Karlsruhe Institute of Technology

Bend Minimization of Ortho-Radial Graph Drawings

Master Thesis of

Matthias Wolf

At the Department of Informatics
Institute of Theoretical Computer Science

Reviewers:	Prof. Dr. Dorothea Wagner
	Prof. Dr. Peter Sanders
Advisors:	Lukas Barth, M.Sc.
	Dipl.-Inform. Benjamin Niedermann
	Dr. Ignaz Rutter

Time Period: 9th June 2016 – 8th December 2016

Statement of Authorship

I hereby declare that this document has been composed by myself and describes my own work, unless otherwise acknowledged in the text.

Karlsruhe, 7th December 2016

Abstract

We study orthogonal drawings of 4-planar graphs on cylinders—so called ortho-radial drawings. Equivalently these can be regarded as drawings on an ortho-radial grid formed by concentric circles and rays from the center of these circles but excluding the center itself. Ortho-radial drawings form a proper extension of orthogonal drawings in the plane.

We present ortho-radial representations as a means to describe the shape of these drawings without fixing the actual coordinates and edge lengths. Additionally, we give conditions on the representation that are both necessary and sufficient for an ortho-radial drawing to exist. Being able to describe the shape by ortho-radial representations is a crucial step in order to apply Tamassia's Topology-Shape-Metrics framework to ortho-radial graph drawing.

Furthermore, we show that it is $\mathcal{NP}$-complete to determine whether a given 4-planar graph without a fixed embedding admits an ortho-radial drawing without edge bends. But when one restricts oneself to cactus graphs, it is possible to find bend-minimal drawings in linear time.

Deutsche Zusammenfassung

In dieser Arbeit werden orthogonale Zeichnungen von 4-planaren Graphen auf Zylindern – so genannte ortho-radiale Zeichnungen – betrachtet. Diese können alternativ auch als Zeichnungen auf ein radiales Gitter angesehen werden. Ein solches Gitter besteht aus konzentrischen Kreisen und Strahlen, die vom gemeinsamen Zentrum der Kreise ausgehen, ohne dabei jedoch das Zentrum selbst zu enthalten. Ortho-radiales Zeichnen ist eine echte Erweiterung des orthogonalen Graphenzeichnens in der Ebene.

Um die Form der Zeichnung zu beschreiben, ohne jedoch Koordinaten oder Kantenlängen festzulegen, werden ortho-radiale Beschreibungen vorgestellt. Für diese werden außerdem Bedingungen aufgezeigt, welche sowohl notwendig als auch hinreichend für die Existenz einer Zeichnung sind. Die Fähigkeit, die Form einer Zeichnung zu beschreiben, ist ein entscheidender Schritt zur Anwendung von Tamassias *Topology-Shape-Metrics-Framework* auf ortho-radiales Graphenzeichnen.

Zusätzlich wird gezeigt, dass es $\mathcal{NP}$-vollständig ist zu entscheiden, ob ein gegebener 4-planarer Graph ohne feste Einbettung eine ortho-radiale Zeichnung ohne Kantenknicke besitzt. Beschränkt man sich jedoch auf Kaktusgraphen, so kann in Linearzeit eine knickminimale Zeichnung bestimmt werden.

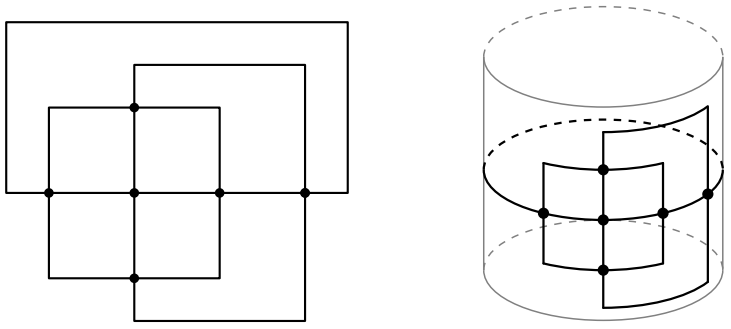
Contents

1. Introduction

Visualization helps to understand the structure of the data and to extract useful information. Moreover, data can often be described by graphs: They may directly represent existing structures, e.g., road networks or other transportation networks, or they may formalize more abstract relations. For instance, each person of a group could be represented by a vertex and an edge between two vertices indicates that the two people know each other. Therefore, automatic or manual drawing of graphs is a common task, for which different drawing styles exist.

One often used style is *orthogonal graph drawing*, that is, edges are drawn as one or more axis-parallel line segments. Figure 1.1(a) contains an orthogonal drawing of the octahedron graph as an example. A *bend* is a point where the direction of an edge changes. During the construction of orthogonal drawings one often seeks to minimize the area of the drawing or the number of bends.

In this thesis we study orthogonal drawings of 4-planar graphs (planar graphs with maximum degree at most 4) on cylinders such that no two edges cross. These drawings are also called *ortho-radial drawings*. In Figure 1.1(b) the octahedron is drawn on the cylinder. Any orthogonal drawing in the plane can be placed on a cylinder. But in ortho-radial drawings there may also be cycles winding themselves around the cylinder—so called *essential cycles*. Hence, ortho-radial drawings are a proper extension of orthogonal drawings. Moreover, using essential cycles to draw a planar graph may reduce the required number of bends. For



(a) (b)

Figure 1.1: (a) An orthogonal drawing of the octahedron. (b) An ortho-radial drawing of the octahedron.

instance, any orthogonal drawing of a triangle in the plane includes at least one bend. On a cylinder however the triangle may be drawn as an essential cycle by placing all vertices on a circle whose center lies on the axis of the cylinder. The edges can then be simply drawn as straight lines. In fact, any cycle may be drawn without bends in this way. Comparing the two drawings of the octahedron in Figure 1.1, one notes that the ortho-radial drawing has fewer bends than the orthogonal drawing. Moreover, no edge has more than two bends in the ortho-radial drawing, whereas any orthogonal drawing of the octahedron needs at least one edge with three bends [BK98]. This already shows that ortho-radial drawings of a graph may have fewer bends than orthogonal drawings of the same graph.

Ortho-radial drawings can equivalently be seen as drawings on an ortho-radial grid, which is formed by concentric cycles and arcs from the center of these cycles but excluding the center. Drawings of this kind are also used to visualize geographic data. For instance, it has been proposed to represent metro networks in this way [RNC16] and automatic creation of such maps has been studied [Bar16, FLW14].

For the creation of orthogonal drawings the *Topology-Shape-Metrics framework* presented by Tamassia [Tam87] is often used. This framework suggests a three step approach: First, an embedding of the graph is fixed, then its shape is determined, and finally the coordinates and edge lengths are assigned to obtain a drawing. A key part of the framework is the ability to describe the shape abstractly by *orthogonal representations* without needing to fix the concrete positions of the vertices and edges. Our goal is to apply the Topology-Shape-Metrics framework to ortho-radial graph drawing. To this end we present *ortho-radial representations*, which describe the shape of a graph. We furthermore characterize those representations that describe shapes which can be drawn. To the best of our knowledge such a characterization was previously only known for cycles and theta graphs, which are formed by three paths between two vertices [HHT09].

1.1 Contribution and Outline

In Chapter 2 we give the definitions of terms that are used throughout this thesis, which includes a formal definition of ortho-radial drawings.

We present ortho-radial representations in Chapter 3 as a means to describe the shape of ortho-radial drawings of 4-planar graphs. Moreover, we characterize those representations that describe shapes that can be drawn. We first prove this characterization for rectangular graphs (cf. Section 3.4). To create drawings of representations of such graphs, we use flow networks similar to those used for drawing rectangular graphs in the plane in the Topology-Shape-Metrics framework [Tam87]. We then characterize the representation for general 4-planar graphs in Section 3.5. By providing a procedure to rectangulate ortho-radial representation we reduce the general characterization to the case of rectangular graphs.

In Chapter 4 we prove that it is $\mathcal{NP}$-complete to decide whether a given 4-planar graph without a prescribed embedding admits an ortho-radial drawing without any edge bends. Therefore, it is also hard to find bend-minimal drawings of arbitrary 4-planar graphs.

But when one restricts oneself to cactus graphs, drawings with the minimum number of bends can be computed in linear time as shown in Chapter 5. The algorithm uses dynamic programming on a tree that represents the structure of the input cactus graph.

Finally, Chapter 6 wraps up all the results of this thesis. Moreover, we state possible directions of future research related to ortho-radial drawings.

1.2 Related Work

Hasheminezhad et al. [HHT09] characterize the drawings of essential cycles: There is a drawing of an essential cycle C with a fixed shape if and only if either all edges of C point right or left, or at least one edge points up and one down. It is clearly necessary that all cycles of a graph can be drawn individually in order for an ortho-radial drawing of the whole graph to exist. They however also show that this is not sufficient, even when one restricts oneself to graphs consisting of three paths with common start and endpoints—so called *theta graphs*. For this graph class they also present a characterization of drawable shapes: They distinguish between five cases, in which certain subpaths contain certain shapes. A shape is then drawable if and only if any of the five cases applies. Note that this approach is different from our characterization since we do not directly require certain subgraph with specific shapes but use more general conditions.

In a *rectangular ortho-radial drawing* the outer and the central face are drawn as circles and the other faces are rectangles. No edge bends occur in these drawings, as otherwise the faces incident to a bend would not be rectangles. For a planar graph with maximum degree 3 and a fixed embedding, it can be determined in linear time whether there is a rectangular ortho-radial drawing of this graph [HHMT10]. This result was extended to a linear time algorithm for rectangular drawings on a sphere using a grid that consists of circles of latitude and meridians [HHM09].

Orthogonal drawings in the plane have been extensively studied with particular respect to minimizing edge bends. Garg and Tamassia [GT95] showed that testing whether an orthogonal drawing without edges exists is $\mathcal{NP}$-complete. For 3-planar graphs and series-parallel graphs however polynomial time algorithms were presented by di Battista et al. [DBLV98]. Any 4-planar graph except for the octahedron can be drawn with at most two bends per edge [BK98]. Testing whether one bend per edge is sufficient is possible in quadratic time [BKRW14]. Moreover, variants in which different edges may have different numbers of bends have been studied [BKRW14, BLR16].

In the case that an embedding of the graph is fixed the total number of bends can be minimized by Tamassia's flow network [Tam87], which can be solved in $\mathcal{O}(n^{3/2} \log n)$ time, where n denotes the number of vertices of the input graph [CK12].

Further topics that are related to graph drawing on a cylinder but which are not studied in this thesis include cylindric visibility representations [TT91] and rectangular duals on a cylinder [HHMT10].

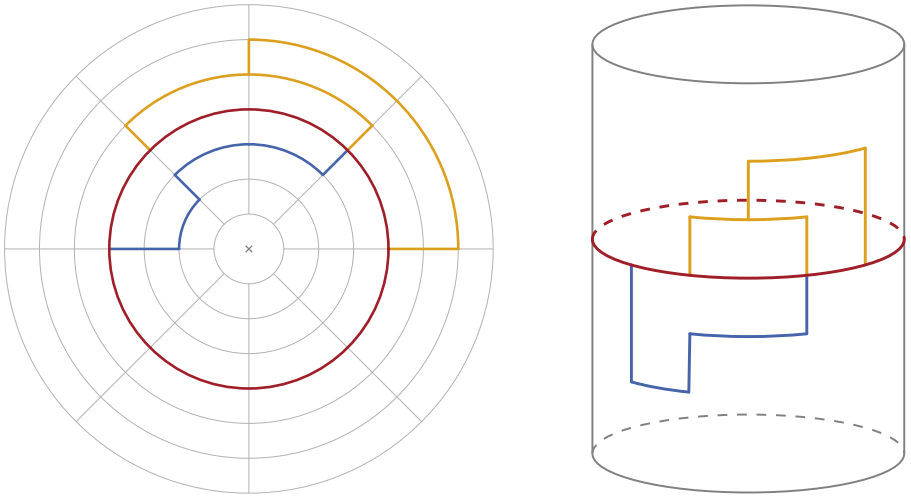
2. Preliminaries

A graph $G = (V, E)$ consists of a set V of vertices and a set E of edges. We refer to the directed edge from a vertex $v \in V$ to another vertex $w \in V$ by vw. If H is a subgraph of G, we denote the set of edges that belong to H by $E(H)$. Similarly, $V(H)$ denotes the set of vertices of H. For a subset $S \subseteq V$ of vertices, we denote the subgraph of G induced by S by $G[S]$. In an *embedding* of a graph, the order of the incident edges around each vertex is fixed. A plane graph is a planar graph with a fixed embedding. If the maximum degree of all vertices in a planar graph is at most 4, it is said to be a *4-planar graph*. Similarly, *4-plane graph* refers to a 4-planar graph with a fixed embedding.

A *path* P in a graph G is a sequence of vertices $v_1 \dots v_k$ such that G contains the edge $v_i v_{i+1}$ and no vertex appears twice. Similarly, a *cycle* is a sequence of vertices $v_1 \dots v_k v_1$ such that G contains the edge $v_i v_{i+1}$ for $i = 1, \dots, k-1$ and $v_k v_1$. If a cycle C contains all vertices only once except that the first and the last vertex are equal, it is a *simple cycle*. Note that we follow the convention that paths cannot contain vertices multiple times but cycles can. The cycles we consider are always directed—usually in clockwise direction. If an embedding is fixed, C divides the plane into two parts. We call the area that is locally to the right of C the *interior* of C and the area to its left the *exterior*. We include the vertices and edges of C in both the interior and the exterior. If the cycles are directed clockwise—as it is mostly the case—the interior is bounded and the exterior is not.

Two paths $P = v_1 \dots v_k$ and $Q = w_1 \dots w_\ell$ with $v_k = w_1$ can be joined together to form $P + Q = v_1 \dots v_k w_2 \dots w_\ell$. For any path $P = v_1 \dots v_k$, we define its reverse $\overline{P} = v_k \dots v_1$. For a path P and vertices u and v on P, we denote the subpath of P from u to v (including these vertices) by $P[u, v]$. For a cycle C that contains any edge at most once (e.g., if C is simple), we extend the notion of subpaths as follows: For two edges e and e' on C, the subpath $C[e, e']$ is defined as the unique path on C that starts with e and ends with e'. If the start vertex v of e identifies e uniquely, i.e., C contains v exactly once, we may write $C[v, e']$ to describe the path on C from v to e'. Analogously, we may identify e' with its endpoint if this is unambiguous.

We represent a face as a cycle f in which the interior of the face lies locally to the right of f. Note that f may not be simple since cut vertices may appear multiple times on f. But no directed edge is used twice by f. Therefore, the notation of subpaths of cycles applies to faces. Note furthermore that the cycle bounding the outer face of a graph is directed counter-clockwise, whereas all other faces are bounded by cycles directed clockwise.



(a) (b)

Figure 2.1: Equivalent drawings of the same graph. (a) Ortho-radial drawing of a graph on a grid. (b) The same drawing interpreted as an orthogonal drawing on a cylinder.

2.1 Orthogonal and Ortho-Radial Drawings

In ortho-radial drawings the graph is placed on an ortho-radial grid. An example of such a drawing is shown in Figure 2.1(a). An *ortho-radial grid* is formed by concentric circles and N spokes, where $N \in \mathbb{N}$ is fixed.

- Concentric circles around the origin:

$$G_C = \left\{(x, y) \in \mathbb{R}^2 \mid \sqrt{x^2 + y^2} \in \mathbb{N}\right\}$$

- N spokes:

$$G_S = \left\{(r \cdot \cos \tfrac{2\pi k}{N}, r \cdot \sin \tfrac{2\pi k}{N}) \mid k \in \{0, \ldots, N-1\}, r \in [1, \infty)\right\}$$

The point $(0, 0)$ is called the *center* of the grid and we represent it by a cross in our figures as for example in Figure 2.1(a). Note however that the center does not belong to the grid. The distance of any point to the center is called the *r-coordinate* of that point.

In an *ortho-radial drawing* of a 4-planar graph $G = (V, E)$ the graph is drawn on an ortho-radial grid such that the vertices are placed at intersections of circles and spokes and the edges are represented by curves on the grid (cf. Figure 2.1(a)). Formally, an ortho-radial drawing $\Delta = (\delta_V, \delta_E)$ consists of two functions:

- $\delta_V : V \to G_C \cap G_S$ maps the vertices to their coordinates.
- $\delta_E : E \times [0, 1] \to G_C \cup G_S$ maps the edges to curves.

These functions must satisfy certain properties:

1. The function δ_V is injective.
2. For $e \in E$ the function δ_e defined by $\delta_e(x) = \delta_E(e, x)$ is continuous.
3. For $vw \in E$ it is either $\delta_E(vw, 0) = \delta_V(v)$ and $\delta_E(vw, 1) = \delta_V(w)$, or $\delta_E(vw, 0) = \delta_V(w)$ and $\delta_E(vw, 1) = \delta_V(v)$.

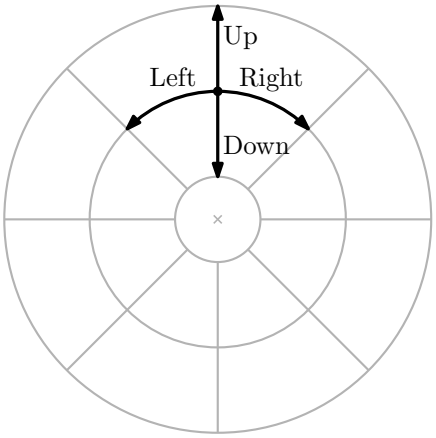


Figure 2.2: The possible edge directions.

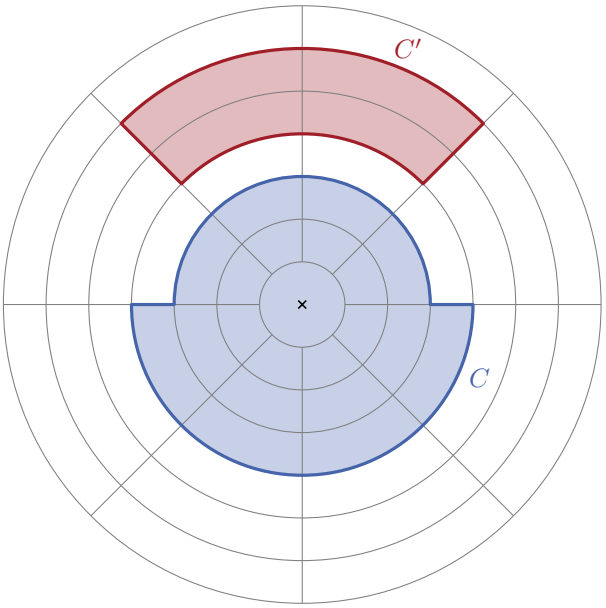


Figure 2.3: The cycle C is essential, whereas C' is non-essential.

The first condition ensures that no two different vertices are drawn at the same point. By the second condition each edge is drawn as one curve without jumps and the third condition makes sure that all edges start and end at the correct vertices.

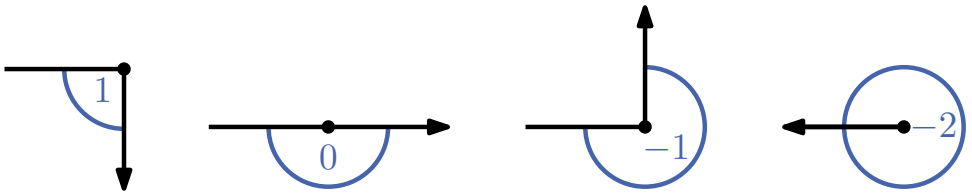
Ortho-radial drawings can also be thought of as orthogonal drawings on a cylinder. Figure 2.1 shows two equivalent ortho-radial drawings of the same graph—one on an ortho-radial grid and one on a cylinder. Edge segments that are originally drawn on rays from the center are parallel to the axis of the cylinder, whereas segments that are originally drawn as arcs on a cycle around the center are drawn on cycles parallel to the ends of the cylinder.

Each directed edge in an ortho-radial drawing can point in four directions as shown in Figure 2.2: Edges directed away from the center point *up*, whereas edges directed towards the center point *down*. Otherwise, the edges are drawn as arcs of circles around the center. They point *right*, if they are directed clockwise, and *left* otherwise. Edges pointing left or right are also said to be *horizontal* and edges pointing up or down are said to be *vertical*.

There are two fundamentally different ways how a cycle C can be drawn: The center may lie in the interior or the exterior of C. In the former case we say a cycle is *essential* and in the latter case it is *non-essential*. For instance, in Figure 2.3 the cycle C is essential and C' is non-essential.

In an ortho-radial drawing Δ of G, there are two special faces: One unbounded face, called the *outer face*, and the *central face* containing the center of the drawing. These two faces are equal if and only if Δ contains no essential cycles. All other faces of G are called *regular*. Ortho-radial drawings without essential cycles are equivalent to orthogonal drawings [HHT09]. That is, any such ortho-radial drawing can be transformed to an orthogonal drawing of the same graph with the same outer face and vice versa.

A face f in an ortho-radial drawing is a *rectangle* if and only if its boundary does not make any left turns. That is, if f is a regular face, there are exactly 4 right turns, and if f is the central or the outer face, there are no turns at all. Note that by this definition f cannot be a rectangle if it is both the outer and the central face.



(a) 90° (b) 180° (c) 270° (d) 360°

Figure 2.4: The rotations for turns of different angles.

2.2 Rotation

For two edges uv and vw that enclose the angle $\alpha \in \{90°, 180°, 270°, 360°\}$ at v (such that the measured angle lies locally to the right of uvw), we define the rotation $\text{rot}(uvw) = 2 - \alpha/90°$. In Figure 2.4 the rotations for different turns are shown: The rotation is 1 if there is a right turn at v, 0 if uvw is straight, and -1 if a left turn occurs at v. If $u = w$, it is $\text{rot}(uvw) = -2$.

