D_{k+1} = Q$ such that:*

- *Each drawing D_i is a planar poly-line drawing of the graph G.*
- *For each pair of drawings D_i, D_{i+1} ($i \in \{1, \ldots, k\}$), either the two drawings are equivalent drawings, or the two drawings have the same number of bends along each edge.*



(a) $P = D_1$ (b) D_2 (c) D_3 (d) $D_4 = Q$

Figure 2.3: An example of a planarity-preserving linear morph sequence of length 3 between two compatible planar straight-line drawings P and Q. The movement of vertices compared to each previous step is denoted with orange arrows. Note that P and Q are the same as in Figure 2.1, so the linear morph directly from P to Q is not planarity-preserving.

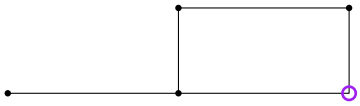
Observe that this definition implies the linear morph from D_i to D_{i+1} is always well-defined, as discussed in the previous section.

Recall also from Chapter 1 that the drawings $D_1, D_2, \ldots, D_{k+1}$ are called the explicit intermediate drawings of the planarity-preserving linear morph sequence. In contrast, the "general" intermediate drawings also include the (infinite) set of drawings during each linear morph from D_i to D_{i+1}. We called these implicit drawings.

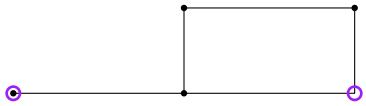
Definition 2.2.3. *A linear morph sequence is a* ***planarity-preserving linear morph sequence*** *if, for every pair of explicit intermediate drawings D_i, D_{i+1} that are not equivalent drawings, the linear morph between them is planarity-preserving.*

Take careful note of the fact that a even linear morph sequence with length two could require an arbitrarily large representation: On its own, a linear morph sequence makes no guarantees of the number of bends per edge nor the size of the grid required to represent the explicit intermediate drawings. Each algorithm that produces linear morph sequences must prove such claims separately.

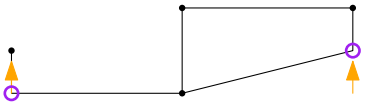
For an example of a planarity-preserving linear morph sequence of poly-line drawings, see Figure 2.5.



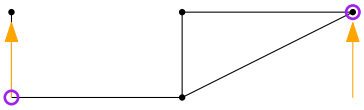
(a) $P = D_1$. One bend, but no degenerate bends.



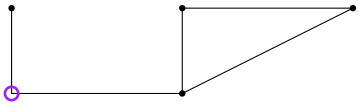
(b) D_2. One degenerate bend, coincident with the leftmost vertex. D_2 is an equivalent drawing to D_1.



(c) $M^{\frac{1}{2}}_{D_2,D_3}$.



(d) D_3. The same number of bends as D_2. The bottom-right bend is coincident with a vertex. The leftmost bend is not.

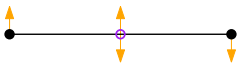


(e) $D_4 = Q$. No degenerate bends, but it is an equivalent drawing to D_3.

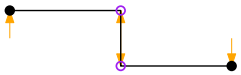
Figure 2.4: An example of a planarity-preserving linear morph sequence D_1, D_2, D_3, D_4 of length 3 between two compatible planar straight-line drawings $P = D_1$ and $Q = D_4$, where the explicit intermediate drawings are planar poly-line drawings. The movement of vertices compared to each previous step is denoted with (orange) arrows. Bends are denoted by large (purple) empty circles.

(a) A simple planar straight-line drawing of K_2 with no bends.

(b) Two (degenerate) bends are both added at the location denoted by the hollow circle. This is an equivalent drawing to Figure 2.5a.



(c) The planar poly-line drawing is undergoing a linear morph, which is just beginning. The vertex and bend trajectories are denoted by the orange arrows.



(d) The planar poly-line drawing is undergoing a linear morph, which has just ended. The two bends are now at different locations, each denoted by a hollow circle.

Figure 2.5: A demonstration of equivalent drawings, bends, degenerate bends, and linear morphs. The bends denoted in this figure are only present for demonstration purposes, and would not be visible in the actual drawings.

2.3 Planarity-Preserving Unidirectional Linear Morphs

For a restricted class of linear morphs, it is quite easy to classify whether a linear morph is planarity-preserving. Recall from Section 1.1 that unidirectional linear morphs are linear morphs for which the directions of movement for all vertices and bends are parallel. Existing work by Alamdari et al. [1] has provided a very simple characterization of unidirectional linear morphs that are planarity-preserving, which will be briefly reviewed in this section.

Theorem 2.3.1 (Proven but not stated by Alamdari et al. [1], Lemma 7 by Kleist et al. [30])**.** *Let P, Q be compatible planar straight-line drawings of the same graph. Suppose that the linear morph $M_{P,Q}$ from P to Q is unidirectional in the direction of a line L. If, for each line L' parallel to L, the sequences of vertices and edge segments intersected in both P and Q are equal, then the linear morph is planarity-preserving.*

We'd like to use a version of the above theorem for morphing planar poly-line drawings. In particular, we wish to apply it for planar poly-line drawings which allow for zero-length edge segments (which implies bends may coincide with vertices and other bends). Hence, we need to be careful how we extend the theorem statement.

For a planar poly-line drawing P, and a line L (with some direction), the line L may intersect vertices, bends, and/or edge segments in P. In particular, it may encounter multiple coinciding bends/vertices simultaneously. We say that the ***partial intersection order*** of L intersecting P is the partial ordering of all vertices/bends/edge segments in P intersected by L along its prescribed direction. In particular, elements which are encountered simultaneously (e.g., two coinciding bends) are incomparable in this partial ordering.

Two partial orderings $<, <'$ over a set S are said to be ***compatible*** if there are no two elements $s, s' \in S$ with $s < s'$ and $s' <' s$. This is enough to extend the theorem for our use-case.

Theorem 2.3.2. *Let P, Q be compatible planar poly-line drawings of the same graph, with the same number of bends along each edge. Suppose that the linear morph $M_{P,Q}$ from P to Q is unidirectional in the direction of a line L. If, for each line L' parallel to L, the partial intersection order of L' intersecting P is compatible with the partial intersection order of L' intersecting Q, then the linear morph is planarity-preserving.*

Proof. The drawings $P = M^0_{P,Q}$ and $Q = M^1_{P,Q}$ are assumed to be planar. It suffices to show that the drawing $M^t_{P,Q}$ is planar for $t \in (0, 1)$.

Bend-bend pairs and bend-vertex pairs that are coincident in both P and Q remain so throughout the morph. Thus, assume without loss of generality that there are no such pairs.

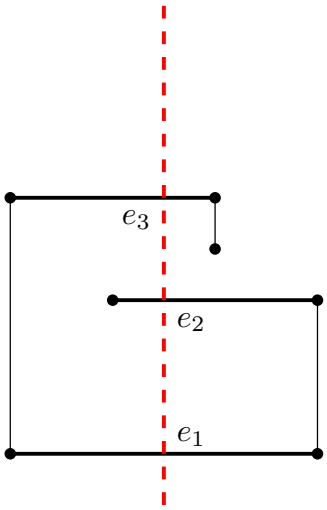
Let b be a bend and let L' be the line parallel to L through b in P and Q. Assume as a first case that b is coincident with some other bend/vertex x in either P or Q. Linear morphs are reversible (i.e., the linear morph from P to Q is planarity-preserving if and only if the linear morph from Q to P is planarity-preserving), so without loss of generality we assume b and x are coincident in P, and not coincident in Q. Assume without loss of generality that L' is oriented so that $b < x$ along L' in Q. Then $b < x$ along L' at all times $t \in (0, 1)$ during the linear morph (by Lemma 7.2.1 by Alamdari et al. [1]). Therefore, no bend is coincident with any other bend/vertex in $M^t_{P,Q}$ for any time $t \in (0, 1)$.

By Observation 2.2.1, there is some sufficiently small time $\delta > 0$ at which $M^{\varepsilon}_{P,Q}$ is planar for all $0 < \varepsilon \leq \delta$. Without loss of generality (since linear morphs are reversible), assume $M^{1-\varepsilon}_{P,Q}$ is also planar for all $0 < \varepsilon \leq \delta$. There are no coincident bends in $M^t_{P,Q}$ for all $t \in (\delta, 1 - \delta)$, so by Theorem 2.3.1, the linear morph from $M^{\delta}_{P,Q}$ to $M^{1-\delta}_{P,Q}$ is planarity-preserving (since we can treat them as straight-line drawings, with a vertex in place of each bend). Hence, all drawings $M^t_{P,Q}$ with $t \in (0, 1)$ are planar, so the linear morph $M_{P,Q}$ is planarity-preserving. □

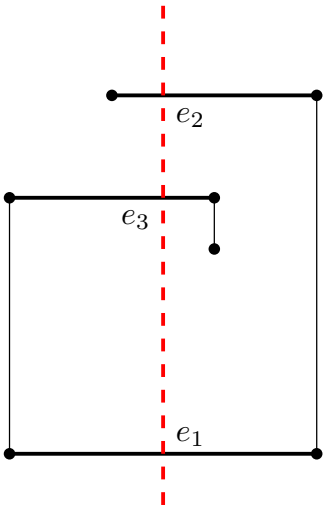
2.4 Linear Morphs of Orthogonal Box Drawings

The main methods of this work will use morphs of planar orthogonal box drawings in order to derive morphs of planar poly-line drawings. Note that a linear morph between two planar orthogonal box drawings has not yet been defined, and in fact the definition must be carefully chosen to coincide with linear morphs of some closely related planar poly-line drawings, though discussion of this relationship will be postponed until Section 3.1. Recall that orthogonal box drawings represent vertices with axis-aligned rectangles (called ***vertex boxes***) of non-zero area, and edges with orthogonal poly-lines. Unlike planar poly-line drawings, orthogonal box drawings require edges to be completely disjoint, even where they attach to vertex boxes. Orthogonal box drawings are promising to work with since they are similar to orthogonal point drawings, for which short sequences of linear morphs are known to exist [47], as discussed in Section 1.1.

Before a discussion of morphs, it should be briefly noted exactly which geometric co-ordinates are required to specify an orthogonal box drawing. In a straight-line drawing,



(a) The drawing P. The order of objects intersecting the red line is e_1, e_2, e_3.



(b) The drawing Q. The order of objects intersecting the dashed red line is e_1, e_3, e_2.

Figure 2.6: An example of two drawings P and Q for which the unidirectional morph between them is not planarity-preserving, and a line parallel to the direction of movement on which the order of objects differs. Note that this is the same pair of drawings as in Figure 2.1.

coordinates are needed only for the vertices. In a poly-line drawing, coordinates are also needed for each bend. In an orthogonal box drawing, coordinates are needed for the endpoint of each edge, called a ***port***, in addition to the corners of each vertex box, and the bends along each edge. Any definition of a linear morph must describe how these values change.

The individual types of linear morphs which will be performed on orthogonal box drawings in this work will actually be quite limited, so there is a lot of flexibility in the definition of both linear morphs, and of planarity-preserving linear morphs. The following is a useful and simple definition, which will suffice for all of our purposes:

Definition 2.4.1. *Let P and Q be orthogonal box drawings of the same graph G with the same number of bends along each edge, The* ***linear morph of orthogonal box drawings*** *P and Q, denoted $\boldsymbol{M_{P,Q}}$, is the morph which linearly interpolates all corners, ports, and bends from their locations in P to their respective locations in Q.*

We let $\boldsymbol{M^t_{P,Q}}$ denote the drawing obtained by performing this linear interpolation with a parameter $t \in [0, 1]$, where $M^0_{P,Q} = P$ and $M^1_{P,Q} = Q$. If s is an edge segment of P/Q, then the motion of S during the morph is determined by the motions of its endpoints. We restrict our attention to linear morphs where horizontal edge segments remain horizontal, and vertical edge segments remain vertical (allowing for segments of zero length when a bend coincides with a port or another bend). Previous work on morphs of orthogonal point drawings calls the analogous restricted linear morph an "orthogonal linear morph" or simply "orthogonal morph" (i.e., a special kind of linear morph), but we primarily use this restriction to make the steps of our orthogonal box morphing algorithm easier to follow.

If B is a vertex box of P/Q, then the motion of B is determined by the motion of its four corners. We restrict our attention to linear morphs where each vertex box remains a rectangle throughout the morph, but we do not require that the rectangle remains axis-aligned. We call these intermediate shapes ***vertex rectangles***, and reserve the term "vertex box" for the axis-aligned rectangles of orthogonal box drawings (such as P and Q). It should be noted that the drawing $M^t_{P,Q}$ (for $t \in (0, 1)$) is not necessarily an orthogonal box drawing.

Definition 2.4.2. *Let $M_{P,Q}$ be a linear morph of orthogonal box drawings P and Q. If all the directions of movement are parallel, then $M_{P,Q}$ is also called a* ***unidirectional linear morph****. We restrict our attention to unidirectional linear morphs which are strictly horizontal or strictly vertical, which are called* ***horizontal linear morphs*** *and* ***vertical linear morphs*** *respectively.*

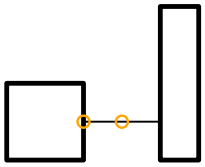
The definition of a linear morph for orthogonal box drawings explicitly permits the vertex boxes to become non-axis-aligned rectangles during the morph (i.e., in the implicit drawings). In particular, it may appear as if we rotated the boxes, even though we performed a linear morph. See Figure 2.7 for an example of a unidirectional linear morph of an orthogonal box drawing, and a linear morph of an orthogonal box drawing which "rotates" a vertex box.

The definition of linear morph we have chosen for orthogonal box drawings leads to a natural extension for the definition of "planarity-preserving".

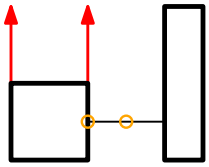
Definition 2.4.3. *Let $M_{P,Q}$ be a linear morph of orthogonal box drawings P and Q. We say that $M_{P,Q}$ is **planarity-preserving** if the following restrictions hold for every drawing $M^t_{P,Q}$ ($t \in (0,1)$):*

- *The four corners corresponding to each vertex form a rectangle (the vertex rectangle). Each vertex rectangle has positive area, and no pair of vertex rectangles intersect.*
- *Each port lies on the boundary of its respective vertex rectangle.*
- *An edge appears as a path of segments joining its two ports, and this path is a **simple path**, meaning that if two positive-length segments intersect at a point p, then p is an endpoint of both segments, there is at least one bend at p, and no other positive-length segment contains the point p.*
- *Apart from its ports, an edge does not share any points with any vertex rectangle.*
- *If a bend-bend pair or bend-port pair are coincident, then they are also coincident in both P and Q.*

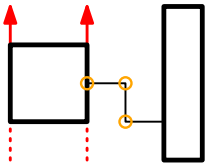
Most of the linear morphs which will be used throughout this work will be horizontal and vertical linear morphs. Under the above natural extension of planarity-preserving linear morphs, the important result characterizing planarity-preserving unidirectional morphs on planar poly-line drawings, specifically Theorem 2.3.2 is also true of horizontal and vertical linear morphs of orthogonal box drawings. Note that vertex boxes remain axis-aligned rectangles throughout horizontal (vertical) linear morphs since each pair of corners along the same box connected by a vertical (horizontal) side of the box has the same pair of starting and ending x-coordinates (y-coordinates).



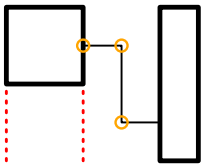
(a) An orthogonal box drawing P_1. The left hollow circle is a bend coinciding with a port, and the right hollow circle is a pair of coinciding bends.



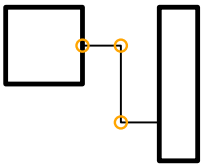
(b) The beginning of a (vertical unidirectional) linear morph from P_1 to the drawing P_2 given in Figure 2.7e at time $t = 0$, with the trajectory of some corners denoted.



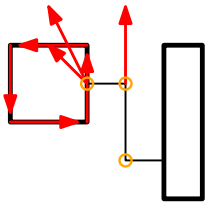
(c) The midpoint of the linear morph at time $t = 0.5$, with some remaining trajectories and trajectories traversed denoted.



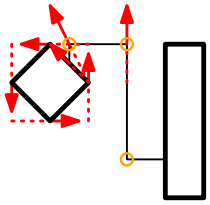
(d) The end of the linear morph, at time $t = 1$.



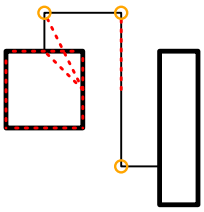
(e) The orthogonal box drawing P_2.



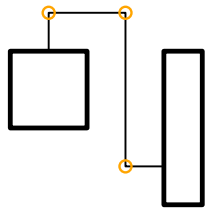
(f) The beginning of another linear morph at time $t = 0$, this time from P_2 to the drawing P_3 given in Figure 2.7i. This morph is not unidirectional. All non-trivial corner/bend/port trajectories are denoted.



(g) The second linear morph at time $t = 0.5$. Note that the leftmost vertex rectangle is not axis-aligned.



(h) The second linear morph at time $t = 1$.



(i) The final drawing P_3, with no degenerate bends.

Figure 2.7: An example of a linear morph sequence using orthogonal box drawings.

2.5 Overview of Methodology

The primary goal of this work is to give an algorithm for generating a morph between compatible straight-line drawings, by adding bends, while also maintaining a few constraints on the intermediate drawings. Recall the statement of the main theorem of this work:

Theorem 1.2.1 (Main)**.** *Let G be a connected planar graph with n vertices. For a compatible pair of planar straight-line drawings P and Q of G, whose vertices lie on an $O(n) \times O(n)$ grid, there exists a planarity-preserving linear morph sequence from P to Q of length $O(n)$, where each explicit intermediate drawing lies on an $O(n) \times O(n)$ grid and has $O(1)$ bends per edge. Moreover, this sequence can be found in $O(n^2)$ time.*

Several constraints must be maintained throughout the long sequence of linear morphs that we wish to construct. We will use an algorithm that carefully maintains all the constraints while progressively constructing the sequence. The algorithm will be broken into several phases:

Phase I: Morph the straight-line drawings into poly-line drawings which are "admitted" by a pair of (straight-line) orthogonal box drawings. These definitions will be discussed thoroughly in Section 3.2. At this point, the problem reduces to that of morphing orthogonal box drawings with zero bends per edge.

Phase II: Morph the pair of orthogonal box drawings so that they are "parallel", essentially meaning that the sequences of ports along each side of each vertex box, as well as the sequences of turns along each edge are all the same in both drawings. This definition will be discussed thoroughly in Chapter 4. This will introduce $O(1)$ bends per edge.

Phase III: Morph the pair of parallel orthogonal box drawings to be equal by appealing to a result for morphing parallel orthogonal points drawings.

See Figure 2.8 for an example of all three phases.

Phase I will be quite short and relatively simple, and will be contained entirely within Section 3.2. Phases II and III are, at a high-level, very similar to the techniques used by Biedl et al. [7] to morph parallel orthogonal point drawings with $O(n^2)$ linear morphs, and correspond to the two phases used in that work. However, Phase II is more complicated because we are morphing orthogonal box drawings, and we are doing so with only $O(n)$ linear morphs. We further break up Phase II as follows:

Phase IIa: Morph the two orthogonal box drawings so that each side of each vertex box contains the same sequence of ports in both drawings. This will introduce bends in the edges. At this point, if the drawings fail to be parallel, then it must be because they have a different sequence of turns along some edges.

Phase IIb: Compute a sequence of "simultaneous twists" (to be defined in Section 5.2) to be performed on the vertices (in one of the drawings), which will guarantee a certain property for a pair of resulting drawings. At a high-level, this property will be that the "net" left and right turns along each edge will be the same in both drawings. In this phase, we only compute what the twists should be, but do not yet perform the corresponding morphs.

Phase IIc: Perform each of the desired simultaneous twists. After each one, compress the drawing to a small grid and simplify so that each edge consists of exclusively left turns or exclusively right turns (and hence the sequence of turns along the edge are the same as its "net" turns).

Each of these phases correspond directly to a phase of the algorithm used by Biedl et al. [7], though phases IIa and IIc involve many additional details. Phases IIa, IIb, and IIc will be completed in Chapter 4, Section 5.3, and Chapter 7 respectively.

The final result of Phase II is the following theorem for morphing orthogonal box drawings to become parallel:

Theorem 2.5.1 (Obtaining Parallel Boxes)**.** *Let G be a connected planar graph on n vertices. If P and Q are a pair of compatible planar orthogonal box drawings of G, both drawn on an $O(n) \times O(n)$ grid, with $O(1)$ bends per edge, then there exists a pair of parallel planar orthogonal box drawings P' and Q' of G and a pair of planarity-preserving linear morph sequences of lengths $k_P, k_Q \in O(n)$, from P to P' and from Q to Q', respectively, such that the explicit intermediate drawings in each morph sequence are all also drawn on an $O(n) \times O(n)$ grid and have $O(1)$ bends per edge. Moreover, these sequences can be found in $O(n^2)$ time.*

This theorem will be proven in Section 8.1.

Phase III will be quite short and simple, since it simply applies some results by Biedl et al. [7] to orthogonal box drawings via a reduction. As such, it will be presented entirely within the proof of Theorem 1.2.1 in Section 8.2.

To summarize: Chapter 3 contains some important definitions/reductions, and Phase I. Chapter 4 contains Phase IIa. Chapter 5 contains some important definitions and discussion, and Phase IIb. Chapter 6 contains one component of Phase IIc ("compressions"

and "simplification"). Chapter 7 contains the other component of Phase IIc ("performing twists"). Chapter 8 puts all the phases together and proves Theorem 2.5.1 (encapsulating Phase II) and then Theorem 1.2.1 (the main result). In addition, it also discusses some simple extensions of Theorem 1.2.1. Chapter 9 discusses some remaining open problems.

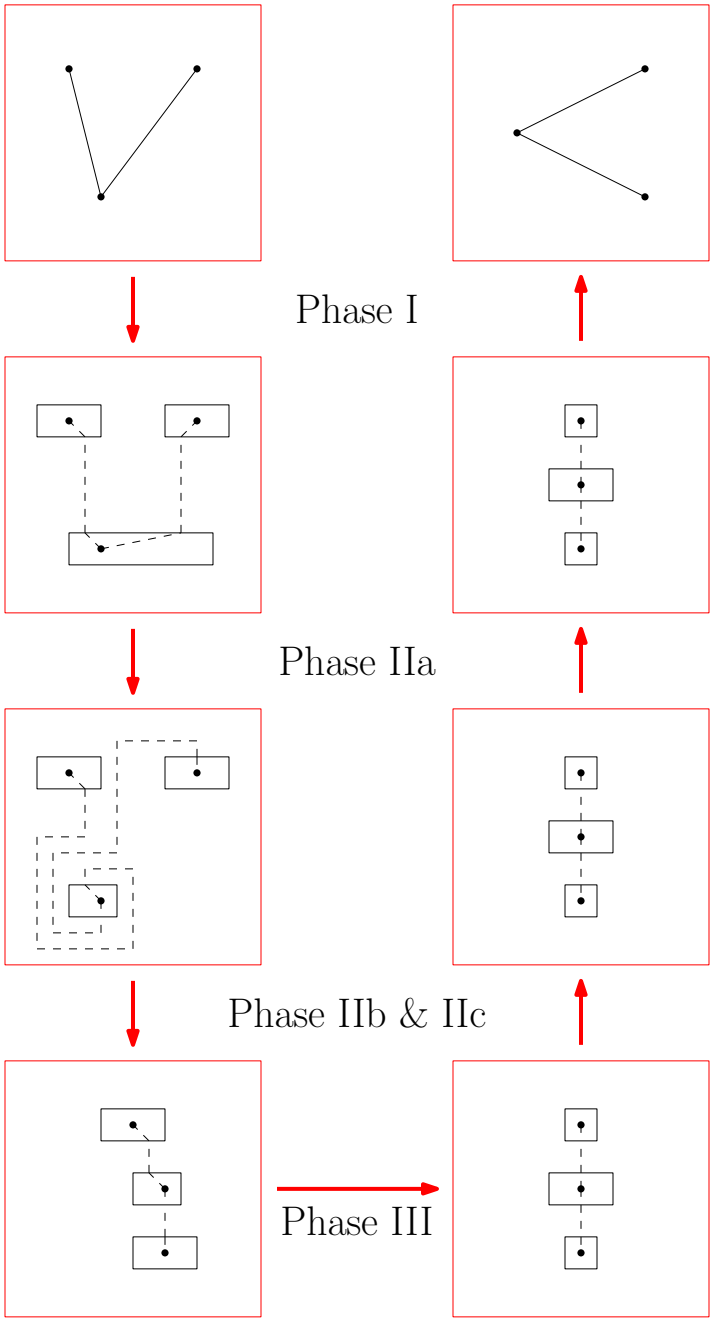


Figure 2.8: An example of all phases on a very simple drawing.

Chapter 3

Reduction to Morphing Orthogonal Box Drawings

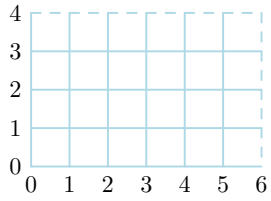
Phase I consists of reducing the problem of morphing planar straight-line drawings to a morphing problem on orthogonal box drawings, and is the topic of this chapter. This reduction has two steps. First, it will be shown that every orthogonal box drawing has an associated planar poly-line drawing called an "admitted drawing", and that a planarity-preserving linear morph of the former induces the same for the latter. Second, it will be shown that any planar straight-line drawing can be morphed to the admitted drawing of some orthogonal box drawing.

Detailed coordinates for orthogonal box drawings will play a particularly prominent role in this chapter, so it is helpful to recall some notation we introduced in Section 2.1.

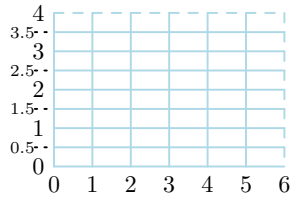
Notation 3.0.1. *For an orthogonal box drawing A and a vertex v, let $A(v)$ denote the vertex box of v in A.*

Discussion of coordinates in this chapter will be aided by an additional concept. We usually say that an $N \times N$ integer grid has N columns and N rows, each numbered from $0, \ldots, N-1$, and a coordinate $(x, y) \in \mathbb{R}^2$ is a grid point if x and y are integers such that $0 \leq x, y < N$. The grids we use can be assumed to have this positioning, since translations of all our drawing types can be handled by a single (planarity-preserving) linear morph.

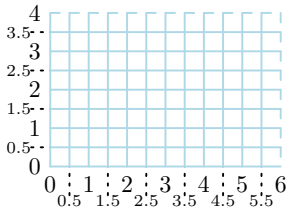
Given an $N \times N$ integer grid Z, we say that a ***(w, h)-refinement*** of Z is a $wN \times hN$ grid with columns numbered $0, \frac{1}{w}, \ldots, \frac{w-1}{w}, 1, 1+\frac{1}{w}, \ldots, N-1+\frac{w-1}{w}$, and rows numbered $0, \frac{1}{h}, \ldots, \frac{h-1}{h}, 1, 1+\frac{1}{h}, \ldots, N-1+\frac{h-1}{h}$, such that a coordinate $(x, y) \in \mathbb{R}^2$ is a grid point if wx and hy are integers and $0 \leq x, y < N$. See Figure 3.1 for examples of refinements.



(a) A 6×4 integer grid Z.



(b) A $(2, 1)$-refinement of Z.



(c) A $(2, 2)$-refinement of Z.

Figure 3.1: Examples of refinements of an integer grid.

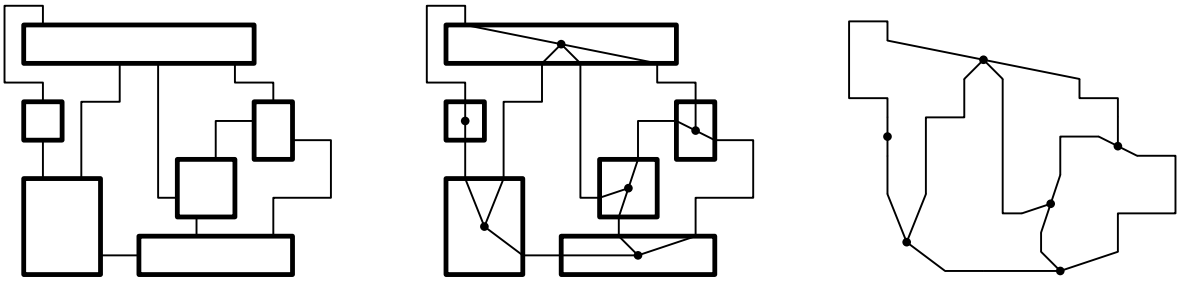
A drawing D is said to be drawn on a (w, h)-refinement Z' of an $N \times N$ integer grid if all of its coordinates are grid points of Z'. Then, there exists a scaled drawing D' of D drawn on a $wN \times hN$ integer grid Z_0 (obtained by multiplying all coordinates component-wise by (w, h)) and the linear morph from D to D' is planarity-preserving, since it simply scales the coordinates. Furthermore, for a pair of drawings D_1, D_2 drawn on Z', there are corresponding scaled drawings D'_1 and D'_2 that are drawn on Z_0 (also obtained by scaling the coordinates in D_1, D_2 component-wise by (w, h)) such that the linear morph from D_1 to D_2 is planarity-preserving if and only if the linear morph from D'_1 to D'_2 is planarity-preserving. Hence, a refined grid should be thought of as merely a tool to simplify presentation for this chapter. All linear morph sequences on refined grids are equivalent to linear morph sequences on integer grids, sometimes with some additional linear morphs that scale coordinates at the beginning and end (which does not change the asymptotic complexity of the sequence length).