We define the rotation of a path $P = v_1 \dots v_k$ as the sum of the rotations at its internal vertices.

$$\text{rot}(P) = \sum_{i=2}^{k-1} \text{rot}(v_{i-1}v_iv_{i+1})$$

Similarly, for a cycle $C = v_1 \dots v_k v_1$, its rotation is the sum of the rotations at all its vertices (where we define $v_0 = v_k$ and $v_{k+1} = v_1$).

$$\text{rot}(P) = \sum_{i=1}^{k} \text{rot}(v_{i-1}v_iv_{i+1})$$

Let v be a vertex on a face f and u and w be the vertices before and after v on f, respectively. Then, we define the *rotation of v in f* by

$$\text{rot}_f(v) = \text{rot}(uvw).$$

When splitting a path at an edge, the sum of the rotations of the two parts is equal to the rotation of the whole path.

Observation 2.1. *Let P be a path from vertex s to vertex t. For all edges $e \in E(P)$ it holds that*

$$\text{rot}(P) = \text{rot}(P[s, e]) + \text{rot}(P[e, t]).$$

Furthermore, reversing a path changes all left turns to right turns and vice versa. Hence, the sign of the rotation is flipped.

Observation 2.2. *For any path P it is*

$$\text{rot}(\overline{P}) = -\text{rot}(P).$$

The third observation analyzes what happens if one makes a detour as shown in Figure 2.5.

Observation 2.3. *Let P be a path from v to w and $xy \in P$ an edge such that x is an internal vertex of P.*

(a) If xy lies locally to the left of P, it is

$$\text{rot}\left(P[v, x] + xy\right) + \text{rot}\left(yx + P[x, w]\right) = \text{rot}\left(P\right) - 2.$$

(b) If xy lies locally to the right of P, it is

$$\text{rot}\left(P[v, x] + xy\right) + \text{rot}\left(yx + P[x, w]\right) = \text{rot}\left(P\right) + 2.$$

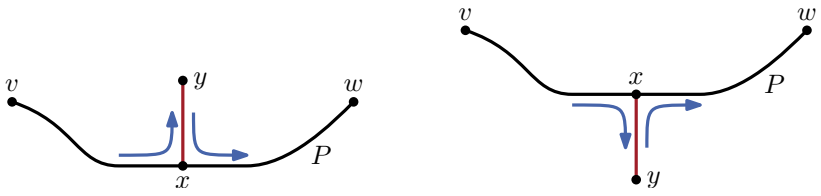


Figure 2.5: The situation of Observation 2.3. If one makes a detour via xy (without accounting for the 360°-turn at y), the rotation changes by -2 if xy is to the left of P, or $+2$ if xy lies to the right.

2.3 Flow Networks

A *flow network* is a directed graph $N = (V, A)$ with

- a demand function $d : V \to \mathbb{Z}$,
- a function for the lower bound of the flow $l : A \to \mathbb{N}_0$, and
- a function for the upper bound of the flow $u : A \to \mathbb{N}_0 \cup \{\infty\}$.

A *feasible flow* in N is a function $f : A \to N_0$ such that the following conditions are met:

1. For each node $v \in V$ it is $\sum_{uv \in A} f(uv) - \sum_{vw \in A} f(vw) = d(v)$.
2. For each arc $a \in A$ it is $l(a) \leq f(a) \leq u(a)$.

The first condition ensures that the flow is preserved at the nodes. By the second condition the amount of flow along an arc respects the capacity of the arc. Note that our definition of flow networks does not include a cost function, since we are only interested in the existence of feasible flows and do not require minimum cost flows. The existence of a feasible flow can be tested in polynomial time as shown, for example, in [CLRS01].

3. Ortho-Radial Representations

Instead of dealing with the drawing of a graph itself, it is often simpler to work with a representation that fixes the general shape of the drawing but not the exact lengths and positions of edges and vertices. In this chapter we present a representation for ortho-radial drawings of 4-planar graphs, which extends the orthogonal representations introduced by Tamassia [Tam87]. We furthermore present a set of conditions for these representations that are both necessary and sufficient for the existence of an ortho-radial drawing. Hence, algorithms for ortho-radial drawings (e.g., for bend minimization) do not need to deal with coordinates on the grid. Instead, they may work with the representations.

3.1 Labelings of Essential Cycles

Before we give the definition of ortho-radial representations, we introduce labelings of essential cycles with respect to an edge on the outer face. These labelings prove to be a valuable tool to ensure that the drawings of all cycles of the graph fit together.

Let Δ be an ortho-radial drawing of a 4-planar graph G. We fix an edge $e^* = rs$, called the *reference edge*, on the outer face directed such that the outer face lies locally to its left.

Definition 3.1 (Labeling)**.** *Let C be a simple, essential cycle in G directed such that it contains the central face in its interior, and let P be a path from s to a vertex v on C. The* labeling ℓ_C^P *of C induced by P is defined for each edge segment e of an edge on C by*

$$\ell_C^P(e) = \mathrm{rot}(e^* + P + C[v, e]).$$

This definition is illustrated in Figure 3.1. Two such paths P and Q from s to vertices on C are *equivalent* (in symbols $P \equiv_C Q$) if the labelings induced by these paths agree on all

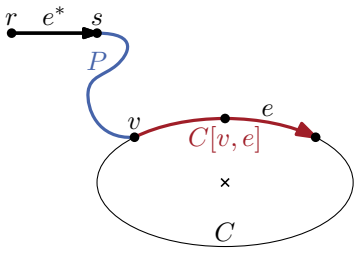


Figure 3.1: The labeling of e induced by P is $\ell_C^P(e) = \mathrm{rot}(e^* + P + C[v, e])$.

edge segments of C, i.e., $\ell_C^P(e) = \ell_C^Q(e)$ for all segments e of C. We are mostly interested in labelings that are induced by paths that intersect C only at their endpoints and we call such paths *elementary*.

Note that the labeling does not depend on the exact coordinates of the vertices or the edge lengths but only on their relative positions. Hence, to determine the labeling of a cycle C induced by a path P, it is sufficient to know the directions of e^* and the edges on C and P. In Section 3.3 we further investigate the properties of labelings.

3.2 Definition of Ortho-Radial Representations

In this section, we define ortho-radial representations, which extend orthogonal representations as described by Tamassia [Tam87]. The main differences are additional constraints to ensure essential cycles can be drawn and minor adjustments to the rotation of the exterior and the central face.

Definition 3.2 (Ortho-Radial Representation)**.** *An* ortho-radial representation *of a connected plane graph G with maximum degree 4 consists of a list $H(f)$ of triples (e, s, a) for each face f, where e is an edge on f, s a binary string and $a \in \{90, 180, 270, 360\}$. Additionally, the outer face and the central face are fixed, and an edge e^* directed such that the outer face lies to its left—called the* reference edge*—is given.*

We interpret the fields of a triple in $H(f)$ in the following way: e denotes an edge on f directed such that f lies to the right of e. The string s describes the bends encountered when moving along e, i.e., the k-th bit of s describes the k-th bend on e beginning at the bend that is closest to the starting point of e. We interpret 0 as a right bend (angle of 90°) and 1 as a left bend (angle of 270°). The field a represents the angle inside f from e to the following edge in degrees.

Using this information we define the *rotation* of such a triple $t = (e, s, a)$ as

$$\operatorname{rot}(t) = \operatorname{zeroes}(s) - \operatorname{ones}(s) + \frac{180 - a}{90},$$

where $\operatorname{zeroes}(s)$ and $\operatorname{ones}(s)$ are the numbers of 0 and 1 in s, respectively. That is, the rotation includes both the turns on the edge and the angle to the next edge. The rotation of a face is then defined as

$$\operatorname{rot}(f) = \sum_{t \in H(f)} \operatorname{rot}(t).$$

Note that an ortho-radial representation fixes the rotation of all paths. Therefore, an ortho-radial representation determines the labelings of essential cycles.

Definition 3.3. *An ortho-radial representation is* valid *if the following conditions hold:*

1. *The sum of the angles around each vertex (as given by the values of the a-fields of all incoming edges) is 360.*
2. *If two triples (e, s, a) and (e', s', a') refer to the same edge but in different directions (i.e., $e' = \bar{e}$), s' can be obtained by reversing s and replacing all ones with zeros and vice versa.*
3. *For each face f, it is*

$$\operatorname{rot}(f) = \begin{cases} 4, & f \text{ is a regular face,} \\ 0, & f \text{ is the outer or the central face but not both,} \\ -4, & f \text{ is both the outer and the central face.} \end{cases}$$

4. *For each simple, essential cycle C in G, there is a labeling ℓ_C^P of C induced by an elementary path P such that either $\ell_C^P(s) = 0$ for all segments s of edges of C, or there are segments s_+ and s_- on C such that $\ell_C^P(s_+) > 0$ and $\ell_C^P(s_-) < 0$.*

Conditions 1 and 3 ensure that the sum of the angles at vertices and inside the faces are correct. Condition 2 makes sure that the two descriptions of the bends of an edge are consistent. The last condition guarantees that all cycles in the graph can be drawn together consistently. For an essential cycle C that does not satisfy this condition, either all labels of segments on C are non-negative or all are non-positive. In the former case C is called *decreasing* and in the latter case *increasing*. Note that an increasing (decreasing) cycle contains an edge with a negative (positive) label. Cycles with label 0 everywhere are neither in- nor decreasing.

If an ortho-radial representation Γ contains no essential cycles, Condition 4 is vacuously true and the other conditions ensure that Γ is also a valid orthogonal representation. Therefore, orthogonal representations can be viewed as a strict subset of ortho-radial representations.

Although the definition of ortho-radial representation supports edge bends, it is often more convenient to assume that no edges have bends. We can easily achieve this by adding dummy vertices at the bends, which yields a subdivision of the graph. From here on, we assume in the remainder of the chapter that no edge bends exist.

An ortho-radial representation fixes the direction of all edges: To determine the direction of an edge e, we consider a path P from the reference edge to e including both edges. Different such paths may have different rotations but we can observe that these rotations differ by a multiple of 4.

Observation 3.4. *For any two paths P and Q starting with the reference edge and ending with the same edge e, we have* $\operatorname{rot}(P) \equiv \operatorname{rot}(Q) \mod 4$.

An edge e is directed *right*, *down*, *left*, and *up* if $\operatorname{rot} P$ is congruent to 0, 1, 2, and 3 modulo 4, respectively. If e lies on an essential cycle C, one can consider the label $\ell_C^Q(e)$ induced by any path Q instead of explicitly choosing a path P since the labels are defined as rotations of such paths.

Having introduced all necessary definitions we are able to state our main theorem, which characterizes those ortho-radial representations for which a drawing exists. The remainder of this chapter is devoted to the proof of this theorem.

Theorem 3.5. *Let G be a 4-planar graph and Γ an ortho-radial representation of G. The representation Γ is valid if and only if there is an ortho-radial drawing of G respecting Γ.*

Before we turn to the proof, we analyze the properties of labelings in the following section. Section 3.4 then contains a proof of the characterization for the case that all faces are rectangles. In Section 3.5 we extend this result to arbitrary graphs by showing how valid ortho-radial representations of arbitrary graphs can be rectangulated.

3.3 Properties of Labelings

Orthogonal and ortho-radial representations mainly differ by the additional constraint on the labelings of essential cycles. Therefore, we need to study the properties of these labelings. Throughout this section G is a 4-planar graph with ortho-radial representation Γ, which satisfies Conditions 1–3 of Definition 3.3. That is, Γ is valid except that it might contain increasing or decreasing cycles. By Observation 3.4 an ortho-radial representation

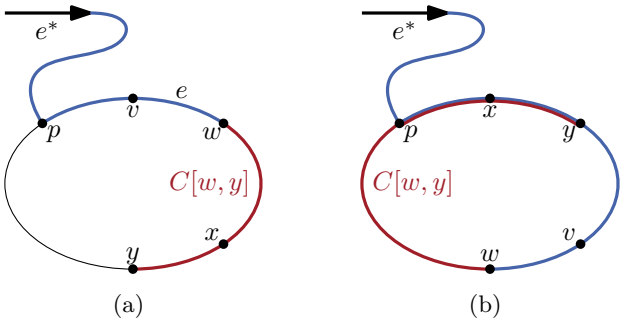


Figure 3.2: The situation of Lemma 3.7. (a) The edge xy does not lie on $C[p, w]$ and P' contains no duplicate edges. (b) The edge xy lies on $C[p, w]$ and P' goes around C completely.

fixes the directions of all its edges. Thus, an ortho-radial representation also fixes the labelings of essential cycles.

We start with a simple observation that follows from the fact that the rotation of essential cycles is 0.

Observation 3.6. *For any two edges e and e' on a simple, essential cycle C, it holds that*

$$\text{rot}(C[e, e']) = \ell_C(e') - \ell_C(e).$$

Our first goal is to show that two elementary paths to the same essential cycle C induce identical labelings of C. As a first step, we prove that two paths are already equivalent if the labels for one edge are equal. Thus, to test whether two paths are equivalent, it suffices to check the labels of only one edge.

Lemma 3.7. *Let C be a simple, essential cycle and P and Q two paths from the endpoint of the reference edge to vertices on C. If there is an edge e on C such that $\ell_C^P(e) = \ell_C^Q(e)$, then P and Q are equivalent.*

Proof. Denote the endpoints of P and Q on C by p and q, respectively. Let $e = vw$ be an edge on C such that $\ell_C^P(vw) = \ell_C^Q(vw)$, which implies by the definition of the labeling

$$\text{rot}(e^* + P + C[p, vw]) = \text{rot}(e^* + Q + C[q, vw]). \tag{3.1}$$

For any edge $e' = xy$ on C, we obtain $P' = P + C[p, vw] + C[w, y]$ and $Q' = Q + C[q, vw] + C[w, y]$ by adding $C[w, y]$ to the end of the two paths to e. Equation 3.1 then implies

$$\text{rot}(e^* + P') = \text{rot}(e^* + Q'). \tag{3.2}$$

If $xy \notin C[p, w]$ (as illustrated in Figure 3.2(a)), it is $C[p, y] = C[p, vw] + C[w, y]$ and therefore $\ell_C^P(e') = \text{rot}(e^* + P')$. Otherwise, P' contains C completely and $\text{rot}(e^* + P') = \text{rot}(e^* + C[p, y]) + \text{rot}(C) = \ell_C^P(e')$, since C is essential and hence it is $\text{rot}(C) = 0$. This case is shown in Figure 3.2(b).

Analogously, we obtain $\ell_C^Q(e') = \text{rot}(e^* + Q')$. In conjunction with Equation 3.2 we get in total

$$\ell_C^P(e') = \text{rot}(e^* + P') = \text{rot}(e^* + Q') = \ell_C^Q(e'). \tag{3.3}$$

Hence, the labelings induced by P and Q are equal. □

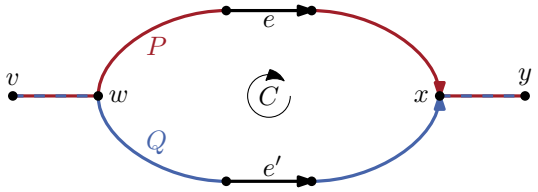


Figure 3.3: The situation of Lemma 3.8. The paths P and Q use different sides of the cycle C and the edges e and e' cut P and Q in two parts.

In the lemma above it is assumed that the labelings coincide at one edge. In order to find such an edge, we analyze the rotations of different paths from the reference edge to one edge on an essential cycle. Under mild conditions all these paths that end with the same edge have the same rotation. To show this, we compare two paths from the reference edge to the cycle. These paths may split at some point and meet again later (cf. Figure 3.3). That is, they use two different paths along a cycle. Note that this cycle is different from the cycle the paths end on. The following lemma shows that the rotation of the paths are equal if the paths leave the cycle on the correct side.

Lemma 3.8. *Let C be a simple cycle in G. Consider two vertices w and x on C and edges $vw \notin E(C)$ and $xy \notin E(C)$ to and from these vertices, respectively. Let $P = vw + C[w,x] + xy$ and $Q = vw + \overline{C}[w,x] + xy$.*

(a) If C is a non-essential cycle and vw and xy lie in the exterior of C, or

(b) if C is an essential cycle and vw lies in the exterior of C and xy in the interior,

then it is $\operatorname{rot}(P) = \operatorname{rot}(Q)$.

Proof. To prove the equality, we show in the following that $\operatorname{rot}(P) - \operatorname{rot}(Q) = 0$. We first cut the paths in two parts each and analyze them separately. To this end we choose arbitrary edges e on $P[w,x]$ and e' on $Q[w,x]$ as shown in Figure 3.3. By Observation 2.1 it is

$$\begin{aligned}\operatorname{rot}(P) - \operatorname{rot}(Q) = &\operatorname{rot}(P[v,e]) - \operatorname{rot}(Q[v,e']) \\ &+ \operatorname{rot}(P[e,y]) - \operatorname{rot}(Q[e',y]).\end{aligned} \tag{3.4}$$

By definition it is $P[v,e] = vw + C[w,e]$ and $Q[v,e'] = vw + \overline{C}[w,e']$. Therefore, we can calculate that difference of the rotations as follows:

$$\begin{aligned}\operatorname{rot}(P[v,e]) - \operatorname{rot}(Q[v,e']) &= \operatorname{rot}(P[v,e]) + \operatorname{rot}(\overline{Q}[\overline{e'},v]) \\ &= \operatorname{rot}(vw + C[w,e]) + \operatorname{rot}(C[\overline{e'},w] + wv) \\ &= \operatorname{rot}(C[\overline{e'},e]) - 2\end{aligned} \tag{3.5}$$

The last equality follows from Observation 2.3, since vw lies in the exterior of C and therefore locally to the left of $C[\overline{e'},e]$. In a similar spirit one obtains an equation for the second part:

$$\operatorname{rot}(P[e,y]) - \operatorname{rot}(Q[e',y]) = \operatorname{rot}(C[e,\overline{e'}]) + c \tag{3.6}$$

Here, c represents a constant, which depends on whether xy lies in the exterior of C (Case (a); $c = -2$) or in the interior (Case (b); $c = 2$).

Substituting Equations 3.5 and 3.6 in Equation 3.4, we get

$$\operatorname{rot}(P) - \operatorname{rot}(Q) = \operatorname{rot}(C[\overline{e'},e]) + \operatorname{rot}(C[e,\overline{e'}]) - 2 + c. \tag{3.7}$$

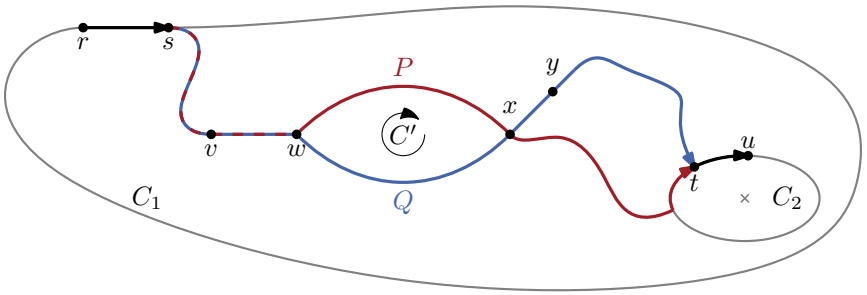


Figure 3.4: Two paths P and Q from s to t, which lie between the essential cycles C_1 and C_2. Here, the cycle C' formed by $P[w, x]$ and $\overline{Q}[x, w]$ is non-essential.

Note that the paths in the equation above together form the cycle C. Hence, it is $\text{rot}(C[\overline{e'}, e]) + \text{rot}(C[e, \overline{e'}]) = \text{rot}(C)$ and the equation simplifies to

$$\text{rot}(P) - \text{rot}(Q) = \text{rot}(C) - 2 + c. \tag{3.8}$$

In (a) the cycle C is non-essential and therefore it is $\text{rot}(C) = 4$. Moreover, we have $c = -2$. Plugging these values in Equation 3.8, we get $\text{rot}(P) - \text{rot}(Q) = 0$.

Similarly, in (b) it is $c = 2$ and $\text{rot}(C) = 0$ since C is essential. Therefore, Equation 3.8 again gives the desired result. □

Applying this result repeatedly, we show that the label of an edge e on an essential cycle C induced by two elementary paths P and Q is the same, i.e., $\ell_C^P(e) = \ell_C^Q(e)$. We can even prove a slightly more general result, where the paths do not need to start at the reference edge but must stay between two essential cycles C_1 and C_2, which is illustrated in Figure 3.4. If one is interested in paths from the reference edge, one can choose the cycle bounding the outer face as C_1.

Lemma 3.9. *Let C_1 and C_2 be two essential cycles in G such that C_2 lies in the interior of C_1. Fix an edge rs on C_1 and an edge $e = tu$ on C_2. Let P and Q be two paths starting at s and ending at t such that both paths lie in the interior of C_1 and in the exterior of C_2. Then, it is*

$$\text{rot}(rs + P + tu) = \text{rot}(rs + Q + tu).$$

Proof. Let k be the number of directed edges on P that do not lie on Q. We prove the equivalence of P and Q by induction on k. If $k = 0$, Q contains P completely. Since both paths have the same start and endpoint, P and Q are equal. Hence, the claim follows immediately.

If $k > 0$, there is a first edge vw on P such that the following edge does not lie on Q. Let x be the first vertex on P after w that lies on Q and let y be the vertex on $Q + tu$ that follows x immediately. This situation is illustrated in Figure 3.4. As both P and Q end at t, these vertices always exist. Consider the cycle $C' = P[w, x] + \overline{Q}[x, w]$. We may assume without loss of generality that the edges of C' are directed such that the outer face lies in the exterior of C' (otherwise we may reverse the edges and work with $\overline{C'}$ instead). Note that $Q[x, t]$ cannot intersect C' (except for x), since the internal vertices of $P[w, x]$ do not lie on Q by the definition of x and Q does not intersect itself. Therefore, xy lies on the same side of C' as C_2.

If C' is non-essential, it does not contain the central face, and hence, C' cannot contain C_2 in its interior. Thus, both vw and xy lie in the exterior of C'. By Lemma 3.8(a), we have

$$\text{rot}(vw + P[w, x] + xy) = \text{rot}(vw + Q[w, x] + xy). \tag{3.9}$$

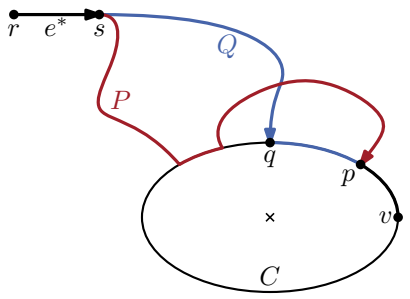


Figure 3.5: The situation of Lemma 3.10. Two paths P and Q from the reference edge to vertices on the essential cycle C. Both paths lie in the exterior of C but only Q is elementary.

If C' is essential, C_2 lies in the interior of C'. Therefore, vw lies in the exterior of C' and xy in its interior or boundary. Thus, Lemma 3.8(b) states that

$$\operatorname{rot}(vw + P[w,x] + xy) = \operatorname{rot}(vw + Q[w,x] + xy). \tag{3.10}$$

In both cases it follows from $P[s,w] = Q[s,w]$ that

$$\operatorname{rot}(rs + P[s,x] + xy) = \operatorname{rot}(rs + Q[s,x] + xy). \tag{3.11}$$

For $Q' = P[s,x] + Q[x,t]$ it therefore holds that

$$\operatorname{rot}(rs + Q + tu) = \operatorname{rot}(rs + Q' + tu). \tag{3.12}$$

As Q' includes the part of P between w and x, it misses fewer edges from P than Q does. Hence, the induction hypothesis implies

$$\operatorname{rot}(rs + P + tu) = \operatorname{rot}(rs + Q' + tu). \tag{3.13}$$

Thus, Equations 3.12 and 3.13 yield the desired result:

$$\operatorname{rot}(rs + P + tu) = \operatorname{rot}(rs + Q + tu) \tag{3.14}$$

□

Combining the previous lemma with Lemma 3.7 shows that the labelings induced by elementary paths are equal.

Lemma 3.10. *Let C be an essential cycle in G and let $e^* = rs$ be the reference edge. If P and Q are paths from s to vertices on C such that P and Q lie in the exterior of C, they are equivalent. In particular, elementary paths are equivalent.*

Proof. First assume that one of the paths, say Q, is elementary as shown in Figure 3.5. Let p and q be the endpoints of P and Q, respectively, and let v be the vertex following p on C when C is directed such that the central face lies in its interior. We define $Q' = Q + C[q,p]$. It is easy to verify that both P and Q' are paths and lie in the exterior of C. Therefore, it follows from Lemma 3.9 that

$$\ell_C^P(pv) = \operatorname{rot}(e^* + P + pv) = \operatorname{rot}(e^* + Q' + pv) = \ell_C^Q(pv).$$

Thus, P and Q are equivalent by Lemma 3.7.

If neither P nor Q are elementary, choose any elementary path R to a vertex on C. The argument above shows that $P \equiv_C R$ and $R \equiv_C Q$, and thus $P \equiv_C Q$. □

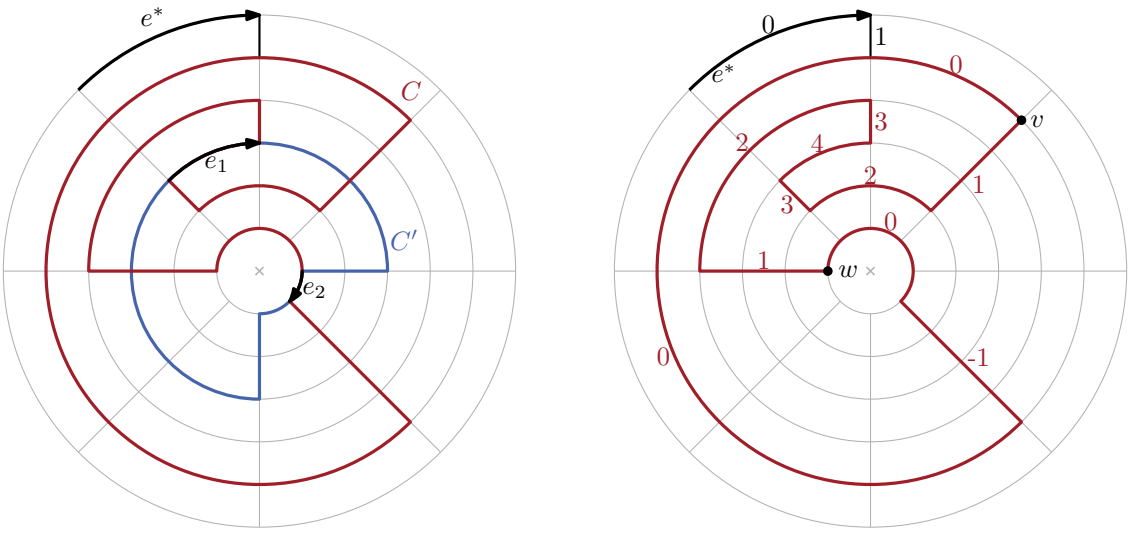


Figure 3.6: (a) The essential cycles C and C' have both common edges with different labels ($\ell_C(e_1) = 4 \neq 0 = \ell_{C'}(e_1)$) and ones with equal labels ($\ell_C(e_2) = \ell_{C'}(e_2) = 0$). (b) The labels of the cycle C. All labels of $C[v, w]$ are positive.

In the remainder of this section all labelings are induced by elementary paths. By the lemma above, the labelings are independent of the choice of the elementary path. Therefore, we drop the superscript P and write $\ell_C(e)$ for the labeling of an edge e on an essential cycle C.

If an edge e lies on two simple, essential cycles C and C', the labels $\ell_C(e)$ and $\ell_{C'}(e)$ may not be equal in general. In Figure 3.6 the labels of the edge e_1 are different but e_2 has label 0 on both cycles. Note that e_2 is incident to the central face of the subgraph of G formed by the two cycles C and C'. In the following lemma we show that this is a sufficient condition for the equality of the labels. It is however not a necessary condition as an edge that is not incident to the central face may be labeled the same for both cycles.

Lemma 3.11 (Intersection Lemma)**.** *Let C and C' be two essential cycles and let $H = C + C'$ be the subgraph of G formed by these two cycles. Let v be a common vertex of C and C' on the central face of H and consider the edge vw on C. Denote the vertices before v on C and C' by u and u', respectively. Then, it is*

$$\ell_C(uv) + \mathrm{rot}(uvw) = \ell_{C'}(u'v) + \mathrm{rot}(u'vw).$$

In particular, if vw belongs to both C and C', the labels of e are equal, i.e., $\ell_C(vw) = \ell_{C'}(vw)$.

Proof. Let f be the cycle bounding the central face of H and we first assume that not only v but the whole edge vw lies on f. Let P and Q be elementary paths from the endpoint s of the reference edge to vertices t and t' on C and C', respectively. Define $\tilde{P} = P + C[t, v]$ and $\tilde{Q} = Q + C'[t', v]$. The paths $\tilde{P}$ and $\tilde{Q}$ lie in the exterior of f. Thus, Lemma 3.9 can be applied to $\tilde{P}$ and $\tilde{Q}$:

$$\begin{aligned}\mathrm{rot}(e^* + \tilde{P}) + \mathrm{rot}(uvw) &= \mathrm{rot}(e^* + \tilde{P} + vw) \\ &= \mathrm{rot}(e^* + \tilde{Q} + vw) = \mathrm{rot}(e^* + \tilde{Q}) + \mathrm{rot}(u'vw) \qquad (3.15)\end{aligned}$$

By the definition of labelings it is $\ell_C(uv) = \text{rot}(e^* + \tilde{P})$ and $\ell_{C'}(u'v) = \text{rot}(e^* + \tilde{Q})$. Substitution into the previous equation gives the desired result:

$$\ell_C(uv) + \text{rot}(uvw) = \ell_{C'}(u'v) + \text{rot}(u'vw) \tag{3.16}$$

If vw does not lie on the central face, then the edge vw' on C' does lie on the central face, where w' denotes the vertex on C' after v. By the argument above we have

$$\ell_C(uv) + \text{rot}(uvw') = \ell_{C'}(u'v) + \text{rot}(u'vw'). \tag{3.17}$$

Since vw lies locally to the left of both uvw' and $u'vw'$, it is

$$\text{rot}(u_ivw) = \text{rot}(u_ivw') - \alpha, \tag{3.18}$$
$$\text{rot}(u_ivw) = \text{rot}(u_ivw') - \alpha \tag{3.19}$$

for the same constant $\alpha \in \{1, 2\}$. Hence, we get

$$\ell_C(uv) + \text{rot}(uvw) + \alpha = \ell_{C'}(u'v) + \text{rot}(u'vw) + \alpha. \tag{3.20}$$

Canceling α on both sides gives the desired result.

If vw lies on both C and C', Observation 3.6 implies $\ell_C(uv) + \text{rot}(uvw) = \ell_C(vw)$ and $\ell_{C'}(u'v) + \text{rot}(u'vw) = \ell_{C'}(vw)$. Hence, the result above can be simplified to $\ell_C(vw) = \ell_{C'}(vw)$. □

Intuitively, positive labels can often be interpreted as going downwards and negative labels as going upwards. In Figure 3.6(b) all edges of $C[v, w]$ have positive labels and in total the r-coordinate decreases along this path, i.e., the r-coordinate of v is greater than the r-coordinate of w. Yet, the edges on $C[v, w]$ point in all possible directions—even upwards. One can still interpret a maximal path with positive labels as leading downwards with the caveat that this is a property of the whole path and does not impose any restriction on the directions of the individual edges.

Using this intuition, we expect that a path P going down (positive labels) cannot intersect a path Q going up (negative labels) such that P starts below Q and ends above it. In the following lemma, we show that this assumption is indeed correct if we restrict ourselves to intersections on the central face—a situation that is depicted in Figure 3.7.

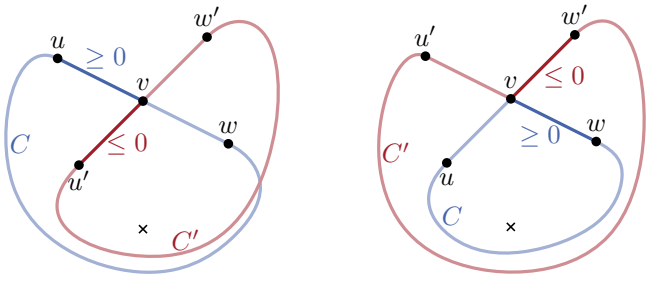


Figure 3.7: Possible intersection of two cycles C and C' at v. (a) The labels of the incoming edges satisfy $\ell_C(uv) \geq 0$ and $\ell_{C'}(u'v) \leq 0$. The edges vw and vw' may be exchanged. (b) The labels of the outgoing edges satisfy $\ell_C(vw) \geq 0$ and $\ell_{C'}(vw') \leq 0$. The edges uv and $u'v$ may be exchanged.

Lemma 3.12. *Let C and C' be two simple, essential cycles in G sharing at least one vertex. Let $H = C + C'$ be the subgraph of G composed of the two cycles C and C' and denote the central face of H by f. Take subpaths uvw of C and $u'vw'$ of C' with the same vertex v in the middle such that v lies on f.*

(a) If $\ell_C(uv) \geq 0$ and $\ell_{C'}(u'v) \leq 0$, $u'v$ lies in the interior of C.

(b) If $\ell_C(vw) \geq 0$ and $\ell_{C'}(vw') \leq 0$, vw' lies in the exterior of C.

Proof. (a) Since the central face f lies in the interior of both C and C' and v on the boundary of f, one of the edges vw and vw' lies on f. We denote this edge by vx and it is either $x = w$ (as in Figure 3.7(a)) or $x = w'$. By the Intersection Lemma we have

$$\ell_C(uv) + \mathrm{rot}(uvx) = \ell_{C'}(u'v) + \mathrm{rot}(u'vx). \tag{3.21}$$

Applying $\ell_C(uv) \geq 0$ and $\ell_{C'}(u'v) \leq 0$, we obtain

$$\mathrm{rot}(uvx) \leq \mathrm{rot}(u'vx). \tag{3.22}$$

Therefore, $u'v$ lies to the right of or on uvx and thus in the interior of C.

(b) Assume that vw' does not lie in the exterior of C. That is, vw' lies locally to the right of uvw. Therefore, vw' lies on f, and the Intersection Lemma implies together with Observation 3.6 that

$$\begin{aligned} 0 \geq \ell_{C'}(vw') &= \ell_{C'}(u'v) + \mathrm{rot}(u'vw') \\ &= \ell_C(uv) + \mathrm{rot}(uvw'). \end{aligned} \tag{3.23}$$

Since vw' lies locally to the right of uvw, it is $\mathrm{rot}(uvw) < \mathrm{rot}(uvw')$ and therefore

$$\begin{aligned} 0 &\geq \ell_C(uv) + \mathrm{rot}(uvw') \\ &> \ell_C(uv) + \mathrm{rot}(uvw) = \ell_C(vw). \end{aligned} \tag{3.24}$$

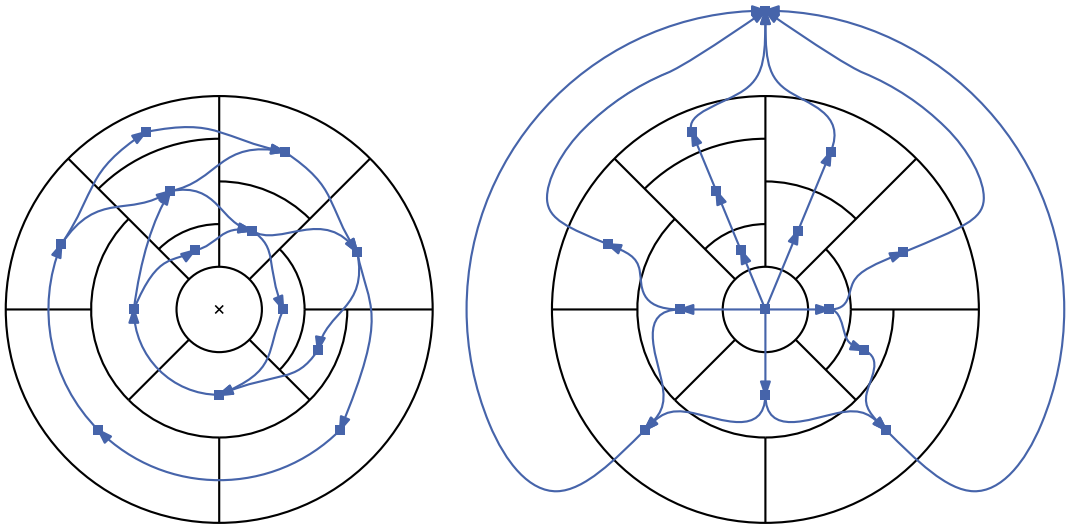
Here, the last equality follows from Observation 3.6. But this contradicts the assumption $\ell_C(vw) \geq 0$. Hence, vw' must lie in the exterior of C. □

3.4 Characterization of Rectangular Graphs

Our overall goal is to show that a graph with a given ortho-radial representation can be drawn if and only if the representation is valid (cf. Theorem 3.5). In this section we prove the characterization for rectangular graphs, i.e., graphs whose faces are all rectangles. This result is extended in Section 3.5 to the characterization for arbitrary graphs.

To produce a drawing from an ortho-radial representation Γ of a graph G, we need to assign consistent edge lengths. To this end we use two flow networks—one for the vertical and one for the horizontal edges. These networks are straightforward adaptions of the networks used for drawing rectangular graphs in the plane, which are presented in [DBETT99], for instance. In the following, the words *vertex* and *edge* refer to the vertices and edges of the graph G, whereas *node* and *arc* are used for the flow networks. Additionally, *face* always refers to a face of G.

The network $N_{\mathrm{ver}} = (F_{\mathrm{ver}}, A_{\mathrm{ver}})$ with nodes F_{ver} and arcs A_{ver} for the vertical edges contains one node for each face of G except for the central and the outer face. All nodes have a demand of 0. For each vertical edge e in G there is an arc $a_e = fg$ in N_{ver}, where f is the face to the left of e and g the one to its right when e is directed upwards. The flow on fg



(a) N_{ver} (b) N_{hor}

Figure 3.8: The flow networks N_{ver} and N_{hor} for an example graph G, which are used to assign the lengths to the vertical and horizontal edges of G, respectively. For simplicity, the edge from the outer to the central face in N_{hor} is omitted.

has the lower bound $l(fg) = 1$ and the upper bound $u(fg) = \infty$. An example of this flow network is shown in Figure 3.8(a).

To obtain a drawing from a flow in N_{ver}, we set the length of a vertical edge e to the flow on a_e. The conservation of the flow at each node f ensures that the two vertical sides of the face f have the same length.

In a similar fashion the network N_{hor} is built to assign lengths to the horizontal edges. There is a node for all faces of G including the central and the outer face. Each horizontal edge e of G induces an arc $a_e = fg$ in N_{hor}, where f is the face to the right of e when e points to the right and g the face to the left of e. Additionally, N_{hor} includes one arc from the outer to the central face. Again, all arcs require a minimum flow of 1 and have infinite capacity. The demand of all nodes is 0. Figure 3.8(b) shows an example of such a flow network.

Theorem 3.13. *Let $G = (V, E)$ be a plane graph with ortho-radial representation Γ satisfying Conditions 1–3 of Definition 3.3 such that all faces are rectangles. Let N_{ver} and N_{hor} be the flow networks as defined above. Then, the following statements are equivalent:*

(i) There is an ortho-radial drawing of Γ.

(ii) There are feasible flows in N_{hor} and N_{ver}.

(iii) No $S \subseteq F_{ver}$ exists such that there is an arc from $F_{ver} \setminus S$ to S in N_{ver} but not vice versa.

(iv) Γ is valid.

Proof. (i) $\Rightarrow$ (iv): Let Δ be an ortho-radial drawing of G preserving the embedding described by Γ. Since Γ already satisfies Conditions 1–3 of Definition 3.3, we only need to show that Γ contains neither increasing nor decreasing cycles. Let C be a simple, essential cycle directed such that the center lies in its interior. We construct a path P from the reference edge to a vertex on C such that the labeling of C induced by P attains both positive and negative values.

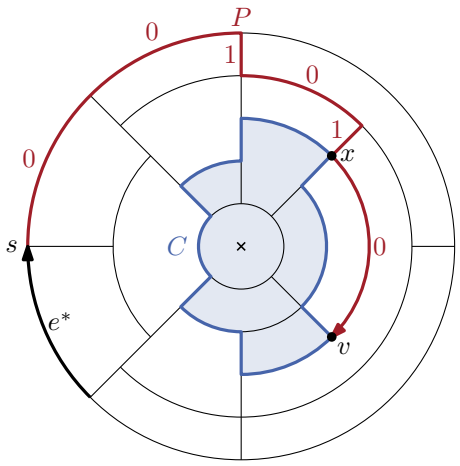


Figure 3.9: The path P from s to v—constructed backwards by going only up or left—does not intersect the interior of C. The rotations of the edges on P relative to e^* are 0 or 1.

In Δ either all vertices of C have the same r-coordinate, or there is a maximal subpath Q of C whose vertices all have the maximum r-coordinate among all vertices of C. In the first case we may choose the endpoint v of the path P arbitrarily, whereas in the second case we select the first vertex of Q as v.

We construct the path P backwards (i.e., the construction yields $\overline{P}$) as follows: Starting at v we choose the edge going upwards from v if it exists, or the one leading left. Since all faces of G are rectangles, at least one of these always exists. This procedure is repeated until the endpoint s of the reference edge is reached. An example of the constructed path is shown in Figure 3.9.

To show that this algorithm terminates, we assume that this was not the case. As G is finite, there must be a first time a vertex w is visited twice. Hence, there is a cycle C' in Δ containing w such that all edges of C' go left or up. As all drawable essential cycles with edges leading upwards must also have edges that go down [HHT09], all edges of C' are horizontal. By construction no edge incident to a vertex of C' leads upwards. But the only cycle with this property is the one enclosing the outer face since G is connected. This cycle however contains the reference edge and therefore the algorithm halts.

This not only shows that the construction of P ends, but also that P is a path (i.e., the construction does not visit a vertex twice). However, P might be not elementary, since it may intersect C multiple times (e.g., in Figure 3.9 the path P contains two vertices of C: v and x). But v has the smallest r-coordinate among all vertices of P and the largest among those on C. Thus, no part of P lies inside C. Hence, Lemma 3.10 guarantees that the labeling ℓ_C^P induced by P coincides with the labeling ℓ_C induced by any elementary path.

Let v' be the vertex following v on C. By construction of P the labeling of vv' induced by P is 0. If all edges of C are horizontal, this implies $\ell_C^P(e) = 0$ for all edges e of C. Otherwise, we claim that the edge $e_- = uv$ directly before the maximal horizontal subpath Q containing v and the edge $e_+ = wx$ directly following Q on C have labels -1 and $+1$, respectively. Since all edges on Q are horizontal and e_+ goes down, we have $\mathrm{rot}(C[v,x]) = 1$ and therefore $\ell_C^P(e_+) = 1$. Similarly, we obtain $\ell_C^P(uv) = \ell_C^P(vv') - \mathrm{rot}(uvv') = -1$ from $\mathrm{rot}(uvv') = 1$.

(iv) $\Rightarrow$ (iii): Instead of proving this implication directly, we show the contrapositive. That is, we assume that there is a set $S \subseteq F_{\mathrm{ver}}$ of nodes in N_{ver} such that S has no outgoing but at least one incoming arc. From this assumption we derive that Γ is not valid, as we find an increasing or decreasing cycle.

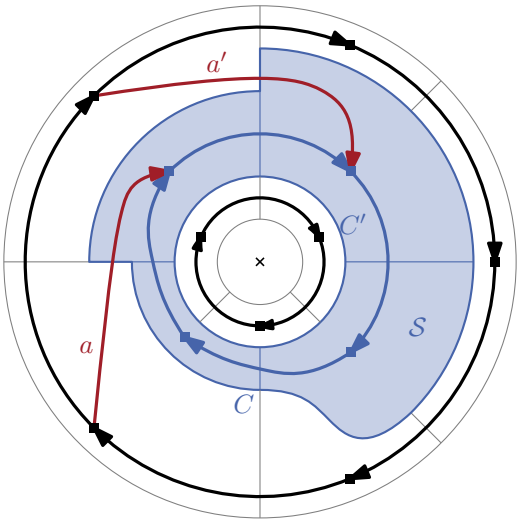


Figure 3.10: A set S of nodes in a graph G such that $N_{\text{ver}}[S]$ has no outgoing but two incoming arcs a and a'. The set of faces $\mathcal{S}$ corresponding to the nodes in S are shaded with blue. Note that the edge on C at the bottom is curved because G does not admit an ortho-radial drawing.