3.1 Admitted Drawings of Orthogonal Box Drawings

Admitted drawings are planar poly-line drawings which "agree" with orthogonal box drawings everywhere in the plane outside each vertex box.

Definition 3.1.1. *Let D be a planar orthogonal box drawing of a graph G. Then the **admitted drawing** of D is the unique planar poly-line drawing P such that:*

- *For each vertex $v \in V(G)$, the location of v in P is at the centre of the vertex box $D(v)$.*
- *For each edge $e = uv \in E(G)$, the segments of e in P agree with D outside the vertex boxes $D(u)$ and $D(v)$, and inside each vertex box they each contain exactly one*



(a) A planar orthogonal box drawing D.

(b) The admitted planar poly-line drawing P of D superimposed onto D.

(c) The admitted planar poly-line drawing P of D.

Figure 3.2: An example of a planar orthogonal box drawing, and an admitted planar poly-line drawing.

segment, joining the vertex and an additional bend at the location of the corresponding port.

With this definition of admitted drawings, an important property can be stated about their underlying grids.

Observation 3.1.2. *Let D be a planar orthogonal box drawing of a graph G drawn on an $N \times N$ integer grid Z. Then the admitted drawing A of D is drawn on the $(2,2)$-refinement of Z.*

Proof. The only additional coordinates in A not present in D belong to the locations of the vertices in A. For each vertex $v \in V(G)$, the location $A(v)$ is found at the centre of the vertex box $D(v)$, whose four corners lie in Z. Therefore, each of the x and y-coordinates of $A(v)$ are either an integer, or a half integer, so $A(v)$ lies in the $(2,2)$-refinement of Z. □

Note that it is possible that the admitted drawing A of D is *also* drawn on Z itself, if all the coordinates of vertex corners in D happen to be even. Regardless, a scaled drawing A' of A must exist that is drawn on a $2N \times 2N$ integer grid, which is obtained simply by multiplying all coordinates of A by 2.

See Figure 3.2 for an example of an admitted drawing. Admitted drawings are particularly useful because of the following result:

Lemma 3.1.3. *Let P and Q be planar orthogonal box drawings of the same graph. Let P' and Q' be the admitted planar poly-line drawings of P and Q, respectively. Suppose that*

the linear morph from P to Q is planarity-preserving. Then the linear morph from P' to Q' is also planarity-preserving.

Note that Lemma 3.1.3 applies even if the linear morph from P to Q is a non-horizontal non-vertical linear morph that transforms the vertex boxes.

Proof. The admitted drawings P' and Q' differ from the orthogonal box drawings P and Q (respectively) only within each vertex box, so it suffices to show that during the morph $M_{P',Q'}$, the location of each vertex $v \in V(G)$ at time t remains in the strict interior of its respective vertex rectangle in $M^t_{P,Q}$.

Let the corners of the vertex boxes $P(v)$ and $Q(v)$ be denoted by $X^P_0, X^P_1, X^P_2, X^P_3$ and $X^Q_0, X^Q_1, X^Q_2, X^Q_3$ respectively. Then, the location of the vertex v in $M^t_{P',Q'}$ is

$$(1-t)\sum_{i=0}^{3} X^P_i/4 + (t)\sum_{i=0}^{3} X^Q_i/4 = \sum_{i=0}^{3}\left[(1-t)X^P_i + (t)X^Q_i\right]/4,$$

which is the centre of v's vertex rectangle in $M^t_{P,Q}$. Therefore, the linear morph is planarity-preserving. □

With this result, morphing admitted drawings can be reduced to the problem of morphing orthogonal box drawings. Given a pair of orthogonal box drawings A and B and a planarity-preserving sequence of linear morphs from A to B, the corresponding sequence of linear morphs between the admitted planar poly-line drawings is also planarity-preserving. Hence, if A' and B' are planar poly-line drawings which are admitted drawings of A and B respectively, then the problem of finding a planarity-preserving linear morph sequence from A' to B' can be reduced to finding a linear morph sequence from A to B.

As an aside, the reader may question why admitted drawings have specifically been defined so as to require the vertex lies at the centre of the vertex box. In fact, with this alternative definition, the statement of Lemma 3.1.3 would no longer be true. If the vertex is placed near the same corner of a vertex box in both drawings, and then the vertex box is "rotated" similarly to the square box in Figure 2.7g, it might lie outside the vertex rectangle, and could intersect something moving at that location.

3.2 Finding an Initial Orthogonal Box Drawing

The previous section showed that admitted planar poly-line drawings of planar orthogonal box drawings can be a useful tool for morphing poly-line drawings. It remains to be shown

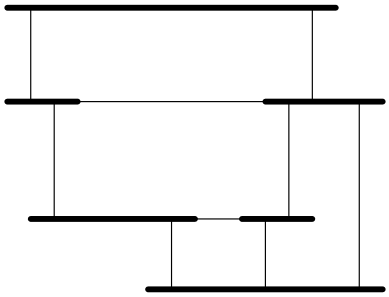


Figure 3.3: An example of a visibility representation of a 6-vertex graph..

that an arbitrary planar straight-line drawing P can be morphed into an admitted planar poly-line drawing P' of some orthogonal box drawing D, which is the purpose of this section. This process will be broken into two substeps. First, we will use P to compute an orthogonal box drawing D with some special properties, by using existing techniques known for so-called "visibility representations". Then, we will take the admitted drawing P' of D, and show that a planarity-preserving linear morph sequence from P to P' can be computed.

3.2.1 Visibility Representations

We introduce another well-studied class of graph drawings. For a planar graph G, a ***visibility representation*** of G represents each vertex v as a horizontal line segment, and each edge as either a vertical or horizontal line segment, such that for each line segment L_{uv} representing an edge uv, one of the endpoints of L_{uv} intersects the horizontal line segment representing u, and the other endpoint intersects the horizontal line segment representing v, and no other forms of intersections are pemitted. Additionally, no pair of horizontal line segments representing vertices may intersect. In some other literature, our "visibility representations" could sometimes be called "flat 2-directional visibility representations", but since we will study no other kinds, we will omit these qualifiers. See Figure 3.3 for an example of this type of drawing.

There is a large body of existing literature on visibility representations and their variations [42, 35, 50, 29, 6], but of particular interest to us is the following result:

Theorem 3.2.1 (Theorems 5 and 6 by Biedl [6])**.** *Let P be a planar straight-line drawing of an n-vertex graph G drawn on an $N \times N$ integer grid. Then there exists a visibility representation R of G, so that the horizontal line segments representing vertices in R have the same y-coordinates as the corresponding points representing vertices in P, and every*

possible horizontal line intersects the same sequence of edges and vertices in both drawings. Furthermore, R is drawn on an $O(n) \times N$ integer grid.

Note that the condition on horizontal lines is (in our notation from Section 2.3) the same as saying that the partial intersection order of horizontal lines is the same in P and R.

For our intended application, we also need to know how efficiently we can find a drawing R given by Theorem 3.2.1. Unfortunately, this was not discussed in the paper, and the naive approach to the construction results in an algorithm which would be too slow for our purposes. However, we will discuss machinery in Chapter 6 which will make the construction of a much faster algorithm both possible and straightforward, given the above existential proof. Hence, we will postpone any detailed discussion of an algorithm and proof for this step until Section 6.2.5. However, the improved result, which we will use in this chapter, is as follows:

Theorem 3.2.2. *Given a planar straight-line drawing P, a visibility representation R with the properties stated in Theorem 3.2.1 can be computed in $O(n \log n)$ time.*

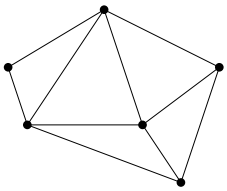
Since we are interested in morphing orthogonal box drawings, and not visibility representations, we are more interested in the following corollary that replaces the visibility representation by a straight-line orthogonal box drawing, i.e., an orthogonal box drawing where edges have no bends:

Corollary 3.2.3. *Let P be a planar straight-line drawing of a graph G drawn on an $N \times N$ integer grid, such that no vertex lies on the row labelled 0. Then there exists an $O(n) \times N$ integer grid Z, and a planar straight-line orthogonal box drawing D of G drawn on a $(1, 3)$-refinement of Z, so that each vertex has the same y-coordinate in both P and the admitted planar poly-line drawing P' of D, and every horizontal line intersects the same sequence of edges and vertices in both P and P'. Moreover, D and P' can be found in $O(n \log n)$ time.*

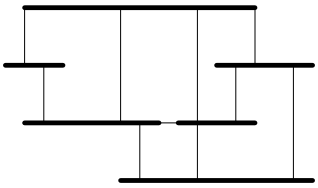
See Figure 3.4 for examples of the constructions given by Theorem 3.2.1 and Corollary 3.2.3.

Note that the assumption that no vertex in P lies on the row labelled 0 is made only to simplify handling the edge case of coordinates for vertex boxes corresponding vertices along the bottommost occupied row. This can be thought of as an implementation detail.

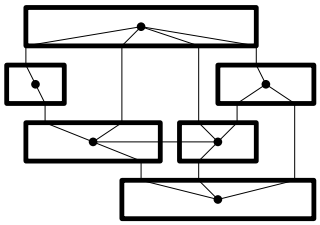
Note also that since D is drawn on a $(1, 3)$-refinement of Z, P' is therefore drawn on a $(2, 6)$-refinement of Z. We will see that P' is more specifically drawn on a $(2, 3)$-refinement of Z, though this fact is not needed for our purposes.



(a) A planar straight-line drawing P.



(b) A visibility representation which would result from applying Theorem 3.2.1 to P.



(c) An orthogonal box drawing, and corresponding admitted drawing P', which would result from applying Corollary 3.2.3 to P.

Figure 3.4: An example of a straight-line (point) drawing, and other forms of drawings as they would be computed via Theorem 3.2.1 and Corollary 3.2.3.

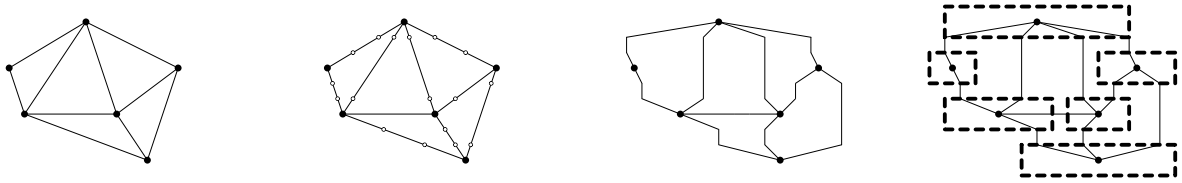
Proof. Apply Theorem 3.2.2 to obtain a visibility representation R in $O(n \log n)$ time, and let Z be the integer grid on which it resides. Construct D on the $(1, 3)$-refinement of Z by replacing each horizontal line segment representing a vertex with a rectangle of height $2/3$ vertically centered at the original y-coordinate, and with the same x-coordinate range. Each vertical line segment representing an edge in R is replaced with a shorter vertical line segment whose endpoints become ports along its incident vertex boxes. Horizontal line segments representing edges can remain identical in both R and D, since the ports for such line segments are now found along the left and right sides of the vertex boxes.

For each vertex v in the admitted drawing P' of D, the point $P'(v)$ lies in the centre of the vertex box $D(v)$, and hence has the same y-coordinate as $P(v)$.

P' can be directly constructed from D in $O(n)$ time, so the time complexity of this construction is $O(n \log n)$ in total. □

3.2.2 Morphing to an Initial Admitted Drawing

The previous subsection used a planar straight-line drawing P to construct an orthogonal box drawing D with some useful properties, and a corresponding admitted drawing P' of D. In this section, we will show that a planarity-preserving linear morph sequence exists from P to P'. Furthermore, we also want our linear morph sequence to have some useful properties. Specifically, we want the explicit intermediate drawings to be drawn on a small $(O(n + N) \times O(n + N))$ grid, and we want the length of the linear morph sequence to be



(a) The initial straight-line drawing P.

(b) An equivalent drawing to P which introduces bends at (potentially non-grid) coordinates.

(c) A planar poly-line drawing P', which is an admitted drawing of an orthogonal box drawing.

(d) The drawing P', with the vertex boxes of the orthogonal box drawing D overlayed.

Figure 3.5: An example of a (simple) technique for morphing to an admitted drawing which allows introducing bends at arbitrary locations (i.e., not on a grid).

small ($O(n)$). We will make careful use of refinements to simplify our presentation and proof.

As a simple aside, if we relax the grid condition so that bends can be introduced at arbitrary locations (rather than only at grid points), similarly to the algorithm of Lubiw and Petrick [33] discussed in Chapter 1, then this morph is quite simple: For any non-horizontal edge, simply introduce bends to P whose y-coordinates are equal to their final locations in the admitted drawing P', after which a horizontal linear morph will be planarity-preserving by Theorem 2.3.2. See Figure 3.5 for an example of this construction.

In order to satisfy the requirement that the bends are introduced at grid points, the bends will instead be introduced at the locations coinciding with their corresponding vertices. However, this requires a longer and more complex planarity-preserving sequence of linear morphs.

Theorem 3.2.4. *Let P be a planar straight-line drawing of a graph G drawn on an $N \times N$ integer grid. Then there exists an $(O(n) + N) \times N$ integer grid Z, and a straight-line orthogonal box drawing D drawn on a $(1, 3)$-refinement of Z, so that a planarity-preserving linear morph sequence of length $O(n)$ exists from P to the admitted planar poly-line drawing P' of D. Furthermore, each explicit intermediate drawing of the linear morph sequence is drawn on a $(2, 6)$-refinement of Z, and has at most 2 bends per edge. Moreover, P', D, and the linear morph sequence can be computed in $O(n^2)$ time.*

Proof. Use Corollary 3.2.3 to compute an orthogonal box drawing D and its admitted planar poly-line drawing P' in $O(n \log n)$ time. Assume for simplicity that P' does not

contain any bends along strictly horizontal edges. They can be re-added after the morph is complete, since the drawing without them is essentially equivalent, so long as y-coordinates are always preserved.

Let Z_P be the $N \times N$ integer grid on which P resides. Let Z_0 be the $O(n) \times N$ integer grid whose $(1,3)$-refinement Z_D supports D and its $(2,6)$-refinement $Z_{P'}$ supports P'. That is, the corners/ports/bends of D lie on grid points of Z_D, and the vertices/bends of P' lie on grid points of $Z_{P'}$.

We let Z be the $(O(n)+N) \times N$ grid obtained by attaching Z_P to the right of Z_0, so that Z_0 forms the leftmost $O(n)$ columns of Z, while Z_P forms the rightmost N columns. Note that all vertices have the same y-coordinate in P' drawn in Z_0 (as a subgrid of Z), and in P drawn in Z_P (as a subgrid of Z).

It suffices to find a planarity-preserving sequence of linear morphs from P to P'. We first morph P to add two bends to each edge, one at each endpoint, and translate the drawing so that it lies on the N rightmost coordinates of Z (i.e., add N to each x-coordinate). Let P^+ be the resulting drawing which is drawn on Z.

Now recall that D lies on the $(1,3)$-refinement of Z_0, which is the $O(n)$ leftmost columns of Z, and hence P' lies on the $(2,6)$-refinement on Z_0. The y-coordinates of each vertex are shared by both P^+ and P'. See Figure 3.6 for an example of P^+ and P drawn simultaneously. For the remainder of the proof, all bends and vertices in each explicit intermediate drawing will use either their locations in P' or their locations in P^+, so every explicit intermediate drawing will be drawn on the $(2,6)$-refinement of Z.

The main idea of this proof is to progressively morph vertices and edges of P^+ to their final locations in P', while being careful of the order chosen for these vertices and edges. First we will find an order of all the vertices and edges. Then we will move each vertex and edge according to this order. When it is an edge's turn, both of its bends are moved. When it is a vertex's turn, the vertex itself is moved.

For a set of points and line segments (in our case, the vertices and non-horizontal edges of P), the ***visibility ordering*** is a partial order defined by $a < b$ (where a, b may be edges or vertices) if there is a horizontal line that, traversed left to right, intersects a and then intersects b. We will move the vertices and non-horizontal edges of P^+ in an order respecting the visibility ordering given by P. Visibility orderings are known to be acyclic [38, 14, 15, 23]. However, we wish to add some additional constraints. Specifically, in our ordering, we will additionally require that each vertex is listed after all its incident non-horizontal edges. All the new constraints are those of the form $e < v$, where e is an non-horizontal edge incident to v. This is in fact still a visibility ordering, but one of a modified set of points and line segments. Specifically, each point for this modified set is obtained by

translating the location of a vertex in P by a sufficiently small amount $0 < \varepsilon < 1$ to the right, while the line segments are obtained again from the non-horizontal edges of P. This partial order can be computed trivially in $O(n^2)$ time by brute force comparison (i.e. check every pair of vertex/edge, edge/edge, and vertex/vertex, and augment the result with the additional constraints for incident vertex/edge pairs), and it can be converted to a total order in the same time complexity using standard topological sorting algorithms [28, 43]. Faster algorithms for computing the visibility order exist, but they do not improve the overall time complexity for computing our linear morph sequence in this proof.

We recommend studying the example in Figure 3.7 while reading the following construction.

For each item in the computed total order, at most one linear morph will be performed. If the item is a non-horizontal edge, we apply one linear morph moving both its bends from their locations in P^+ to their locations in P'. If the item is a vertex, we apply one linear morph, moving it from its location in P^+ to its final location in P'. A horizontal edge will be implicitly moved with its endpoints, and so it is not included in the total order. The length of this linear morph sequence is $O(n)$, since there are $O(n)$ edges and vertices. This construction takes $O(n^2)$ time to construct as a result.

It remains only to be shown that the constructed linear morph sequence is planarity-preserving. We show this by induction. Assume that for the first $k - 1 \geq 0$ vertices and non-horizontal edges, the linear morphs performed are planarity-preserving. Let the explicit intermediate drawing at this stage be denoted P_{k-1}. For the first $k-1$ vertices and non-horizontal edges in the order, the locations of the vertices and bends in P_{k-1} are equal to the corresponding locations in P'. For the remaining vertices and non-horizontal edges, the locations of the vertices in bends in P_{k-1} are equal to the corresponding locations in P^+. Then, we consider the kth step, and split into two cases.

As a first case, assume the kth step moves a non-horizontal edge e. Then, the only components of the drawing P_{k-1} which we will change are the locations of the bends in e. Note that, since the y-coordinates of the first and last segment in e do not overlap in either P^+ or P', the edge e cannot intersect itself during the linear morph. Since each bend moves to the left during the morph, a simple 4-sided polygon Q is formed by the line segments representing each edge segment along e in both of P^+ and P' (not including those with zero-length). See Figure 3.7b for an example of Q during a linear morph step. By Observation 2.2.1, it suffices to show that Q is empty of other vertices or bends of P_{k-1}. Let x be a vertex or bend of P_{k-1} not found along e. Let H be the horizontal line intersecting x. If H does not intersect Q, then x is not contained in Q. If, when H is followed from left to right, x is found before e in P_{k-1}, then x is at its location in P' (it

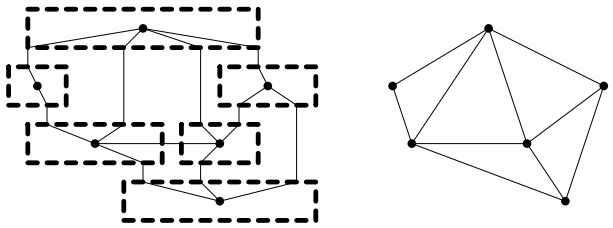
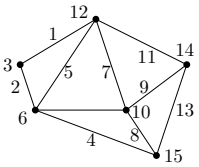


Figure 3.6: The drawings P' (with D as an overlay) and P^+ drawn together. Note that they do not overlap.

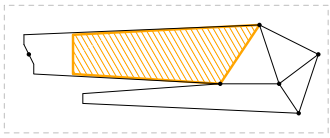
is earlier in the visibility ordering). Since P and P' have the same intersection order of horizontal lines by Corollary 3.2.3, we know that x is to the left of e in P' along H, and is therefore to the left of Q along H. If x is found after e in P_{k-1} along H, it is therefore to the right of Q along H, since Q extends leftward from e in P_{k+1}.

For the second case, assume the kth step moves a vertex v. Then, the only component of the drawing P_{k-1} which we will change is the location of v. The kth step is a unidirectional linear morph. Consider the horizontal line h whose y-coordinate is that of v (in all drawings). All non-horizontal edges to the right of v along h in P^+ have not yet been moved, and all non-horizontal edges to the left of v along h in P' have already been moved (by the visibility ordering). Therefore, the same order along h will be preserved by the linear morph of the kth step, so by Theorem 2.3.2, the linear morph is planarity-preserving. During this step, the endpoint of a horizontal edge incident to v may also be moved, and this also does not violate planarity since v will not intersect any other vertices during the morph (since the ordering along every horizontal line is preserved). □

As another aside, it may be interesting to readers to ask why a simpler approach doesn't work. Specifically, let P_0 be an equivalent drawing to P augmented with two (degenerate) bends per edge, one at each endpoint. Then, if the linear morph from P_0 to P' was planarity-preserving, we would be done. Unfortunately, this morph is not always planarity-preserving. In Figure 3.8 we show a 3-vertex counter-example that was found with a constraint formulation using the SMT solver Z3 [17]. Additionally, P_0 and P^+ (as defined in the above proof) differ only by a uniform horizontal translation of all vertices, and as a result, a similar counter-example can be found which shows that the linear morph directly from P^+ to P' is not always planarity-preserving.



(a) An order of the edges and vertices generated from the visibility ordering.



(b) An example of the 4-sided polygon Q (in Step 5 below) for an edge used to prove that the linear morph sequence is planarity-preserving.

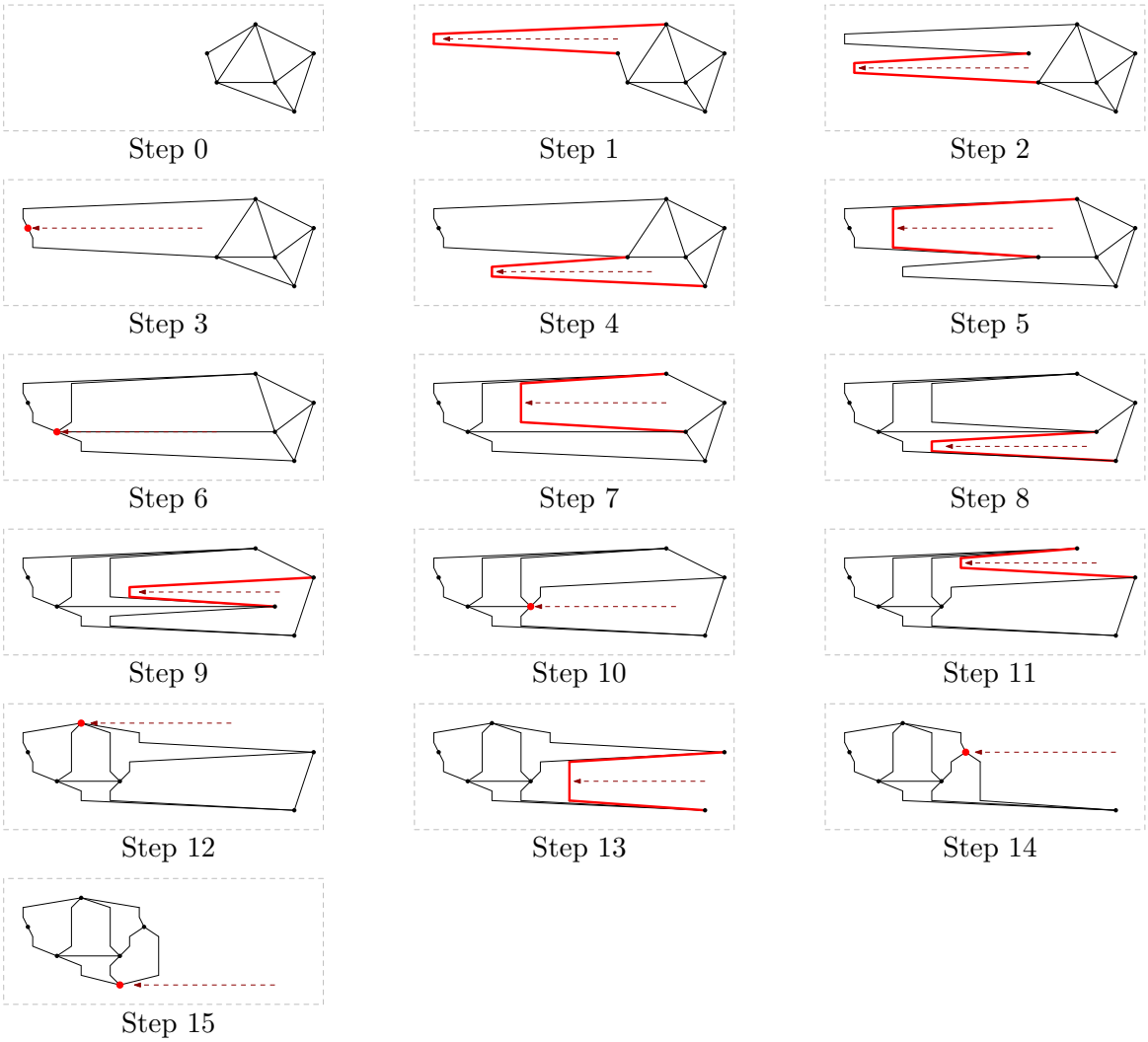
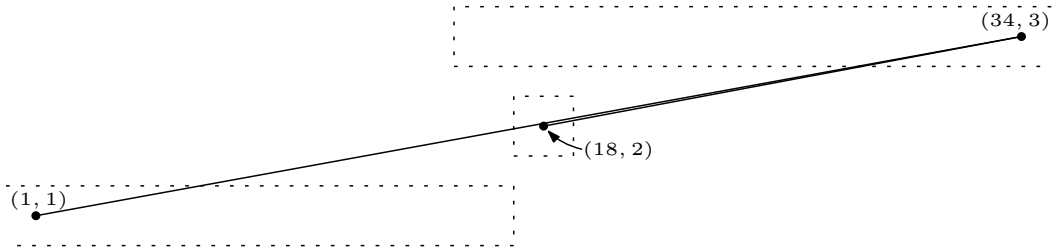
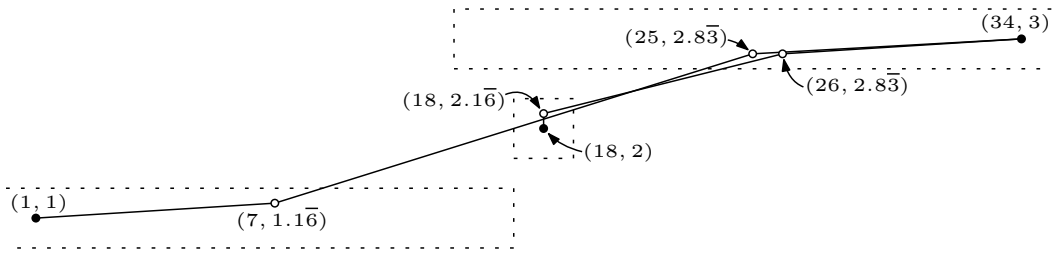


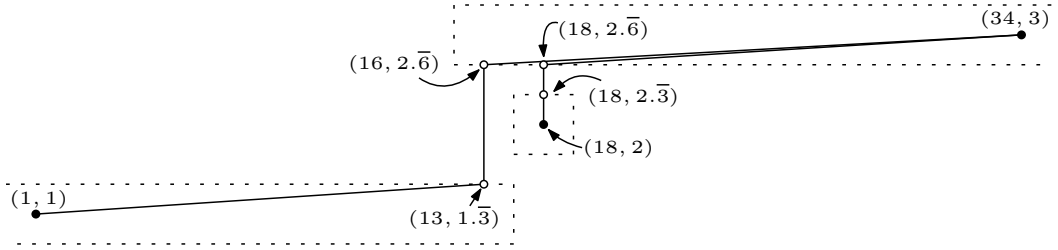
Figure 3.7: An example of a few steps in the construction of Theorem 3.2.4. At each step, the bolded (red) object is being moved, according to the order computed from the visibility ordering. Figure 3.7b denotes Q for the step that morphs the third edge.



(a) An example of a planar straight-line drawing P, with coordinates denoted. P_0 is the equivalent drawing to P that introduces bends coincident with each of the three vertices displayed. The number of coincident bends introduced at each vertex is equal to the degree of that vertex.