Without loss of generality, S can be chosen such that $N_{\text{ver}}[S]$ is weakly connected. If $N_{\text{ver}}[S]$ is not weakly connected, at least one weakly-connected component of $N_{\text{ver}}[S]$ possesses an incoming arc and we can work with this component instead. As each node of S corresponds to a face of G, S can also be considered as a collection of faces of G. To distinguish between the two interpretations of S, we refer to this collection of faces by $\mathcal{S}$. Our goal is to show that the outermost or the innermost boundary of $\mathcal{S}$ forms an increasing or decreasing cycle in the original graph G. Figure 3.10 shows an example of such a set S of nodes. Here, the arcs a and a' lead from a node outside of S to one in S. These arcs cross edges on the outer boundary C of $\mathcal{S}$, which point upwards.

Each node of S has at least one outgoing arc, which must end at another node of S. Hence, S contains a cycle, and therefore $\mathcal{S}$ separates the outer and the central face of G. Thus, the cycle C that constitutes the outermost boundary of $\mathcal{S}$, i.e., the smallest cycle containing all faces of $\mathcal{S}$ in its interior, is essential. Similarly, we define C' as the cycle forming the innermost boundary of $\mathcal{S}$. By assumption there is an arc a from a node entering S from the outside. Since $N_{\text{ver}}[S]$ is weakly connected, $\mathcal{S}$ has no holes and the edge e crossed by a either lies on C or C'. In the former case e points up, whereas in the latter case e points down. If e lies on C, we prove in the following that C is an increasing cycle. Similarly, one can show that C' is a decreasing cycle if e lies on C'. We only present the argument for C as the one for C' is similar.

We first observe that no edge on C is directed downwards, since the arc a_e corresponding to a downward edge e would lead from a node in S to a node outside of S. To restrict the possible labels for edges on C, we construct an elementary path P from the endpoint s of the reference edge to a vertex on C. The construction is similar to the one we used above, but this time the construction works forwards starting at s. If the current vertex lies on C, the path is completed. Otherwise, we append the edge going down if it exists, or the one that goes to the right. As above one can show that this procedure produces a path P. It is even elementary, because we stop when a vertex on C is reached.

Let v be the endpoint of P and w the vertex on C following v. By construction it is $\text{rot}(e^* + P) \in \{0, 1\}$. Therefore, we have $\ell_C(vw) = 0$ if vw points right or $\ell_C(vw) = -1$ if it points up. Since no edge on C has a label that is congruent to 1 modulo 4 (i.e., no

edge points down) and the labels of neighboring edges differ by -1, 0 or 1, we obtain $\ell_C(e') \in \{-2, -1, 0\}$ for all edges $e' \in C$. In particular, it is $\ell_C(e') \leq 0$. But the edge e crossed by the arc a points upwards and therefore $\ell_C(e) = -1$. Hence, C is an increasing cycle and Γ is not valid.

(iii) $\Rightarrow$ (ii): We claim that N_{hor} always possesses a feasible flow. To construct a flow, note that N_{hor} without the arc from the outer face g to the central face f is a directed acyclic graph with f as its only source and g as its only sink. For each arc $a \neq gf$ in N_{hor} there is a directed path P_a from f to g via a. Adding the arc gf, we obtain the cycle $C_a = P_a + gf$. We let one unit flow along each of these cycles C_a. Adding all these flows gives the desired flow in N_{hor}. By construction each arc a lies on at least one cycle (namely C_a), and hence, the minimum flow of 1 on a is satisfied.

To construct a feasible flow in N_{ver} we again compose cycles of flow. In order to find a directed cycle containing an arc fg, we define the set S_g of all nodes h for which there exists a directed path from g to h in N_{ver}. By definition there is no arc from a vertex in S_g to a vertex not in S_g. As N_{ver} satisfies (iii), S_g does not have any incoming arcs either. Hence, it is $f \in S_g$ and there is a directed path from g to f. Closing this path with the arc fg results in a cycle of N_{ver}, which we denote by C_{fg}.

Repeating this process for all arcs of N_{ver}, we obtain the set $\mathcal{C}$ of the cycles C_a for $a \in A_{\text{ver}}$. We set the flow $\varphi(a)$ on each arc a as the number of the cycles in $\mathcal{C}$ containing a. Since a lies on C_a, the flow on a is at least 1. As φ is the sum of unit flows along cycles, the flow is conserved at all nodes, i.e., the sum of the flows on incoming arcs of a node f is equal to the flow on arcs leaving f. Therefore, φ is a feasible flow in N_{ver}.

(ii) $\Rightarrow$ (i): Given a feasible flow φ in N_{ver}, we set the length of each vertical edge e of G to the flow $\varphi(a_e)$ on the arc a_e that crosses e. For each face f of G the total length of its left side is equal to the total amount of flow entering f. Similarly, the length of the right side is equal to the amount of flow leaving f. As the flow is preserved at all nodes of N_{ver}, the left and right sides of f have the same length. Similarly, one obtains the length of the horizontal edges from a flow in N_{hor}. □

By [HHT09] any ortho-radial drawing of a graph satisfies Conditions 1–3 of Definition 3.3. Therefore, the theorem above implies the characterization of drawable ortho-radial representations for rectangular graphs.

Corollary 3.14 (Special Case of Theorem 3.5 for Rectangular Graphs)**.** *Let G be a 4-planar graph and Γ an ortho-radial representation of G such that all faces are rectangles. The representation Γ is valid if and only if there is an ortho-radial drawing of G respecting Γ.*

3.5 Rectangulation

In the previous section we proved that there is an ortho-radial drawing if and only if the ortho-radial representation is valid for the case of rectangular graphs. We extend this result to arbitrary graphs by reduction to the rectangular case. Throughout the section, Γ is a valid ortho-radial representation of a 4-planar graph G. As before we assume that Γ does not contain any edge bends, which can be achieved by adding dummy vertices at the bends.

In order to use the characterization of rectangular graphs from the previous section, we augment G such that all faces become rectangles. The algorithm for this rectangulation is presented in the following section. In Section 3.5.2 we prove some preliminary results, which are needed for the correctness proof in Section 3.5.3.

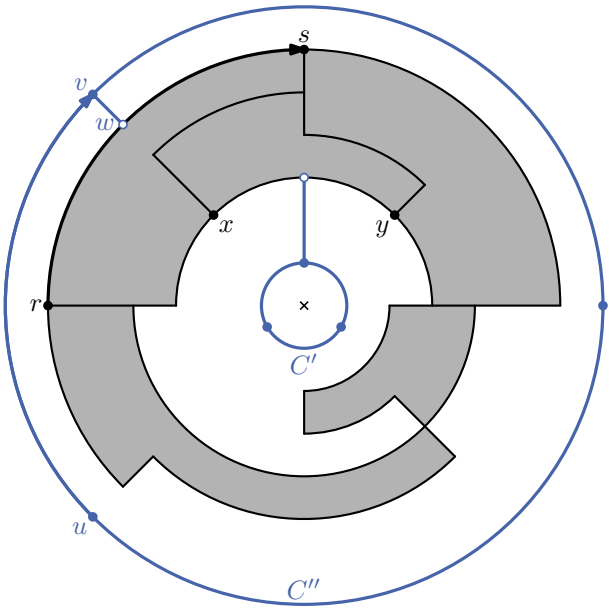


Figure 3.11: The outer and the central face are rectangulated by adding cycles of length 3. The cycle C' is connected to an edge xy that has label 0 and C'' is connected to a new vertex w on the old reference edge rs. The edge uv is selected as the new reference edge.

3.5.1 The Rectangulation Algorithm

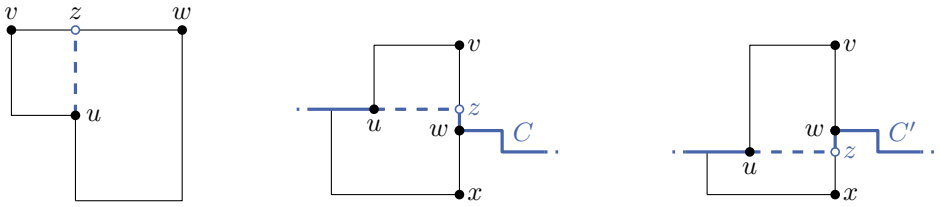
The rectangulation algorithm gets a graph G and an ortho-radial representation Γ of G as input and outputs an augmentation of Γ whose faces are all rectangles. This augmentation works solely with the ortho-radial representation and does not fix any coordinates or edge lengths.

The first step of the rectangulation deals with the outer and the central face. By definition they are rectangular if they are bounded by essential cycles without any turns. For the central face g we identify an edge e on the simple cycle C bounding g such that it is $\ell_C(e) = 0$. Since Γ is valid and C is an essential cycle, such an edge must exist. We then insert a new essential cycle C' of length 3 without turns inside g and connect one of its vertices to a new vertex on e. The new cycle C' now forms the boundary of the central face (cf. Figure 3.11).

Similarly, we surround the graph by a new cycle C'' of length 3 such that all edges of the cycle point right. We furthermore insert an edge from a vertex $v \in V(C'')$ to a new vertex w on the reference edge such that the new edge points downwards (cf. Figure 3.11). But in the resulting representation the reference edge does not lie on the outer face anymore. Hence, we need a new reference edge and we choose that edge of C'' which ends at v. This does not change the labelings of the essential cycles because any elementary path from the old reference edge rs can be transformed into an elementary path from the new reference edge by prepending vws.

Having rectangulated the outer and the central face, we proceed to the regular faces. By definition a face is a rectangle if and only if it does not have any left turns. Hence, each left turn must get a new incident edge. We add these edges one after another by first picking an arbitrary non-rectangular face f and then a suitable left turn at a vertex u on f. Finally, we add an edge from u to a suitable—possibly new—vertex z of f.

Before we describe how to choose these vertices, we define an augmented ortho-radial representation, which is obtained from Γ by adding one edge. For a vertex u at a left



(a) Γ^u_{vw} (b) Γ^u_{vw} (c) Γ^u_{wx}

Figure 3.12: Examples of augmentations. (a) The inserted edge uz points upwards and Γ^u_{vw} is valid. (b) The representation Γ^u_{vw} is not valid, since the insertion of the new edge introduces a decreasing cycle C. (c) Using the candidate wx instead gives the valid representation Γ^u_{wx}. The cycle C', which uses the same edges outside of f as C before, is neither in- nor decreasing.

turn of f and any edge e of f, the *augmentation* is an ortho-radial representation Γ^u_e of the augmented graph $G + uz$, where z is a new vertex on e: The augmentation Γ^u_e is obtained from Γ by inserting the edge uz, which points in the same direction as the edge of f entering u. Examples of such augmentations are shown in Figure 3.12. The new edge uz lies in the interior of f and splits f in two faces. As the rotation of these new faces must be 4, one cannot arbitrarily select e.

Observation 3.15. *The representation Γ^u_{vw} satisfies Conditions 1–3 of Definition 3.3 if and only if* $\mathrm{rot}(f[u, vw]) = 2$.

To find a suitable edge, we define *candidate edges* as the set $\mathcal{C}$ of all edges e on f, for which it is $\mathrm{rot}(f[u, e]) = 2$. According to Observation 3.15 the candidates are exactly those edges e for which Γ^u_e might be valid, and when choosing a candidate we only have to ensure that no increasing or decreasing cycles exist.

Using the definitions given above, we describe in the following how to insert one edge. Repeating this process eventually yields a representation, in which all faces are rectangles. Note that the description below is based on several properties of valid ortho-radial representations. For now, we assume them without proof. We justify these assumptions in Section 3.5.3, where we prove the correctness of the algorithm.

Let f be a regular face that is not a rectangle. As it is $\mathrm{rot}(f) = 4$ in valid ortho-radial representations, there are 4 more right turns on f than left turns. Hence, there is a left turn on f at a vertex u such that the two following turns on f are right turns. We then determine the candidate edges for u. Note that since f makes a left turn at u, there is exactly one edge e on f entering u. If this edge e points up or down, we pick the first candidate vw and build the augmentation Γ^u_{vw}. Figure 3.12(a) illustrates this situation for uz pointing up. Note that v is the vertex at which f makes the second right turn after u. Thus, the face of Γ^u_{vw} including v and w is a rectangle.

If e points left or right, one cannot simply pick the first candidate as this might introduce increasing or decreasing cycles. For example, Γ^u_{vw} contains a decreasing cycle in Figure 3.12(b). Therefore, the new vertex z must lie on another edge—in this example one can choose wx as shown in Figure 3.12(c). Therefore, we consider the candidates in the order in which they appear on f after u. The following description assumes that e points right. If e points left, the roles of in- and decreasing cycles must be exchanged.

For a candidate vw we first check whether Γ^u_{vw} contains a decreasing cycle. If this is the case, we move on to the next candidate. Otherwise, we additionally check for increasing cycles in Γ^u_{vw}. If there are no increasing cycles either, the augmentation Γ^u_{vw} is valid. In that

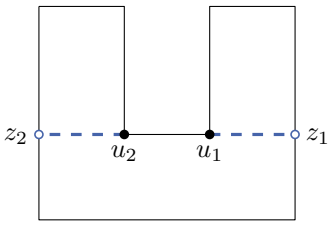


Figure 3.13: The face f is shaped such that the edge that shall be inserted from a left turn must point to the left or to the right. The left turn at u_1 is followed by two right turns and the left turn at u_2 is preceded by two right turns.

case we continue with subdividing the next face. However, if Γ^u_{vw} contains an increasing cycle, we go back to the candidate $v'w'$ before vw. One of v and w' has no incident edge to its left and we call this vertex z. If there is no edge to the left of both v and w', it is $v = w'$, so z is uniquely determined. We augment Γ by inserting the edge uz such that this edge points to the right.

Degree-1 vertices are treated as an edge e between two left turns. Note that e may be a candidate. If we insert an edge to a new vertex on e, e.g., when building the augmentation Γ^u_e, we use the degree-1 vertex as an endpoint instead.

In the rectangulation algorithm we distinguish between inserting a new edge that points up or down and inserting an edge that points left or right. In the first case we always choose the first candidate vw, whereas in the second case more checks are necessary. Therefore, one might always prefer the first case. However, faces can be shaped such that only the second case occurs, even if one additionally searches for left turns that are preceded (and not only followed) by two right turns. For example, the U-shaped face in Figure 3.13 only admits the insertion of new edges that point left or right.

3.5.2 Preparation

In this section we study some properties of ortho-radial representations, which are used in the correctness proof in Section 3.5.3. The rectangulation algorithm heavily uses augmentations of Γ. Such an augmentation Γ^u_{vw} is obtained from Γ by adding a new vertex z on vw and the edge uz subdividing a face f. In order to study the properties of Γ^u_{vw}, we often restrict ourselves to subgraphs of a certain structure: Namely, we work with the subgraph $H = C + f'$ that consists of an essential cycle C and a regular face f' of Γ^u_{vw}, where f' is one of the two new faces that were created by subdividing f.

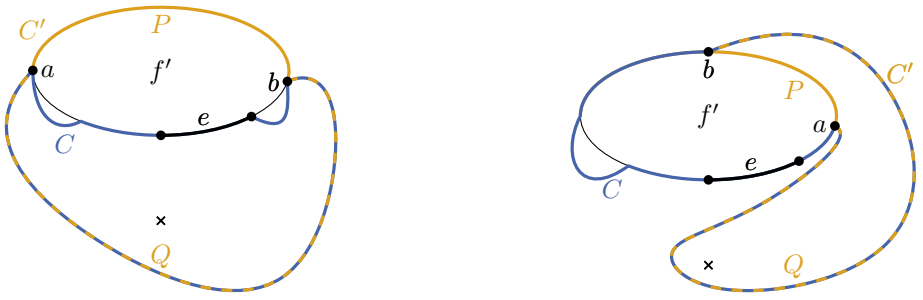


Figure 3.14: The situation in Lemma 3.16. (a) The edge e does not lie on the outer face. (b) The edge e does not lie no the central face.

Lemma 3.16. *If an edge e belongs to both a simple, essential cycle C and a regular face f', there is a simple essential cycle C' not containing e such that C' can be decomposed into two paths P and Q, where P or $\overline{P}$ lies on f' and $Q = C \cap C'$.*

Proof. Consider the graph $H = C + f'$ composed of the essential cycle C and the regular face f'. In H the edge e cannot lie on both the outer and the central face. If e does not lie on the outer face, we define C' as the cycle bounding the outer face but directed such that it contains the center in its interior (see Figure 3.14(a)). Otherwise, C' denotes the cycle bounding the central face, which is illustrated in Figure 3.14(b).

Since C lies in the exterior of f', the intersection of C with C' forms one contiguous path Q. Setting $P = C - Q$ yields a path that lies completely on f' (it is possible though that P and f' are directed differently). In Figure 3.14 the paths P and Q are separated by the vertices a and b. □

Using this lemma, we can construct an essential cycle C' without the new edge uz from an essential cycle C including uz. Moreover, C and C' have a common path P, which lies on the central face of H. Hence, the Intersection Lemma implies that the labelings of C and C' are equal on P.

Corollary 3.17. *For essential cycles C, C' and the path $P = C \cap C'$ from the previous lemma, it is $\ell_C(e) = \ell_{C'}(e)$ for all edges e on P.*

Another useful observation deals with two intersecting essential cycles, where the labeling of one cycle is identically 0.

Lemma 3.18. *Let C and C' be two essential cycles that have at least one common vertex. If all edges on C' are labeled with 0, C is neither increasing nor decreasing.*

Proof. The situation is illustrated in Figure 3.15. If the two cycles are equal, the claim clearly holds. Otherwise, we show that one can find two edges on C such that the labels of these edges have opposite signs.

Let vw be an edge of C but not of C' such that v lies on the central face $\tilde{f}$ of $H = C + C'$, and denote the vertex before v on C' by u. By the Intersection Lemma it is

$$\ell_C(vw) = \ell_{C'}(uv) + \mathrm{rot}(uvw) = \mathrm{rot}(uvw). \tag{3.25}$$

The second equality follows from the assumption $\ell_{C'}(uv) = 0$. Let y be the first common vertex of C and C' on the central face $\tilde{f}$ after v. That is, $\tilde{f}[v, y]$ is a part of one of the

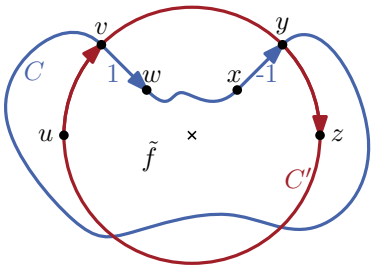


Figure 3.15: The situation of Lemma 3.18. All edges of C' are labeled with 0. In this situation there are edges on C with labels -1 and 1.

cycles C and C' and intersects the other cycle only at v and y. We denote the vertex on C before y by x and the vertex after y on C' by z. Again by the Intersection Lemma we have

$$\ell_C(xy) = \ell_{C'}(yz) - \operatorname{rot}(xyz) = -\operatorname{rot}(xyz). \tag{3.26}$$

By construction vw and xy lie on the same side of C'. Hence, uvw and xyz both make a right turn if vw and xy lie in the interior of C' and a left turn otherwise. Thus, it is $\operatorname{rot}(uvw) = \operatorname{rot}(xyz) \neq 0$, and Equations 3.25 and 3.26 imply that $\ell_C(uv)$ and $\ell_C(xy)$ have opposite signs. Hence, C is neither in- nor decreasing. □

In the following lemma we give a sufficient condition when a candidate exists after a given edge.

Lemma 3.19. *Let f be a regular face of G and u be any vertex on f. If e is an edge on f such that $\operatorname{rot}(f[u,e]) \leq 2$, there is a candidate on $f[e,u]$, i.e., an edge e' such that $\operatorname{rot}(f[u,e']) = 2$.*

Proof. For each edge e' on f we determine the value of $r(e') = \operatorname{rot}(f[u,e'])$. By assumption it is $r(e) \leq 2$. Moreover, for the last edge e'' on $f[e,u]$ it is $r(e'') = \operatorname{rot}(f) - \operatorname{rot}_f(u) \geq 3$. Here, we use that f is a regular face (i.e., $\operatorname{rot}(f) = 4$) and the rotation $\operatorname{rot}_f(u)$ at the vertex u in f is at most 1.

When going from an edge e_i to the next edge e_{i+1}, the value assigned to these edges increases by at most 1, i.e., $r(e_{i+1}) \leq r(e_i) + 1$. Therefore, there exists an edge e' between e and e'', i.e., on $f[e,u]$, such that $r(e') = 2$. □

3.5.3 Correctness of the Rectangulation

In this section we prove that the rectangulation procedure described in Section 3.5.1 produces a valid ortho-radial representation whose faces are all rectangles. The structure of this section follows the structure of the algorithm.

The algorithm fixes a regular face f and a vertex u on f such that f makes a left turn at u and the two following turns are right ones. Then, a set of candidate edges is calculated and two cases are distinguished: Either the edge of f entering u points up or down, or it points left or right. In the first case, we always choose the first candidate edge vw and construct Γ^u_{vw}, which is valid by the following lemma:

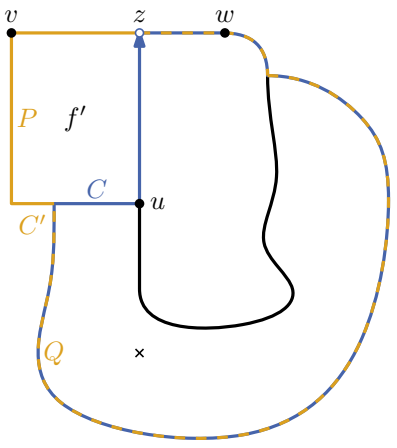


Figure 3.16: The situation in Lemma 3.20. The edge uz is inserted such that it points upwards and z lies on the first candidate vw.

Lemma 3.20. *Let vw be the first candidate edge after u. If the edge on f entering u points up or down, Γ_{vw}^{u} is a valid ortho-radial representation of the augmented graph $G + uz$.*

Proof. Assume that Γ_{vw}^{u} contains a simple increasing or decreasing cycle C. As Γ is valid, C must contain the new edge uz in either direction (i.e., uz or zu). Let f' be the new rectangular face of $G + uz$ containing u, v and z, and consider the subgraph $H = C + f'$ of $G + uz$. According to Lemma 3.16 there exists a simple essential cycle C' that does not contain uz. Moreover, C' can be decomposed into paths P and Q such that P lies on f' and Q is a part of C (cf. Figure 3.16). The goal is to show that C' is increasing or decreasing. We present a proof only for the case that C is an increasing cycle. The proof for decreasing cycles can be obtained by flipping all inequalities.

For each edge e on Q the labels $\ell_C(e)$ and $\ell_{C'}(e)$ are equal by the Intersection Lemma, and hence it is $\ell_{C'}(e) \leq 0$. For an edge $e \in P$, there are two possible cases: e either lies on the side of f' parallel to uz or on one of the two other sides. In the first case, the label of e is equal to the label $\ell_C(uz)$ (or to $\ell_C(zu)$ if C contains zu instead of uz). In particular, the label is negative.

In the second case we first note that $\ell_{C'}(e)$ is even, since e points left or right. Assume for the sake of contradiction that $\ell_{C'}(e)$ was positive and therefore at least 2. Then, let e' be the first edge on C' after e that points to a different direction. Such an edge exists, since otherwise C' would be an essential cycle whose edges all point to the right but they are not labeled with 0. This edge e' lies on Q or is parallel to uz. Hence, the argument above implies that $\ell_{C'}(e') \leq 0$. However, $\ell_{C'}(e')$ differs from $\ell_{C'}(e)$ by at most 1, which requires $\ell_{C'}(e') \geq 1$. Therefore, $\ell_{C'}(e)$ cannot be positive.

We conclude that all edges of C' have a non-positive label. If all labels were 0, C would not be an increasing cycle by Lemma 3.18. Thus, there exists an edge on C' with a negative label and C' is an increasing cycle in Γ. But as Γ is valid, such a cycle does not exist, and therefore C does not exist either. Hence, Γ_{vw}^{u} is valid. □

If the edge entering u points left or right, multiple candidates may be checked. In the following we assume that the edge entering u points right. Starting with the first candidate after u we try the candidates in the order in which they appear on f. When considering a candidate vw, we first check whether Γ_{vw}^{u} contains a decreasing cycle. If a decreasing cycle exists, we move on to the next candidate and repeat this process. This process comes to an end because there is no decreasing cycle in the augmented graph for the last candidate edge.

Lemma 3.21. *Let vw be the last candidate edge before u. Then, Γ_{vw}^{u} contains no decreasing cycle.*

Proof. The face f is split in two parts when uz is inserted. Let f' be the face containing v and f'' the one containing w. Assume that there is a simple decreasing cycle C in Γ_{vw}^{u}. Then, either uz or zu lies on C. We use a similar strategy as in the proof of Lemma 3.20 to find a decreasing cycle in Γ, which contradicts that Γ is valid.

Consider the graph $H = f'' + C$ composed of the decreasing cycle C and f''. Lemma 3.16 shows that there exists an essential cycle C' in H that can be decomposed into a path P on f'' and $Q = C \cap C'$ (see Figure 3.17). We show in the following that C' is a decreasing cycle. But all edges of C' are already present in Γ, contradicting the assumption that Γ is valid. For all edges $e \in E(Q)$ we have $\ell_C(e) = \ell_{C'}(e) \geq 0$ by Corollary 3.17.

To show that edges on P also have non-negative labels, we assume this was not the case, i.e., there is an edge $xy \in P$ such that $\ell_{C'}(xy) < 0$. We present a detailed argument for the

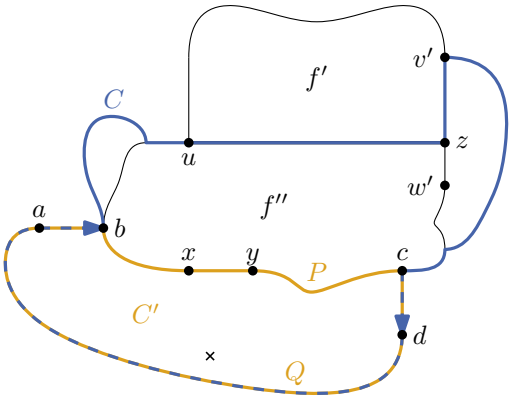


Figure 3.17: The situation in the proof of Lemma 3.21. The cycle C is decreasing and it is assumed that $\ell_{C'}(xy) < 0$.

case that C uses uz (and not zu). Then, P is directed such that f'' lies to the left of P. A similar argument applies to the case that C uses zu.

Our goal is to show that there must be a candidate on f after y and in particular after the last candidate vw. We first assume $\operatorname{rot}(f[u, yx]) < 3$ to derive the contradiction and prove this bound afterwards. Then, there is a candidate on $f[yx, u]$ by Lemma 3.19. In particular, this candidate comes after vw, contradicting the assumption that vw is the last candidate.

Moreover, not all labels of edges on C' can be 0, since C would not be decreasing (cf. Lemma 3.18). Thus, C' is a decreasing cycle that consists solely of edges of G. But as Γ is valid such a cycle cannot exist, showing that Γ^u_{vw} contains no decreasing cycle either.

It remains to show the upper bound $\operatorname{rot}(f[u, yx]) < 3$. Let ab and cd be the last and the first edge of Q, respectively. In order to simplify the descriptions of the paths we use below, we assume without loss of generality that ab and cd have no common endpoints. This property does not hold only if Q has at most two edges, in which case we may subdivide an edge of Q to lengthen Q.

Applying Lemma 3.8 to $C[a, d]$ and $C'[a, d]$, we get

$$\operatorname{rot}(C[a, d]) = \operatorname{rot}(C'[a, d]). \tag{3.27}$$

Splitting the paths at uz and xy, respectively, the total rotation does not change (cf. Observation 2.1).

$$\operatorname{rot}(C[a, d]) = \operatorname{rot}(C[a, uz]) + \operatorname{rot}(C[uz, d]) \tag{3.28}$$
$$\operatorname{rot}(C'[a, d]) = \operatorname{rot}(C'[a, xy]) + \operatorname{rot}(C'[xy, d]) \tag{3.29}$$

Combining the previous equations gives

$$\operatorname{rot}(C[a, uz]) + \operatorname{rot}(C[uz, d]) - \operatorname{rot}(C'[a, xy]) - \operatorname{rot}(C'[xy, d]) = 0. \tag{3.30}$$

As $C'[xy, d] = \overline{f''}[xy, c] + cd$, we get

$$\operatorname{rot}(C'[xy, d]) = \operatorname{rot}(\overline{f''}[xy, c] + cd) = -\operatorname{rot}\left(dc + f''[c, yx]\right). \tag{3.31}$$

The last equality follows from the fact that the rotation of the reverse of a path is the negative rotation of the path (cf. Observation 2.2). Applying Lemma 3.9 to $C[uz, d]$ and $f''[uz, c] + cd$ with C as outer cycle and C' as inner cycle, we get

$$\operatorname{rot}(C[uz, d]) = \operatorname{rot}(f''[uz, c] + cd). \tag{3.32}$$

Substituting Equations 3.31 and 3.32 into Equation 3.30 yields

$$\operatorname{rot}(C[a,uz]) + \operatorname{rot}(f''[uz,c] + cd) - \operatorname{rot}(C'[a,xy]) + \operatorname{rot}(dc + f''[c,yx]) = 0. \tag{3.33}$$

Note that if one joins $f''[uz,c] + cd$ and $dc + f''[c,yx]$ together, the resulting path is almost $f''[uz,yx]$ except for the detour via d. Observation 2.3 therefore implies

$$\operatorname{rot}\left(f''[uz,c] + cd\right) + \operatorname{rot}\left(dc + f''[c,yx]\right) = \operatorname{rot}(f''[u,yx]) - 2. \tag{3.34}$$

Substituting this equality into Equation 3.33 results after rearranging in

$$\operatorname{rot}(f''[u,yx]) = 2 + \operatorname{rot}(C'[a,xy]) - \operatorname{rot}(C[a,uz]). \tag{3.35}$$

By Observation 3.6 the rotations on an essential cycle can be expressed as the difference of labels. Here, we have $\operatorname{rot}(C'[a,xy]) = \ell_{C'}(xy) - \ell_{C'}(ab)$ and $\operatorname{rot}(C[a,uz]) = \ell_C(uz) - \ell_C(ab)$. Additionally ab lies on Q, and therefore it is $\ell_C(ab) = \ell_{C'}(ab)$. Hence, it is

$$\begin{aligned}\operatorname{rot}(f''[u,yx]) &= 2 + \ell_{C'}(xy) - \ell_{C'}(ab) - \ell_C(uz) + \ell_C(ab)\\ &= 2 + \ell_{C'}(xy) - \ell_C(uz)\\ &< 2.\end{aligned} \tag{3.36}$$

Here, the last inequality follows from the assumptions $\ell_{C'}(xy) < 0$ and $\ell_C(uz) \geq 0$. By the construction of Γ^u_{vw} the rotations of $f[u,yx]$ and $f''[u,yx]$ differ by exactly 1:

$$\operatorname{rot} f[u,yx] = \operatorname{rot} f''[u,yx] + 1 < 3 \tag{3.37}$$

Above, we assumed that C uses uz in that direction. If zu lies on C, a similar argument shows that C' would also be a decreasing cycle, contradicting that assumption that Γ is valid. □

If a candidate vw was found such that there is no decreasing cycle in Γ^u_{vw}, we additionally check for increasing cycles. If also no increasing cycle is found, we permanently insert the edge and continue the augmentation process with Γ^u_{vw}. As Γ^u_{vw} has neither increasing nor decreasing cycles, it satisfies Condition 4 of Definition 3.3. Together with Observation 3.15 this shows that Γ^u_{vw} is valid.

If Γ^u_{vw} contains an increasing cycle, we prove in the following that there exists a vertex z on f between the previous and the current candidate such that the insertion of uz completes a cycle whose edges only point to the right. As a first step towards this result, we show that the previous candidate exists, i.e., the current candidate is not the first one.

Lemma 3.22. *Let u be a vertex on f at which f has a left turn, and let vw be the first candidate edge. If vw points down, Γ^u_{vw} does not contain any increasing cycles.*

Proof. Let f' be the new rectangular face of Γ^u_{vw} containing u, v and z, and assume that there is an increasing cycle C in Γ^u_{vw}. This cycle must use either uz or zu. Similar to the proof of Lemma 3.20, we find an increasing cycle C' in Γ, contradicting the validity of Γ.

To this end, we construct the graph H as the union of C and f'. By Lemma 3.16, there exists a simple essential cycle C' without uz and zu that can be decomposed into a path P on f' and a path $Q \subseteq C \setminus f'$. All edges of Q have non-positive labels by the Intersection Lemma. We show in the following that the edges of P also have non-positive labels.

If C contains uz, there are three possibilities for an edge e of P, which are illustrated in Figure 3.18: The edge e lies on the left side of f' and points up, e is parallel to uz,

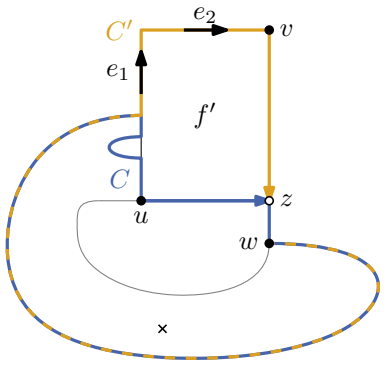


Figure 3.18: Situation of Lemma 3.22: The increasing cycle C contains uz. There are three possibilities for edges on C' that lie not on C: They lie on the left side of f' (like e_1), on the top (like e_2), or on the right side, which is formed by vz.

or $e = vz$. In the first case it is $\ell_{C'}(e) = \ell_C(uz) - 1 < 0$ and in the second case it is $\ell_{C'}(e) = \ell_C(uz) \leq 0$. If $e = vz$, C cannot contain zv and therefore we have $zw \in E(C)$ and $\ell_{C'}(e) = \ell_C(zw) < 0$. In all three cases the label of e is at most 0.

If C contains zu, the label of zu has to leave a remainder of 2 when it is divided by 4, since zu points to the right. As the label is also at most 0, we conclude $\ell_C(zu) \leq -2$. The edges of P lie either on the left, top or right of f'. Therefore, the label of any edge e on P differs by at most 1 from $\ell_C(zu)$, and thus we get $\ell_{C'}(e) \leq 0$.

Summarizing the results above, we see that all edges on C' are labeled with non-positive numbers. The case that all labels of C' are equal to 0 can be excluded, since C would not be an increasing cycles by Lemma 3.18. Hence, C' is an increasing cycle, which was already present in Γ, contradicting the validity of Γ. Thus, no such increasing cycle C in Γ^u_{vw} exists. □

Having ensured that the previous candidate exists, we show in the following that we always find a vertex z on f between the previous and the current candidate and a path P from z to u whose edges point right. In other words, the insertion of uz completes the cycle $P + uz$, whose edges all point to the right.

Lemma 3.23. *Let vw and $v'w'$ be two candidate edges such that no candidate is between them, i.e., the path Q on f between these candidates does not contain any candidate edge. Furthermore, assume that Γ^u_{vw} contains a decreasing cycle and $\Gamma^u_{v'w'}$ an increasing cycle. Then, there is a path P in G containing w, v' and u such that the edges point to the right. More precisely, P starts at w or v' and ends at u, and either Q or $\overline{Q}$ forms the first part of P. Moreover, the start vertex of P has no incident edge to its left.*

Proof. Let z be the new vertex inserted in Γ^u_{vw} and z' the one in $\Gamma^u_{v'w'}$. Since both uz and uz' point to the right, there is no augmentation of Γ containing both edges. We need to compare Γ^u_{vw} and $\Gamma^u_{v'w'}$ though. Therefore, we use the following construction, which models all important aspects of both representations: Starting from Γ we insert new vertices z on vw and z' on $v'w'$. We connect u and z by a path of length 2 that points to the right and denote its internal vertex by x. Furthermore, a path of length 2 from x via a new vertex y to z' is added. The edge xy points down and yz' to the right. This construction is depicted in Figure 3.19. In the resulting ortho-radial representation $\tilde{\Gamma}$ the edge uz is modeled by the path uxz and uz' by $uxyz'$.

Take any simple decreasing cycle in Γ^u_{vw}. As Γ is valid, this cycle must contain either uz or zu. We obtain a cycle C in $\tilde{\Gamma}$ by replacing uz with uxz (or zu with zxu). Note that ux

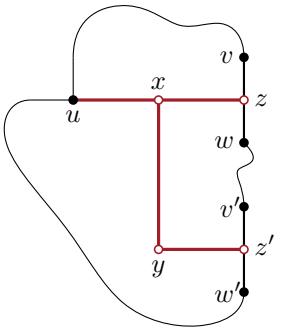


Figure 3.19: The structure that is used to simulate the insertion of both uz and uz' at the same time. The edge uz is replaced by the path uxz and uz' by $uxyz'$.

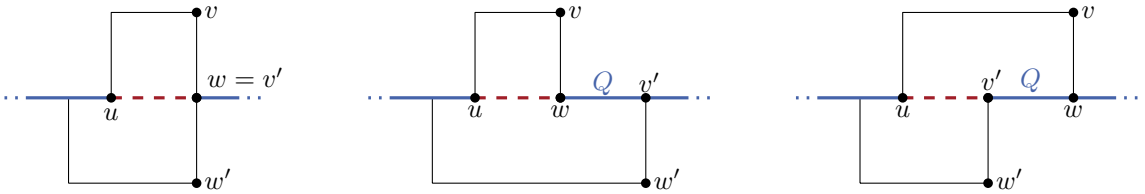


Figure 3.20: There are three possibilities how the path between w and v' can look like: (a) $w = v'$, (b) all edges point right, and (c) all edges point left. In the first two cases the edge uw is inserted and in (c) uv' is added.

and xz have the same label as uz, and the labels of all other edges on the cycles stay the same. Therefore, C is a decreasing cycle.

Similarly, there exists a simple increasing cycle in $\Gamma^u_{v'w'}$, which contains uz' or $z'u$. Replacing uz' with $uxyz'$ (or $z'u$ with $z'yxu$) we get a cycle C' in $\tilde{\Gamma}$. Note that C' might not be an increasing cycle as $\ell_C(xy)$ might be positive. But the labels of all other edges are at most 0 and there exists an edge with a negative label. In other words, outside of f the cycle C' behaves exactly like an increasing cycle.

For now, we assume that the original cycles use uz and uz' in these directions. At the end of the proof we shall see that this is in fact the only possibility. The proof is structured as follows: First, we show $\ell_C(ux) = \ell_{C'}(ux) = 0$. This also determines the labels of C and C' in the interior of f. In a second step, we find that at least one of the vertices w and v' has no incident edge to the left and the other vertex lies on both C and C'. Note that $w = v'$ is possible. In this case, $w = v'$ has all the mentioned properties. Moreover, we prove that the path Q on f between w and v' (of length 0 if $w = v'$) is straight and can be used as the first part of the desired path P. From this information we infer that there are three possibilities as shown in Figure 3.20: Either it is $w = v'$, all edges on f between w and v' point right, or they all point left. Finally, we show that outside of f, the cycles C and C' are equal. In particular, all their edges point to the right and we can use them as the second part of the desired path P.

To show $\ell_C(ux) = \ell_{C'}(ux)$, we consider the graph $H = C + C'$ formed by the two cycles C and C' and denote its central face by $\tilde{f}$. By the Intersection Lemma it suffices to show that ux lies on $\tilde{f}$. Assume for the sake of contradiction that this was not the case. Then, xy, xz and yz' do not lie on $\tilde{f}$ either. Hence, $\tilde{f}$ is formed completely by edges in $E(C[z,u]) \cup E(C'[z',u])$. As C and C' were constructed by subdividing edges of simple cycles, they are simple themselves. Therefore, the edges of $\tilde{f}$ do not all belong to the same

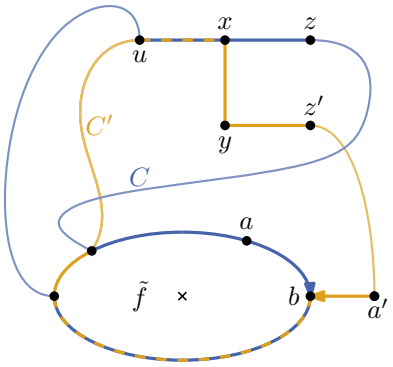


Figure 3.21: The hypothetical situation that ux does not lie on the central face $\tilde{f}$. The edge ab lies on both $\tilde{f}$ and C but not on C' and $a'b$ is the edge on C' that ends at b.

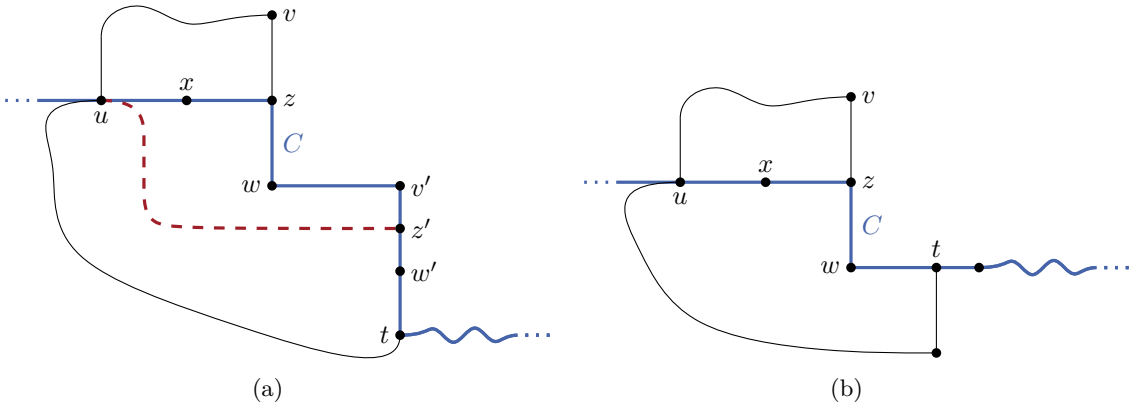


Figure 3.22: The face f makes a left turn at w. (a) The path $C[w,t]$ makes a right turn at v'. But then $C[z',u]+uz'$ would be a decreasing cycle in $\Gamma^u_{v'w'}$. (b) The path $C[w,t]$ is straight and it is $t=v'$.

cycle. Hence, there is an edge ab on $\tilde{f}$ such that b lies on both C and C' but ab only belongs to C and not to C' (cf. Figure 3.21). Since C is a decreasing cycle, it is $\ell_C(ab) \geq 0$.

Moreover, let a' be the vertex before b on C'. Since C' is almost an increasing cycle, we get $\ell_{C'}(a'b) \leq 0$ unless it was $a'b = xy$. But this is impossible since y does not lie on $\tilde{f}$. Lemma 3.12 therefore implies that $a'b$ lies in the interior of C. But then ab would not lie on $\tilde{f}$, contradicting the choice of ab. Thus, ux is part of $\tilde{f}$.

Hence, the Intersection Lemma applies to ux and we obtain

$$0 \leq \ell_C(ux) = \ell_{C'}(ux) \leq 0.$$

Thus, we have $\ell_C(ux) = \ell_{C'}(ux) = 0$ and therefore also $\ell_C(xz) = \ell_{C'}(yz') = 0$. Hence, C contains zw, because otherwise zv would be labeled with -1. Similarly, we see that $z'v'$ lies on C'.