(b) At the time $t = 1/2$ in the linear morph from P_0 to P', an intersection occurs.



(c) The admitted planar poly-line drawing P' of D.

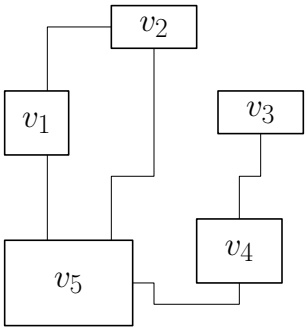
Figure 3.8: A small example of why a linear morph directly from P_0 to P' may not be planarity-preserving. The vertex boxes of D are also drawn at each step (clipped for better visibility).

Chapter 4

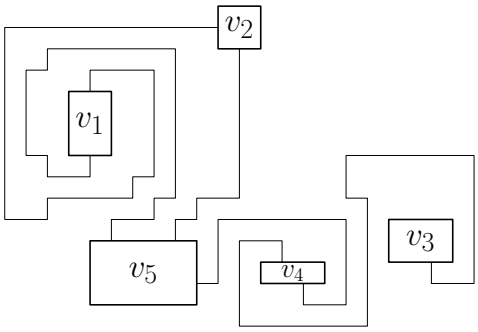
Port Alignment of Orthogonal Box Drawings

The goal of Phase II of the morphing procedure is to arrive at a pair of "parallel" drawings. We say that a pair of orthogonal box drawings P^* and Q^* with no degenerate bends are ***parallel*** if the sequences of edges exiting each vertex box side, and the sequences of turns along each edge, are all the same in P^* and Q^*. If P^* and Q^* have degenerate bends, then we consider them to be parallel if the drawings obtained by removing degenerate bends are parallel. In Phase II, we are given a pair of orthogonal box drawings P and Q, and we wish to morph to a pair of parallel orthogonal box drawings P^* and Q^*. The goal of Phase IIa is to satisfy the first class of conditions of parallel drawings. That is, we are given a pair of orthogonal box drawings P and Q, and we wish to morph to a pair of orthogonal box drawings P' and Q' such that the sequences of ports along each vertex box side is the same in P' and Q'.

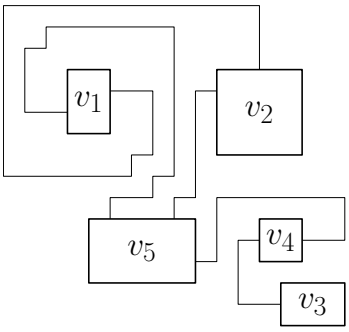
For a vertex v of a graph G, and two compatible planar orthogonal box drawings P and Q of G with no degenerate bends, we say P and Q are ***port aligned*** at v if the sequences of ports along each side of the vertex boxes $P(v)$ and $Q(v)$ are the same. If, dependent on v, we can find an orthogonal rotation (i.e., by $0°$, $90°$, $180°$, or $270°$) of Q such that the sequence of ports along each side of $P(v)$ and each corresponding side of the rotated $Q(v)$ is the same, then we say that P and Q are ***angle aligned*** at v. Note that this is a weaker condition. We also simply say that P and Q are ***port aligned*** if they are port aligned at all vertices $v \in V(G)$, and likewise we say that P and Q are ***angle aligned*** if they are angle aligned at all vertices (with potentially different rotations for each vertex). Clearly, if P and Q are parallel, then they are also port aligned (and hence angle aligned). Examples of these conditions are demonstrated in Figure 4.1. In this chapter, we will focus



(a) An orthogonal box drawing D.



(b) An orthogonal box drawing which is port aligned with D.



(c) An orthogonal box drawing which is angle aligned with D, but not port aligned with D.

Figure 4.1: Examples of port aligned and angle aligned drawings, with labelled vertices.

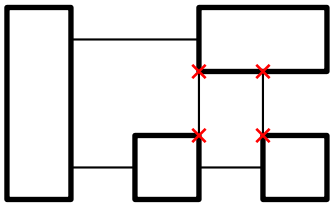
on achieving port alignment, but in the next chapter we will emphasize maintaining angle alignment after having achieved port alignment.

4.1 Obtaining Orthogonal Box Drawings with no Ports at Corners

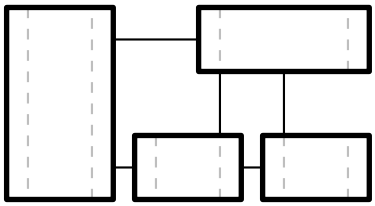
Before directly completing Phase IIa, we will take a brief tangent to discuss a helpful property that can be achieved. Throughout the remainder of the thesis, including this chapter, there are various definitions and algorithms for orthogonal box drawings whose steps are simplified by assuming that there are no ports that coincide with corners. In this short section, we provide a routine that allows us to assume this.

Theorem 4.1.1. *Let P be an orthogonal box drawing of a graph G with n vertices, drawn on an $N \times N$ integer grid Z. Then, there exists an orthogonal box drawing P' with no port-corner coincidences, drawn on an $O(N) \times O(N)$ integer grid Z' and parallel to P, and a planarity-preserving linear morph sequence from P to P' with length $O(1)$, such that every explicit intermediate drawing is also drawn on Z'. Moreover, P' and the linear morph sequence can be computed in $O(n)$ time.*

Whenever we have an orthogonal box drawing, this result will allow us to replace it with an orthogonal box drawing that does not contain any port-corner coincidences.



(a) P



(b) P', with the differing box-sizes of P as gray dashed lines.

Figure 4.2: An example of the main construction step used in Theorem 4.1.1. The crosses mark port-corner coincidences.

We will accomplish the proof by modifying the coordinates of the ***features***—the ports, bends, and corners of an orthogonal box drawing with explicitly defined coordinates.

Proof. We will first remove all port-corner coincidences for ports incident to vertical edge segments, and then argue that by using a symmetric algorithm we can remove port-corner coincidences involving horizontal edge segments.

Let Z be the grid on which P lies, and redraw P using a $(3, 1)$-refinement of Z (equivalently, use a single linear morph to scale P onto a larger grid).

We will move each feature of P to a new x-coordinate on this refinement. We define a drawing P' parallel to P that uses these new x-coordinates, while preserving all y-coordinates. All the following operations are perfomed simultaneously to arrive at the new drawing P':

- For each top-left or bottom-left vertex box corner, decrease its x-coordinate by $\frac{1}{3}$.
- For each top-right or bottom-right vertex box corner, increase its x-coordinate by $\frac{1}{3}$.
- For every port or degenerate bend along the left side of a vertex box that isn't incident to a vertical edge, decrease its x-coordinate by $\frac{1}{3}$.
- For every port or degenerate bend along the right side of a vertex box that isn't incident to a vertical edge, increase its x-coordinate by $\frac{1}{3}$.
- For all other features, maintain the same x-coordinates.

See Figure 4.2 for an example of this construction. Note that, in particular, the top and bottom side of each vertex box are extended by $\frac{1}{3}$ to the left and right, but the locations of

ports along the top and bottom sides (including those that might have been coincident with the corners in P) remain unchanged. Hence, P' contains no port-corner coincidences that involve ports with incident vertical edge segments. By Theorem 2.3.2, the linear morph from P to P' is planarity-preserving.

The previous routine removed all port-corner coincidences involving ports that were incident to vertical edge segments, and it did not introduce any port-corner coincidences at all. Therefore, by the symmetry of x and y-coordinates, we may perform the symmetric routine to remove port-corner coincidences involving ports coincident to horizontal edge segments. In total, this involves $O(1)$ linear morphs (including those used by refinements). Moreover, each drawing can clearly be constructed in linear time. □

As an aside, the above proof is quite similar to the proof of Corollary 3.2.3.

4.2 Port Alignment of Orthogonal Box Drawings with no Ports at Corners

Using the result of the previous section, the main result of Phase IIa will be:

Theorem 4.2.1. *Let A and B be compatible orthogonal box drawings of a graph G drawn on an $N \times N$ grid. Then there exists an orthogonal box drawing A' and a sequence of linear morphs of length $O(n)$ from A to A' such that A' and B are port aligned. Furthermore, A' and all the explicit intermediate drawings in the sequence are drawn on an $(N + O(n)) \times (N+O(n))$ grid, and have at most $O(1)$ additional bends per edge compared to A. Moreover, A' and the sequence can be found in $O(n^2)$ time.*

For the remainder of chapter, since the main construction will work closely with ports, it is helpful to briefly introduce some notation. For an orthogonal box drawing D with a vertex v and an edge e incident to v, we let $\boldsymbol{D(e, v)}$ denote the port of e along the vertex box $D(v)$.

The technique for proving this theorem will be to morph ports around corners. Without loss of generality, it will suffice to assume that a port is being morphed around the top left corner counter-clockwise, since all other cases of a port being morphed around a corner can be achieved by symmetry.

Lemma 4.2.2. *Let D be an orthogonal box drawing of a graph G drawn on an $N \times N$ grid, such that D has no port-corner coincidences. Let $e \in E(G)$ be an edge incident*

to a vertex $v \in V(G)$ so that the port $D(e, v)$ is the leftmost port along the top side of the vertex box $D(v)$. Then, there exists an orthogonal box drawing D' of G drawn on an $(N + O(1)) \times (N + O(1))$ grid, with $O(1)$ additional bends along the edge e (and no other additional bends), so that the sequence of ports along each side of each vertex box is the same in both drawings, except that $D'(e, v)$ is instead the topmost port along the left side of the vertex box $D'(v)$.

Furthermore, there is a planarity-preserving sequence of linear morphs from D to D' of length $O(1)$, such that every explicit intermediate drawing in the sequence is also drawn on an $(N + O(1)) \times (N + O(1))$ grid, and also has $O(1)$ additional bends along the edge e (and no other additional bends). Moreover, both D' and the sequence can be computed in $O(n)$ time.

Proof. The final planarity-preserving sequence of linear morphs will be labelled $D = D_1$, $D_2, D_3, D_4, D_5, D_6, D_7 = D'$. The explicit construction is given here, and a visual example can be found in Figure 4.3.

- Let x^* be the x-coordinate of the left side of the vertex box of v. Create D_2 by shifting every feature (bend, port, or vertex box corner) strictly to the left of the line $x = x^*$ to the left by 1, so no feature will have x-coordinate $x^* - 1$ afterwards. The linear morph from D_1 to D_2 is a horizontal linear morph, and it is planarity-preserving by Theorem 2.3.2.

- Let y^* be the y-coordinate of the top side of the vertex box of v. Create D_3 by shifting every feature strictly above the line $y = y^*$ up by 1, and every feature below the line $y = y^*$ down by 1, so no feature will have y-coordinate $y^* - 1$ or $y^* + 1$ afterwards. The linear morph from D_2 to D_3 is a vertical linear morph, and it is also planarity-preserving by Theorem 2.3.2.

- Let D_4 be the equivalent drawing to D_3 that introduces the bends b_1, b_2, b_3 just after the port $p = D_3(e, v)$, so that the location of b_1 coincides with p, and the locations of b_2 and b_3 coincide with each other, at a unit distance upwards from p.

- D_5 can be constructed by modifying only the locations of p, b_1 and b_2. Specifically, they should all be shifted horizontally to align with the left side of the vertex box. The linear morph from D_4 to D_5 is a horizontal linear morph, and it is planarity-preserving since e was assumed to be the leftmost edge along the top side of the vertex box, and since there are guaranteed to be no conflicting bends, ports, or vertex box corners along the path.

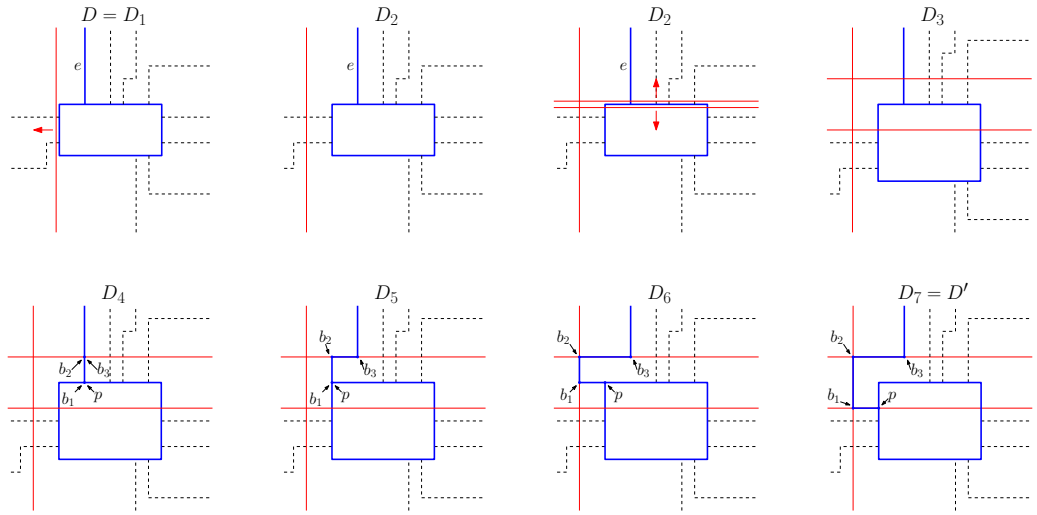


Figure 4.3: A port can be morphed around a corner using exactly 6 linear morphs.

- D_6 can be constructed by modifying only the locations of b_1 and b_2. Specifically, they should both be shifted horizontally to the left by a unit distance. The linear morph from D_5 to D_6 is again a planarity-preserving linear morph since there are guaranteed to be no conflicting bends, ports, or vertex box corners along the path.
- Finally, $D_7 = D'$ can be constructed by modifying only the locations of b_1 and p. Specifically, they should be shifted down by a unit distance (i.e., to $y^* - 1$). By the construction of D_3, no feature has y-coordinate equal to $y^* - 1$ in D_6 (since none were moved to occupy that y-coordinate), so this does not cause any intersections. The linear morph from D_6 to D_7 is a vertical linear morph, and it is planarity-preserving again for the same reason.

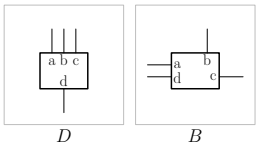
The first two steps increase the size of the grid by $O(1)$. The third step adds $O(1)$ bends. Each explicit intermediate drawing can be computed in $O(n)$ time, and there are $O(1)$ such drawings, so the total time complexity is $O(n)$ to compute the linear morph sequence. □

Repeatedly applying this lemma on appropriate vertices will result in a proof of Theorem 4.2.1:

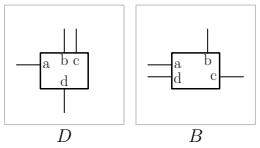
Proof. Without loss of generality by Theorem 4.1.1, we may assume A does not have any coinciding port-corner pairs.

The algorithm is quite simple: Let $D_0 := A$. For increasing values of i, consider each side of each vertex box in D_i with some edges along it, and consider the most counter-clockwise edge along that side (i.e., leftmost along top, bottommost along left, rightmost along bottom, topmost along right). If e is not the most counter-clockwise edge along that side in both D_i and B (or if that side does not contain edges in B), then modify D_i using Lemma 4.2.2 to morph e around the next corner counter-clockwise, and call the result D_{i+1}. After performing this for enough values of i, we guarantee that, for each side of the vertex box in D that is non-empty, the most counter-clockwise edge along that side is the same in both D_i and B. Once this condition is satisfied, repeat the procedure using the most clockwise edges along each side. Once the second procedure has completed, output $A' := D_k$, where k is the largest value for which D_k is computed (i.e., the final value of i). Note that the second procedure won't invalidate the guarantee of the first procedure along any side (the only unusual case is when a side has only one edge in B, in which case it is both the most clockwise and counter-clockwise edge on that side). So, since the most clockwise and most counter-clockwise edges along each side match, the final drawings A' and B must be port aligned.
During each of the two procedures, no port will move around more than 4 corners, since at least one port along the vertex box must be the most-counter-clockwise (or most-clockwise) edge along some side, blocking other ports from moving around that corner. Hence, each port will move around at most $8 = O(1)$ corners, so Lemma 4.2.2 will be applied at most $O(1)$ times per edge, or $O(n)$ times in total. Thus, there will be at most $O(1)$ bends additional bends per edge in the final drawing D_k, and at most $O(n)$ additional coordinates for a final grid size of $N + O(n) \times N + O(n)$. Moreover, the total time to compute all these linear morphs is $O(n^2)$. □

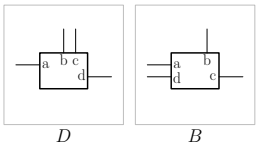
See Figure 4.4 for an example of this construction.



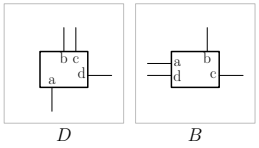
(a) The ports along D will first be greedily rotated counter-clockwise.



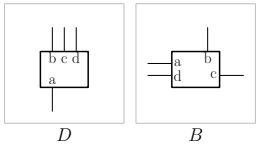
(b) The port a is moved counter-clockwise around a corner.



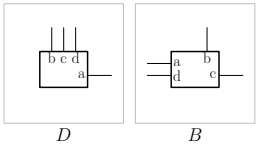
(c) The port d is moved counter-clockwise around a corner.



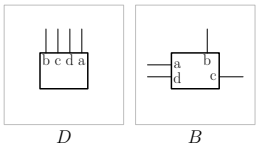
(d) The port a is moved.



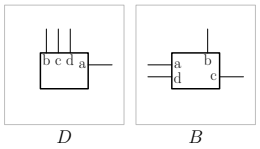
(e) The port d is moved.



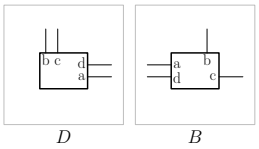
(f) The port a is moved.



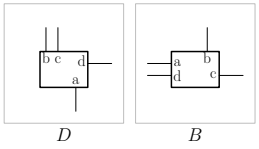
(g) Counter-clockwise moves have been completed. Clockwise moves now begin.



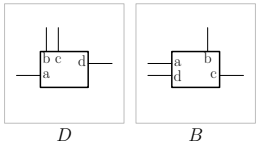
(h) The port a is moved clockwise around a corner.



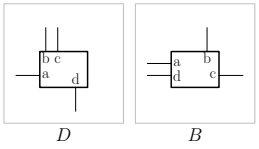
(i) The port d is moved clockwise around a corner.



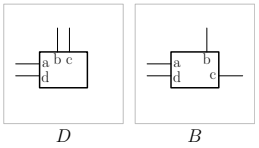
(j) The port a is moved.



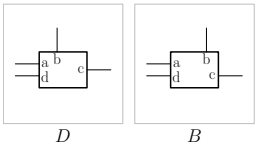
(k) The port a is moved.



(l) The port a is moved.



(m) The port d is moved.



(n) The final port c is moved The resulting drawings D and B are port aligned at the vertex.

Figure 4.4: An example of the construction used for proving Theorem 4.2.1, used at a single vertex. This a representation of the relative port locations along the vertex box, but not the exact linear morphs or explicit drawings.

Chapter 5

Spirality and Twists of Orthogonal Box Drawings

The goal of Phase II of the morph is to obtain parallel orthogonal box drawings. In addition to port alignment, the second and final condition for a pair of orthogonal box drawings to be parallel is that the sequence of turns along the path defined by each edge (along some arbitrary direction) is the same. In order to unify these sequences of turns, it is possible to adapt machinery developed by Biedl et al. [7] for morphing orthogonal point drawings. Most of this machinery is adapted to form Phase IIb, discussed in this chapter, while the rest is adapted to form Phase IIc.

There are three main steps for discussing Phase IIb. First, a (signed) measure of similarity between two edges will be defined, called "spirality". Second, a high-level operation for adjusting the spirality of edges will be discussed, called "twists". In addition, a new method for adjusting the spirality of every edge simultaneously will be discussed, called "simultaneous twists". Lastly, a method for choosing twists will be discussed which unifies the spirality of all edges. Besides the use of simultaneous twists, each of these steps is also performed in a similar manner by Biedl et al. [7] for orthogonal point drawings. However, a large number of small adaptations needed to be made for orthogonal box drawings and simultaneous twists, so these techniques are explained in full in this chapter. Later, in Section 6.2, it will be shown that equal spirality along each edge can be used to obtain equal sequences of turns along each edge.

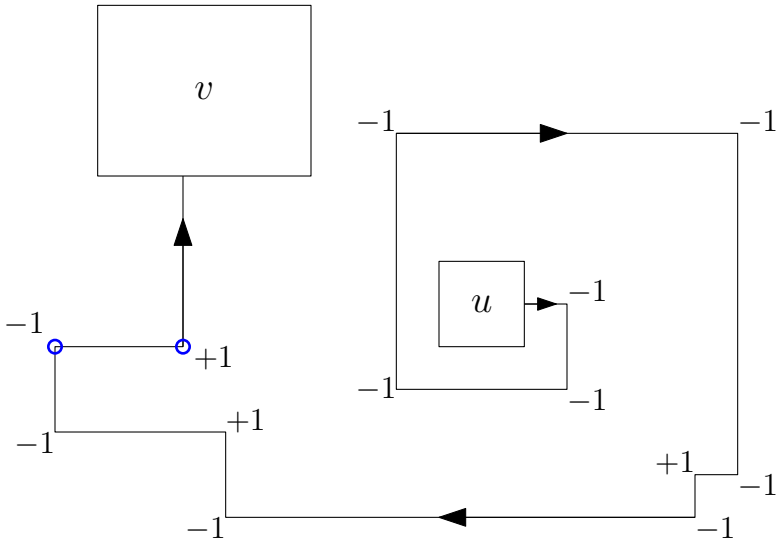


Figure 5.1: An example of how the spirality is computed for an edge uv, oriented from u to v. The spirality of the edge is $s_D(u, v) = -7$.

5.1 Spirality

Spirality is a method of aggregating the sequence of turns along an edge, and it can be used to define a measure of similarity between edges in a pair of orthogonal box drawings. This section revises concepts devised by Biedl et al. [7] for orthogonal point drawings into corresponding concepts for orthogonal box drawings.

Notation 5.1.1. *Let D be an orthogonal box drawing with no degenerate bends. Let $e = uv$ be an edge. Then the **spirality** $s_D(u, v)$ of e in D is the number of left turns minus the number of right turns encountered while traversing e from u to v.*

See Figure 5.1 for an example of how spirality is computed along an edge. Spirality is a discrete analogue to the turning number or total curvature of a smooth curve.

In order to discuss the relationship between spirality and pairs of parallel drawings, the following notion is also required:

Definition 5.1.2. *Let D be an orthogonal box drawing. D is said to be **zig-zag-free** if it has no degenerate bends, and for each edge $e = uv$, there are either only left turns or only right turns along e.*

The name "zig-zag-free" comes from an equivalent condition: In the sequence of turns along the edge, there is no pair of adjacent left and right turns (a ***zig-zag***). For zig-zag-free drawings, the spirality of an edge is enough to determine its exact sequence of left and right turns. The drawing in Figure 5.1 is not zig-zag-free—the two circled bends form a zig-zag (and there are several others too).

Now, in order to compare two different drawings, the spirality of each edge needs to be compared:

Definition 5.1.3. *For any two vertices u and v sharing an edge in G, and any two compatible orthogonal box drawings P and Q of G, the* ***difference in spirality*** *from u to v is* $\boldsymbol{\Delta s_{P,Q}(u,v) := s_Q(u,v) - s_P(u,v)}$.

The following simple definition is also of importance:

Definition 5.1.4. *The* ***maximum absolute difference in spirality*** *between two orthogonal box drawings P and Q of the same graph G is* $\max_{uv \in E(G)} |\Delta s_{P,Q}(u,v)|$.

There is a key relationship between the difference in spirality and the property of port alignment:

Observation 5.1.5. *Let P and Q be compatible port aligned orthogonal box drawings of the same graph G. Then, for each edge $e = uv$, $\Delta s_{P,Q}(u,v) \equiv 0 \pmod 4$.*

As a second observation, there is an important relationship between port alignment, zig-zag-free, and the maximum absolute difference in spirality:

Lemma 5.1.6. *Let P and Q be two compatible port aligned zig-zag-free orthogonal box drawings of the same graph. If the maximum absolute difference in spirality between P and Q is 0, then P and Q are parallel.*

Proof. Let $e = uv$ be an edge. Since P and Q are zig-zag-free, the sequence of turns along e is given by the spirality of e in P and Q respectively. The maximum absolute difference in spirality is 0, so $\Delta s_{P,Q}(u,v) = 0$, and hence the spirality of e is the same in P and Q. Thus, the sequence of turns along e is the same in both P and Q. Since the two drawings are port aligned, they are also parallel. □

In order to apply Lemma 5.1.6, one needs to be able to find zig-zag-free drawings, and to be able to reduce the maximum absolute difference in spirality. Reducing the maximum absolute difference in spirality will be discussed in the succeeding sections of this chapter, while finding zig-zag-free drawings is a nearly-solved problem in previous literature, and will be discussed in Section 6.2.

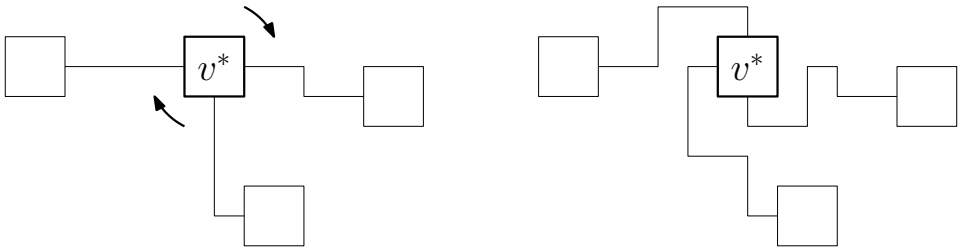


Figure 5.2: An example of two drawings forming a clockwise twist at a vertex.

5.2 Twists

In order to reduce the maximum absolute difference in spirality between a pair of drawings, the spirality of edges in one drawing needs to be changed. A ***twist*** will be a type of linear morph that changes the spirality of the edges around some vertex. A ***simultaneous twist*** will be a type of linear morph that changes the spirality of edges around multiple vertices. The former was used by Biedl et al. [7] for orthogonal point drawings, while the latter is new.

A twist is just what it sounds like: It is a method for (locally) twisting a vertex box (or set of vertex boxes), without changing the drawing globally, in a sense.

Definition 5.2.1. *Let P and Q be angle aligned orthogonal box drawings of a graph G. Let $v^* \in V(G)$. P and Q are said to form a* ***clockwise twist*** *at v^* if:*

- *For each vertex $v \in V(G) \setminus \{v^*\}$, P and Q are port aligned at v.*
- *Let $\hat{P}$ be a drawing obtained by rotating the entirety of P clockwise $90°$. Then $\hat{P}$ and Q are port aligned at v^*.*
- *For each edge $e = uv$ with $u, v \neq v^*$, the spirality of e is unchanged, i.e., $\Delta s_{P,Q}(u, v) = 0$.*
- *For each edge $e = uv^*$, the spirality of e is changed so that one additional net left turn is introduced, i.e., $\Delta s_{P,Q}(v^*, u) = 1$.*

P and Q are said to form a ***counter-clockwise twist*** *at v^* if the analogous conditions hold (rotation becomes $90°$ counter-clockwise and additional right turns are introduced).*

It turns out that applying twists as defined above can result in a linear morph sequence with length up to $\Theta(n^2)$, even for the analogous problem of morphing orthogonal point drawings [7]. Such a sequence would be too long for our purposes. Van Goethem et al. [47] overcame this issue for orthogonal point drawings, but their techniques would be difficult (perhaps impossible) to use in a manner that would achieve our desired time complexity and grid size in the final linear morph sequence. Instead, we will perform several twists at different vertices simultaneously to overcome this issue.

In order to define a "simultaneous twist", we add one necessary piece of input: A ***twist assignment*** is a function $t : V(G) \to \{-1, 0, 1\}$ that represents the desired direction to twist each vertex. A -1 represents a clockwise twist, a 1 represents a counter-clockwise twist, and a 0 represents no twist.

Definition 5.2.2. *Let P and Q be angle aligned compatible orthogonal box drawings of a graph G. Let $t : V(G) \to \{-1, 0, 1\}$ be a twist assignment. P and Q are said to form a* ***simultaneous twist*** *of the twist assignment t if:*

- *For each vertex $v \in V(G)$, let $\hat{P}_v$ be the drawing obtained by rotating the entire drawing P 90° clockwise, 90° counter-clockwise, or not at all, in the cases of $t(v) = -1$, $t(v) = 1$, and $t(v) = 0$ respectively. Then $\hat{P}_v$ and Q are port aligned at v.*
- *For each edge $e = uv$, the spirality of e is changed to respect the (possible) rotations of both u and v according to t, i.e., $\Delta s_{P,Q}(u, v) = t(u) - t(v)$.*

The reason for separating out an abstract definition of a "twist" like this, rather than presenting an explicit algorithm to construct a twist, is two-fold: Firstly, the algorithms and results which actually select which twists need to be performed do not need all the additional complexity of a specific implementation, and can be presented quite abstractly. Secondly, the actual "implementations" of twists that have all the qualities that are necessary for proving Theorem 2.5.1 are quite technical (details will be discussed in Chapter 7), so the reader may find it helpful to be able to consider the two problems separately.

Using the abstract notion of a twist, it becomes much simpler to discuss which twists should be chosen. Given a pair of compatible orthogonal box drawings P and Q, the goal of using twists will not exactly be to strictly reduce the maximum absolute difference in spirality between some P' and Q at each step. Rather, the goal will be to find a sequence of twists which *eventually* result in a maximum absolute difference in spirality of 0 between a final drawing P' and Q, while also guaranteeing that P' is port aligned with Q. In other words, the goal is to satisfy the pre-conditions of Lemma 5.1.6. Choosing twists that accomplish this is the topic of the next section.

5.3 Choosing Twists

It has been established in the previous section that twists (and simultaneous twists) change the spirality of edges, and the manner in which spirality is changed for each type of twist has been determined. The goal of Phase IIb is to show that, for a pair of port aligned orthogonal box drawings P and Q, a sequence of twist assignment functions exists so that any corresponding sequence of drawings starting with P, forming a sequence of corresponding simultaneous twists, is guaranteed to end with a drawing parallel to Q. In other words, twists will be chosen which guarantee a resulting pair of parallel drawings, satisfying the conditions of Lemma 5.1.6.

In order to obtain such a sequence of twist assignment functions, we will show that all solutions to a carefully chosen system of equations have the intended guarantees. Then, it will be shown that the system of equations admits a solution consisting of $O(n)$ twist assignment functions. Biedl et al. [7] used a similar system of equations and corresponding proofs to derive a sequence of $O(n^2)$ twists for orthogonal point drawings.

The system of equations will involve a sequence of twist assignment functions denoted $t_1, t_2, \cdots : V(G) \to \{-1, 0, 1\}$. It involves two types of constraints:

- For each edge $uv \in E(G)$, $\Delta s_{P,Q}(u, v) + \sum_i t_i(v) - \sum_i t_i(u) = 0$ (constraint type 1).
- For each vertex $v \in V(G)$, $\sum_i t_i(v) \equiv 0 \pmod 4$ (constraint type 2).

Let P' be some drawing that is the result of a sequence of simultaneous twists, starting from P, corresponding to the sequence of twist assignment functions. The constraints of type 1 imply that the maximum absolute difference in spirality between P' and Q is 0, and the constraints of type 2 imply that P' and Q are port aligned. The latter can be seen by the fact that $\sum_i t_i(v)$ expresses how many times the box of v is rotated 90° counter-clockwise (or counter-clockwise if it is negative).

Observe that the constraints do not place any restrictions on individual twist assignment functions, but rather operate only on the sums $\sum_i t_i(v)$ of twist assignment functions for each vertex $v \in V(G)$, indicating the (net) number of times and direction that each vertex is twisted. Rather than computing a sequence of twist assignment functions directly, it is simpler to compute and discuss these values, which will be referred to as the values of a ***cumulative twist assignment function*** $\dot{t}$, and will be denoted $\dot{t}(v) \in \mathbb{Z}$ for each $v \in V(G)$. Given a cumulative twist assignment function $\dot{t}$, it's trivial to greedily compute a sequence of twist assignment functions $\{t_i\}_i$ satisfying $\sum_i t_i(v) = \dot{t}(v)$. A more careful choice of functions will need to be chosen in order to keep the number of bends along each

edge low, but that choice will be discussed in the proof of Theorem 5.3.3. The constraints of both types can now be restated for a cumulative twist assignment function:

- For each edge $uv \in E(G)$, add a constraint that $\Delta s_{P,Q}(u,v) + \dot{t}(v) - \dot{t}(u) = 0$ (constraint type 1).
- For each vertex $v \in V(G)$, add a constraint that $\dot{t}(v) \equiv 0 \pmod 4$ (constraint type 2).

It remains to be shown that the system of equations is feasible, and that it can be solved efficiently. We first show that the constraints of type 1 are satisfied for every edge provided that they are satisfied for the edges of a maximal spanning forest (i.e., a spanning tree for each connected component).

Lemma 5.3.1. *Let P and Q be compatible port aligned orthogonal box drawings of the same graph G. Let $\dot{t}$ be a cumulative twist assignment function satisfying the constraints of type 1 for each edge in a maximal spanning forest of G. Then, $\dot{t}$ satisfies the constraints of type 1 for every edge.*

Proof. Let $u^*v^* \in E(G)$ be some edge that is not part of the maximal spanning forest of G. Together with some minimal subset of edges from the maximal spanning forest of G, it must form a cycle C within G. Direct C in G so that it becomes counterclockwise in any admitted drawing of P or Q, i.e., the bounded region enclosed by C lies to its left (when traversed according to the edge directions) in each. Note that this is possible because P and Q are compatible drawings, and hence clockwise-oriented faces must be the same (while C is assembled from such faces).

For each drawing P and Q, consider the closed orthogonal path in the plane formed by repeatedly moving along each edge in C, and then moving counter-clockwise around each vertex box encountered until the port of the next edge in C is reached. See Figure 5.3 for an example of this traced path. This path is simple in both drawings, by planarity.

A theorem of Vijayan and Wigderson [48] states that any planar orthogonal cycle has 4 more left turns than right turns when traversed counter-clockwise. Both the traced paths are planar orthogonal cycles traversed counter-clockwise. Note that the left and right turns encountered at each vertex box are identical along both traced paths, since P and Q are port aligned. Therefore, if these turns are ignored, the number of left turns minus the number of right turns must still be the same along both paths. Omitting these terms results in the sum of spiralities along each edge of C within each of P and Q. Thus, if the sequence of

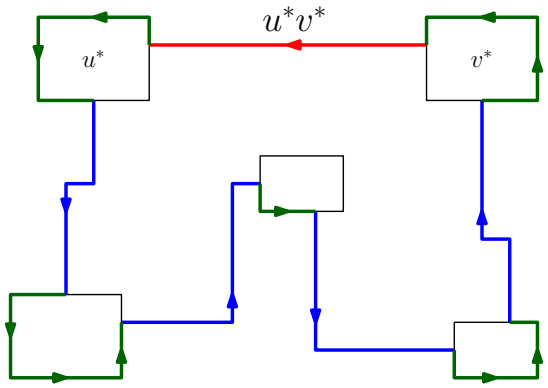


Figure 5.3: The planar orthogonal cycle which is used in the proof of Lemma 5.3.1 to show that the edge u^*v^* has the same spirality in both drawings.

vertices in C is $v^* = v_1, v_2, \ldots, v_k = u^*$, we have $\Delta s_{P,Q}(v_k, v_1) + \sum_{i=1}^{k-1} \Delta s_{P,Q}(v_i, v_{i+1}) = 0$. This implies

$$\begin{aligned}\Delta s_{P,Q}(u^*, v^*) &= -\sum_{i=1}^{k-1} \Delta s_{P,Q}(v_i, v_{i+1}) \\ &= \big(-\dot{t}(v_1) + \dot{t}(v_2)\big) + \big(-\dot{t}(v_2) + \dot{t}(v_3)\big) + \cdots + \big(-\dot{t}(v_{k-1}) + \dot{t}(v_k)\big) \\ &= \dot{t}(u^*) - \dot{t}(v^*),\end{aligned}$$

so the constraint of type 1 on the edge u^*v^* is satisfied. □

In addition, the constraints of type 2 can be satisfied easily as well.

Lemma 5.3.2. *Let P and Q be compatible port aligned orthogonal box drawings of the same graph G. Let $\dot{t}$ be a cumulative twist assignment function satisfying the constraints of type 1 for every edge, and satisfying the constraints of type 2 for at least one vertex in each connected component of G. Then, $\dot{t}$ satisfies the constraints of type 2 for all vertices.*

Proof. The proof is inductive. Let v be a vertex whose neighbour u satisfies its constraint of type 2. By assumption, the edge uv satisfies its constraint of type 1. P and Q are port aligned, so Observation 5.1.5 implies that $\Delta s_{P,Q}(u, v) \equiv 0 \pmod 4$. Therefore, $\dot{t}(v) = \dot{t}(u) - \Delta s_{P,Q}(u, v) \equiv 0 + 0 \pmod 4$. □

The main result of Phase IIb for computing a sequence of twist assignment functions is as follows:

Theorem 5.3.3. *Let P and Q be compatible port aligned orthogonal box drawings of the same graph G. Assume that each of P and Q have $O(1)$ bends per edge. Then, there exist a sequence of $k \in O(n)$ twist assignment functions $t_1, t_2, \ldots, t_k$, such that any sequence of orthogonal box drawings $P = P_0, P_1, \ldots, P_k$, where each P_{i-1}, P_i form a simultaneous twist of the twist assignment t_i, has the following properties:*

- *P_k and Q are port aligned.*
- *The maximum absolute difference in spirality between the final drawing P_k and Q is zero.*
- *For each edge $uv \in E(G)$, the absolute value of the difference in spirality with Q monotonically decreases throughout the sequence (i.e., $|\Delta s_{P_{i-1},Q}(u,v)| \geq |\Delta s_{P_i,Q}(u,v)|$ for all $0 \leq i \leq k$).*

Moreover, the sequence of twist assignment functions can be computed in $O(n^2)$ time.

Note that the requirement that each adjacent pair forms a simultaneous twist also implies that all drawings in the sequence $P_0, \ldots, P_k$ are angle aligned. In this sense, we must "maintain" angle-alignment when using this result, as mentioned in Chapter 4.

Proof. Given a pair of compatible port aligned orthogonal box drawings P and Q of the same graph G, Lemma 5.3.1 and Lemma 5.3.2 together imply a straightforward algorithm for computing a cumulative twist assignment function $\dot{t}$ satisfying all constraints of types 1 and 2: Start with a maximal spanning forest of G, and root each of its trees. For each root $r \in V(G)$, choose $\dot{t}(r) := 0$, so the constraint of type 2 at r is satisfied. Then, perform a pre-order traversal of each tree, and set $\dot{t}(v) := \dot{t}(u) - \Delta s_{P,Q}(u,v)$ for a vertex v with parent u. This choice satisfies all constraints of both types by Lemma 5.3.1 and Lemma 5.3.2. This choice of $\dot{t}$ can be computed in $O(n)$ time, and its maximum absolute value is bounded above by $\sum_{uv \in E(G)} |\Delta s_{P,Q}(u,v)|$.

Choose $k = \max_{v \in V(G)} |\dot{t}(v)|$. For every $uv \in E(G)$, $\Delta s_{P,Q}(u,v)$ is at most the maximum number of bends in P and Q across the edge uv, which is $O(1)$. Therefore, $|\dot{t}(v)| \leq \sum_{uv \in E(G)} |\Delta s_{P,Q}(u,v)| \in O(n)$ for every vertex $v \in V(G)$. For the the ith twist assignment function t_i (where $i \in \{1, \ldots, k\}$), choose

$$t_i(v) := \begin{cases} 1 & \text{if } \dot{t}(v) \geq i \\ -1 & \text{if } \dot{t}(v) \leq -i \\ 0 & \text{if } -i < \dot{t}(v) < i \end{cases}.$$

It can be verified that $\sum_{i=1}^{k} t_i(v) = \dot{t}(v)$, so all the constraints of type 1 and 2 are satisfied for the twist assignment functions. Therefore, the maximum absolute difference in spirality between P_k and Q is zero, and P_k and Q are port aligned. It remains to be shown that the absolute value of the difference in spirality with Q of each edge is monotonically decreasing. Let $uv \in E(G)$ be an arbitrary edge, and assume without loss of generality that $\dot{t}(u) \geq \dot{t}(v)$, else swap u and v. Consequently, by the constraint of type 1 on uv, $\Delta s_{P,Q}(u, v) = \dot{t}(u) - \dot{t}(v) \geq 0$. We claim that $t_i(u) \geq t_i(v)$ for all $i \in \{1, \ldots, k\}$:

- If $\dot{t}(v) \geq 0$, then $t_i(u) = t_i(v)$ for all $i \in \{1, \ldots, \dot{t}(v)\} \cup \{\dot{t}(u) + 1, \ldots, k\}$. For $i \in \{\dot{t}(v) + 1, \ldots, \dot{t}(u)\}$, $t_i(v) = 0 < 1 = t_i(u)$.
- If $\dot{t}(u) \leq 0$, then $t_i(u) = t_i(v)$ for all $i \in \{1, \ldots, -\dot{t}(u)\} \cup \{-\dot{t}(v) + 1, \ldots, k\}$. For $i \in \{-\dot{t}(u) + 1, \ldots, -\dot{t}(v)\}$, $t_i(v) = -1 < 0 = t_i(u)$.
- If $\dot{t}(u) \geq 0 \geq \dot{t}(v)$, then $t_i(u) \geq 0 \geq t_i(v)$ for all $i \in \{1, \ldots, k\}$.

Hence, for all $i \in \{1, \ldots, k\}$,

$$\begin{aligned}
\Delta s_{P_i,Q}(u, v) &= s_Q(u, v) - s_{P_{i-1}}(u, v) + s_{P_{i-1}}(u, v) - s_{P_i}(u, v) \\
&= \Delta s_{P_{i-1},Q}(u, v) + t_i(v) - t_i(u) \\
&\leq \Delta s_{P_{i-1},Q}(u, v).
\end{aligned}$$

Therefore, since $\Delta s_{P_0,Q}(u, v) \geq \Delta s_{P_1,Q}(u, v) \geq \cdots \geq \Delta s_{P_{k-1},Q}(u, v) \geq \Delta s_{P_k,Q}(u, v) = 0$, we have $|\Delta s_{P_i,Q}(u, v)| \geq \left|\Delta s_{P_{i-1},Q}(u, v)\right|$, as desired. □

As an aside, note that the length k of this computed linear morph sequence would be equal to the maximum absolute value of all the cumulative twist counts, and such a sequence can then be computed in time $O(kn)$. For general planar graphs, we have shown that $k \in O(n)$ is possible. Prior work has shown that this is tight, i.e., that there are examples where $k \in \Omega(n)$ always (as discussed in Chapter 1). However, there are special cases of graphs (e.g., those with low-diameter spanning trees) where smaller bounds are possible.

Chapter 6

Simplifying and Compressing Orthogonal Box Drawings

Phase IIc consists of two parts: Performing twists, and the simplification and compression of the drawing. This chapter discusses the latter. During the process of generating a sequence of linear morphs, it is possible to blow up the complexity of the drawings over time. For example, if an operation (such as performing twists) takes an orthogonal box drawing A_i drawn on an $N \times N$ grid with at most c bends per edge, and produces an orthogonal box drawing A_{i+1} drawn on an $(N+\Omega(n))\times(N+\Omega(n))$ grid with $\Omega(1)$ additional bends per edge, then a mere $\omega(1)$ applications of this operation will result in a drawing already on an $\omega(N+n) \times \omega(N+n)$ grid, with $c+\omega(1)$ bends per edge. As an extreme (but relevant) example, $\Omega(n)$ applications of this operation will result in a drawing on an $\Omega(N+n^2) \times \Omega(N+n^2)$ grid, with $c+\Omega(n)$ bends per edge. Since it is a primary goal of this work (in particular, for proving Theorem 2.5.1) to maintain an $O(n) \times O(n)$ grid and $O(1)$ bends per edge, these values are too large.

In this chapter, it will be shown how to simplify orthogonal box drawings by reducing the number of bends per edge. This will be accomplished using a sequence of linear morphs produced by repeatedly applying an operation called ***zig-zag elimination*** to make drawings zig-zag-free (as defined in Section 5.1). Additionally, it will be shown how to reduce the size of the grid on which an orthogonal box is drawn by ***compressing*** the drawings. In fact, the methods for accomplishing each of these are similar to each other.

The techniques used in this chapter could be classified as a simple form of "refinement" techniques for orthogonal box drawings, which constitute a large field (see the fairly comprehensive experimental paper by Six, Kakoulis, and Tollis [37], the "4M-algorithm" by

Fößmeier and Heß and Kaufmann [25], or Lengauer's textbook [32] for a broader discussion of integrated circuit layout). Crucially however, the techniques in this chapter additionally enable us to find morphs, which is not the problem of interest for refinement techniques.

6.1 Compressions

The main idea of performing compressions is that, given an orthogonal box drawing A of a graph G with at most $O(1)$ bends per edge, drawn on an $N \times N$ grid, there exists a sequence of $O(1)$ linear morphs from A into an orthogonal box drawing A' of the same graph G with the same number of bends along each edge, except that A' is drawn on an $O(n) \times O(n)$ grid. Specifically, such a sequence will involve two unidirectional linear morphs, one horizontal and one vertical.

In other words, if some operation (e.g., performing twists) takes a drawing drawn on an $N \times N$ grid and produces one drawn on an $N + \Theta(n) \times N + \Theta(n)$ grid, then a compression can be performed afterwards to obtain a (parallel) drawing on an $O(n) \times O(n)$ grid. The methods we will use to accomplish this are similar to "retraction" linear morphs used by Biedl et al. [7], except that we use the language of orthogonal box drawings, and discuss time complexity.

This is not a surprising result: The number of distinct x or y coordinates used by features (bends, ports, vertex box corners) in an orthogonal box drawing with $O(1)$ bends per edge must already be $O(n)$. So, if $N \in \omega(n)$, then there must be unused values smaller than the maximum, leaving parts of the grid unused.

Lemma 6.1.1. *Let P be an orthogonal box drawing of an n-vertex graph G drawn on an $N \times N$ grid with $O(1)$ bends per edge. Then, there exists an orthogonal box drawing P' of the same graph G drawn on an $O(n) \times N$ grid which is parallel to P, such that the linear morph from P to P' is planarity-preserving, and it is a horizontal linear morph. Moreover, P' can be found in $O(\min\{N + n, n \log n\})$ time.*

Proof. The idea for this proof is simply to "delete" the unused x-coordinates. For each feature, store the value of its x-coordinate (along with a reference to the feature) in an array called XCoords. There are $O(n)$ features and hence $\min\{O(n), N\}$ unique x-coordinates in total. Next, sort XCoords. This takes $O(\min\{n \log n, n + N\})$ time, which is the minimum time to perform either a comparison sort or a bucket sort. Iterate through XCoords and mark groups with identical x-coordinates. This data structure gives us a simple way to assign new x-coordinates: We will create a new drawing P' by modifying

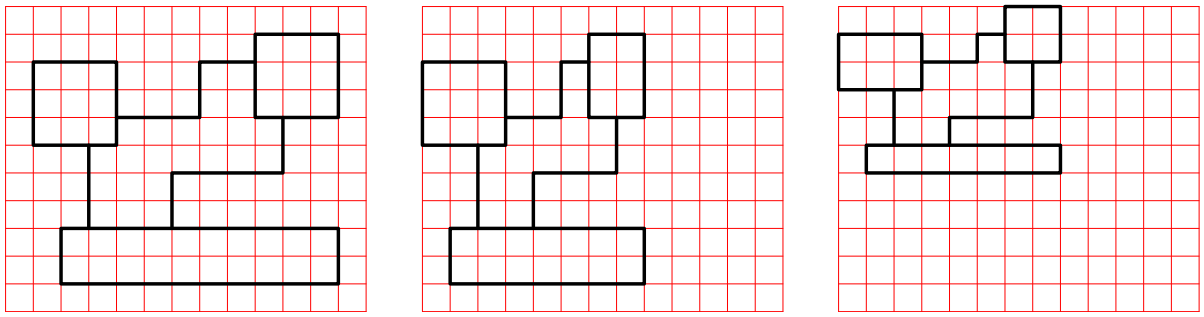


Figure 6.1: An example of horizontal, and then vertical compression.

the x-coordinates of each features in P, while preserving y-coordinates. Specifically, if a feature's x-coordinate in P corresponds is part of the ith group in XCoords after sorting, then its x-coordinate in P' is equal to i exactly. The maximum group index will be $O(n)$, so P' is drawn on an $O(n) \times N$ grid. We have preserved the order of all x-coordinates, so by Theorem 2.3.2 the linear morph from P to P' is planarity-preserving. □

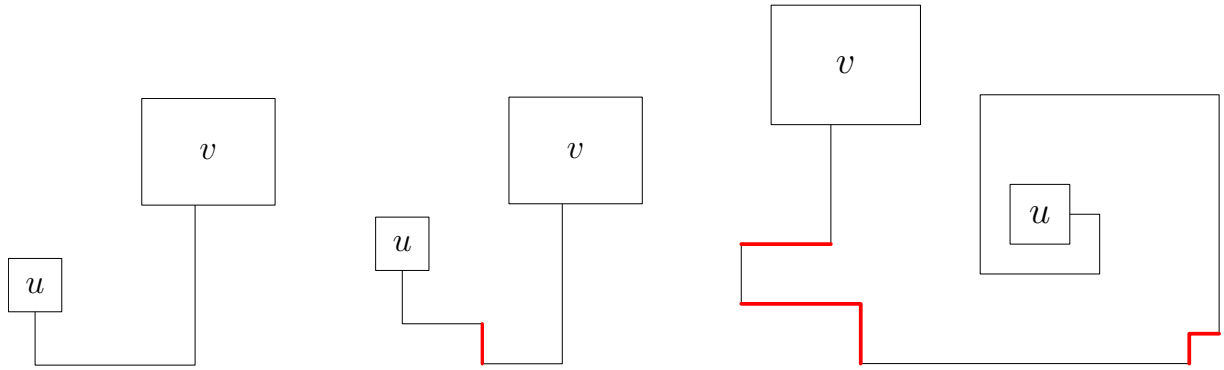
Theorem 6.1.2. *Let P be an orthogonal box drawing of an n-vertex graph G drawn on an $N \times N$ grid with $O(1)$ bends per edge. Then, there exists an orthogonal box drawing P' of the same graph G drawn on an $O(n) \times O(n)$ grid which is parallel to P. Furthermore, a planarity-preserving linear morph sequence from P to P' of length 2 exists, such that the unique explicit intermediate drawing is also parallel to P and drawn on an $(N + O(n)) \times (N + O(n))$ grid. Moreover, P' and the linear morph sequence can be found in $O(\min\{N + n, n \log n\})$ time.*

Proof. Apply Lemma 6.1.1 first to the x-coordinates, then apply a modified symmetric version of the lemma to the y-coordinates via the symmetry of x and y coordinates. □

An example of the algorithm for Theorem 6.1.2 can be found in Figure 6.1.

6.2 Zig-Zag Elimination and Trapezoidal Maps

The previous section discussed compression, a method for limiting the number of coordinates used by a drawing. This section discusses zig-zag elimination, which is a method for limiting the number of bends used by a drawing. Recall from Section 5.1 that an orthogonal box drawing is zig-zag-free if every edge consists of only left or only right turns when traversed.



(a) An example of an edge with no zig-zags.

(b) An example of an edge with one (vertical) zig-zag.

(c) An example of an edge with five zig-zags, three horizontal and two vertical.

Figure 6.2: Examples of edges with and without zig-zags. The zig-zags are highlighted as bolder and differently coloured.

Previous works have devised methods for using linear morphs to obtain zig-zag-free drawings in orthogonal point drawings [7, 47], and these methods could also be applied to orthogonal box drawings. However, they do not have sufficient simultaneous guarantees of final grid size, algorithmic complexity, and linear morph sequence length. We will discuss their guarantees in more detail, and a strengthened version of their results for our purposes will be obtained.

Recall from Section 5.1 that a zig-zag is an adjacent pair of left and right turns along the traversal of an edge in a drawing. We expand this definition to include the orientation.

Definition 6.2.1. *Let P be an orthogonal box drawing of an n-vertex graph G. Let $e = uv \in E(G)$ be an edge. Let b, b' be adjacent (non-coincident) bends of e in P forming a zig-zag. If the segment from b to b' is horizontal, then the zig-zag is said to be a **horizontal zig-zag**, otherwise it is a **vertical zig-zag**.*

See Figure 6.2 for examples.

Our primary purpose in discussing zig-zags is to eliminate them. That is, we wish to perform a sequences of linear morphs that will remove all of them. As a simple example, observe that the linear morph from the drawing in Figure 6.2b to the drawing in Figure 6.2a is planarity-preserving.

Biedl et al. [7] were the first to show that a (unidirectional) linear morph can be used to eliminate a zig-zag. In other words, a drawing with a zig-zag can be transformed to

one without that zig-zag using a single linear morph. This method will be very important to this chapter, as it has numerous useful properties. An example/explanation of their construction (which can be applied to orthogonal box drawings as well) can be found in Figure 6.3, though the useful properties will be discussed below.

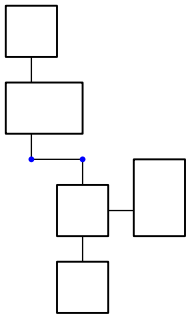
Lemma 6.2.2 (Based on the text of Section 3.2 by Biedl et al. [7])**.** *Let P be an orthogonal box drawing of a graph G. Let $e = uv \in E(G)$ be an edge. Let b, b' be adjacent bends of e in P forming a horizontal zig-zag. Then there is an orthogonal box drawing P' of G parallel to P except for the coincidence of the newly-degenerate bends b and b', with several additional properties:*

- *The locations of b and b' in P' coincide and the incident segments are vertical.*
- *No other pair of bends becomes or ceases to be coincident. That is, no pair of bends $(b_0, b'_0) \neq (b, b')$ is coincident in exactly one of P and P'.*
- *The linear morph from P to P' is a planarity-preserving horizontal linear morph.*
- *Let $x^b, x^{b'}$ be the x-coordinates of b and b' respectively in P. Assume without loss of generality that $x^{b'} > x^b$. Let x^* be the location of a port/bend/corner in P. Then its x-coordinate in P' is either x^* again or $x^* + \left(x^{b'} - x^b\right)$.*

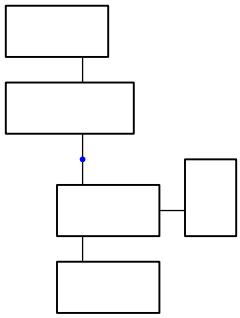
Note that the original discussion by Biedl et al. [7] of the above lemma was in the language of orthogonal point drawings. However, the result also applies to orthogonal box drawings, which is more useful to us.

Applications of the above lemma to obtain new drawings are referred to as ***zig-zag elimination***s (or in this case, ***horizontal zig-zag elimination***s), since they remove the bends of a zig-zag.

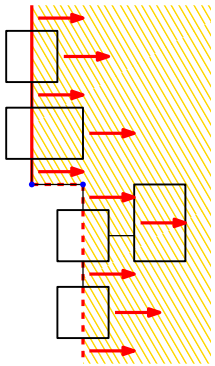
Van Goethem et al. [47] made the observation that repeat applications of horizontal zig-zag elimination can be accomplished with a single linear morph (though without any guarantees on final grid size). By repeatedly applying Lemma 6.2.2 to eliminate all horizontal zig-zags, followed by applying Lemma 6.1.1 to compress the drawing, a new drawing can be obtained that has no horizontal zig-zags and is drawn on a grid with width $O(n)$. Moreover, Van Goethem et al.'s lemma implies that one can morph directly to this drawing using only two horizontal linear morphs (one to eliminate horizontal zig-zags, and one to compress to a small grid). Unfortunately, the computation for this method can take $\Omega(n^2)$ time, which is not fast enough for our purposes. We want an algorithm that computes a drawing with the same bounds, but that runs in only $O(n)$ time. We will devise such



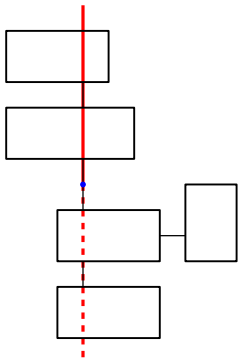
(a) An orthogonal box drawing P with a horizontal zig-zag.



(b) The drawing P' which results from applying the zig-zag elimination routine described by Biedl et al. [7] on P.



(c) Applying their method involves partitioning the features (bends/ports/corners) of the drawing into two halves, depending on where they lie in relation to the zig-zag. One of the sides is visualized here with the hatched (gold) fill and the solid (red) line exiting the top of the zig-zag. The left-side boundary is inclusive above the zig-zag (where it is further to the left), and exclusive below the zig-zag (where it is further to the right). The features on this side will be moved in the direction of the arrows.



(d) Every feature was moved a distance to the right exactly equal to the length of the deleted zig-zag.

Figure 6.3: An example of the zig-zag elimination linear morph method used by Biedl et al. [7], applied to orthogonal box drawings.

an algorithm by applying techniques of Doenhardt and Lengauer [20] for one-dimensional compaction in VLSI design (explained below). The correctness of this application will use properties of the zig-zag elimination process described above, even though the algorithm itself will not.