As a next step we prove that one of v' and w has no incident edge to its left and the other vertex lies on both C and C'. This is the case if $v' = w$, since any subgraph to the left of this vertex would contain a candidate. Remember that we treat degree-1 vertices as two left turns with an edge in between and this edge can be a candidate. Therefore, even in the extreme case, where the subgraph to the left of $v' = w$ is just one path that points left, we find a candidate—namely the leftmost endpoint of the path.

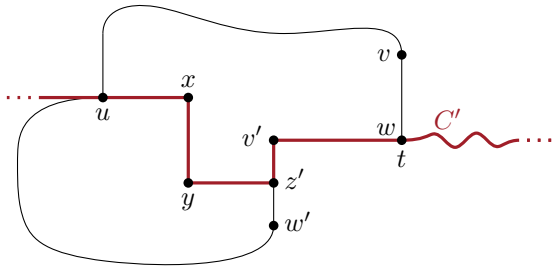


Figure 3.23: If f makes a right turn at w, it makes a left turn at v'. The longest common path $C'[v', t]$ on C' and $\overline{f}$ contains w and all its edges point to the right.

If $w \neq v'$, f makes either a left or a right turn at w. If f makes a left turn at w, C makes a left turn there as well. Let t be the vertex at which C leaves f after w. In other words, t is the last vertex of C such that $C[w, t]$ lies on f. This situation is illustrated in Figure 3.22.

We show that $C[w, t]$ contains v' and that all edges of $C[w, v']$ point to the right. If $C[w, t]$ makes turns, the first turn cannot be a left turn, since the edge following the turn would lie on C and be labeled with -1. If the first turn is a right turn, the edge following the turn is the candidate $v'w'$. But then $\Gamma^u_{v'w'}$ would contain the decreasing cycle $C[z', u] + uz'$, which contradicts the choice of $v'w'$ (cf. Figure 3.22(a)). Hence, it remains the case in which $C[w, t]$ is straight, which is illustrated in Figure 3.22(b). The edge following t on C must point right. Therefore, f makes a right turn at t implying $t = v'$. In all possible cases, v' lies on both C and C' and the path $Q = C[w, v']$ is straight and points to the right. Note that there is no edge incident to the start vertex w of Q, since f makes a left turn at w.

If f makes a right turn at w, we consider the edge $wa \in E(f)$ following this turn.

$$\operatorname{rot}(f[u, wa]) = \operatorname{rot}(f[u, vw]) + \operatorname{rot}(vwa) = 2 + 1 = 3$$

Since $v'w'$ is a candidate, it is $\operatorname{rot}(f[u, v'w']) = 2$. When walking along f the rotation changes by at most 1 per step, when degree-1 vertices are treated as two steps. Hence, it is $\operatorname{rot}(f[u, e]) \geq 3$ for all edges e between vw and $v'w'$ and f makes a left turn at v'. Otherwise, there would be another candidate in between.

As C' enters v' on the edge $z'v'$ from below, C' must leave v' to the right. Thus, there is a part of C' starting at v' that lies on $\overline{f}$. Let t be the last vertex on C' such that $C[v', t] = \overline{f}[v', t]$, which is illustrated in Figure 3.23. Similar to the case above, we analyze the first turn of $C'[v', t]$ if it exists, and show that w lies on this path. Moreover, we shall see that all edges of this path point to the right. Because C' is an increasing cycle and the first edge of $C'[v', t]$ is labeled with 0, the first turn cannot be a right turn. If it is a left turn, the left turn must occur at w. Note that in this case Γ^u_{vw} contains an increasing cycle. We shall see that this situation actually cannot occur, but we cannot exclude this possibility yet. If $C'[v', t]$ makes no turns, the edge on C' after t also points to the right and the edge on f incident to t is the candidate vw. Hence, the path $Q = C'[v', w]$ points completely to the right. Furthermore, there is no edge incident to the left of v' because f makes a left turn at v'.

In all cases we found a (possibly empty) path Q from v' to w or vice versa, whose edges point to the right. Moreover, the endpoint t of this path lies on both C and C'. The path Q is the initial part of the desired path P. To construct the remaining part of P, we prove in the following that it is $C[t, u] = C'[t, u]$. Moreover, we show that all edges on this path point to the right.

Assume for the sake of contradiction that it was $C[t, u] \neq C'[t, u]$. Let ab be the last edge of $C[t, u]$ that does not lie on C' and let $a'b$ be the edge of C' entering b. By Lemma 3.12

the edge $a'b$ lies in the interior of C. But then consider the last common vertex c of C and C' before b. We denote the edges of C and C' starting at c by cd and cd', respectively. As $a'b$ lies in the interior of C, so does cd'. But according to Lemma 3.12 the edge cd' lies in the exterior of C, a contradiction. Thus, all edges of $C[t, u]$ lie on $C'[t, u]$ as well. Since these are both paths from t to u, this means that they are equal. To shorten the notation, we refer to $C[t, u]$ as R.

Since $\ell_C(ux) = \ell_{C'}(ux)$, it is $\ell_C(e) = \ell_{C'}(e)$ for all edges e on R. Moreover, as C is a decreasing and C' an (almost) increasing cycle, these labels are 0. In particular, all edges on R point to the right.

Thus, $P = Q + R$ is a path containing both v' and w and ends at u, such that all edges of P point to the right, which concludes the proof for the case when both C and C' use the edge ux in this direction.

It remains to show that neither C nor C' can contain xu. By an argument similar to the one above, we know that xu lies on the boundary of the central face $\tilde{f}$ of $H = C + C'$. Since C and C' contain $\tilde{f}$ in their interiors, any edge e on one of them incident to $\tilde{f}$ is directed such that $\tilde{f}$ lies locally to the right of e. If C and C' used ux in different directions, this implies that $\tilde{f}$ lies locally to the right of both ux and xu, which is impossible.

Hence, if xu lies on one cycle, it lies on the other one, too. In this case, it is $\ell_C(xu) = \ell_{C'}(xu) = 0$. But xu points to the left and therefore its label must leave a remainder of 2 when divided by 4. Thus, the assumption that both C and C' contain ux is justified as none of the other cases can occur. □

There are three possible ways how the vertices w and v' can be arranged on P (cf. Figure 3.20): Either $w = v'$, w comes before v', or v' comes before w. In any case we denote the start vertex of P by z. According to the lemma above, no edge is incident to the left of z. Hence, the insertion of the edge uz such that it points right gives a new ortho-radial representation Γ', which is valid by the following lemma.

Lemma 3.24. *Let Γ' be the ortho-radial representation that is obtained from Γ by adding the edge uz pointing to the right. Then, Γ' is valid.*

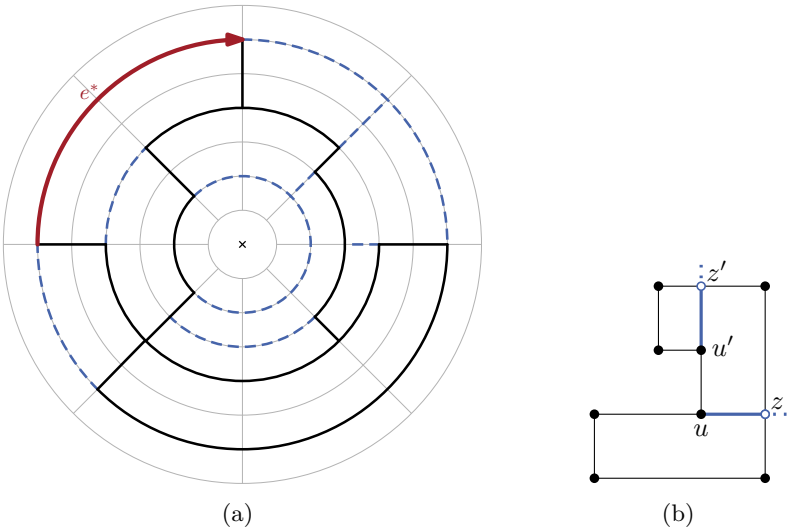
Proof. By construction, Γ' satisfies Conditions 1–3 of Definition 3.3.

Let $C' = P + uz$ be the new cycle whose edges point right. It is $\ell_{C'}(e) = 0$ for each edge e of C'. No essential cycles without uz or zu is increasing or decreasing, since they are already present in the valid representation Γ. If an essential cycle C contains uz or zu and in particular the vertex u, Lemma 3.18 states that C is neither increasing nor decreasing. Thus, Γ' also satisfies Condition 4 and is therefore valid. □

Putting all results of this section together we see that the rectangulation algorithm presented in Section 3.5.1 works correctly. That is, given a valid ortho-radial representation Γ, the algorithm produces another valid ortho-radial representation Γ' such that all faces of Γ' are rectangles and Γ is contained in Γ'. Combining this result with the characterization for rectangular graphs (Corollary 3.14) we get one implication of Theorem 3.5:

Theorem 3.25. *Let Γ be a valid ortho-radial representation of a graph G. Then, there is a drawing of G respecting Γ.*

This theorem shows one implication of the characterization of ortho-radial drawings by valid ortho-radial representations presented in Theorem 3.5. The other implication is proved by the following theorem:



(a) (b)

Figure 3.24: (a) The result of the geometric rectangulation of an ortho-radial graph drawing. The edges of the original graph are drawn in solid black and the edges inserted during the rectangulation in dashed blue. The reference edge e^* is chosen as an edge on the outermost circle. (b) The rectangulation of a face. The edges uz and $u'z'$ are added to divide the face into rectangles.

Theorem 3.26. *For any drawing Δ of a 4-planar graph G there is a valid ortho-radial representation of G.*

Proof. We first note that any drawing Δ fixes an ortho-radial representation up to the choice of the reference edge. Let Γ be such an ortho-radial representation where we pick an edge e^* on the outer face as the reference edge such that e^* points to the right and lies on the outermost circle that is used by Δ (as in Figure 3.24(a)). By [HHT09] the representation Γ satisfies Conditions 1–3 of Definition 3.3. To prove that Γ also satisfies Condition 4, i.e., Γ does not contain any increasing or decreasing cycles, we show how to reduce the general case to the more restricted one, where all faces are rectangles. By Corollary 3.14 the existence of a drawing and the validity of the ortho-radial representation are equivalent for rectangular graphs.

Given the drawing Δ we augment it such that all faces are rectangles. This rectangulation is similar to the one described in Section 3.5.1 but works with a drawing and not with a representation. We first insert the missing parts of the innermost and outermost circle that are used by Δ such that the outer and the central face are already rectangles. For each left turn on a face f at a vertex u, we then cast a ray from v in f in the direction in which the incoming edge of u points (cf. Figure 3.24(b)). This ray intersects another edge in Δ. Say the first intersection occurs at the point p. Either there already is a vertex z drawn at p or p lies on an edge. In the latter case, we insert a new vertex, which we call z, at p. We then insert the edge uz in G and update Δ and Γ accordingly.

Repeating this step for all left turns we obtain a drawing Δ' and an ortho-radial representation Γ' of the augmented graph G' (see Figure 3.24(a) for an example of Δ'). As the labelings of essential cycles are unchanged by the addition of edges elsewhere in the graph, any increasing or decreasing cycle in Γ would also appear in Γ'. But by Corollary 3.14 the representation Γ' is valid, and hence neither Γ nor Γ' contain increasing or decreasing cycles. Thus, Γ satisfies Condition 4 and is valid. □

4. Complexity of Ortho-Radial Embeddability

Garg and Tamassia [GT95] showed that it is $\mathcal{NP}$-complete to decide whether a 4-planar graph admits an orthogonal drawing without any edge bends (Orthogonal Embeddability). In this chapter, we study the analogous problem for ortho-radial drawings and prove that it is $\mathcal{NP}$-complete as well. We say a graph G admits an *ortho-radial (or orthogonal) embedding* if there is an embedding of G such that G can be drawn ortho-radially (or orthogonally) without bends.

Definition 4.1 (Ortho-Radial Embeddability)**.** *Does a 4-planar graph G admit an ortho-radial embedding?*

To show that Ortho-Radial Embeddability is $\mathcal{NP}$-hard, we reduce Planar Monotone 3-SAT, which was shown to be $\mathcal{NP}$-hard by de Berg and Khosravi [dBK12], to Ortho-Radial Embeddability.

Definition 4.2 (Planar Monotone 3-SAT)**.** *Given a Boolean formula Φ in conjunctive normal form such that each clause contains exactly three literals, which are either all positive or all negative and a planar representation of its variable-clause-graph in which the variables are placed on one line, the clauses with only positive literals above that line and the clauses with only negative literals below this line, is Φ satisfiable?*

To reduce from Planar Monotone 3-SAT to Ortho-Radial Embeddability we first construct an equivalent instance G of Orthogonal Embeddability as described by Bläsius et al. [BBR14]. We then build a structure around G yielding a graph G' such that in any ortho-radial representation of G' the representation Γ of G does not contain any essential cycles. In other words, Γ is actually an orthogonal representation of G. Hence, an ortho-radial embedding of G' can only exist if G admits an orthogonal embedding. We may assume without loss of generality that G is connected, as otherwise, we handle each component separately.

The construction of G' from G is based on the fact that there is only one way to ortho-radially draw a triangle C, i.e., a cycle of length 3, without bends: as an essential cycle on one circle of the grid. We build a graph H consisting of three triangles called C_1, C_2 and C_3 and denote the vertices on C_i by u_i, v_i and w_i. Furthermore, H contains the edges u_1u_2 and u_2u_3. In Figure 4.1 the black edges belong to H.

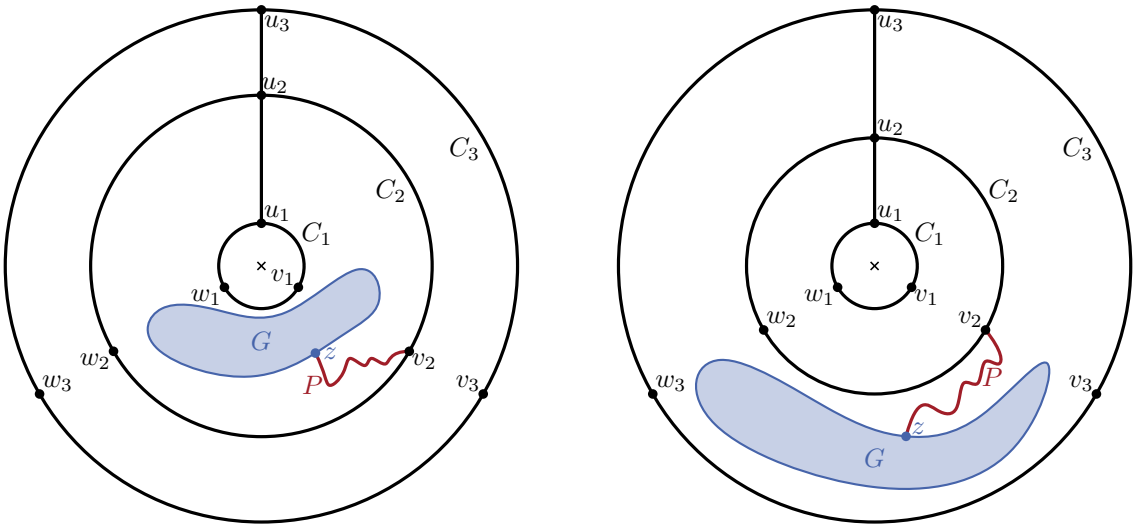


Figure 4.1: Possible embeddings of G': (a) G lies between C_1 and C_2. (b) G lies between C_2 and C_3. In both cases G contains no essential cycles. The roles of C_1 and C_3 can be exchanged.

To connect H and G, we identify a vertex z in G, called the *port* of G such that either there is an orthogonal embedding of G with z on the outer face and the angle at z in the outer face is at least 180° or G does not admit an orthogonal embedding. We shall see later how such a vertex can be identified in polynomial time. We connect z and v_2 by a path P whose length is equal to the number of edges in G and denote the resulting graph by G' (cf. Figure 4.1).

Lemma 4.3. *The graph G' admits an ortho-radial embedding if and only if G admits an orthogonal embedding with the port z on the outer face such that the angle at z in the outer face is at least* 180°.

Proof. Let Γ' be a valid ortho-radial representation of G' without edge bends. In Γ' all three cycles C_1, C_2 and C_3 are essential cycles. Hence, C_2 lies between C_1 and C_3. Furthermore, G either lies inside the area enclosed by C_1, C_2 and u_1u_2 (as in Figure 4.1(a)), or in the area enclosed by C_2, C_3 and u_2u_3 (as in Figure 4.1(b)). Therefore, G cannot contain any essential cycles. Hence, the ortho-radial representation of G can be interpreted as an orthogonal representation, and thus, G admits an orthogonal embedding. Since P ends at the port z and intersects G only at z, the angle at z in the outer face of G must be at least 180°.

Let Γ be an orthogonal representation of G without bends such that z lies on the outer face and the angle at z in the outer face is at least 180°. We interpret Γ as an ortho-radial representation of G and extend it to a representation of G' as follows: We embed H such that C_1 lies in the interior of C_2, which in turn lies in C_3. We place G between C_1 and C_2 as shown in Figure 4.1(a). Since Γ has no bends, the path P connecting G and H needs to make at most as many turns as there are edges incident to the outer face of G. As the length of P is equal to the number of edges of G, we can place P such that all turns occur at vertices, i.e., P contains no edge bends. Moreover, P can be placed completely in the exterior of G because the angle at the port z has at least 180°. □

The construction of G' relies on the fact that we find a vertex z in G such that placing z on the outer face of G does not restrict the possible embeddings too much. In order to show

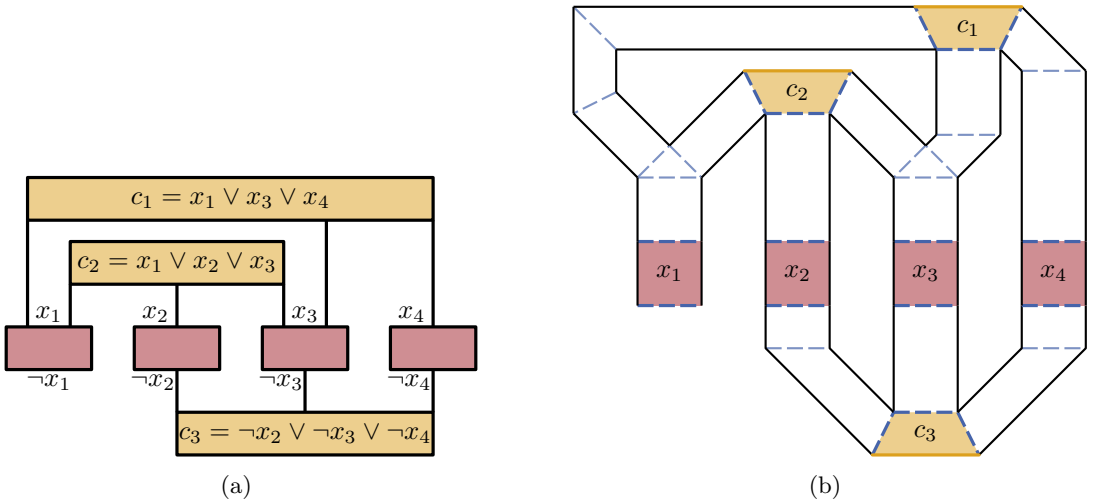


Figure 4.2: (a) An instance Φ of Planar Monotone 3-SAT. (b) A schematic drawing of the graph G constructed from Φ. The orange-shaded parts correspond to the clauses and the red-shaded ones to the variables. The dashed blue lines refer to edges with exactly one bend.

how z can be chosen, we present some details of the reduction from Planar Monotone 3-SAT to Orthogonal Embeddability by Bläsius et al. [BBR14].

The idea behind the reduction is to rebuild the variable-clause graph of Φ and represent the truth values by edge bends. Figure 4.2 provides an example of this construction. Bläsius et al. introduce edges that must have exactly one bend (*1-edges*) and represent true and false as bends in different directions. In our figures, 1-edges are drawn as dashed blue lines (e.g., in Figure 4.2(b)). However, instances of Orthogonal Embeddability cannot contain 1-edges and the gadget shown in Figure 4.3(c) is used instead. Flipping the embedding of this gadget changes the direction of the bend.

We choose the port z on a part of G that corresponds to one of the outermost clauses (e.g., on c_1 or c_3 in Figure 4.2). The gadget they use to represent clauses, is shown in Figure 4.3(a). Note that the boxes are not vertices but occurrences of the graph in Figure 4.3(b). Consider the gadget for one of the outermost clauses in the variable-clause graph. We choose any degree-2 vertex of the box in the corner as out port z (cf. Figure 4.3(a)). By the following lemma this vertex has the desired properties.

Lemma 4.4. *Let z be the port of G as selected above. Then, exactly one of the following is true:*

1. *There is an orthogonal embedding of G in which z lies on the outer face and the angle at z in the outer face has at least 180°.*
2. *The graph G does not admit any orthogonal embedding.*

Proof. If G admits an orthogonal embedding, then Φ is satisfiable [BBR14]. But then, G can be embedded like the variable-clause graph of Φ. In particular, z lies on the outermost clause gadget. By the choice of z this implies that z also lies on the outer face of the whole graph G. Moreover, the angles in the drawing can be chosen such that the angle at z in the outer face has at least 180°. □

Finding such a vertex z was the missing piece in the reduction from Planar Monotone 3-SAT to Orthogonal Embeddability.