6.2.1 Simplified Results by Doenhardt and Lengauer for One-Dimensional Layout Compaction

In their work, Doenhardt and Lengauer [20] formalized a very general one-dimensional compaction problem for VLSI design, which they show to be NP-hard. Doenhardt and Lengauer give an efficient algorithm for a tractable case of their one-dimensional compaction problem. For clarity and simplicity, we will limit our discuss of their work to a carefully chosen further simplified case handled by their algorithm that is most relevant to our application. That case is as follows:

Definition 6.2.3 (Special case of of Doenhardt and Lengauer's [20] one-dimensional layout compaction model)**.** *Let $L = \{l_1, \ldots, l_n\}$ be a set of n vertical line segments in the plane that do not intersect, with endpoints lying on a grid. The goal of the* ***simplified one-dimensional layout compaction problem*** *is to find new x-coordinates $x(l_i)$ for each line segment $l_i \in L$ so that:*

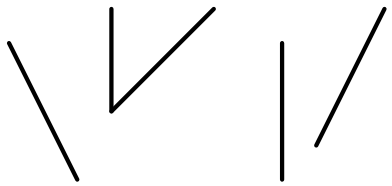
- *For each pair $l_i, l_j \in L$ such that l_i lies strictly to the right of l_j along some horizontal line intersecting both, it is required that $x(l_i) - x(l_j) \geq 1$. These constraints are denoted $C_{\leq}$.*
- *The* ***width****, defined as $\max_i x(l_i) - \min_i x(l_i)$, is minimized.*

The output of an algorithm that solves this problem is a set L' of line segments which are constructed by shifting each line segment $l_i \in L$ horizontally to its assigned x-coordinate $x(l_i)$. It can be additionally required that $\min_i x(l_i) = 0$ without changing the minimum width of an instance.

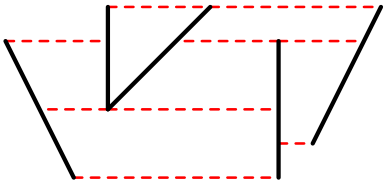
See Figure 6.5a and Figure 6.5f for examples of the input and output of Doenhardt and Lengauer's problem (respectively).

Doenhardt and Lengauer refer to $C_{\leq}$ as *II*.

Doenhardt and Lengauer give an efficient algorithm that we will refer to as "longest path compaction" which computes a solution to the simplified one-dimensional layout



(a) The set of line segments L.



(b) The trapezoidal map of L is drawn as dotted (red) lines (rays omitted).

Figure 6.4: An example of the trapezoidal map of a set of line segments L that includes non-vertical line segments.

compaction problem in $O(n \log n)$ time. Their solution is also guaranteed to use integer coordinates. Their algorithm does not compute $C_{\leq}$, since the size of $C_{\leq}$ could be $\Theta(n^2)$ for some instances. Doenhardt and Lengauer instead bypass this issue by using two phases in their algorithm: The first phase computes a carefully chosen subset of $C_{\leq}$ with size $O(n)$ in $O(n \log n)$ time. The second phase uses a topological sort to compute the final set of line segments in $O(n)$ time.

Doenhardt and Lengauer first observe that the constraints in $C_{\leq}$ correspond to a directed acyclic graph whose vertex set is L. Each constraint of the form $x(l_i) - x(l_j) \geq 1$ corresponds to an edge from l_j to l_i in this graph. They call this graph $G_0(L)$. The output of the first phase in the algorithm by Doenhardt and Lengauer is another directed acyclic graph, with $O(n)$ edges. They call this graph $G_0^r(L)$, but we will call this the "trapezoidal graph" of L, and denote it $G_T(L)$. Importantly, Doenhardt and Lengauer show that the transitive closure of $G_T(L)$ is the same as the transitive closure of $G_0(L)$.

Towards the definition of $G_T(L)$, consider a set of (potentially not just vertical) non-horizontal line segments L that are disjoint except for (possibly) shared endpoints. For each endpoint x of a line segment in L, extend horizontal line segments to the left and right of x until they each hit a line segment in L not incident to x. Note that some of these horizontal line segments are extended infinitely, and become rays. We call this set of horizontal line segments (and rays) the ***trapezoidal map*** [13] of L, and we will denote it H. See Figure 6.4 for an example. Although the above definition allows sets of line segments that may share endpoints, we will mostly deal with sets of disjoint line segments. Under this more restricted class, we can define the ***trapezoidal graph*** $\boldsymbol{G_T(L)}$ of a set of disjoint non-horizontal line segments L to be the (simple) directed graph whose vertex-set is L and whose edges correspond to line segments (but not rays) in H (always directed from left to right). In particular, each edge in $G_T(L)$ can be said to be ***generated*** by some endpoint of some line segment $l_i \in L$ (specifically, the endpoint of the line segment l_i

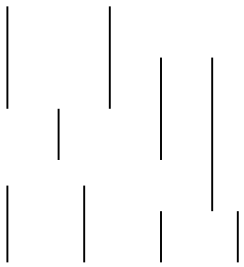
with which the horizontal line segment in the trapezoidal map shared an endpoint). Some edges may be generated multiple times. We omit the additional edges for simplicity, but record all line segment endpoints that generate each edge as additional data alongside the trapezoidal graph itself. In this subsection and the following two subsections, we will focus on trapezoidal maps and graphs of sets of vertical line segments. However, in Section 6.2.5, we will use trapezoidal maps and graphs for sets of disjoint non-horizontal line segments.

We take a brief tangent to discuss some well-known results on computing trapezoidal maps that will be useful. For an arbitrary set of disjoint non-horizontal line segments (not just vertical line segments), it is known that its trapezoidal map can be computed in $O(n \log n)$ time by standard methods [13]. In some cases, this time complexity can be improved. Chazelle [12] describes a method which can be used to compute a trapezoidal map in linear time for the non-horizontal segments of any simple polygon (when the simple polygon, including horizontal segments, is given as input). We will also be able to make use of this faster time complexity in our use-case, by using Chazelle's algorithm as a subroutine, as we will discuss in the next subsection.

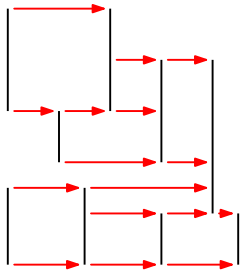
We now return to the discussion of the algorithm by Doenhardt and Lengauer. For the set of disjoint vertical line segments L, recall that Doenhardt and Lengauer compute the trapezoidal graph $G_T(L)$ by computing the trapezoidal map of L. Specifically, they create a directed edge corresponding to each (non-ray) edge of the trapezoidal map, directed from left to right (this is a slight simplification of Doenhardt and Lengauer's algorithm, but it is equivalent for our purposes).

The second phase of the algorithm by Doenhardt and Lengauer takes as input any directed acyclic graph $G(L)$ whose transitive closure is the same as that of $G_0(L)$ (in particular, they show that $G_T(L)$ suffices). It takes time $O(|V(G(L))| + |E(G(L))|)$ to compute the coordinates $x(l_i)$ satisfying every constraint in $C_{\leq}$ with minimum width. In particular, the second phase of their algorithm does not require the x-coordinates of each line segment in L as an input, nor does it require the constraints $C_{\leq}$. The resulting x-coordinates are also guaranteed to have a width no larger than the number of edges in $G_T(L)$, which we will call the ***compactness property***. This is a consequence of a fact given for using the "critical-path method" (typically for scheduling) in the second phase of Doenhardt and Lengauer's algorithm [20]. We will omit discussion of the details of their second phase (and the critical-path method), since we use it as a black-box, but essentially they use a topological sort similar to the one we use in the proof of Theorem 3.2.4.

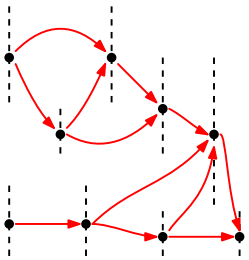
See Figure 6.5 for an example of all the different components of their algorithm which have been discussed.



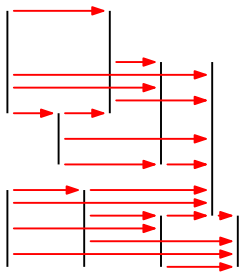
(a) A set of vertical line segments L.



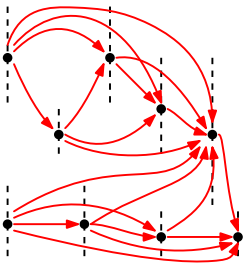
(b) The trapezoidal map of L (rays omitted).



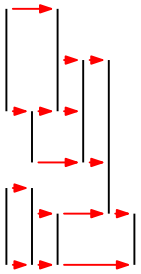
(c) The graph $G_T(L)$ computed from the trapezoidal map of L.



(d) The pairs of line segments corresponding to $C_{\leq}$.



(e) The graph $G_0(L)$, with the same transitive closure as $G_T(L)$.



(f) The compacted set of line segments, along with their trapezoidal map. Note that this is (combinatorially) the same trapezoidal map as the one of L.

Figure 6.5: Examples of the constructions for the algorithm used by Doenhardt and Lengauer.

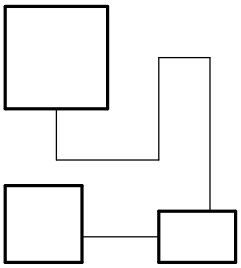
6.2.2 Applying Doenhardt and Lengauer's Results for Compressions

For an orthogonal box drawing P, Doenhardt and Lengauer's algorithm can be used to find a compression of P, which could be used as an alternative proof of the compression lemma (Lemma 6.1.1). This alternative method will be an important warm-up for the next subsection. More importantly, in the process we will make use of some important concepts for applying Doenhardt and Lenguaer's algorithm to zig-zag elimination.

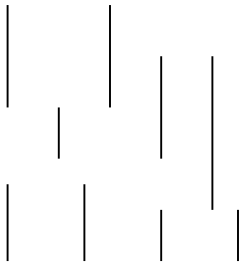
We will assume that P does not contain any successive horizontal edge segments in the order along any edge, or zero length edge segments connecting successive horizontal edge segments (note that this is a weaker property than P having no degenerate bends). The ***vertical line segments of P***, denoted $V(P)$, are the vertical edge segments along with the set of minimal vertical line segments joining two ports/corners along the left and right sides of each vertex box. Note that this is a set of line segments that are disjoint except for (possibly) shared endpoints. We define the ***set of maximal vertical line segments covering P***, which we will denote as $L(P)$, to be the set of maximal disjoint vertical line segments following along P. Equivalently, to define $L(P)$, start with the vertical line segments $V(P)$ of P, and exhaustively merge all vertical segments sharing endpoints until the result is the set of disjoint vertical line segments. Note that these segments may cover more than one element of the drawing P (see Figure 6.6d and Figure 6.6e), and that for any two equivalent orthogonal drawings P^0, P^1, $L(P^0) = L(P^1)$. Also note that every port and every corner of P is covered by some element of $L(P)$, and our assumption that P does not contain successive horizontal edge segments or zero length edge segments connecting successive horizontal edge segments guarantees that every bend of P is covered as well.

By processing the set $L(P)$ with the algorithm of Doenhardt and Lengauer, we obtain new x-coordinates $x(l)$ for each segment $l \in L(P)$. Hence, we can recover a new drawing: For every feature of P covered by a vertical line segment l in $L(P)$, change its x-coordinate to the new value $x(l)$, to obtain a new drawing P'. These x-coordinates respect the relative x-order of vertical line segments (and therefore also features of P and P') crossed by any horizontal line. Therefore, the linear morph from P to P' is planarity-preserving by Theorem 2.3.2 (the unidirectional morph characterization theorem). It can be assumed that all the x-coordinates of P' are non-negative and no larger than $O(n)$ by the compaction property (since there are $O(n)$ segments) and hence this gives the same guarantees on grid size as Lemma 6.1.1. See Figure 6.6 for some examples.

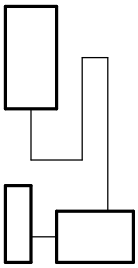
This "alternate" algorithm for performing compressions has a worse time complexity—it takes $O(n \log n)$ time to compress the drawing. This is due to the time it takes to compute the trapezoidal map of $L(P)$. If this time complexity could be improved to $O(n)$



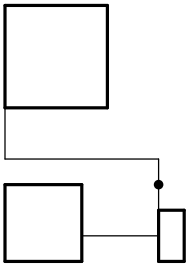
(a) An orthogonal box drawing D.



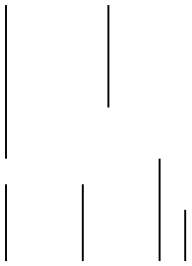
(b) The set of vertical line segments $L(D)$ corresponding to D (the same as the one used in Figure 6.5).



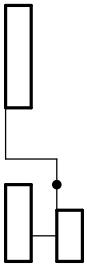
(c) A horizontally compressed orthogonal box drawing D' parallel to D.



(d) An orthogonal box drawing C. The filled circle denotes an arbitrary number of coincident bends.



(e) The set of vertical line segments $L(C)$ corresponding to C. This set is the same (always these six segments) regardless of how many coincident bends are in C.

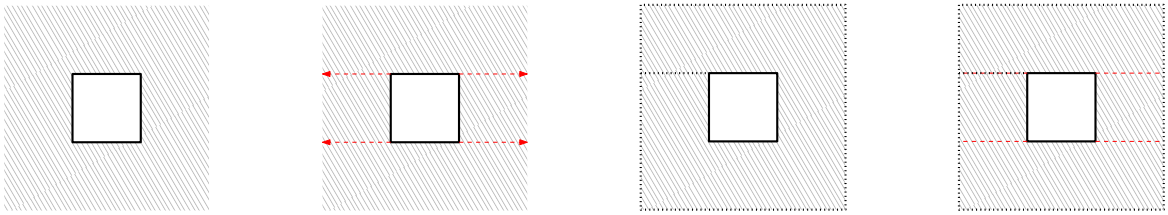


(f) A horizontally compressed orthogonal box drawing C' parallel to C.

Figure 6.6: Two examples of how Doenhardt and Lengauer's longest-path compaction algorithm can be applied to orthogonal box drawings.

time, then we would be able to perform compressions in $O(n)$ time. In fact, as we will see, this is possible by making use of Chazelle's algorithm for simple polygons. While we are already able to perform compressions in $O(n + N)$ time (on an $N \times N$ grid), and we have no need to perform them faster for large grids, this time complexity improvement for computing trapezoidal maps will also be useful for zig-zag eliminations in the next subsection.

Lemma 6.2.4. *Let D be an orthogonal box drawing of an n-vertex connected graph with $O(1)$ bends per edge, containing no degenerate bends, drawn on an $O(n) \times O(n)$ grid. Then the trapezoidal maps of $V(D)$ and $L(D)$ can be computed in $O(n)$ time.*



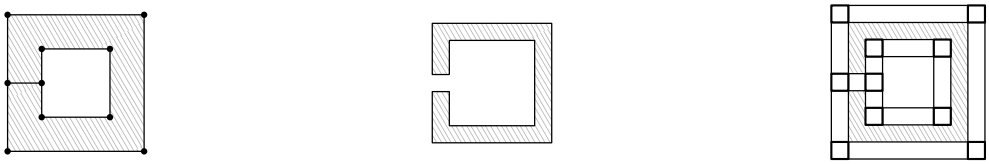
(a) The outer face of a square.

(b) The trapezoidal map of the outer face of a square consists of 4 rays.

(c) A bounding box added with a connecting edge segment.

(d) The trapezoidal map now consists of 4 line segments.

Figure 6.7: An example of how an infinite face can be converted to a finite face for the purposes of computing a trapezoidal map.



(a) A straight-line orthogonal point drawing D with a marked (hatched) face with interior R.

(b) M^* (hatched) and its simple polygonal boundary (solid).

(c) A visualization of how M may be computed.

Figure 6.8: An example of the construction used to replace weakly simple orthogonal polygons with simple orthogonal polygons, for the purpose of computing trapezoidal maps.

Proof. We first discuss computing the trapezoidal map of $V(D)$, and afterwards we use the trapezoidal map of $V(D)$ to compute the trapezoidal map of $L(D)$.

We will compute the trapezoidal map within each face of D independently. Since the underlying graph is connected, the boundary of each face is a weakly simple polygon. Moreover, since there are $O(1)$ bends per edge, the total number of line segments among all these weakly simple polygons is $O(n)$. Let P be one such weakly simple polygon, of a face f, so that P is the boundary an open region R (the interior of the set f, in the point-set topology sense). Without loss of generality, we may assume that f is not an outer face, since otherwise we may add a bounding box and a single horizontal segment to P, to P, so that all rays of the trapezoidal map that intersect the bounding box become segments hitting the bounding box. See Figure 6.7 for an example.

We wish to apply Chazelle's result for (strongly) simple polygons to P. However, our polygon P may only be weakly simple. Fortunately, since D is an orthogonal box drawing, we know that P consists of only horizontal and vertical line segments between points on an integer grid, which allows us to use the following construction:

- Let R^* denote the complement of R (a closed and connected polygonal region).
- Let M be the Minkowski sum of R^* with a (filled and closed) square of side-length $1/3$. This can be computed in linear time (in the size of R^*) [13, Theorem 13.11].
- Let M^* be the complement of M (an open region).
- Then boundary(M^*) is a (strongly) simple polygon. Compute the trapezoidal map of the vertical segments along the boundary of M^* using Chazelle's algorithm.
- This trapezoidal map can be combinatorially mapped to the trapezoidal map of the vertical segments along the boundary of R in linear time (in the size of the face). The method for doing this is fairly straightforward:
  - If p and q are connected in the trapezoidal map for M^*, then there must be some endpoint p' of a line segment in $V(D)$, and some point q' along another line segment in $V(D)$, such that p' and q' have the same y-coordinate, and furthermore that p and q are respectively within the Minkowski sums of p' and q' with a (filled and closed) square of side-length $1/3$.
  - Create a horizontal line segment between every such pair of points p', q' to obtain the trapezoidal map of $V(D)$.

See Figure 6.8 for an example.

Since we can compute the trapezoidal map within each face f in linear time (in the size of f), the whole trapezoidal map of $V(D)$ can be computed in $O(n)$ time.

Using the trapezoidal map of $V(D)$, we can obtain the trapezoidal map of $L(D)$ as follows: Let x be an (arbitrary) endpoint of a line segment in $L(D)$. It suffices to cast two horizontal rays (one in each direction) out from x until they hit something in $L(D)$ (or out to infinity). We can simply check for the same point x among the endpoints of $V(D)$, and see what it hits among $V(D)$ by using the trapezoidal map of $V(D)$. Since $L(D)$ is the result of taking the union of segments in $V(D)$, we need only find the corresponding segment in $L(D)$. We can also perform these operations for all such endpoints x in $O(n)$ total time, so we can obtain the trapezoidal map of $L(D)$ in $O(n)$ time as well. □

As an aside, the condition that all features lie on an $O(n) \times O(n)$ grid could be relaxed to requiring that they lie on some polynomial grid (i.e., sufficiently small so that linear time is the same as $O(n)$ time in the word RAM model), and the time complexity would not be affected. However, in the next subsection, we will only use this lemma in a circumstance where we have an $O(n) \times O(n)$ grid, so we have presented it this way for consistency and ease-of-understanding.

6.2.3 Specializing Doenhardt and Lengauer's Results for Simultaneous Horizontal Zig-Zag Elimination

Let P be an orthogonal box drawing, and let P^* be the result of several horizontal zig-zag eliminations on P. The drawing P^* is expensive to compute (sometimes taking $\Omega(n^2)$ time), and as such we do not wish to compute it directly. Recall that the second phase of the algorithm by Doenhardt and Lengauer does not use the x-coordinates of its input segments L, and it instead only uses the y-coordinates and the trapezoidal graph $G_T(L)$ induced by the trapezoidal map of L. In this subsection, it will be shown that the graph $G_T(L(P^*))$ (where $L(P^*)$ is the set of maximal vertical line segments covering P^*, defined in the previous subsection) can be computed without having to compute P^* itself. This will be enough to find a faster algorithm for performing simultaneous horizontal zig-zag eliminations.

Lemma 6.2.5. *Let P be an orthogonal box drawing of a graph on n vertices, with at most $O(1)$ bends per edge. Assume that P also contains no degenerate bends. Let P^* be the result of sequentially applying horizontal zig-zag elimination on every horizontal zig-zag*

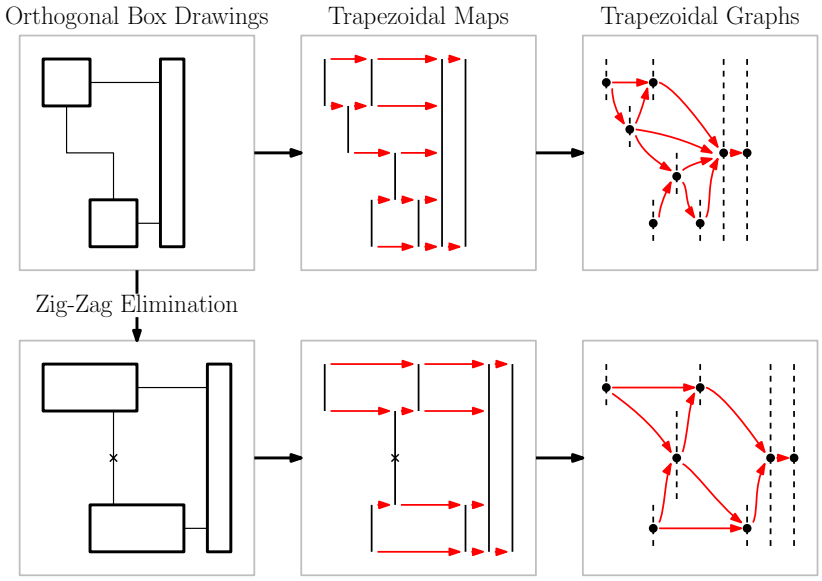


Figure 6.9: An example of how zig-zag elimination alters the formation of a trapezoidal map from a drawing. The pair of coinciding bends resulting from the zig-zag elimination is denoted with a cross.

in P, using Lemma 6.2.2. Then, given only $G_T(L(P))$ and P itself, $G_T(L(P^*))$ can be computed in $O(n)$ time.

See Figure 6.9 for a simple example of the above lemma.

Proof. Recall that we assume that the trapezoidal graph $G_T(L(P))$ tells us, for each edge, from which line segment endpoints and directions it was generated (i.e., which rays cast from which points in which directions generated the segment in the trapezoidal map that corresponds to the edge in the trapezoidal graph).

We first consider one horizontal zig-zag elimination at a time, and discuss some important structure. Let the sequence of drawings in the horizontal zig-zag elimination sequence be denoted $P = P_1, P_2, \ldots, P_k = P^*$.

For the ith horizontal zig-zag elimination, let b, b' be two adjacent (non-coincident) bends along an edge e in P_i forming a horizontal zig-zag. Let l, l' denote the (maximal) vertical segments along e incident to b and b' (respectively) in P_i. We use the same notation for the corresponding vertices in the graph $G_T(L(P_i))$. Let l'' denote the unique maximal vertical segment covering both b and b' in P_{i+1}. Once again, use the same notation for the corresponding vertex in the graph $G_T(L(P_{i+1}))$.

The horizontal zig-zag elimination lemma (Lemma 6.2.2) leaves b and b' coincident, and does not change the coincidence of any other pair of bends. Moreover, the linear morph from P_i to P_{i+1} is a planarity-preserving horizontal linear morph, and hence by Theorem 2.3.2 (which characterizes planarity-preserving unidirectional linear morphs) the order of vertical line segments intersected by any horizontal line crossing $L(P_i)$ or $L(P_{i+1})$ must be the same, with the exception that the line segments l, l' in $L(P_i)$ are replaced with one segment l'' in $L(P_{i+1})$ (and hence the unique horizontal line which crosses both l and l', in order, now only crosses l''). Therefore, all edges of $E(G_T(L(P_i)))$ that are not incident to either l or l' appear in $E(G_T(L(P_{i+1})))$. Moreover, $G_T(L(P_{i+1}))$ can be obtained from $G_T(L(P_i))$ by first deleting all (at least one, up to three) edges generated exclusively by the bends b, b', and then merging the vertices l, l' to obtain the vertex l'' (and relabelling the endpoints of all edges containing l or l').

Essentially, we have shown that we need only perform a set of vertex merge and edge deletion operations on the graph $G_T(L(P))$, where our 'vertices' correspond to vertical segments. We now use this structure to prove the result. By recording all the vertical segments that are to be merged during all operations, a lookup table can be formed that maps indices referencing vertical segments in P to indices referencing vertical segments in P^*. By recording all vertical segment endpoints in $L(P)$ that are to be deleted (that is, bends forming horizontal zig-zags), those that are not to be deleted can check their generated edges in the trapezoidal graph of $L(P)$, and find their corresponding generated edges in $L(P^*)$ by using the lookup table. By doing this, $G_T(L(P^*))$ can be computed from $G_T(L(P))$ in $O(n)$ time. □

At a high level using the established notation (P is the input drawing, P^* is the drawing after all horizontal zig-zag eliminations have been performed), we will compute $G_T(P^*)$ directly from $G_T(P)$ by using Lemma 6.2.5. We will then use Doenhardt and Lengauer's longest-path compaction algorithm to compute vertical line segments and hence a drawing. This drawing will not be P^*, but it will be parallel to P^* (and hence contain no horizontal zig-zags), which will suffice for our purposes.

Lemma 6.2.6. *Let P be an orthogonal box drawing drawn on an $O(n) \times O(n)$ grid of a connected graph G on n vertices with at most $O(1)$ bends per edge. Assume that P also contains no degenerate bends. Let P^* be the result of sequentially applying horizontal zig-zag eliminations on every horizontal zig-zag in P. Then there exists an orthogonal box drawing $\hat{P}$ with no horizontal zig-zags drawn on an $O(n) \times O(n)$ grid, such that $\hat{P}$ is parallel to P^*. Moreover, $\hat{P}$ can be computed from P alone in $O(n)$ time. Furthermore, the linear morph from P to $\hat{P}$ is a horizontal planarity-preserving linear morph.*

Proof. Since G is a connected graph, we can use Lemma 6.2.4 to compute the the trapezoidal map of $L(P)$ in $O(n)$ time. Hence, we also get the trapezoidal graph $G_T(L(P))$.

Next, by applying Lemma 6.2.5, $G_T(L(P^*))$ can be computed in $O(n)$ time using $G_T(L(P))$. Note that the y-coordinates of each line segment endpoint in $L(P^*)$ can also be trivially computed, since they are exactly the y-coordinates of the corresponding endpoints in $L(P)$.

Finally, with $G_T(L(P^*))$ the second phase of Deonhardt and Lengauer's longest-path compaction algorithm can be used to obtain shifted x-coordinates $x(l^*)$ for each line segment $l^* \in L(P^*)$, even though we haven't computed the original x-coordinates of $L(P^*)$. For each feature in P covered by a vertical line segment $l \in L(P)$ that corresponds to a vertical line segment $l^* \in L(P^*)$ (note that multiple line segments in $L(P)$ may correspond to a single line segment in $L(P^*)$, since they can be merged), the x-coordinate of that feature can be modified to be $x(l^*)$. This creates a new orthogonal box drawing which we will call $\hat{P}$. There are $O(n)$ edges in $G_T(P^*)$, so this application of their algorithm runs in $O(n)$ time. Also, $\hat{P}$ can be assumed to have non-negative x-coordinates no larger than $O(n)$ by the compaction property. Therefore, $\hat{P}$ is drawn on an $O(n) \times O(n)$ grid. By definition, $\hat{P}$ is also parallel to P^*.

Since the constraints $C_{\leq}(P^*)$ are respected by both P and $\hat{P}$ (even though they are not computed directly), and each pair of bends forming a horizontal zig-zag is coincident in $\hat{P}$, the ordering along any horizontal line is preserved in the linear morph from P to $\hat{P}$. Hence, Theorem 2.3.2 (the theorem characterizing planarity-preserving unidirectional linear morphs) guarantees that the linear morph from P to $\hat{P}$ is planarity-preserving.

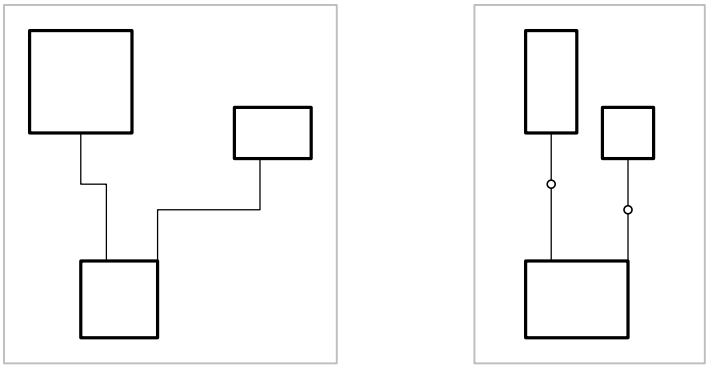
Since there are $O(1)$ bends per edge in P, there are also $O(1)$ bends per edge in $\hat{P}$, since there are an equal number of bends along each edge. □

See Figure 6.10 for an example of the above lemma.

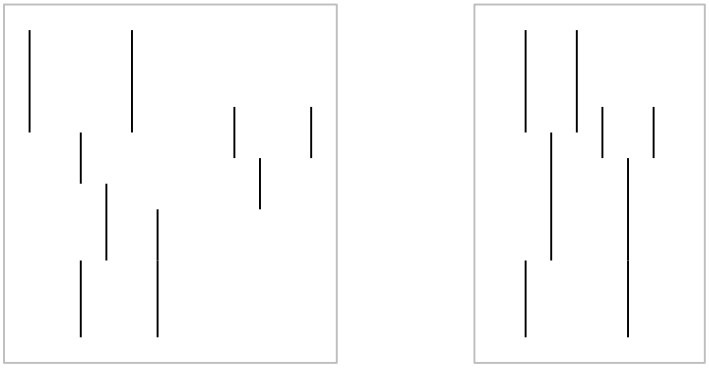
Note that the newly constructed drawing $\hat{P}$ contains many degenerate bends which come from each horizontal zig-zag (recall from Chapter 1 that degenerate bends include both coincident bends and bends that do not form turns). These can be identified and removed to find an equivalent drawing which does not contain any degenerate bends.

6.2.4 Obtaining Zig-Zag-Free Drawings using Simultaneous Horizontal/Vertical Zig-Zag Elimination

Lemma 6.2.6 gives an efficient method for removing all horizontal zig-zags from a drawing using a single linear morph. It can also be applied symmetrically to remove all vertical zig-

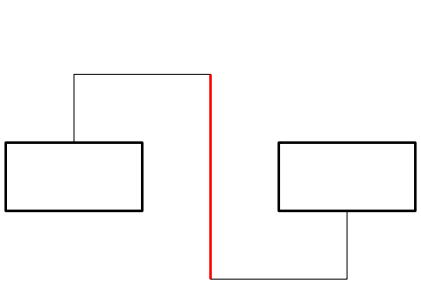


(a) P (b) $\hat{P}$

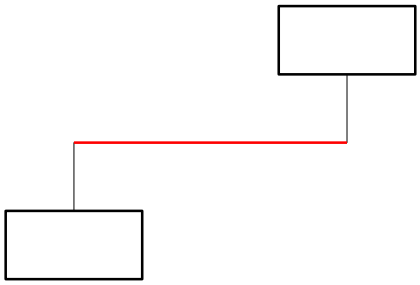


(c) $L(P)$ (d) $L(\hat{P})$

Figure 6.10: An example of simultaneous horizontal zig-zag elimination using the methods of Doenhardt and Lengauer. Coinciding bends are denoted with a hollow circle.



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



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

Figure 6.11: An example of a vertical zig-zag elimination that results a new horizontal zig-zag.

zags from a drawing. However, removing horizontal zig-zags may introduce new vertical zig-zags, and vice versa. See Figure 6.11. Fortunately, this method is still guaranteed to reduce the number of bends along each edge containing a zig-zag.

Theorem 6.2.7. *Let P be an orthogonal box drawing of a connected graph G drawn on an $N \times N$ grid with $O(1)$ bends per edge. Then there exists a compatible zig-zag-free orthogonal box drawing P' of G drawn on an $O(n) \times O(n)$ grid and a planarity-preserving linear morph sequence from P to P' of length $O(1)$, such that every explicit intermediate drawing in the sequence is drawn on an $(N + O(n)) \times (N + O(n))$ grid and has $O(1)$ bends per edge. Furthermore, P' is port aligned to P, and the maximum absolute difference in spirality between P and P' is 0. Moreover, P' and the linear morph sequence can be computed in $O(\min\{n + N, n \log n\})$ time. Additionally, if P contained no port-corner coincidences, then P' does not either.*

Proof. Apply Theorem 6.1.2 to obtain an orthogonal box drawing $\overline{P}$ drawn on an $O(n) \times O(n)$ grid, and a linear morph sequence from P to $\overline{P}$ of length $O(1)$ where every explicit intermediate drawing is drawn on an $(N + O(n)) \times (N + O(n))$ grid.

Starting with $\overline{P}$, alternately apply Lemma 6.2.6 on all vertical zig-zags, and then all horizontal zig-zags, while taking the (combinatorially smaller) equivalent drawing which eliminates degenerate bends after each application. Repeat this until the resulting drawing is zig-zag free. Since each edge with a zig-zag should have its number of bends reduced after each pair of applications, and each edge starts with at most $O(1)$ bends, this will terminate after $O(1)$ iterations. Call the final result P'. Since this process uses only $O(1)$ iterations, each explicit intermediate drawing in the linear morph sequence from $\overline{P}$ to P' (as well as P' itself) is also drawn on an $O(n) \times O(n)$ grid.

The first step takes $O(\min\{n+N, n\log n\})$ time, and the second step uses $O(1)$ iterations each taking $O(n)$ time. Hence, the full algorithm also runs in $O(\min\{n+N, n\log n\})$ time.

Finally, neither step modified the coincidence of any feature pairs besides bend-bend pairs. Therefore, if P contained no port-corner coincidences, neither does P'. □

This result is what allows us to fully solve the issues discussed at the beginning of this chapter. Essentially, Theorem 6.2.7 implies that any orthogonal box drawing D can be replaced with another orthogonal box drawing D' for which the number of bends along each edge is bounded above by the absolute value of the spirality along that edge. This will be useful for proving the Phase II theorem (Theorem 2.5.1) in Chapter 8.

6.2.5 Fast Computation of Visibility Representations

One of the minor results in Chapter 3, regarding the time complexity of Theorem 3.2.1, was left unproven, since it required some techniques that we had not yet discussed. Those techniques were the topic of this section, and so we will prove the result here. Recall the statements of Theorem 3.2.1 and Theorem 3.2.2:

Theorem 3.2.1 (Theorems 5 and 6 by Biedl [6])**.** *Let P be a planar straight-line drawing of an n-vertex graph G drawn on an $N \times N$ integer grid. Then there exists a visibility representation R of G, so that the horizontal line segments representing vertices in R have the same y-coordinates as the corresponding points representing vertices in P, and every possible horizontal line intersects the same sequence of edges and vertices in both drawings. Furthermore, R is drawn on an $O(n) \times N$ integer grid.*

Theorem 3.2.2. *Given a planar straight-line drawing P, a visibility representation R with the properties stated in Theorem 3.2.1 can be computed in $O(n\log n)$ time.*

We say that a drawing D is a ***a straight-line box drawing*** if it represents each vertex with an orthogonal rectangle (i.e., a vertex box), and each edge with a (potentially non-orthogonal) line segment. Note that this generalizes the notion of a straight-line orthogonal box drawing, which further restricts the edges to be orthogonal line segments. See Figure 6.12a for an example.

For a straight-line box drawing D with no ports along left or right sides of vertex boxes, we say that the ***non-horizontal line segments of D***, denoted $N(D)$, is the set of (disjoint) non-horizontal line segments corresponding to each non-horizontal edge,

(a) D (b) $N(D)$

Figure 6.12: An example of a straight-line box drawing D, and the non-horizontal segments $N(D)$.

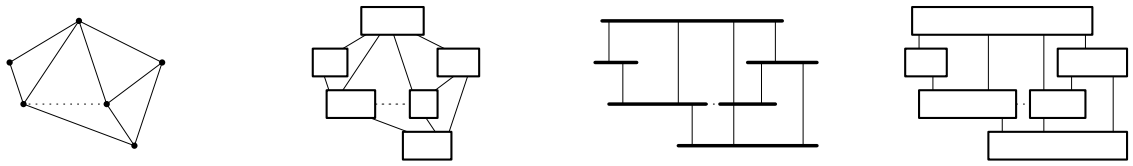
together with the vertical line segments corresponding to the left and right sides of each vertex box. See Figure 6.12b for an example.

We now prove Theorem 3.2.2 using Theorem 3.2.1.

Proof. Let R denote a visibility representation given by Theorem 3.2.1. We will *not* compute R. Instead, we will present an algorithm to compute a drawing R' with all the same properties, using only P as input. Then, similarly to the proof of Lemma 6.2.6, we will use observations about trapezoidal maps (namely, that they capture information about intersection orders of horizontal lines) to show that the algorithm is correct, and this correctness proof will make use of the existence of R.

Without loss of generality, since every horizontal line intersects the same sequence of edges and vertices in P and R, we may assume that P and R do not contain horizontal edges, since we may always add them afterwards. Also, again without loss of generality, we may assume that no edge endpoint in R coincides with the endpoint of a horizontal line segment representing a vertex in R. In some sense, our task will be to "straighten" the non-horizontal edges of P to become vertical while expanding vertices of P to horizontal segments—while ensuring that every horizontal line intersects the same sequence of edges and vertices. We will accomplish this by using some "approximations" of P and R, in the form of straight-line box drawings.

For a value $\varepsilon > 0$, let P_ε be the straight-line box drawing obtained by starting with P and expanding each point representing a vertex to a box of height ε while contracting each edge so that it attaches along the sides of its incident vertex boxes. Assume further that each box (minimally) has sufficient width so that all ports are along the top and bottom sides, but not the corners (recall that we assumed there are no horizontal edges in P, so this is possible). Similarly, let R_ε be the straight-line box drawing obtained by starting



(a) P (b) P_ε (c) R (d) R_ε

Figure 6.13: Examples of P_ε and R_ε, given P and R. The horizontal edges are dotted since they are handled via a reduction in the proof.

with R and expanding each horizontal line segment of length l representing a vertex to a box with height ε and width l, and contracting each edge accordingly. For some sufficiently small value of ε, both P_ε and R_ε are planar (though not necessarily drawn on a grid). See Figure 6.13 for an example of these drawings.

By construction, $N(P_\varepsilon)$ consists of disjoint segments. Hence, the trapezoidal graph $G_T(N(P_\varepsilon))$ is well-defined. Moreover, $G_T(N(P_\varepsilon))$ is identical for all sufficiently small ε, so we can compute it using standard symbolic perturbation techniques using a standard $O(n \log n)$ time trapezoidal map algorithm [13], without computing ε itself. We will later show that $G_T(N(P_\varepsilon)) = G_T(N(R_\varepsilon))$. For now, we will assume equality, and finish discussing the algorithm. Given $G_T(N(R_\varepsilon))$, we can compute compacted x-coordinates for a (compacted and orthogonal) straight-line box drawing R'_ε by running the algorithm of Doenhardt and Lengauer on R_ε (again, without computing ε, since it is only required for the y-coordinates). By the existence of R_ε, the maximum compacted x-coordinate of R'_ε is $O(n)$. Note that R'_ε has the y-coordinates of P_ε (computable in linear time with a symbolic representation). We finally let R' be the visibility representation sharing x-coordinates with R'_ε and y-coordinates with P (also computable in linear time), completing the algorithm.

In order to prove that this algorithm is correct, it remains only to show that $G_T(N(P_\varepsilon)) = G_T(N(R_\varepsilon))$. Recall that a trapezoidal graph $G_T(N)$ of a set of disjoint non-horizontal line segments N is induced by the trapezoidal map of N. After excluding the rays to infinity, this is itself the set of maximal non-crossing horizontal line segments that intersect elements of N at both their endpoints (and no other location), where at least one endpoint is shared with an element of N. Consider the horizontal lines intersecting at least one endpoint in N. The full intersection order (that is, the order in which every element of N is possibly intersected) of all such horizontal lines is enough to reconstruct $G_T(N)$. Denote this set of horizontal lines by $H(N)$. Specifically, if $H(N(P(_\varepsilon)))$ and $H(N(R_\varepsilon))$ have the same intersection orders in their respective drawings, then $G_T(N(P_\varepsilon)) = G_T(N(R_\varepsilon))$. Note that these are the same set, since y-coordinates are shared, so we will henceforth refer to

this set as H. Note also that we have chosen/constructed both P_ε and R_ε in a manner such that no two elements of $N(P_\varepsilon)$ or $N(R_\varepsilon)$ share endpoints, since no ports were placed along corners in either, so the intersection orders are total orders.

Now, observe that all endpoints of segments in $N(P_\varepsilon)$ and $N(R_\varepsilon)$ have y-coordinates equal to either $y(v) + \varepsilon$ or $y(v) - \varepsilon$ for some vertex v with y-coordinate $y(v)$ in P and R. Hence, any horizontal line in H defined by $y = y(v) + \varepsilon$ or $y = y(v) - \varepsilon$ for some vertex v passes through the same sequence of line segments corresponding to edges as it would in P (resp. R), but additionally passes through the line segments corresponding to the left and right sides of the vertex box for v in P_ε (resp. R_ε), and any other vertices sharing the y-coordinate $y(v)$. Therefore, since any horizontal line intersects the same sequence of edges and vertices in R and P, the horizontal lines in H must also intersect the same sequence of segments in $N(P_\varepsilon)$ and $N(R_\varepsilon)$, so $G_T(N(P_\varepsilon)) = G_T(N(R_\varepsilon))$. □

If we assumed that the graph was connected and assumed to be drawn on an $O(n)\times O(n)$ grid, then the run-time could be reduced to $O(n)$ time using Lemma 6.2.4. However, the application for this theorem in Chapter 3 is bottle-necked by a step that takes $O(n^2)$ time, and permitting a disconnected graph is somewhat helpful for discussing an extension to our main theorem in Section 8.3.

Chapter 7

Linear Morph Sequences that Perform Twists

Twists were discussed in Chapter 5 already, but an important step for using them (actually finding the sequence of linear morphs that form a twist) was postponed. This chapter fills that hole.

The exact locations and sizes of vertex boxes will play a very prominent role in this chapter. Recall the notation introduced at the beginning of Chapter 2: For an orthogonal box drawing A and a vertex v, we let $A(v)$ denote the vertex box of v in A.

In this chapter we will deal with both twist assignment functions, and times during linear morphs. We have previously denoted both of these with t. In order to differentiate, we will consistently denote the former with a bold $\mathbf{t}$, and the latter with simply t.

Recall from Section 5.2 that a simultaneous twist is a pair of (compatible) orthogonal box drawings A and B of the same graph G, together with a twist assignment function $\mathbf{t} : V(G) \to \{-1, 0, 1\}$, such that B effectively has each vertex box rotated in a direction prescribed by t (in a way that agrees with expected changes in spirality). Given an orthogonal box drawing A and a twist assignment function $\mathbf{t}$, it is important for the proof of Theorem 2.5.1 (and remains to be shown) that we can compute a corresponding orthogonal box drawing B such that A and B form a simultaneous twist given by t. Additionally, there should be a (short) sequence of linear morphs from A to B. In other words, we wish to design algorithm that takes A and t as input, and produces B.

There are two main steps for performing a twist with all the necessary constraints:

1. Morph the initial drawing A into a parallel orthogonal box drawing A^* with some additional useful properties.

2. Introduce bends to A^* carefully and explicitly define a final drawing B so that A^* can be morphed to B. Then, show that A and B (or A^* and B) form a simultaneous twist of the twist assignment $\mathbf{t}$.

The specific additional properties that will be obtained by A^* come from the following definitions:

Definition 7.0.1. *Let A^* be an orthogonal box drawing of a graph G. If all the vertex boxes of A^* are square, then A^* is said to be have **square boxes**.*

Definition 7.0.2. *For an orthogonal box drawing A^*, a vertex v, and an integer $k \geq 1$, the **k-proximal region** of v in A^* is the (two-dimensional) region of the plane with L_∞ distance at most $k \cdot d(v)$ from $A^*(v)$, where $d(v)$ denotes the degree of v. The k-proximal region of v in A^* can also be partitioned into several subregions. For any $i \in \{1, \ldots, k\}$, the **ith annulus around v** is the region of space with L_∞ distance in the half-open range $((i-1)d(v), i \cdot d(v)]$ from $A^*(v)$.*

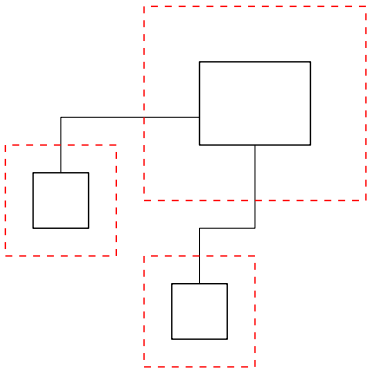
Definition 7.0.3. *Let A^* be an orthogonal box drawing of a graph G. For an integer k, suppose that for each vertex $v \in V(G)$, there are no bends, other vertex boxes, or non-incident edge segments in the k-proximal region of v in A^*, and that no two distinct k-proximal regions overlap. Then A^* is said to have **k-spaced boxes**.*

After the first step (to be handled in Section 7.1), the drawing A^* will have square and 2-spaced boxes. The second step (to be handled in Section 7.2) will use both of these properties in order to carefully choose a drawing B so that the (non-unidirectional) linear morph from A^* to B is planarity-preserving.

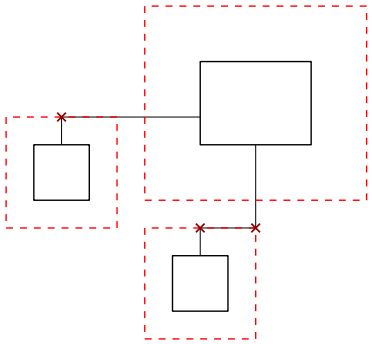
7.1 Obtaining Square and 2-Spaced Boxes

In order to obtain square and 2-spaced boxes, an additional intermediate property will be used. For an orthogonal box drawing A and a vertex v, $A(v)$ is said to be a ***thin box*** if one of its side lengths is less than $d(v) + 2$.

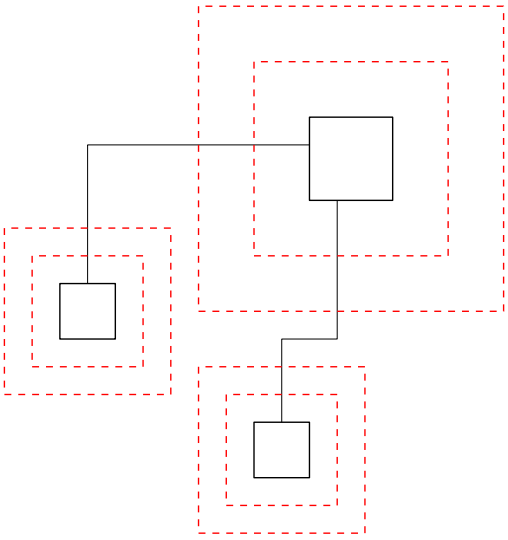
Step 1 (obtaining square and 2-spaced boxes) will be accomplished in two substeps 1a and 1b. In Step 1a, a drawing with 3-spaced boxes and no thin boxes will be obtained using



(a) An orthogonal box drawing with 1-spaced boxes, with its 1-proximal regions drawn.



(b) An orthogonal box drawing without 1-spaced boxes, with its 1-proximal regions drawn, and intersections violating the 1-spaced boxes property denoted with crosses.



(c) An orthogonal box drawing with 2-spaced square boxes, with its 1-proximal and 2-proximal regions drawn.

Figure 7.1: Examples of k-proximal regions and k-spaced boxes.

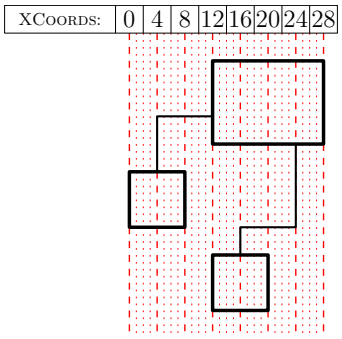
a simple coordinate modification technique, similar to the one used to prove Lemma 4.2.2. In Step 1b, additional bends will be added in the third annulus in order to obtain 2-spaced and square boxes simultaneously.

Step 1a can be accomplished with the following lemma:

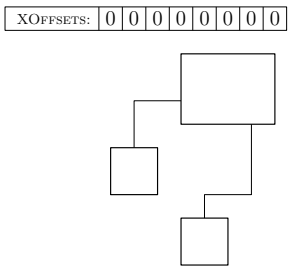
Lemma 7.1.1. *Let A be an orthogonal box drawing of a graph G drawn on an $N \times N$ grid with $O(1)$ bends per edge, and no port-corner coincidences or degenerate bends. Then there exists an orthogonal box drawing A' with no port-corner coincidences, 3-spaced boxes and no thin boxes drawn on an $(N + O(n)) \times (N + O(n))$ grid that is parallel to A, and a planarity-preserving linear morph sequence of length 2 from A to A' whose explicit intermediate drawing is also parallel to A and drawn on an $(N + O(n)) \times (N + O(n))$ grid. Moreover, A' and this linear morph sequence can be found in $O(\min\{n \log n, n + N\})$ time.*

Proof. See Figure 7.2 for an example of the construction. The x and y coordinates will be modified separately. This construction will use similar components to the proof of Lemma 6.1.1. Store all the values of the x-coordinates used by features (bends, ports, and vertex box corners) of A in an array called XCoordsWithDupes (along with a reference to the corresponding feature). There are $O(n)$ features and hence $\min\{O(n), N\}$ unique x-coordinates in total. Next, sort XCoordsWithDupes. This takes $O(\min\{n \log n, n+N\})$ time, which is the minimum time to perform either a comparison sort or a bucket sort. Iterate through XCoordsWithDupes and mark groups with identical x-coordinates. Create another array XCoords whose entries correspond to the groups of XCoordsWithDupes (i.e, an array of the unique x-coordinates), in the same relative order. This data structure can now be used to modify all x-coordinates in a uniform manner.

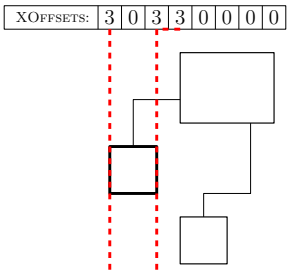
The computation of the modified values for XCoords can be accomplished using a prefix sum, as follows. Create an additional array called XOffsets with the same length as XCoords, initialized to all zeroes. For a vertex v, add $3d(v)$ to the entry in XOffsets corresponding to its left side's x-coordinate. Add $d(v) + 2$ to the entry in XOffsets corresponding to its right side's x-coordinate. Add $3d(v)$ to the entry in XOffsets corresponding to the next entry after its right side's x-coordinate (if one exists). See Figures 7.2b, 7.2c, 7.2d and 7.2e. Compute the prefix sum PSumXOffsets of XOffsets. That is, PSumXOffsets$[0]$ = XOffsets$[0]$ and PSumXOffsets$[i]$ = XOffsets$[i]$ + PSumXOffsets$[i-1]$ for all $i > 0$. Finally, add PSumXOffsets to XCoords to obtain XCoordsNew. This procedure runs in $O(n)$ time. Since all values of XOffsets were non-negative, the entries of XCoordsNew are in strictly increasing order. Additionally, for any index $i > 0$, XCoordsNew$[i]$ − XCoordsNew$[i - 1] \geq$ XOffsets$[i]$.



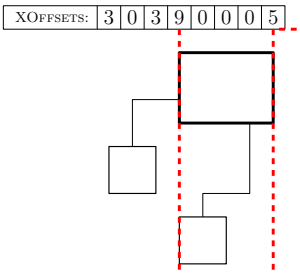
(a) Base drawing with XCoords and x-coordinate grid lines.



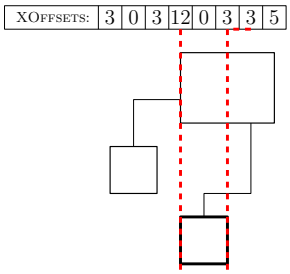
(b) Base drawing with XOffsets initialized.



(c) Base drawing with XOffsets computation done for one vertex.



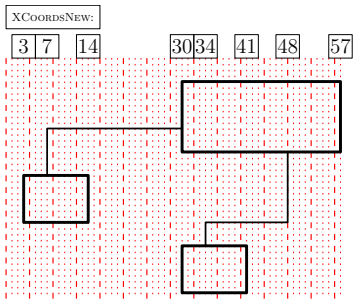
(d) Base drawing with XOffsets computation done for two vertices.



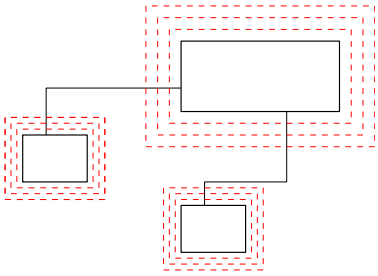
(e) Base drawing with XOffsets computation done for all three vertices.

XOffsets:	3	0	3	12	0	3	3	5
PSumXOffsets:	3	3	6	18	18	21	24	29
XCoords:	0	4	8	12	16	20	24	28
XCoordsNew:	3	7	14	30	34	41	48	57

(f) The computation of PSumOffsets and XCoordsNew given XCoords and XOffsets.



(g) The drawing with its final x-coordinates, drawn with x-coordinate grid lines.



(h) The final drawing with its 3-proximal regions drawn.

Figure 7.2: An example of how to add padding around each vertex-box (Step 1a).

Let A'_x be the drawing obtained by using these new x-coordinates and the same y-coordinates. The order of these new x-coordinates is the same, so by Theorem 2.3.2, the horizontal linear morph from A to A'_x is planarity-preserving. Note that the largest x-coordinate will be increased by an amount proportional to the sum of the degrees, which is $O(n)$. Apply the same technique again on y-coordinates to obtain a final drawing A'. The final linear morph sequence is A, A'_x, A'.

It remains only to show that A' has 3-spaced boxes that are not thin. The latter property is easy to show. Let v be a vertex in $V(G)$. Let $x_{\min}(v), x_{\max}(v)$ denote the indices of the left and right coordinates (respectively) of $A(v)$ in XCoords. Then $\text{XOffsets}[x_{\max}(v)] \geq d(v) + 2$, so $\text{XCoordsNew}[x_{\max}(v)] - \text{XCoordsNew}[x_{\min}(v)] \geq d(v) + 2$. The same is true for the y-coordinates, so there are no thin vertex boxes in A'.

Finally, we show that A' has 3-spaced boxes. Let u, v be distinct vertices in $V(G)$. Let $x_{\min}(u)$ and $x_{\max}(u)$ again denote the indices of the left and right coordinates (respectively) of $A(u)$ in XCoords. Let $x_{\min}(v)$ and $x_{\max}(v)$ denote the same thing for $A(v)$. Let $y_{\min}(u), y_{\max}(u), y_{\min}(v), y_{\max}(v)$ denote the symmetric definitions corresponding to the top and bottom coordinates of $A(u)$ and $A(v)$ (note that A and A'_x share the same y-coordinates). Since the vertex boxes $A(u)$ and $A(v)$ do not overlap (nor do the corresponding pairs of vertex boxes in A'_x and A'), either $[x_{\min}(u), x_{\max}(u)] \cap [x_{\min}(v), x_{\max}(v)] = \emptyset$ or $[y_{\min}(u), y_{\max}(u)] \cap [y_{\min}(v), y_{\max}(v)] = \emptyset$ (or both). The arguments in both cases will be symmetric, so assume that the former holds, and furthermore that $x_{\max}(u) < x_{\min}(v)$ (by choice of ordering on u and v). Vertex v added $3d(v)$ to $\text{XOffsets}[x_{\min}(v)]$, and vertex u added $3d(u)$ to $\text{XOffsets}[x_{\max}(u) + 1]$, so, since $x_{\min}(v) \geq x_{\max}(u) + 1$, we have $\text{PSumXOffsets}[x_{\min}(v)] \geq \text{PSumXOffsets}[x_{\max}(u)] + 3d(u) + 3d(v)$. Hence, since XCoords is strictly increasing, and PSumXOffsets is non-decreasing, it is also true that $\text{XCoordsNew}[x_{\min}(v)] \geq \text{XCoordsNew}[x_{\max}(u)] + 3d(u) + 3d(v)$. Since this is true for at least one of the x-coordinates or y-coordinates for every pair u, v, the 3-proximal regions do not overlap. By a similar argument, since there are no degenerate bends (and hence no bends coinciding with ports), no bend/vertex box/non-incident edge segment intersects any 3-proximal region, and hence A' has 3-spaced boxes.