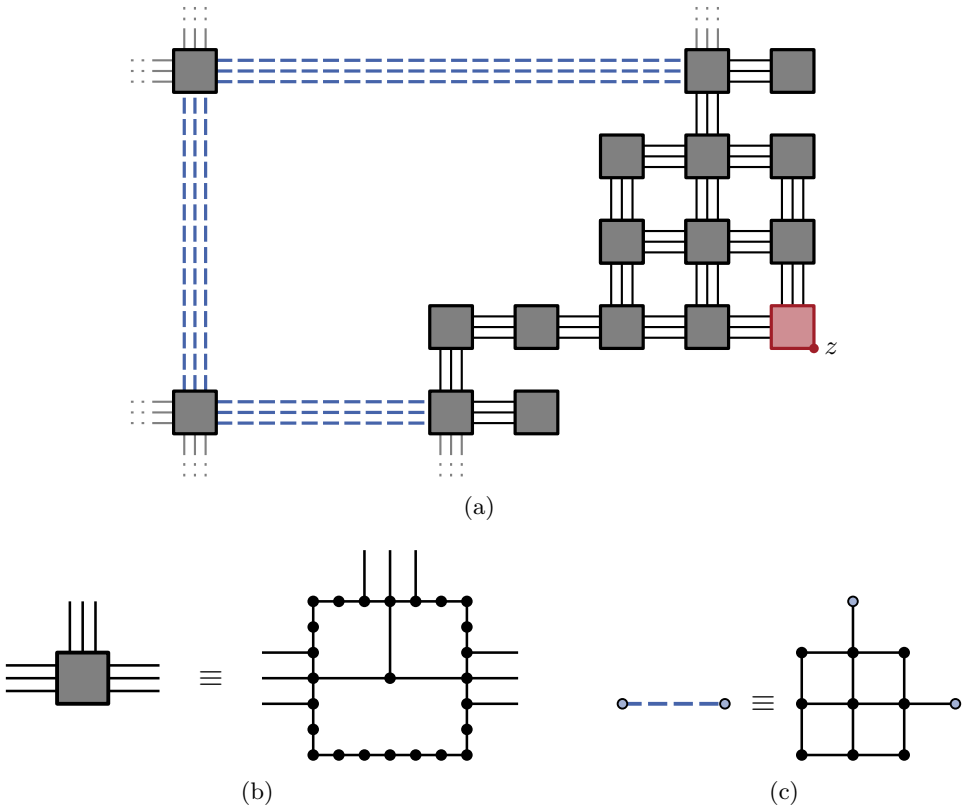


Figure 4.3: (a) The clause gadget. The dashed blue edges must have exactly one bend. The dotted edges indicate where the clause gadget is connected to the remainder of G. (b) The subgraph each box stands for. (c) The gadget that represents edges that must have exactly one bend. By flipping the embedding one can achieve both bends to the left and to the right.

Theorem 4.5. Ortho-Radial Embeddability *is $\mathcal{NP}$-complete.*

Proof. Clearly, Ortho-Radial Embeddability lies in $\mathcal{NP}$.

The construction presented above first transforms an instance Φ of Planar Monotone 3-SAT to an instance G of Orthogonal Embeddability and then to an instance G' of Ortho-Radial Embeddability. By Lemma 4.3 the graph G' admits an ortho-radial embedding if and only if G admits an orthogonal embedding with z on the outer face. According to Lemma 4.4 this is in turn equivalent to the existence of an orthogonal embedding of G without requiring z to lie on the outer face. Bläsius et al. [BBR14] prove that G admits an orthogonal embedding if and only if Φ is satisfiable. In total, this implies that Φ and G' are equivalent. Furthermore, the reduction runs in polynomial time. As Planar Monotone 3-SAT is $\mathcal{NP}$-hard [dBK12], Ortho-Radial Embeddability is $\mathcal{NP}$-hard as well. □

One might wonder why we do not directly reduce Orthogonal Embeddability to Ortho-Radial Embeddability but instead start from Planar Monotone 3-SAT. This is due to the fact that we need to connect the triangles to a vertex of the instance G of Orthogonal Embeddability. Therefore, we must find a vertex z on G that can lie on the outer face. That is, if G admits an orthogonal drawing, there is an orthogonal drawing of G such that z lies on the outer face. We cannot identify such a vertex easily on arbitrary instances. For instances created by the reduction from Planar Monotone 3-SAT, however, we can exploit the structure to find a suitable vertex.

5. Drawing Cactus Graphs with the Minimum Number of Bends

As we have seen in the previous chapter, it is $\mathcal{NP}$-complete to decide whether an arbitrary 4-planar graph can be drawn ortho-radially without bends when one is allowed to choose the embedding. In particular, minimizing the number of bends in an ortho-radial drawing is hard. But if one restricts oneself to certain graph classes, there are polynomial time algorithms that minimize the number of bends. For instance, any tree can be drawn orthogonally—and therefore also ortho-radially—without any bends.

In this chapter we present a linear time algorithm that solves the bend minimization problem for cactus graphs with maximum degree 4. A connected graph is a *cactus graph* if and only if no pair of vertices lies on more than one simple cycle. In other words, two simple cycles in a cactus graph share at most one vertex. These properties ensure that no vertex with degree at most 4 lies on more than two simple cycles.

Observation 5.1. *In a cactus graph with maximum degree 4 no three cycles share a common vertex.*

We first describe how the structure of cactus graphs can be captured in a tree—the so called cycle tree. In Section 5.2 we present the Cycle Drawing Algorithm, which draws individual cycles ortho-radially with the minimum number of bends. We then show how these cycles can be combined when it is given which cycle may be drawn as essential cycles and which only as non-essential cycles (cf. Section 5.3). Finally, we present a dynamic program to decide which cycles shall be drawn as essential cycles in Section 5.4.

5.1 The Cycle Tree

Clearly, each tree is a cactus graph. But even the structures of general cactus graphs are quite similar to trees. We formalize this resemblance by the following construction: Given a cactus graph G with maximum degree 4, let $\mathcal{C} = \{C_1, \ldots, C_k\}$ be the set of all simple cycles in G. In the *cycle tree* each simple cycle $C_i \in \mathcal{C}$ is represented by a *cycle vertex* c_i. Each vertex of G that is not part of any simple cycle also belongs to the cycle tree as a *regular vertex*. The cycle tree is an undirected graph $\mathcal{T} = (V_\mathcal{T}, E_\mathcal{T})$, where $V_\mathcal{T}$ is the set of cycle and regular vertices; see Figure 5.1 for an example of a cycle tree. There is an edge between two different vertices a and b of $\mathcal{T}$ if and only if at least one of the following conditions holds:

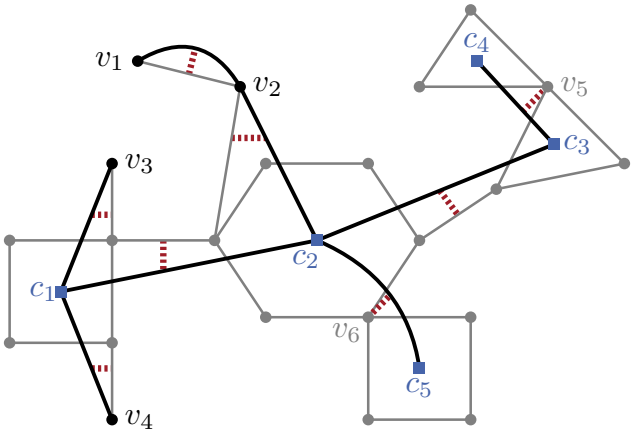


Figure 5.1: The cycle tree of a cactus graph G. The original graph is drawn in gray. Here, the regular vertices are $v_1, \ldots, v_4$ and the cycle vertices are $c_1, \ldots, c_5$. The correspondences of edges in the cycle tree and edges and vertices of G are indicated by red dashed lines.

1. Both a and b are regular vertices and they are adjacent in G (e.g., v_1v_2 in Figure 5.1).
2. Exactly one of a and b is a cycle vertex for a cycle C and the other vertex is adjacent to a vertex of C in G (e.g., c_1v_3, c_1v_4 and c_2v_2 in Figure 5.1).
3. Both a and b are cycle vertices for cycles C and C', resepectively, and there is an edge e in G between a vertex of C and one of C' that does not belong to any cycle of G (e.g., c_1c_2 and c_2c_3 in Figure 5.1).
4. Both a and b are cycle vertices for cycles C and C' and these cycles have a common vertex (e.g., c_2c_5 and c_3c_4 in Figure 5.1).

Note that if any of the Conditions 1–3 holds for an edge $e \in E(\mathcal{T})$, there is exactly one edge $e' \in E(G)$ of the original graph G that induces e. We say that e' is the *original edge corresponding to* e. These correspondences are indicated by dashed red lines in Figure 5.1. If Condition 4 holds for an edge e (e.g., c_3c_4 in Figure 5.1), there is no such corresponding original edge and we call e a *virtual edge*. There is however a vertex $v \in V(G)$ that lies on both cycles represented by the endpoints of e and we call v the *original vertex corresponding to* e. For instance, in Figure 5.1 the vertex v_5 corresponds to the virtual edge c_3c_4.

Lemma 5.2. *The cycle tree of a connected cactus graph with a maximum degree of at most 4 is a tree.*

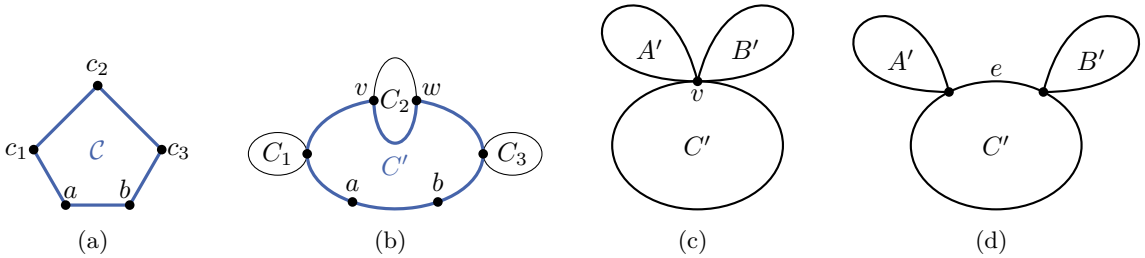


Figure 5.2: Illustrations for the proof of Lemma 5.2. (a) A hypothetic cycle $\mathcal{C}$ in the cycle tree. (b) The cycle C' in the graph G constructed from $\mathcal{C}$. (c) The edge $a'b'$ between the cycle vertices of A' and B' is virtual and corresponds to v. (d) The edge $a'b'$ is not virtual and corresponds to the edge $e \in E(G)$.

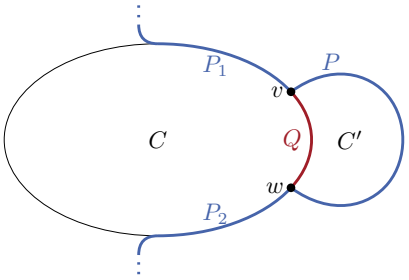


Figure 5.3: Situation of Lemma 5.3. The path P intersects the cycle C twice. The paths $P[v, w]$ and Q together form another cycle C'.

Proof. Let $\mathcal{T}$ be the cycle tree of a 4-planar cactus graph G and assume for the sake of contradiction that there is a simple cycle $\mathcal{C}$ in $\mathcal{T}$. To derive a contradiction, we first construct a cycle in G by replacing all cycle vertices on $\mathcal{C}$ with paths as follows: Let $b \in V(\mathcal{C})$ be a cycle vertex for a cycle B and consider the edges ab and bc on $\mathcal{C}$ that are incident to b. If these edges are non-virtual, they correspond to edges of G, which end at vertices of B. If they are virtual, they directly correspond to vertices of B. In any case, they determine two (possibly equal) vertices v and w of B. In Figure 5.2(b) these vertices are shown for C_2. We arbitrarily choose a path on B from v to w and replace b with that path. After replacing all cycle vertices in this fashion, we obtain a simple cycle C' in G as shown in Figure 5.2(b).

Since regular vertices do not lie on simple cycles of G by definition, there cannot be any regular vertices on C'. Therefore, all vertices of $\mathcal{C}$ are cycle vertices. Since $\mathcal{C}$ is simple and has at least length 3, we can choose an edge $a'b' \in E(\mathcal{C})$ such that neither a' nor b' are cycle vertices for C'. Let A' and B' be the cycles represented by a' and b', respectively. If $a'b'$ is virtual, there is a common vertex v on both A' and B' as illustrated in Figure 5.2(c). By construction of C' this vertex v must also lie on C'. As A', B' and C' are three distinct simple cycles and any two cycle have at most one vertex in common, v must have a degree of at least 6—two edges per cycle—contradicting the maximum degree of 4.

Hence, $a'b'$ is not virtual and corresponds to an edge e of G (cf. Figure 5.2(d)). By construction, this edge lies on C', and therefore, it does not induce the edge $a'b'$, contradicting the correspondence between $a'b'$ and e. In this situation the cycle tree would only contain the edge $a'c'$ and $b'c'$ but not the edge $a'b'$. Thus, the assumption that there is a simple cycle in $\mathcal{T}$ is false. As G is connected and the construction of $\mathcal{T}$ preserves this connectivity, we conclude that $\mathcal{T}$ is a tree. □

The following lemma investigates the relationship between paths and simple cycles of G that will allow us to define a projection of paths from G to $\mathcal{T}$.

Lemma 5.3. *Fix a simple cycle C in a cactus graph G. For any path P in G the intersection $P \cap C$ forms a subpath of P (or is empty).*

Proof. Assume for the sake of contradiction that $C \cap P$ is not one path. Then $C \cap P$ contains (at least) two paths P_1 and P_2. We may assume without loss of generality that P_1 occurs before P_2 on P and no vertex of P between these paths lies on C. Let v be the last vertex of P_1 and w be the first vertex of P_2 (cf. Figure 5.3). By construction the path $P[v, w]$ intersects C only at v and w. As both v and w lie on C, there is a path Q on C from w to v. Joining $P[v, w]$ and Q together yields a simple cycle C'. The intersection of C and C' contains at least two vertices—namely v and w—but $C \neq C'$ since C' contains $P[v, w]$, which does not belong to C. This contradicts the properties of cactus graphs, and thus, $C \cap P$ must be one path. □

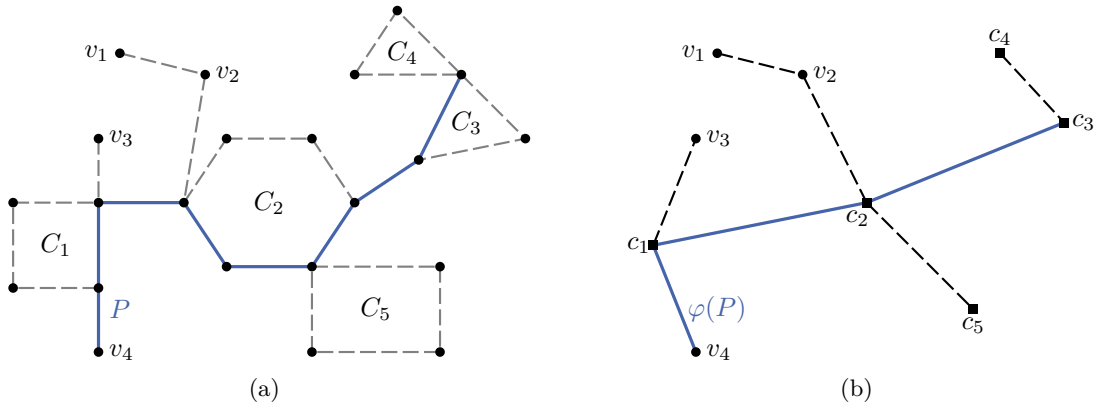


Figure 5.4: (a) A path P in the cactus graph G. (b) Its projection $\varphi(P)$ to the cycle tree $\mathcal{T}$. The projection does not contain c_4 and c_5 although P contains vertices of C_4 and C_5, since $P \cap C_4$ and $P \cap C_5$ are part of longer subpaths of P on C_3 and C_2, respectively.

For any path $P = v_1 \dots v_k$ in G with $k \geq 2$ vertices we define its *projection* $\varphi(P)$ to $\mathcal{T}$ by replacing subpaths on one cycle of G by the corresponding cycle vertex. More precisely, the projection is constructed as follows: For each simple cycle $C \in \mathcal{C}$ the intersection $P \cap C$ is one contiguous path or empty (cf. Lemma 5.3). Consider the set of all such non-empty paths $P \cap C$ and let $\mathcal{P}_{\text{cycle}}$ be this set after removing all paths that are a proper subpath of another path in this set. For instance, for the path in Figure 5.4 we have $\mathcal{P}_{\text{cycle}} = \{P \cap C_1, P \cap C_2, P \cap C_3\}$. Note that $P \cap C_4$ and $P \cap C_5$ were removed as they are a part of $P \cap C_3$ and $P \cap C_2$, respectively. Let $\mathcal{P}_{\text{regular}}$ be the set of all vertices of P that do not belong to any simple cycle. We interpret the elements of $\mathcal{P}_{\text{regular}}$ as paths of length 0. In Figure 5.4 it is $\mathcal{P}_{\text{regular}} = \{v_4\}$.

We order the paths in $\mathcal{P} = \mathcal{P}_{\text{cycle}} \cup \mathcal{P}_{\text{regular}}$ by their endpoints and get the sequence $P_1, \dots, P_\ell$. Note that no two paths in $\mathcal{P}$ have the same endpoint as then one path would be a proper subpath of another path and would have been removed. The projection $\varphi(P) = a_1 \dots a_\ell$ contains one vertex a_i per element $P_i \in \mathcal{P}$: If P_i corresponds to a cycle C, a_i is the cycle vertex for C. Otherwise, it is $P_i = v$ and v occurs as a regular vertex in $\mathcal{T}$. In this case we set $a_i = v$.

Lemma 5.4. *The projection $\varphi(P)$ of any path in G is a path in the cycle tree $\mathcal{T}$.*

Proof. Let P be a path in G and $\varphi(P) = a_1 \dots a_\ell$ its projection to $\mathcal{T}$. We first show that two neighboring vertices in $\varphi(P)$ are adjacent in $\mathcal{T}$: Each vertex a_i corresponds to a subpath P_i of P, namely $P_i = P \cap C_j$ if $a_i = c_j$ is the cycle vertex for a cycle C_j, or the path consisting of the single vertex a_i if a_i is regular. By the definition of the projection, the last vertex of P_i and the first vertex of P_{i+1} are either adjacent or identical. In both cases the vertices a_i and a_{i+1} are adjacent in $\mathcal{T}$.

To see that the projection uses no vertex more than once, we note that each occurrence of a regular vertex v in $\varphi(P)$ corresponds to an occurrence of v in P. Since P is a path, $\varphi(P)$ contains v exactly once. For a simple cycle C Lemma 5.3 states that the intersection $P \cap C$ forms a path. Hence, $\varphi(P)$ contains the cycle vertex for C at most once. Therefore, $\varphi(P)$ is a path. □

In any ortho-radial drawing of G we observe that the essential cycles must lie inside each other and their cycle vertices lie on one path in $\mathcal{T}$.

Lemma 5.5. *Let $\mathcal{C}$ be the set of all simple, essential cycles in an ortho-radial drawing of a connected cactus graph G with maximum degree 4. Then, there is a path in the cycle tree of G containing all cycle vertices for cycles in $\mathcal{C}$.*

Proof. Since any two cycles in $\mathcal{C}$ share at most one vertex and all these cycles contain the center in their interiors, one of the cycles lies in the interior of the other. Thus, there is an order $C_1, \dots, C_k$ of the cycles in $\mathcal{C}$ such that C_i contains all cycles C_j with $j \leq i$ in its interior. That is, C_1 is the innermost cycle and C_k the outermost.

As G is connected, there is a path P_i from a vertex s_i on C_i and to a vertex t_{i+1} on C_{i+1}. We may choose the start and end vertices such that P_i contains no other vertices from C_i and C_{i+1}. We join all these paths by adding a path from t_i to s_i on C_i for $i = 2, \dots, k-1$. We furthermore add a neighbor of s_1 on C_1 at the beginning and a neighbor of t_k on C_k at the end. We denote the result of the construction by P. As each path P_i lies completely in the exterior of C_i and the interior of C_{i+1}, P does not contain any vertex twice and hence is a path.

By construction the path P has length at least 2. Hence, the projection $\varphi(P)$ is well-defined and a path by Lemma 5.4. We furthermore claim that $\varphi(P)$ contains all cycle vertices for cycles in $\mathcal{C}$. The path P contains at least one vertex per cycle $C \in \mathcal{C}$. Hence, $\varphi(P)$ contains the cycle vertex for C unless all vertices of $P \cap C$ also lie on another (possibly non-essential) cycle C' in G. In this case it is $P \cap C = \{v\}$, since no two cycles share more than one vertex, and it is $C \neq C_1, C_k$, as P contains two vertices of these cycles. But then v would have at least 5 neighbors—two on C, two on C' and the neighbor of v on P that does not lie on C'—contradicting the maximum degree of 4. Thus, $\varphi(P)$ contains the cycle vertices for all vertices of $\mathcal{C}$. □

5.2 Drawing Cycles

One important subtask of drawing cactus graphs is to decide how the individual cycles are drawn. In this section we present an algorithm that produces an ortho-radial drawing of a given cycle and the edges incident to vertices of that cycle with the minimum number of bends. More precisely, we are given a cactus graph G, a cycle C in G and two edges e_{in} and e_{out} incident to C but not on C. We denote the (possibly identical) vertices on the cycle C incident to these edges by v_{in} and v_{out}, respectively. In the following, we consider e_{in} to be directed towards v_{in} and e_{out} directed away from v_{out}. Additionally, we are given labels $d_{\text{in}}, d_{\text{out}} \in \{H, V\}$ or the edges e_{in} and e_{out}, respectively. The labels describe in which direction the edges e_{in} and e_{out} must be drawn. We say an ortho-radial drawing of C and all edges incident to a vertex in C is *extensible* if it satisfies the following conditions:

1. The edge e_{in} lies in the exterior of C.
2. The edge e_{out} lies in the interior of C if C is drawn as an essential cycle, and in its exterior otherwise.
3. If $d_{\text{in}} = H$, the edge e_{in} is drawn horizontally. Otherwise, it points downwards.
4. If $d_{\text{out}} = H$, the edge e_{out} is drawn horizontally. Otherwise, it points downwards.
5. If two edges incident to a vertex on C but not on C belong to a common simple cycle, they are drawn on the same side of C.

These conditions ensure that it is possible to combine the drawings of cycles to obtain a drawing of the whole graph G, i.e., to extend these drawings to a drawing of G.

5.2.1 The Cycle Drawing Algorithm

In this section we present an algorithm that produces an extensible drawing of a simple cycle C with the minimum number of bends. The cycle may be drawn as an essential or a non-essential cycle depending on which produces fewer bends. In the description of the algorithm we use the word *turn* to describe positions at which the boundary of C changes its direction (e.g., from going left to going down). Turns may occur at vertices or on edges. In the latter case we also call them *bends*.