Finally, A contained no port-corner coincidences, so A' also does not contain any, since no two features that had different x-coordinates or different y-coordinates in A will have the same location in A'. □

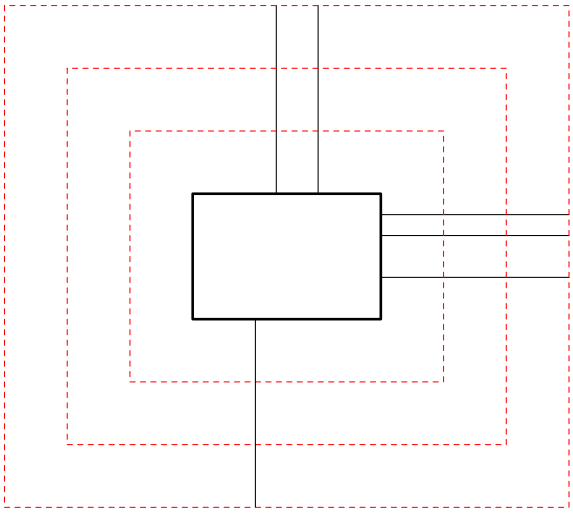
The coordinate scaling method used in Lemma 7.1.1 can be used to add spacing and remove all thin boxes, but it's not obvious how it can be used to make square boxes, which are required for Step 1b (and Step 1 as a whole). Instead, we complete Step 1b by adding a small number of bends to each edge, with the following lemma:

Lemma 7.1.2. *Let A' be an orthogonal box drawing with no port-corner coincidences, 3-spaced boxes and no thin boxes of a graph G drawn on an $N \times N$ grid, with at most $O(1)$ bends per edge. Then there exists an orthogonal box drawing A^* with 2-spaced square boxes that is drawn on an $N \times N$ grid and has at most $O(1)$ bends per edge, and a planarity-preserving linear morph sequence of length $O(1)$ from A' to A^*, such that every explicit intermediate drawing is also drawn on an $N \times N$ grid and also has at most $O(1)$ bends per edge. Furthermore, A^* is port aligned with A' and the maximum absolute difference in spirality between A' and A^* is 0. Moreover, A^* and the linear morph sequence can be computed in $O(n)$ time.*

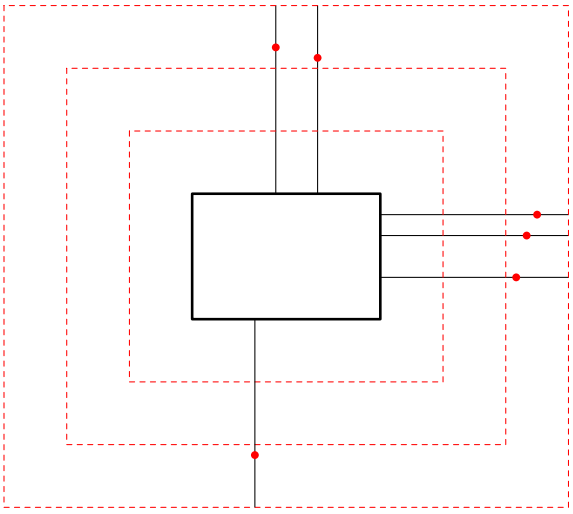
Proof. The orthogonal box drawing A^* will be computed by making changes to the locations of ports and corners, while adding bends within the 3rd annulus around each vertex box, so the 2-proximal regions will remain free of bends. This will only involve changes within the 3-proximal region around each vertex box, and reducing the vertex box to a subset of itself, so the changes for each vertex can be performed simultaneously. In more detail, consider some specific vertex v. We will shrink the width of the vertex box $A'(v)$ to $d(v) + 2$ while moving all of the ports along the top and bottom of $A'(v)$ by carefully introducing bends in the 3rd annulus that allow us to move the ports. Then we will similarly shrink the height of the resulting vertex box to $d(v) + 2$ while moving all of the ports along its left and right sides by introducing more bends in the 3rd annulus. These two high level steps will be symmetric, and each will be performed using a linear morph sequence of length 2. We proceed by describing the former high-level step.

First, a new drawing $\hat{A}^*_x$ equivalent to A' will be formed by introducing bends along the edges connected to the top and bottom of $A'(v)$ (and all other vertices simultaneously). Let the edges along the top of $A'(v)$ be denoted $e_1, e_2, \ldots, e_k$, in left-to-right order. Edge e_i will have two coinciding bends added at a distance $2d(v) + (k - i + 1)$ above its port. Since $0 < k - i + 1 \leq k \leq d(v)$, these bends lie in the 3rd annulus. The edges along the bottom will have bends added in a reflectively symmetric manner, so that the bends in edges further to the right are further from the port. This only adds 4 bends per edge globally, and does not change the grid size. See Figure 7.3b for an example.

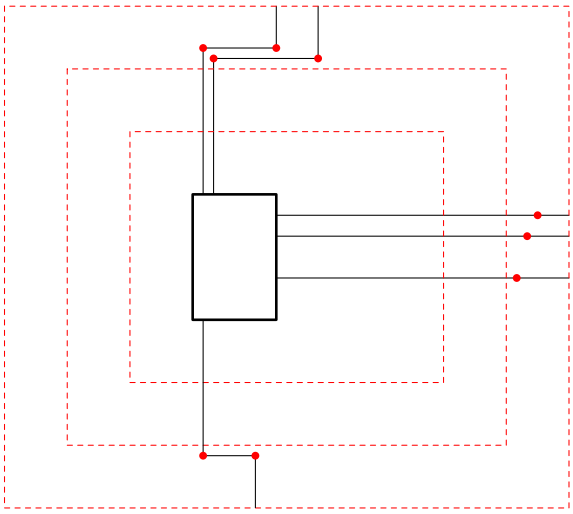
Next, we construct a new drawing A^*_x which moves these bends and the corners of $\hat{A}^*_x$. The top-left corner of the vertex box $A^*_x(v)$ should be given the same location p as it has in $\hat{A}^*_x$. The port of e_i should then be placed at the location $p + (i, 0)$. Note that this is (non-strictly) to the left of its location in $\hat{A}^*_x$, since there were no port-corner coincidences in A. The first bend along e should be placed at the same x-coordinate as its port with its y-coordinate unchanged from $\hat{A}^*_x$, while the second bend should remain at the exact



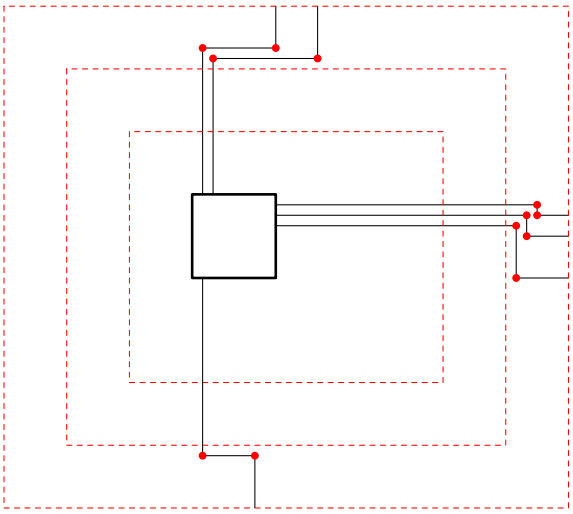
(a) The drawing A' with its annuli.



(b) The drawing $\hat{A}_x^*$ with its annuli and bends.



(c) The drawing A_x^* with its bends (and the annuli of A').



(d) The drawing A^* with its bends (and the annuli of A').

Figure 7.3: An example of how to make a vertex box square (step 1b).

same location it had in $\hat{A}^*_x$. The top-right corner of $A^*_x(v)$ should be given the location $p + (d(v) + 2, 0)$. Corners and points/bends for edges along the bottom of $\hat{A}^*_x(v)$ should be defined in a reflectively symmetric manner. All the ports along the right side of $\hat{A}^*_x(v)$ should have their x-coordinate changed to match the right side of $A^*_x(v)$. See Figure 7.3c for an example.

For each edge e_i, the port/first bend of e_i lies to the left of (or is possibly coincident with) its second bend. Moreover, the y-coordinate of the first bend along e_i is higher than the y-coordinate of any bend e_j with $j > i$, so the two edges do not intersect. By a reflectively symmetric argument, the same holds for the edges along the bottom side. Hence, the drawing is planar. Moreover, the resulting linear morph is a horizontal linear morph, since y-coordinates are preserved. For each pair of edges e both connected to the top (or bottom) of $A^*_x(v)$ and $\hat{A}^*_x(v)$, the relative horizontal order is the same in both drawings at all horizontal lines that intersect both. Hence, by Theorem 2.3.2, the horizontal linear morph is planarity-preserving within the 3-proximal region of each vertex box $A'(v)$. Each of these morph constructions occur simultaneously, and none of them move anything outside their respective 3-proximal regions, so the full simultaneous morph is also planarity-preserving.

Finally, repeat this same sequence of linear morphs on the left and right sides of the vertex box $A^*_x(v)$ to obtain an orthogonal box drawing A^* with square boxes that is port-aligned with A'. See Figure 7.3d for an example.

The 2-proximal region of v in A^* is a subset of the 2-proximal region of v in A', since $A^*(v)$ itself is a subset of $A'(v)$ (which was not a thin box, and hence had side length at least $d(v) + 2$). Since all the new bends were inserted in the 3-annulus, the final drawing A^* must have 2-spaced boxes.

Each edge e has 4 bends added to it in A^* relative to A' (possibly fewer if degenerate bends are removed). Therefore there are $O(1)$ bends per edge in A^*. In a traversal of e, exactly half of these bends form left turns and the other half of them form right turns. Therefore, the maximum absolute difference in spirality between A^* and A' is 0. □

7.2 Performing Twists using Linear Morphs

We have shown in the previous section how to achieve an orthogonal box drawing with 2-spaced square boxes. In this section, we will show that these special boxes can be used to perform a twist.

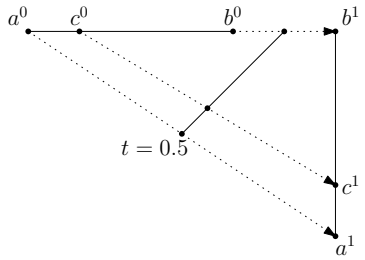


Figure 7.4: An example of Lemma 7.2.1 with $\alpha = 0.25$ and $t = 0.5$.

This section will involve a non-unidirectional linear morph, which is not fundamentally necessary in any other section of this thesis (we will briefly summarize this in Chapter 9). Hence, it will be useful to have some awareness of the structure of more general linear morphs. In particular, the following lemma will be helpful:

Lemma 7.2.1. *Let $a^0, a^1, b^0, b^1, c^0, c^1$ be points in $\mathbb{R}^1$. Assume that there is some parameter α such that $c^0 = \alpha a^0 + (1-\alpha)b^0$, and $c^1 = \alpha a^1 + (1-\alpha)b^1$. Then $(1-t)c^0 + tc^1 = \alpha\left((1-t)a^0 + ta^1\right) + (1-\alpha)\left((1-t)b^0 + tb^1\right)$.*

Proof. By expanding the definitions,

$$\begin{aligned}(1-t)c^0 + tc^1 &= (1-t)\left(\alpha a^0 + (1-\alpha)b^0\right) t\left(\alpha a^1 + (1-\alpha)b^1\right) \\ &= (1-t)\alpha a^0 + (1-t)(1-\alpha)b^0 + t\alpha a^1 + t(1-\alpha)b^1 \\ &= \alpha\left((1-t)a^0 + ta^1\right) + (1-\alpha)\left((1-t)b^0 + tb^1\right),\end{aligned}$$

which is our goal. □

This is relevant to non-unidirectional linear morphs: The time-parametrized points a, b, c can be used to represent features of the drawing, where (for example) $a^t = (1-t)a^0 + ta^1$ for a time t. In particular, if the point c can be formed by a specific convex combination of a and b at both the beginning and end of the morph, then it can be formed by that same convex combination throughout the morph (see Figure 7.4).

Lemma 7.2.2. *Let A^* be an orthogonal box drawing of a graph G drawn on an $N \times N$ grid with $O(1)$ bends per edge, no port-corner coincidences, and 2-spaced square boxes. Let $\boldsymbol{t} : V(G) \to \{-1, 0, 1\}$ be a twist assignment function. Then there exists an orthogonal box drawing B of G, also drawn on an $N \times N$ grid and with $O(1)$ bends per edge, such that A^* and B form the simultaneous twist of the twist assignment $\boldsymbol{t}$. Furthermore, there is a planarity-preserving linear morph sequence of length 2 from A^* to B, where each*

explicit intermediate drawing is also drawn on an $N \times N$ grid and has $O(1)$ bends per edge. Moreover, B and the linear morph sequence can be computed in $O(n)$ time.

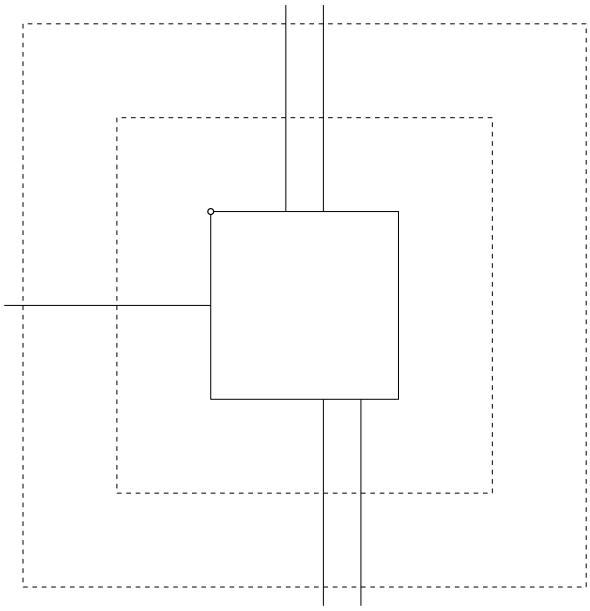
Proof. The construction has two steps. First, an orthogonal box drawing A^{**} equivalent to A^* will be constructed that adds bends in the 2-proximal region of each vertex. Second, B will be constructed by simultaneously moving some of these new bends, and the corners of each vertex box, in a way that effectively rotates each vertex box.

Let v be an arbitrary vertex in G. The drawings A^{**} and B will differ only within the 2-proximal regions of each vertex box. As a consequence, it will suffice to describe the structure of each within the 2-proximal region of the arbitrarily chosen v only, and the prescribed linear morphs can be performed for all vertices simultaneously. If $\mathbf{t}(v) = 0$, then we choose that A^{**} and B do not differ in the 2-proximal region of v, so we need only consider the case where $\mathbf{t}(v) \neq 0$. Without loss of generality, we will assume $\mathbf{t}(v) = -1$, so the twist assigned for v is clockwise.

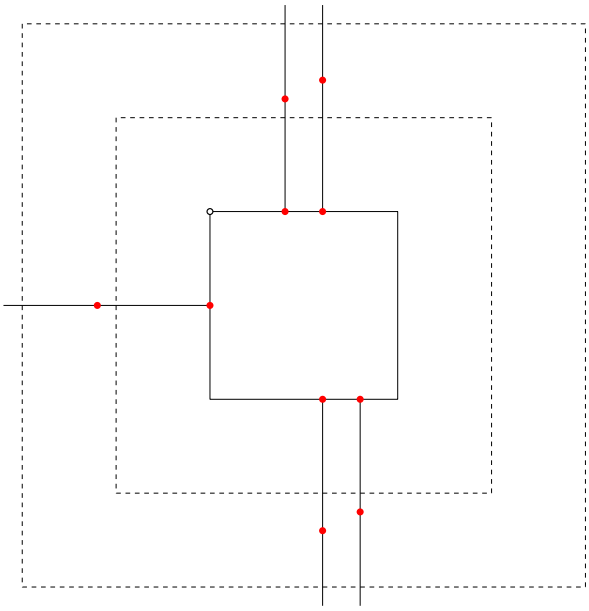
High-level overview. At a high-level, the goal of this construction will be to allow for the corners of the vertex box $A^{**}(v)$ to permute in a (clockwise) cyclic manner while maintaining planarity of the drawing (that is, each corner moves to the original location of the next one in clockwise order). Figure 7.5 gives an example of the construction. In particular, the corner denoted by a hollow circle, as well as Figure 7.6, should give the reader an understanding of the intended morph for the vertex box itself (and that it is a linear morph). Each port will also stay at the same relative location along its respective side throughout the morph, attached to the box, which we will show using Lemma 7.2.2. We will be adding bends to A^{**} so that they have carefully chosen paths during the linear morph from A^{**} to B. These bends serve to accommodate the morph of the vertex box and its ports while maintaining planarity.

Algorithmic construction. In order to construct A^{**}, six new bends will be added to each edge, with three of those in the 2-proximal region of each incident vertex. Traverse an arbitrary side of $A^*(v)$ in the clockwise direction, and let $e_1, e_2, \ldots$ denote the order of the incident edges encountered along that side. Add three new bends to e_i. The first should be coincident with the port of e_i at $A^*(v)$. The second and third should be coincident with each other at a distance of $d(v) + i$ from the port and first bend, so they lie in the second annulus. See Figure 7.5b for an example of this step.

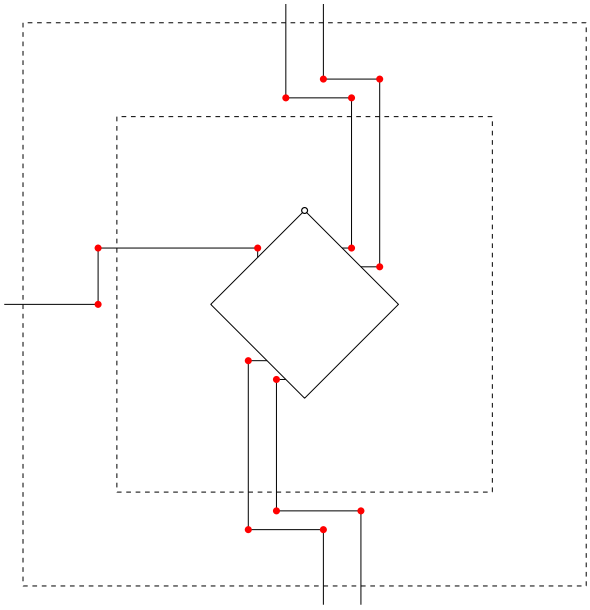
As discussed at a high-level, B is constructed by rotating all the ports and corners of $A^{**}(v)$ clockwise by 90°, so that the ports are at the same relative locations along their



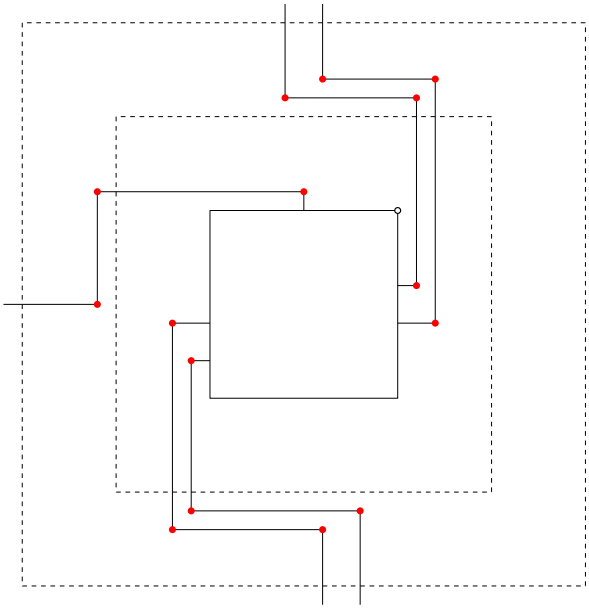
(a) A drawing A^* local to a vertex-box.



(b) The drawing A^{**} local to a vertex-box.



(c) The drawing of $M^{1/2}_{A^*,A^{**}}$ local to a vertex-box.



(d) The drawing B local to a vertex-box.

Figure 7.5: An example of how to perform a twist's (effective) rotation step. Bends are drawn as small red vertices. One corner is denoted with a hollow circle throughout so that the direction of the twist is easier to follow visually.

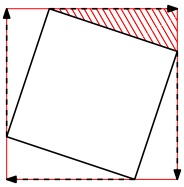


Figure 7.6: A vertex-box mid-morph, with the directions of motion drawn. The triangle in the striped area has 3 similar triangles that are rotationally symmetric around the moving box.

respective sides. The first two bends of each edge are also given new coordinates in B, while the third bend will remain at the same location in A^{**} and B. Without loss of generality, assume that the arbitrary side of the vertex box $A^{**}(v)$ is the top side, which will become the right side of the vertex box $B(v)$. Then, the first bend along e_i will be placed at a distance i to the right of the port's new location. The second bend will be placed directly above this port, using the y-coordinate of the third bend and the x-coordinate of the first bend. See Figure 7.5d for an example of this step.

Steps to prove that the morph is planarity-preserving. It remains only to show that the linear morph from A^{**} to B is planarity-preserving, which will additionally imply that B is planar. Observe first that the drawings A^{**} and B are equivalent outside the vertex boxes and 2-proximal regions. Additionally, this is also true throughout the linear morph from A^{**} to B. Hence, it suffices to prove that the linear morph is planarity-preserving within the 2-proximal region of v, since the same construction can be applied to all 2-proximal regions simultaneously. This differs from the methods we have used to prove that other linear morphs are planar throughout the thesis.

Let v again be an arbitrary vertex in G. In order to show that the linear morph from A^{**} to B is planarity-preserving in the 2-proximal region of $A^{**}(v)$ and $B(v)$, we will prove important properties about each feature of the drawing. In order, we will consider the vertex boxes, the ports, edges along the same side, and then pairs of edges along different sides. During this process, we will prove that no pair of these features intersects in a manner that causes the linear morph to violate the planarity-preserving constraint.

Vertex boxes and ports. Observe first that the vertex box remains a box throughout the morph, since similar triangles can be found between adjacent moving corners of the box, and final locations (see Figure 7.6). Additionally, the ports are each at the same relative position along their respective sides in both A^{**} and B, and hence by applying

Lemma 7.2.1 for each port p (with c as the location of p, and a and b as the corners of the vertex box side containing p), the ports remain along the vertex box $M^t_{A^{**},B}(v)$ at all times throughout the morph for $t \in [0,1]$.

Edges and the vertex box. We next show that edges along the same side of $M^t_{A^{**},B}(v)$ cannot intersect each other. Without loss of generality, we consider only the top side (that is, the top side of $A^{**}(v)$ that morphs to the right side of $B(v)$). Observe first that the portions of the edges along the top side, restricted to the second annulus of v, form a unidirectional (horizontal) linear morph (see Figure 7.7), and so do not intersect by Theorem 2.3.2.

We now characterize the structure of each edge e within the 1-proximal region of v. Note that we consider the 1-proximal region, rather than just the first annulus, since the port of e at v (and possibly the first bend along e) will go through $A^{**}(v)$ during the morph (see Figure 7.5c). Let p^t, b_1^t, b_2^t respectively denote the locations of the port, first bend, and second bend of e at time t. We will now briefly show the following claim:

Claim 7.2.2.1. *For all $t > 0$, the portion of e that is contained within the 1-proximal region of v consists of a segment leaving the port p^t heading rightwards, and a left turn at b_1^t into a segment heading up towards b_2^t.*

For the initial and final times $t \in \{0,1\}$, b_1^t has the same y-coordinate as p^t, and so they also share a y-coordinate for all intermediate times $t \in [0,1]$. Similarly, for the initial and final times $t \in \{0,1\}$, b_1^t has the same x-coordinate as b_2^t, and so they also share an x-coordinate for $t \in [0,1]$ as well. Moreover, the x-coordinate of b_1^1 is strictly greater than the x-coordinate of p, while b_1^0 is coincident with p^0. Thus, for all times $t \in (0,1]$, the x-coordinate of b_1^t is strictly greater than the x-coordinate of p^t. A similar argument can be used to show that for all times $t \in (0,1]$, the y-coordinate of b_2^t is strictly greater than the y-coordinate of b_1^t. These statements together are enough to prove Claim 7.2.2.1. Moreover, this implies the interior of edges do not intersect with the vertex box.

Edges along the same side. We can now show that no intersections occur between edges of the same side within the 1-proximal region of v. By Observation 2.2.1, it suffices to show that no bend b intersects any of the vertical or horizontal edge segments not incident to b (we need not consider intersections with the ports, since Claim 7.2.2.1 implies that no edge segment interior intersects the vertex box). In fact, by the previous paragraph, the x-order of the vertical segments is preserved by the morph, and so is the y-order of the horizontal segments. Therefore, no pair of vertical segments nor pair of horizontal

segments intersects. Since each bend is incident to one of each of these types of segments, no bend intersects any non-incident edge segments.

Edges along different sides. It remains only to show that no two edges incident to different sides of the vertex box intersect. We will show that the edges for each side lie in disjoint regions at all times throughout the morph.

For the top side u^* of the vertex box $A^{**}(v)$, let $R^2_{u^*}$ denote the subregion of the second annulus directly above the first annulus (see Figure 7.7). Let R^2_u denote the symmetric subregions for all other sides u, and note that these subregions are interior-disjoint for different sides. For each edge e whose port lies along a side u, the portion of e restricted to the second annulus is contained entirely within R^2_u in all drawings $M^t_{A^{**},B}$ (for $t \in [0,1]$). Therefore, it suffices to verify there are no intersections between edges on different sides within the 1-proximal region of v.

Let u denote a side of the vertex box $A^{**}(v)$, and let $u(t)$ (for $t > 0$) denote the corresponding side of the vertex box $M^t_{A^{**},B}(v)$. We will define time-parametrized subregions $R_u(t)$ of the 1-proximal region of v in $M^t_{A^{**},B}$, so that the set $\{R_u(t)\}_u$ partitions the 1-proximal region at all times. In particular, if u is the top side of $A^{**}(v)$, then let $R_u(t)$ be the set of all points p lying in the 1-proximal region of v that do not also lie in the interior of the vertex box $M^t_{A^{**},B}(v)$, such that there is some orthogonal path starting from p that goes down and then left, before hitting $u(t)$. Define $R_u(t)$ in a rotationally symmetric manner for all other sides u. By carefully letting each $u(t)$ include only one corner of the respective sides (whose choice will not be important for the remainder of the proof, so long as they are disjoint, since we have no port-corner coincidences), this partitions the 1-proximal region of v in $M^t_{A^{**},B}$ into 5 parts (the four regions $R_u(t)$ plus the interior of the vertex box). See Figure 7.8 for an example.

By Claim 7.2.2.1, for each edge e along side u of $A^{**}(v)$, the portion of e contained within the 1-proximal region of v in $M^t_{A^{**},B}$ is contained entirely within $R_u(t)$.

For every pair of edges whose ports are along different sides $u(t)$ and $u'(t)$ of $M^t_{A^{**},B}(v)$, the portions of the edges contained within the 1-proximal region of v are themselves (fully) contained within $R_u(t)$ and $R_{u'}(t)$. Therefore, they cannot intersect, and so no pair of edges intersects at any time t.

Since the linear morph from A^{**} to B is planar within the 2-proximal region of each vertex, it is planar globally as well. □

To wrap up, the main result of this chapter is as follows:

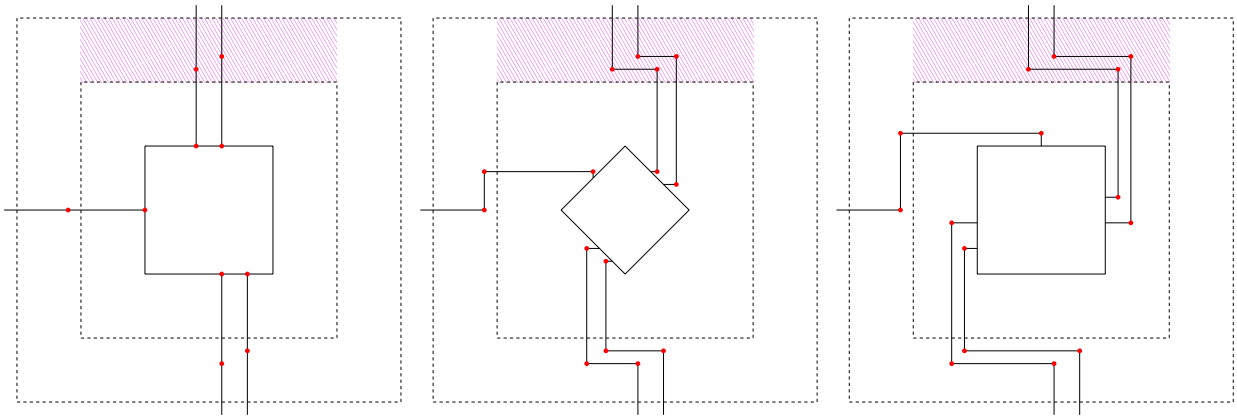


Figure 7.7: The static region R_u^2 throughout the morph, for u equal to the top side of $A^{**}(v)$.

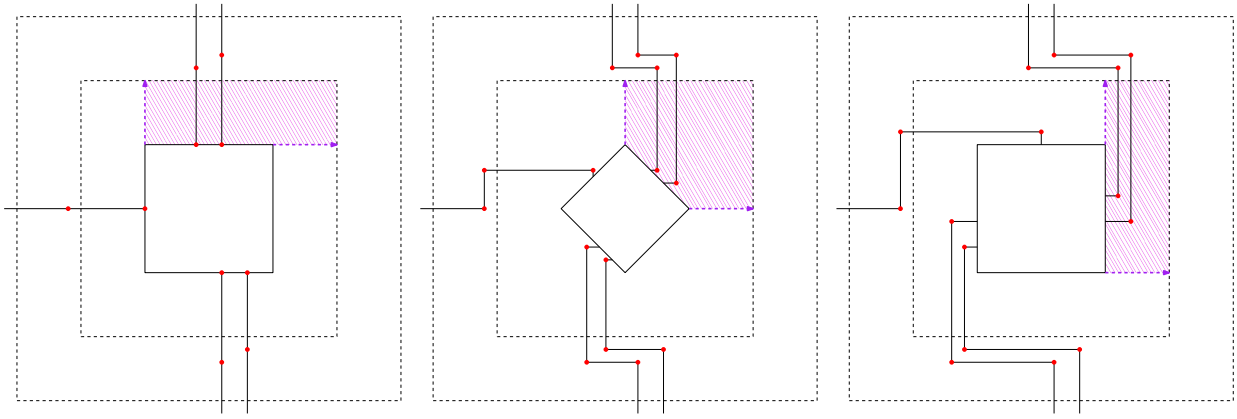


Figure 7.8: The dynamic region $R_u^1(t)$ throughout the morph, for u equal to the top side of $A^{**}(v)$.

Theorem 7.2.3. *Let A be an orthogonal box drawing of a graph G drawn on an $N \times N$ grid with $O(1)$ bends per edge and no port-corner coincidences. Let $\boldsymbol{t} : V(G) \to \{-1, 0, 1\}$ be a twist assignment function. Then there exists an orthogonal box drawing B of G drawn on an $(N + O(n)) \times (N + O(n))$ grid and with $O(1)$ bends per edge and no port-corner coincidences, such that A and B form the simultaneous twist of the twist assignment $\boldsymbol{t}$, and there is a planarity-preserving linear morph sequence of length $O(1)$ from A to B, where each explicit intermediate drawing is also drawn on an $(N + O(n)) \times (N + O(n))$ grid and has $O(1)$ bends per edge. Moreover, B and the linear morph sequence can be computed in $O(\min\{n \log n, n + N\})$ time.*

Proof. If necessary, we may replace A with a equivalent drawing that has no degenerate bends. Apply Lemma 7.1.1 to A to obtain a drawing A' with 3-spaced boxes that are not thin boxes. Apply Lemma 7.1.2 to A' to obtain a drawing A^* which has 2-spaced square boxes. Finally, apply Lemma 7.2.2 to A^* to get a drawing B for which A and B form the simultaneous twist assigned by t.

Since each of these lemmas adds $O(n)$ grid coordinates and $O(1)$ bends per edge, each explicit intermediate drawing is drawn on an $(N + O(n)) \times (N + O(n))$ grid and has $O(1)$ bends per edge in total. Moreover, the computations done for each of these lemmas take at most $O(\min\{n \log n, n + N\})$ time (the bottleneck is Lemma 7.1.1), so this is also the total time complexity of the construction. □

Chapter 8

Assembling the Phases

Each of the individual pieces necessary to complete each of the three phases (as discussed in Section 2.5) has now been completed. In this chapter, these pieces will be assembled in order to prove the main result of the thesis Theorem 1.2.1.

The main result of Phase I, Theorem 3.2.4, which reduces polyline morphing to morphing of orthogonal box drawings, was both stated and proven in Chapter 3. The main result of Phase II, Theorem 2.5.1, which morphs two orthogonal box drawings to two parallel orthogonal box drawings, will be proven in this chapter. The main result of Phase III, which morphs between two parallel orthogonal box drawings, will be stated and proven in this chapter, though it is mostly a black-box result from existing literature. Once all three phases are complete, the proof of Theorem 1.2.1 will be presented.

8.1 Assembling Phase II

Recall the statement of Theorem 2.5.1:

Theorem 2.5.1 (Obtaining Parallel Boxes)**.** *Let G be a connected planar graph on n vertices. If P and Q are a pair of compatible planar orthogonal box drawings of G, both drawn on an $O(n) \times O(n)$ grid, with $O(1)$ bends per edge, then there exists a pair of parallel planar orthogonal box drawings P' and Q' of G and a pair of planarity-preserving linear morph sequences of lengths $k_P, k_Q \in O(n)$, from P to P' and from Q to Q', respectively, such that the explicit intermediate drawings in each morph sequence are all also drawn on an $O(n) \times O(n)$ grid and have $O(1)$ bends per edge. Moreover, these sequences can be found in $O(n^2)$ time.*

The proof of Theorem 2.5.1 follows directly from the proofs of each sub-phase.

Proof of Theorem 2.5.1. Phase IIa morphs P and Q to port aligned orthogonal box drawings, and was completed in Chapter 4 by Theorem 4.2.1. With this, it can be assumed without loss of generality that the input drawings are port aligned. By Theorem 4.1.1, we may also assume that the input drawings have no ports that lie on corners.

Phase IIb computes a sequence of twist assignment functions, and was completed in Chapter 5 by Theorem 5.3.3. Let $t_1, t_2, \ldots, t_{k^*}$ be the sequence of $k^* \in O(n)$ twist assignment functions.

Phase IIc repeatedly finds morphs which perform the twists given by each twist assignment function computed in Phase IIb, and after each twist it simplifies (i.e., makes it zig-zag-free) and compresses the resulting drawing. These two routines in Phase IIc were completed in Chapter 7 and Chapter 6, by Theorem 7.2.3 and Theorem 6.2.7 respectively. This results in a planarity-preserving linear morph sequence $P = D_1, D_2, \ldots, D_{k+1} = P'$, where each drawing D_i has $O(1)$ bends per edge and is drawn on an $O(n) \times O(n)$ grid. A combination of Theorem 5.3.3 and Theorem 6.2.7 guarantees that the number of bends is always $O(1)$ in each explicit intermediate drawing in the linear morph sequence, since the maximum absolute spirality is always $O(1)$, and the number of bends along an edge is always on the order of the maximum absolute spirality. In particular, this sequence contains a subsequence $P = D_{i_1}, D_{i_2}, \ldots, D_{i_{k^*+1}} = P'$ for which each consecutive pair $D_{i_j}, D_{i_{j+1}}$ form the simultaneous twist of the twist assignment function t_j. This subsequence can be obtained by choosing each drawing which is the result of applying an iteration of Theorem 7.2.3 and Theorem 6.2.7 (i.e., choose one drawing after each twist is performed).

Theorem 6.2.7 can be applied to Q in order to find a linear morph sequence from Q to an orthogonal box drawing Q' that is zig-zag-free, port aligned with Q, and has the same spirality along each edge as Q (and additionally is drawn on an $O(n) \times O(n)$ grid and has $O(1)$ bends per edge, which also applies to every explicit intermediate drawing). Theorem 5.3.3 and Lemma 5.1.6 then guarantee that P' and Q' are parallel, completing the pair of linear morph sequences.

Each of these three sub-phases can be completed in time $O(n^2)$, and so this algorithm runs in time $O(n^2)$ in total. □

8.2 Proof of the Main Result

Phases I and II discussed in Section 2.5 have been completed, and Phase III is essentially a use of an existing result, so the main result of this thesis can now be proven. Recall the

statement of the main result, Theorem 1.2.1:

Theorem 1.2.1 (Main)**.** *Let G be a connected planar graph with n vertices. For a compatible pair of planar straight-line drawings P and Q of G, whose vertices lie on an $O(n) \times O(n)$ grid, there exists a planarity-preserving linear morph sequence from P to Q of length $O(n)$, where each explicit intermediate drawing lies on an $O(n) \times O(n)$ grid and has $O(1)$ bends per edge. Moreover, this sequence can be found in $O(n^2)$ time.*

The proof simply applies all the existing results for Phases I and II, and uses a result by Biedl et al. [7] for Phase III.

Proof of Theorem 1.2.1. For Phase I, apply Theorem 3.2.4, Lemma 3.1.3 and Observation 3.1.2 to reduce to the problem of morphing orthogonal box drawings P' and Q', neither of which has any bends at all. This phase takes $O(n^2)$ time and uses $O(n)$ planarity-preserving linear morphs, and every explicit intermediate drawing lies on an $O(n) \times O(n)$ grid since the original straight-line drawings P and Q were drawn on an $O(n) \times O(n)$ grid. For Phase II, apply Theorem 2.5.1 to morph P' and Q' into parallel drawings P'' and Q'' using $O(n)$ planarity-preserving linear morphs, while maintaining $O(1)$ bends per edge and an $O(n) \times O(n)$ grid for all explicit intermediate drawings. We may also assume that P'' and Q'' do not contain any degenerate bends. By Theorem 4.1.1, we may also assume that P'' and Q'' have no ports that lie on corners.

Finally, for Phase III, a pair of orthogonal straight-line (point) drawings $S(P'')$ and $S(Q'')$ can be introduced which are equivalent to P'' and Q'' respectively. These are constructed by creating a vertex for each feature (corner/port/bend) in the respective orthogonal box drawings, and making the horizontal and vertical segments of the orthogonal box drawings into edges (see Figure 8.1 for an example). Since P'' and Q'' are parallel and do not contain degenerate bends or ports that lie on corners (and hence contain no coinciding features at all), the resulting drawings $S(P'')$ and $S(Q'')$ are therefore both drawings of the same (labelled) graph G' as each other, and are also parallel. Note also that G' is connected since G is connected.

Any sequence of planarity-preserving linear morphs from $S(P'')$ to $S(Q'')$ that do not change the slopes of any edges (and do not introduce any bends or degeneracies) must then also induce a sequence of planarity-preserving linear morphs from P'' to Q''. Biedl et al. [7] give an algorithm that finds such a sequence of $O(n)$ linear morphs from $S(P'')$ to $S(Q'')$ (Theorem 5.3 from their paper), while maintaining an $O(n) \times O(n)$ grid (separate discussion in Section 7.1 of their paper). In particular, they call the property that the slopes of edges do not change the "preservation of orthogonality" throughout the morph. This satisfies all the necessary guarantees for Phase III.

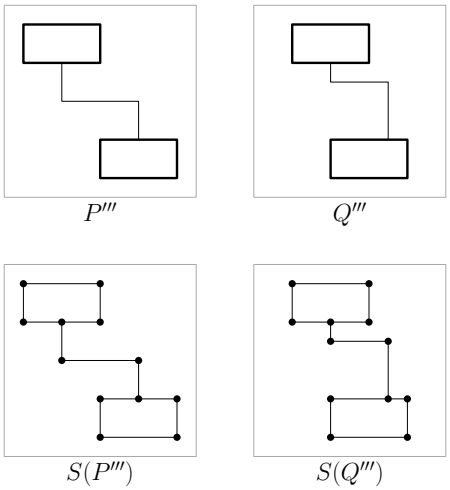


Figure 8.1: An example of how the straight-line orthogonal drawings $S(P'')$ and $S(Q'')$ are obtained from the orthogonal box drawings P''' and Q'''. Observe that $S(P'')$ and $S(Q'')$ are parallel drawings of the same graph.

Each of these results guarantees that all $O(n)$ explicit intermediate drawings will have $O(1)$ bends per edge and are drawn on a $O(n) \times O(n)$ grid, as desired.

Phases I and II run in $O(n^2)$ time each. While not explicitly stated in the paper of Biedl et al. [7], it can be determined from following their definitions that our Phase III runs in $O(n^2)$ time too. Hence, the total runtime is $O(n^2)$, as desired. □

8.3 Extensions

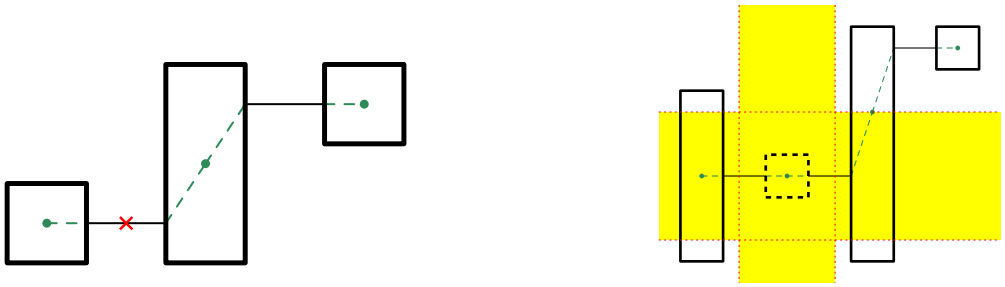
The statement of Theorem 1.2.1 applies only to planar straight-line drawings of connected graphs which are drawn on an $O(n) \times O(n)$ grid. However, the methods of this thesis can be extended to relax these constraints in several ways without any difficult constructions. We briefly outline these extensions here. In particular, this section will not give detailed constructions, since the constructions are quite straightforward given an understanding of the proof of Theorem 1.2.1.

Poly-line Input Drawings. First of all, the methods of this thesis extend to poly-line drawings as well. Let P and Q be a pair of compatible planar poly-line drawings

of the same graph G. Each drawing can be converted a planar straight-line drawing by replacing each bend with a vertex of degree 2. The resulting pair of drawings P_s and Q_s are no longer drawings of the same graph, since an edge of G may be drawn with different numbers of bends in P and Q. We denote their respective graphs as G_P and G_Q. However, Theorem 3.2.4 can still be applied to morph P_s and Q_s to orthogonal box drawings of G_P and G_Q, respectively. We call the respective resulting orthogonal box drawings P_b and Q_b. Let $\beta_P, \beta_Q : E(G) \to \mathbb{Z}^+$ be the functions that count the number of bends along each edge $e \in E(G)$ in the drawings P and Q, respectively. We will add vertices to G_P and G_Q to make them the same graph G', and add corresponding vertex boxes to P_b and Q_b.

Each edge in G corresponds to a path in G_P and G_Q. For each edge $e \in E(G)$ with $\beta_P(e) < \beta_Q(e)$ (resp. $\beta_P(e) > \beta_Q(e)$), modify P_b (and the corresponding graph) as follows: Choose some canonical edge segment e' in P_b (resp. Q_b) that is along the path corresponding to e. Use $O(1)$ linear morphs to add $\beta_Q(e) - \beta_P(e)$ (resp. $\beta_P(e) - \beta_Q(e)$) vertex boxes to e', so that the path now has $\max\{\beta_P(e), \beta_Q(e)\}$ vertices in both drawings (see Figure 8.2 for the construction that adds these vertex boxes). After this has been completed for all edges in $E(G)$, call the resulting drawings P'_b and Q'_b. They are now orthogonal box drawings of the same graph G'. Let $B = \sum_{e \in E(G)} \max\{\beta_P(e), \beta_Q(e)\}$. So, in particular, $|V(G')| = |V(G)| + B$. We continue to let n denote $|V(G)|$. Neither P'_b nor Q'_b contains any bends, and both are drawn on an $O(n+B) \times O(n+B)$ grid. Any planar orthogonal box drawing of G' (including implicit intermediate drawings) induces a corresponding admitted planar poly-line drawing of G (see Chapter 3 for a discussion of admitted drawings), so it suffices to find a linear morph sequence from P'_b to Q'_b. Thus, the remaining steps in the proof of Theorem 1.2.1 can be applied in order to find a linear morph sequence from P to Q that maintains an $O(n+B) \times O(n+B)$ grid and $O(\beta_P(e) + \beta_Q(e))$ bends for each edge $e \in E(G)$. The length of this linear morph sequence is $O(n+B)$, and the whole morph can be computed in $O((n+B)^2)$ time.

Disconnected Input Graphs. Theorem 1.2.1 can also be extended to handle disconnected graphs, at a cost to the time complexity, as well as either the grid size or the length of the linear morph sequence. There are two steps in the proof which make use of the connectivity of the underlying graph. First is the proof of Theorem 6.2.7, which uses connectivity to construct a trapezoidal map in linear time by applying Chazelle's algorithm [12]. This instead can be done in $O(n \log n)$ time for disconnected graphs by using standard methods [13]. It can even be done in $O(n \log \log n)$ time [10, discussion preceding Theorem 2.1] by swapping the balanced tree data structure used in the standard proof for a Van Emde Boas tree [46] (since the trapezoidal map is of vertical line segments on an $O(n) \times O(n)$ grid). The second step that makes use of the connectivity of the underlying



(a) An orthogonal box drawing P, with the (green and dashed) edge segments of its admitted drawing. The new vertex will be introduced at the location of the red cross

(b) The coordinates of P are modified so that no y-coordinate lies in the displayed horizontal strip, and no x-coordinate lies in the displayed vertical strip. A new vertex box can be added afterwards.

Figure 8.2: An example of how an additional vertex box can be added inside an edge segment using $O(1)$ linear morphs.

graph is in Phase III, which applies a result by Biedl et al. [7] that assumes connectivity. Biedl et al. observe that there is a linear morph sequence of length $O(n^2)$ (which can also be computed in $O(n^3)$ time) that accomplishes the same result for disconnected graphs. The total run-time for this approach with $O(n^2)$ linear morphs is $O(n^3)$. Separately, we note that Van Goethem et al. [47] extend their results in a manner that allows the handling of disconnected graphs, but without giving a method for analyzing the resulting morph's grid size, or the time complexity to compute the morph. Their approach is very different from ours, so there is no obvious analogous extension, but we can apply their algorithm directly to our drawings in place of the algorithm by Biedl et al. for Phase III to obtain some linear morph sequence of length $O(n)$. Due to some details of the construction of their algorithm, this linear morph sequence will also induce a linear morph sequence of the orthogonal boxes. However, since they do give a detailed analysis of their grid size or runtime, we only know them each to be polynomial with this approach.

Input Drawings on a Large Grid. Theorem 1.2.1 can be extended to handle arbitrarily large input grids. If the initial grid is of size $N \times N$ (for $N \in \Omega(n)$), then the grid size in Theorem 1.2.1 can be amended to be $O(N) \times O(N)$. The same construction applies. In particular, after Phase I, the problem can be reduced to one with an $O(n) \times O(n)$ grid by applying Theorem 6.1.2 to compress both drawings. This does not increase the time complexity to compute the whole morph, since Phase I can still be computed in $O(n^2)$ time (assuming that the input coordinates can fit in a single word in the word RAM model).

Morphs of Orthogonal Box Drawings. By applying only Phases II and III, we also get a nice result for morphs of orthogonal box drawings:

Theorem 8.3.1. *Let G be a connected planar graph on n vertices. If P and Q are a pair of compatible planar orthogonal box drawings of G, both drawn on an $O(n) \times O(n)$ grid, with $O(1)$ bends per edge, then there exists a planarity-preserving linear morph sequence from P to Q with length $O(n)$, such that the explicit intermediate drawings in each morph sequence are all also drawn on an $O(n) \times O(n)$ grid and have $O(1)$ bends per edge. Moreover, this sequence can be found in $O(n^2)$ time.*

Chapter 9

Conclusion

In proving Theorem 1.2.1, this thesis addressed the problem of morphing planar straight-line drawings using only $O(n)$ linear morphs, while maintaining at most $O(1)$ bends per edge, and also requiring that all drawings lie on an $O(n) \times O(n)$ grid. This improved over the analogous results of Lubiw and Petrick [33] which achieved a linear morph sequence of length $O(n^6)$, whose explicit intermediate drawings lie on an $O(n^3) \times O(n^3)$ grid, and whose edges have up to $O(n^5)$ bends. This also improved over the results of Alamdari et al. [1] and Erickson and Lin [21] (which achieved $O(n)$ linear morphs for straight-line drawings, but did not use a grid), with a small tradeoff of allowing $O(1)$ bends per edge. In order to accomplish this, we used the novel technique of reducing the straight-line morphing problem to a morphing problem on orthogonal box drawings.

In Section 8.3, we also outlined some simple extensions of Theorem 1.2.1 for polyline input drawings, disconnected input drawings, and input drawings lying on an arbitrarily large grid.

9.1 Open Problems

There is one obvious *big* open problem relevant to this thesis: Can Theorem 1.2.1 be improved to use 0 bends per edge throughout the morph while still keeping to an $O(n) \times O(n)$ grid? Recall that the state-of-the-art in planar straight-line morphing accomplishes exactly this *without* any guarantee of even a polynomial-sized grid.

The methods of this thesis also leave many interesting small open problems, which are much more approachable.

First, very few of the constant factors have been optimized. In many cases this is intentional, since it significantly improves the readability of the techniques. However, there are several interesting open questions regarding these constants. What is the minimum number of bends per edge using these techniques, what is the minimum grid size, and what is the minimum number of linear morphs required? Some easy calculations (not given here) show that for an n vertex graph, the techniques in this thesis use up to 36 bends per edge, use an approximately $154n \times 154n$ grid, and approximately $715n$ linear morphs (or $397n$ if steps which introduce/remove bends are ignored). During Phase II, the maximum absolute spirality of each edge is also bounded above by 8 (obtained at the end of Phase IIa). Hence, the maximum absolute difference in spirality is at most 16.

Secondly, almost every linear morph used in the construction was a unidirectional linear morph. The only significant exceptions were the linear morphs used to perform twists in Lemma 7.2.2. We also used one non-unidirectional linear morph in Phase III (hidden inside the black box), and many non-unidirectional linear morphs in Phase I (Theorem 3.2.4) but each of these could be replaced by a pair of unidirectional linear morphs. So, does there exist a sequence of only unidirectional linear morphs with all the same properties as Theorem 1.2.1? It should be noted that, while this is yet-unknown for orthogonal box drawings, this is true for orthogonal point drawings, where a twist can be replaced by two unidirectional morphs [7].

References

[1] Soroush Alamdari, Patrizio Angelini, Fidel Barrera-Cruz, Timothy M Chan, Giordano Da Lozzo, Giuseppe Di Battista, Fabrizio Frati, Penny Haxell, Anna Lubiw, Maurizio Patrignani, et al. How to morph planar graph drawings. *SIAM Journal on Computing*, 46(2):824–852, 2017.

[2] Soroush Alamdari, Patrizio Angelini, Timothy M Chan, Giuseppe Di Battista, Fabrizio Frati, Anna Lubiw, Maurizio Patrignani, Vincenzo Roselli, Sahil Singla, and Bryan T Wilkinson. Morphing planar graph drawings with a polynomial number of steps. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1656–1667. SIAM, 2013.

[3] Noga Alon and Raphael Yuster. Matrix sparsification and nested dissection over arbitrary fields. *Journal of the ACM (JACM)*, 60(4):1–18, 2013.

[4] Patrizio Angelini, Fabrizio Frati, Maurizio Patrignani, and Vincenzo Roselli. Morphing planar graph drawings efficiently. In *Graph Drawing: 21st International Symposium, GD 2013, Bordeaux, France, September 23-25, 2013, Revised Selected Papers 21*, pages 49–60. Springer, 2013.

[5] Fidel Barrera-Cruz, Penny Haxell, and Anna Lubiw. Morphing Schnyder drawings of planar triangulations. *Discrete & Computational Geometry*, 61:161–184, 2019.

[6] Therese Biedl. Height-preserving transformations of planar graph drawings. In Christian Duncan and Antonios Symvonis, editors, *Graph Drawing: 22nd International Symposium, GD 2014, Würzburg, Germany, September 24-26, 2014, Revised Selected Papers 22*, pages 380–391, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[7] Therese Biedl, Anna Lubiw, Mark Petrick, and Michael Spriggs. Morphing orthogonal planar graph drawings. *ACM Transactions on Algorithms (TALG)*, 9(4):1–24, 2013.

[8] Steward S Cairns. Deformations of plane rectilinear complexes. *The American Mathematical Monthly*, 51(5):247–252, 1944.

[9] Stewart S Cairns. Isotopic deformations of geodesic complexes on the 2-sphere and on the plane. *Annals of Mathematics*, pages 207–217, 1944.

[10] Timothy M Chan. Persistent predecessor search and orthogonal point location on the word RAM. *ACM Transactions on Algorithms (TALG)*, 9(3):1–22, 2013.

[11] Steven Chaplick, Philipp Kindermann, Jonathan Klawitter, Ignaz Rutter, and Alexander Wolff. Morphing rectangular duals. In *Graph Drawing and Network Visualization: 30th International Symposium, GD 2022, Tokyo, Japan, September 13–16, 2022, Revised Selected Papers*, pages 389–403. Springer, 2023.

[12] Bernard Chazelle. Triangulating a simple polygon in linear time. *Discrete & Computational Geometry*, 6(3):485–524, 1991.

[13] Mark de Berg, Otfried Cheong, Mark van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin Heidelberg, 3 edition, 2008.

[14] N. G. de Bruijn. Problems 17 and 18. *Nieuw Archief voor Wiskunde*, 2:67, 1954. In: Nieuwe Opgaven, Deel XX, no. 1-40.

[15] N. G. de Bruijn. Solutions to problems 17 and 18 from [14]. In *Wiskundige Opgaven met de Oplossingen*, volume 20, pages 19–20. Noordhoff, 1955.

[16] Hubert De Fraysseix, János Pach, and Richard Pollack. How to draw a planar graph on a grid. *Combinatorica*, 10:41–51, 1990.

[17] Leonardo De Moura and Nikolaj Bjørner. Z3: An efficient SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems: 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings 14*, pages 337–340. Springer, 2008.

[18] Giuseppe Di Battista and Fabrizio Frati. From Tutte to Floater and Gotsman: on the resolution of planar straight-line drawings and morphs. In *Graph Drawing and Network Visualization: 29th International Symposium, GD 2021, Tübingen, Germany, September 14–17, 2021, Revised Selected Papers 29*, pages 109–122. Springer, 2021.

[19] Reinhard Diestel. Graph theory 3rd ed. *Graduate texts in mathematics*, 173(33):12, 2005.

[20] Jürgen Doenhardt and Thomas Lengauer. Algorithmic aspects of one-dimensional layout compaction. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 6(5):863–878, 1987.

[21] Jeff Erickson and Patrick Lin. Planar and toroidal morphs made easier. *Journal of Graph Algorithms and Applications*, 27(2):95–118, 2023.

[22] István Fáry. On straight-line representation of planar graphs. *Acta scientiarum mathematicarum*, 11(229-233):2, 1948.

[23] L. Fejes Tóth and A. Heppes. Über stabile Körpersysteme. *Compositio Mathematica*, 15(2):119–126, 1963.

[24] Michael S Floater and Craig Gotsman. How to morph tilings injectively. *Journal of Computational and Applied Mathematics*, 101(1-2):117–129, 1999.

[25] Ulrich Fößmeier, Carsten Heß, and Michael Kaufmann. On Improving Orthogonal Drawings: The 4M-Algorithm. In Sue H. Whitesides, editor, *Graph Drawing: 6th International Symposium, GD 1998, Montréal, Canada, August 13-15, 1998, Proceedings*, pages 125–137, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[26] Jonas Gomes, Lucia Darsa, Bruno Costa, and Luiz Velho. *Warping & morphing of graphical objects*. Morgan Kaufmann, 1999.

[27] Craig Gotsman and Vitaly Surazhsky. Guaranteed intersection-free polygon morphing. *Computers & Graphics*, 25(1):67–75, 2001.

[28] Arthur B Kahn. Topological sorting of large networks. *Communications of the ACM*, 5(11):558–562, 1962.

[29] Goos Kant. A more compact visibility representation. *International Journal of Computational Geometry & Applications*, 7(03):197–210, 1997.

[30] Linda Kleist, Boris Klemz, Anna Lubiw, Lena Schlipf, Frank Staals, and Darren Strash. Convexity-increasing morphs of planar graphs. *Computational Geometry*, 84:69–88, 2019.

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

[32] Thomas Lengauer. Combinatorial algorithms for integrated circuit layout, 1990.

[33] Anna Lubiw and Mark Petrick. Morphing planar graph drawings with bent edges. *Journal of Graph Algorithms and Applications*, 15(2):205–227, 2011.

[34] Jirí Matousek. Intersection graphs of segments and $\exists\mathbb{R}$. *arXiv preprint arXiv:1406.2636*, 2014.

[35] Pierre Rosenstiehl and Robert E Tarjan. Rectilinear planar layouts and bipolar orientations of planar graphs. *Discrete & Computational Geometry*, 1:343–353, 1986.

[36] Walter Schnyder. Embedding planar graphs on the grid. In *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 138–148, 1990.

[37] Janet M Six, Konstantinos G Kakoulis, and Ioannis G Tollis. Techniques for the refinement of orthogonal graph drawings. In *Graph Algorithms And Applications 2*, pages 109–137. World Scientific, 2004.

[38] Jack Snoeyink and Jorge Stolfi. Objects that cannot be taken apart with two hands. In *Proceedings of the Ninth Annual Symposium on Computational Geometry (SoCG)*, pages 247–256, 1993.

[39] Sherman K Stein. Convex maps. *Proceedings of the American Mathematical Society*, 2(3):464–466, 1951.

[40] Vitaly Surazhsky and Craig Gotsman. High quality compatible triangulations. *Engineering with Computers*, 20:147–156, 2004.

[41] Roberto Tamassia. *Handbook of graph drawing and visualization*. CRC press, 2013.

[42] Roberto Tamassia and Ioannis G Tollis. A unified approach to visibility representations of planar graphs. *Discrete & Computational Geometry*, 1:321–341, 1986.

[43] Robert Endre Tarjan. Edge-disjoint spanning trees and depth-first search. *Acta Informatica*, 6(2):171–185, 1976.

[44] Carsten Thomassen. Deformations of plane graphs. *Journal of Combinatorial Theory, Series B*, 34(3):244–257, 1983.

[45] William Thomas Tutte. How to draw a graph. *Proceedings of the London Mathematical Society*, 3(1):743–767, 1963.

[46] P. van Emde Boas. Preserving order in a forest in less than logarithmic time and linear space. *Information Processing Letters*, 6(3):80–82, 1977.

[47] Arthur van Goethem, Bettina Speckmann, and Kevin Verbeek. Optimal morphs of planar orthogonal drawings. *Journal of Computational Geometry*, 13(1):263–297, 2022.

[48] Gopalakrishnan Vijayan and Avi Wigderson. Rectilinear graphs and their embeddings. *SIAM Journal on Computing*, 14(2):355–372, 1985.

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

[50] Stephen K Wismath. Characterizing bar line-of-sight graphs. In *Proceedings of the First Annual Symposium on Computational Geometry (SoCG)*, pages 147–152, 1985.

Index