We first classify the vertices of C as follows: If a vertex v belongs to two different simple cycles (i.e., to C and another cycle), in any drawing C must have a turn at this vertex. Hence, we say that v *forces a turn*. Similarly, if we have $v_{\text{in}} = v_{\text{out}}$ and $d_{\text{in}} \neq d_{\text{out}}$ (as in Figure 5.5(b)), there must be a turn at v_{in} and we also say that v_{in} *forces a turn*.

A vertex v that is not part of two simple cycles is called *free*, unless it is $v = v_{\text{in}} = v_{\text{out}}$. Drawings of C may make turns at free vertices but are not required to do so. For instance, there are turns at some but not all free vertices in Figure 5.5. Note that if $v = v_{\text{in}} = v_{\text{out}}$ and $d_{\text{in}} = d_{\text{out}}$, the vertex v neither forces a turn nor is free (cf. Figure 5.5(c)). In this case there must not be a turn at v.

Let $T \subseteq V(C)$ be the set of vertices forcing a turn and $t = |T|$ its cardinality. The algorithm works in two steps: First, we select a set T' with $T \subseteq T'$ of *turn vertices*. These are the vertices of C at which there shall be turns in the resulting drawing. In this step we may introduce some *dummy vertices* as turn vertices that correspond to bends on edges of C. In a second step we decide for each turn vertex whether there should be a left or a right turn.

If $t = 0$ and $d_{\text{in}} = d_{\text{out}} = V$, we draw the cycle as an essential cycle without any turns by placing all vertices of C on the same circle of the grid. In this case we set $T' = \emptyset$ and the drawing of C has no bends. An example of such a drawing is shown in Figure 5.6.

In all other cases, the drawing of the cycle needs turns. If $t < 4$, we arbitrarily select $4 - t$ free vertices on C. If there are not enough free vertices on C, we create a sufficient number of dummy vertices on any edge of C. Adding these selected vertices to T gives T'. By construction we have $|T'| = 4$.

If $t \geq 4$ and t is odd, we select one free vertex v on C and set $T' = T \cup \{v\}$. If there is no free vertex on C, we again create a dummy vertex.

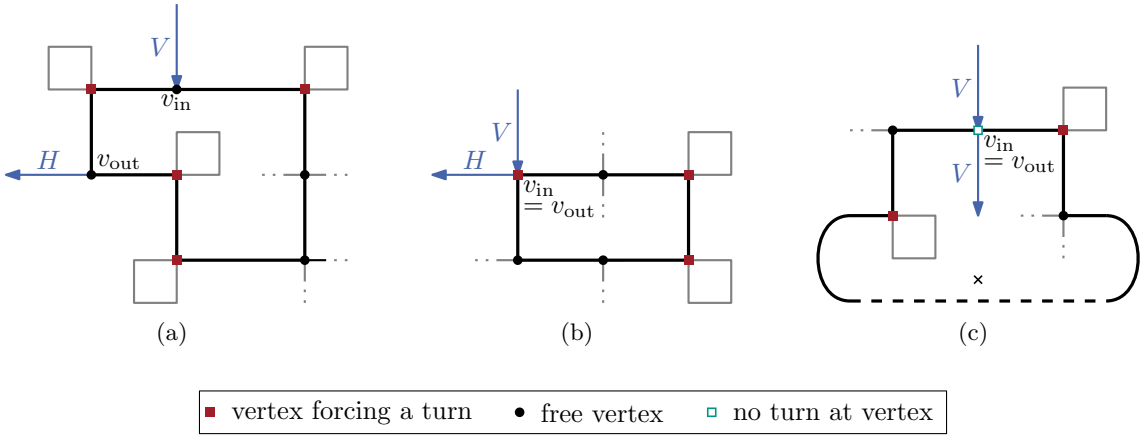
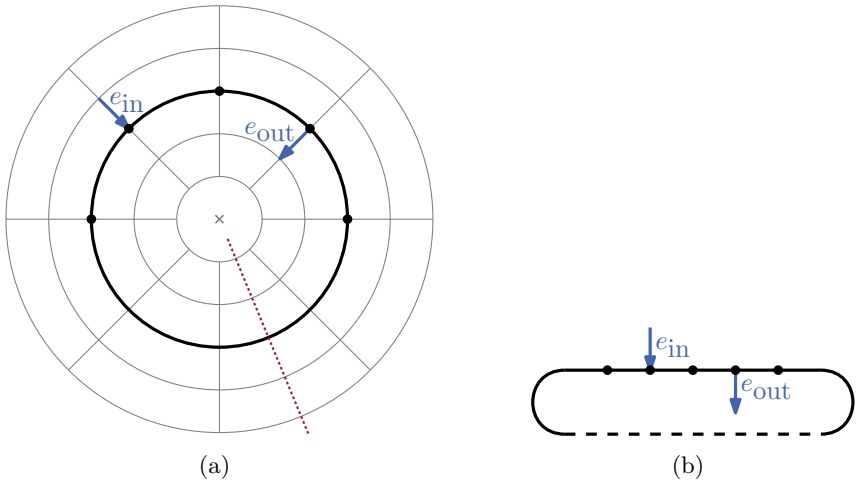
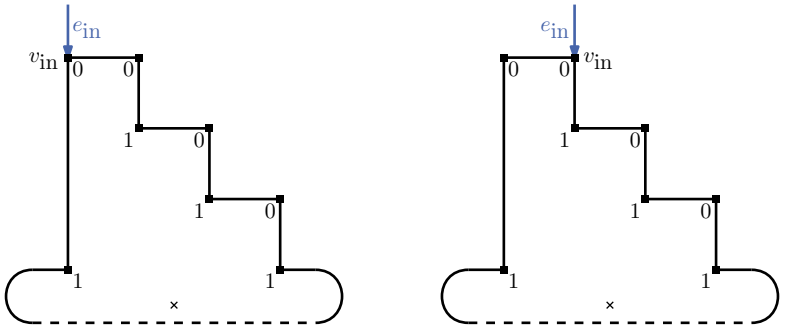


Figure 5.5: Three examples of the vertex classification. All vertices that are on two simple cycles force a turn. At free vertices there may or may not be a turn.



(a) (b)

Figure 5.6: If $t = 0$ and $d_{\text{in}} = d_{\text{out}} = V$, the cycle is drawn as an essential cycle without turns. (a) Drawing of the cycle on the grid. (b) A simplified representation of the drawing, in which the cycle is cut at one edge. In (a) this cut is indicated by a dotted red line.



(a) (b)

Figure 5.7: The shapes of the cycles in Case 1, which only differ by the position of v_{in}. The edge e_{in} may also be horizontal. (a) The shape described by $w_1 = 00(10)^k11$. (b) The shape described by $w_2 = 0(10)^2110$.

If none of the cases above applies, we set $T' = T$. By construction $|T'|$ is even and $|T'| \geq 4$. In order to draw the cycle C with turns at the vertices in T', we distinguish between four cases:

1. $v_{\text{in}} \neq v_{\text{out}}$ and v_{in} is a turn vertex.
2. $v_{\text{in}} \neq v_{\text{out}}$ and v_{in} is not a turn vertex, but v_{out} is a turn vertex.
3. $v_{\text{in}} \neq v_{\text{out}}$ and neither v_{in} nor v_{out} is a turn vertex.
4. $v_{\text{in}} = v_{\text{out}}$.

We describe the turns one encounters when walking around C in clockwise order starting from v_{in} with a binary string w of length $|T'|$. In this representation a 0 corresponds to a right turn and a 1 to a left turn. The direction of the turn at the i-th turn vertex after v_{in} is described by the i-th bit of w. If v_{in} is a turn vertex, the first bit of w refers to the turn at v_{in}.

In Case 1 we describe the turns with one of the strings $w_1 = 00(10)^k11$ and $w_2 = 0(10)^k110$, where $k = (|T'| - 4)/2$. Note that these two strings only differ by a cyclic rotation and

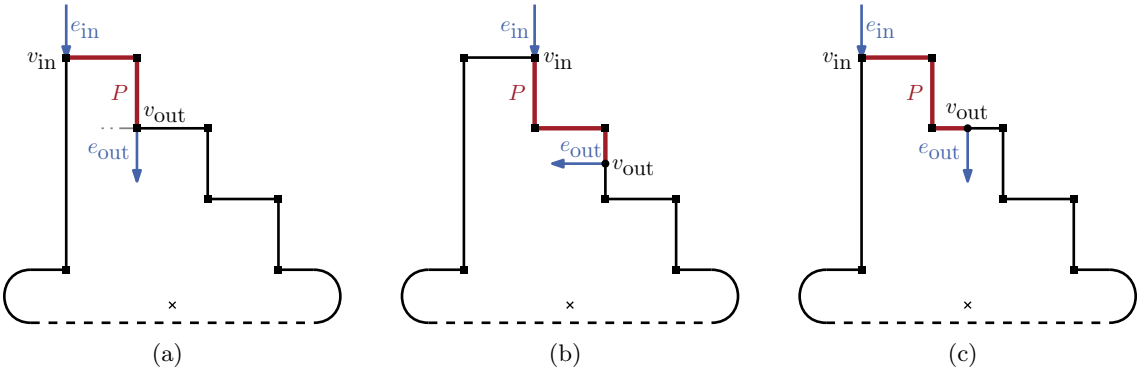


Figure 5.8: Drawings of C if the path P between v_{in} and v_{out} contains an odd number of turn vertices. Here, there are three turns vertices on P. (a) The vertex v_{out} is a turn vertex. (b) It is $d_{\text{out}} = H$. (c) It is $d_{\text{out}} = V$.

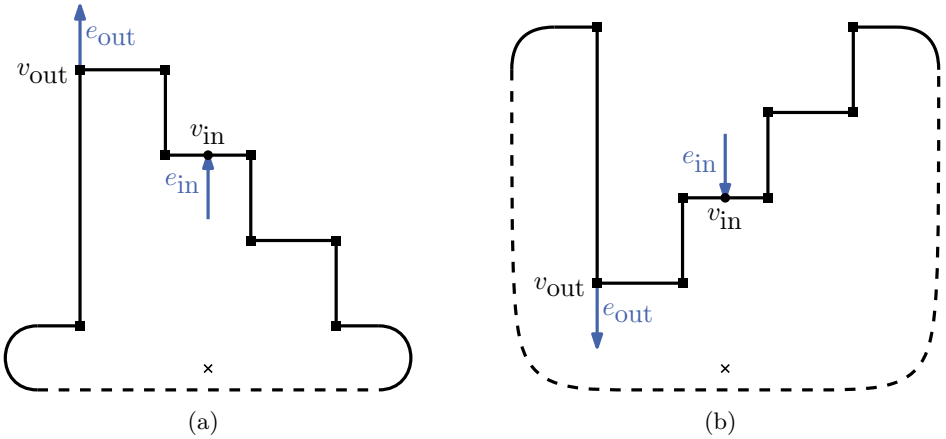


Figure 5.9: The flipping of the drawing in Case 2. (a) The drawing with reversed roles of v_{in} and v_{out} before the flipping. (b) The resulting drawing after the flipping.

therefore describe the same shape, which resembles a staircase (cf. Figure 5.7). The cyclic rotation only changes which turns are mapped to which turn vertices.

We determine which string we use by considering the directed path P on C from v_{in} to v_{out} with the fewest turn vertices. If both paths have the same number of turn vertices, we may choose one arbitrarily. We denote the number of turn vertices on P (including v_{in}) by t_P. Depending on d_{out}, the parity of t_P and whether v_{out} is a turn vertex, we set $w = w_1$ or $w = w_2$ as presented in Table 5.1. While distributing the turns we consider C to be directed such that the edges on P are directed towards v_{out}. Figure 5.8 shows examples of the resulting drawings for an odd number of turn vertices on P.

Table 5.1: Selection of the binary string describing the turns in Case 1

	d_{out}	t_P even	t_P odd
$v_{\text{out}} \in T'$	H, V	w_2	w_1
$v_{\text{out}} \notin T'$	H	w_1	w_2
$v_{\text{out}} \notin T'$	V	w_2	w_1

In Case 2 we temporarily change the roles of e_{in} and e_{out} and proceed as in Case 1. The resulting drawing Γ of C is not yet correct though, since e_{out} lies in the exterior of C and e_{in} in the interior, but it should be the other way round. We correct this by flipping the drawing horizontally as shown in Figure 5.9.

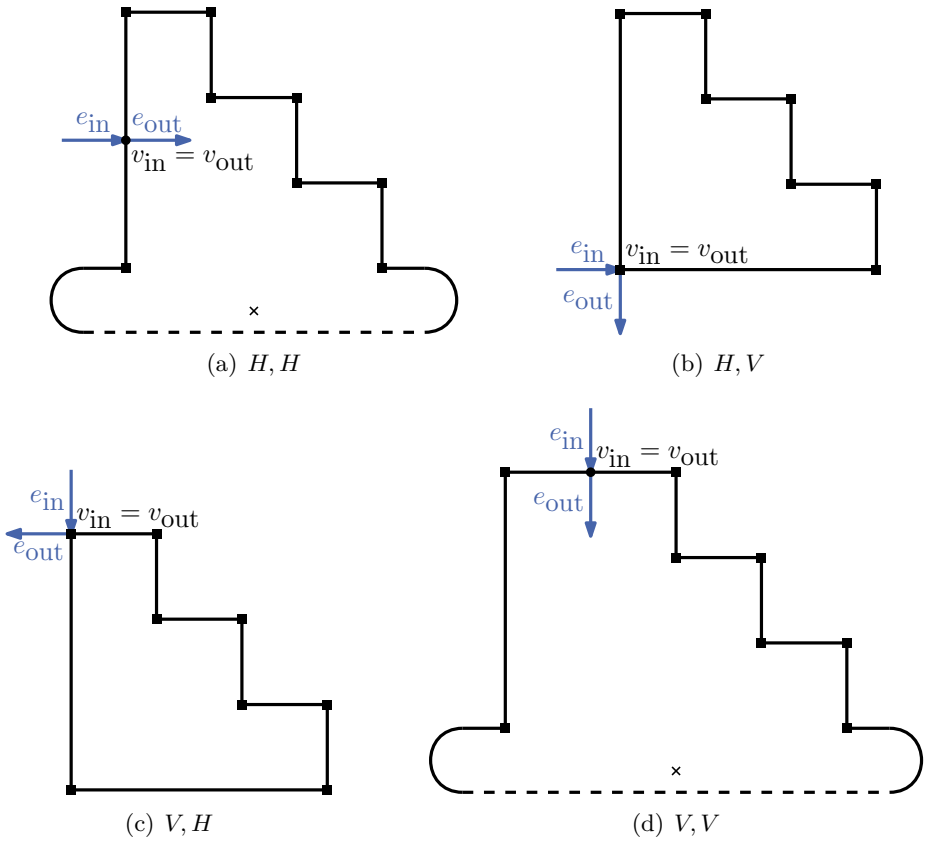


Figure 5.10: Drawings of C in Case 4, where it is $v_{\text{in}} = v_{\text{out}}$. (a),(d) It is $d_{\text{in}} = d_{\text{out}}$ and the cycle is drawn as an essential cycle. (b),(c) It is $d_{\text{in}} \neq d_{\text{out}}$ and the cycle is drawn as a non-essential cycle.

Case 3 can be reduced to Case 1 by setting $T'' = T' \cup \{v_{\text{in}}, v_{\text{out}}\}$. As v_{in} and v_{out} are distinct and not turn vertices, we have $|T''| = |T'| + 2$. Thus, $|T''|$ is even and $|T''| \geq 4$. Hence, as $v_{\text{in}} \in T''$, we can proceed as in Case 1 using T'' as the set of turn vertices instead of T'.

Finally, in Case 4 we use the binary strings in Table 5.2. The shapes described by these strings are illustrated in Figure 5.10. If $d_{\text{in}} = d_{\text{out}}$, we reuse the shapes from Case 1, which describe essential cycles. Otherwise, the shapes describe non-essential cycles.

Table 5.2: The shapes of the cycle in Case 4. It is $k = (|T'| - 4)/2$.

d_{in}	d_{out}	w	
H	H	$00(10)^k11$	$= w_1$
H	V	$000(10)^k0$	
V	H	$00(10)^k00$	
V	V	$0(10)^k110$	$= w_2$

One can observe a useful structure for the case in which one of the labels is H, which helps us to handle the case if the incoming edge e_{in} or the outgoing edge e_{out} is part of a simple cycle.

Observation 5.6. *If $d_{in} = H$ and v_{in} is part of two cycles, one of the edges on C incident to v_{in} is drawn below v_{in} and the other one horizontally. Similarly, for $d_{out} = H$ one of the edges is drawn above v_{out} and the other one is drawn horizontally.*

5.2.2 Optimality of the Cycle Drawing Algorithm

We now show that the algorithm presented in the previous section produces extensible drawings with the minimum number of bends among all extensible drawings. We first analyze at which vertices of C there must be turns.

Observation 5.7. *In any extensible ortho-radial drawing of the cycle C there is a turn at each vertex in T.*

We continue with an observation on the number of turns a cycle can have:

Observation 5.8. *The number of turns on a cycle is 0 or at least 4 and even. If the number of turns is 0, the cycle is essential.*

With these observations we are now prepared to show the optimality of the drawings produced by the Cycle Drawing Algorithm.

Theorem 5.9. *Given a cactus graph G with maximum degree 4, a cycle C in G and two edges e_{in} and e_{out} with labels $d_{in}, d_{out} \in \{H, V\}$, the Cycle Drawing Algorithm creates an extensible drawing Γ of C with the minimum number of bends.*

Proof. By the choice of T' all vertices that belong to two simple cycles are placed at a turn. Moreover, one can verify that in all cases the directions edges e_{in} and e_{out} obey their labels d_{in} and d_{out}, respectively. Hence, the drawings produced by the Cycle Drawing Algorithms are extensible.

In order to show the minimality of the number of bends, we consider any extensible ortho-radial drawing $\tilde{\Gamma}$ of C. Let b and $\tilde{b}$ be the number of bends in the drawings Γ and $\tilde{\Gamma}$. We want to show $b \leq \tilde{b}$. This is clearly the case if $b = 0$, that is, Γ does not contain any bends. Therefore, we may assume $b > 0$ in the following. In particular, there are vertices forcing turns, i.e., $T \neq \emptyset$.

Let $\tilde{t}$ be the number of turns in $\tilde{\Gamma}$. Each turn either occurs at a vertex in T, at a free vertex or is a bend. Moreover, there is a turn at each vertex in T by Observation 5.7. Denoting the number of turns at free vertices in $\tilde{\Gamma}$ by $\tilde{f}$, we therefore obtain

$$\tilde{t} = |T| + \tilde{f} + \tilde{b}. \tag{5.1}$$

For the drawing Γ, we similarly get

$$|T'| = |T| + f + b. \tag{5.2}$$

Here, f denotes the number of free vertices that have turns. Note that by construction the number of turns is equal to the number of turn vertices $|T'|$. The algorithm only uses dummy vertices as turn vertices (and thus creates bends) if there are not enough free vertices and $0 < |T| < 4$ or $|T|$ is odd. In the other cases (i.e., $|T| = 0$ or $|T| \geq 4$ and $|T|$ is even) the drawing produced by the Cycle Drawing Algorithm would not contain any bends, but we assumed $b > 0$.

If $0 < |T| < 4$, the algorithm adds $4 - |T|$ vertices to T such that $|T'| = 4$. Since we assumed $b > 0$, at least one of these vertices is a dummy vertex. However, a dummy vertex is only created if there are not enough free vertices. Thus, f is maximal, and hence it is $f \geq \tilde{f}$. By Observation 5.8 we know that $\tilde{t} \geq 4 = |T'|$. Combining these results with Equations 5.1 and 5.2, we get

$$b = |T'| - |T| - f \leq \tilde{t} - |T| - \tilde{f} = \tilde{b}. \tag{5.3}$$

If $|T| \geq 4$ and $|T|$ is odd, exactly one vertex is added to T. By the assumption $b > 0$ this must be a dummy vertex and it is $b = 1$. As the algorithm prefers to select free vertices, this implies that there are no free vertices, and thus it is $f = \tilde{f} = 0$. Hence, Equations 5.1 and 5.2 can be simplified:

$$\tilde{t} = |T| + \tilde{b} \tag{5.4}$$

$$|T'| = |T| + b \tag{5.5}$$

As $\tilde{t}$ is even (by Observation 5.8) and $|T|$ is odd, it holds that

$$|T'| = |T| + 1 \leq \tilde{t}. \tag{5.6}$$

Combining the Equations 5.4–5.6, we obtain

$$b = 1 = |T'| - |T| \leq \tilde{t} - |T| = \tilde{b}. \tag{5.7}$$

We have shown $b \leq \tilde{b}$ for all possible cases. Thus, the Cycle Drawing Algorithm produces drawings of C with the minimum number of bends. □

Not fixing the directions of e_{in} and e_{out} only changes anything if $v_{\text{in}} = v_{\text{out}}$, since in this case the directions of e_{in} and e_{out} determine whether there is a turn at this vertex or not. If it is $v_{\text{in}} \neq v_{\text{out}}$, there must be turns at the vertices in T in any drawing of C. Combining this observation with the restrictions on the number of turns of cycles (cf. Observations 5.7 and 5.8), we obtain the following result:

Corollary 5.10. *If it is $v_{in} \neq v_{out}$, the drawing of C produced by the Cycle Drawing Algorithm for any choice of directions of e_{in} and e_{out} has the minimum number of bends among all drawings of C in which edges incident to C that belong to the same simple cycle lie on the same side of C.*

Moreover, the decision whether we need bends or not in the algorithm does not directly depend on the actual edges chosen as e_{in} and e_{out}.

Observation 5.11. *The minimum number of bends on an extensible drawing of C depends only on*

- *the labels d_{in} and d_{out}, and*
- *whether $v_{in} = v_{out}$ or $v_{in} \neq v_{out}$.*

In other words, if we change e_{in} and e_{out} but keep the properties mentioned above, the required number of bends does not change. The drawing itself however does change.

5.2.3 Variants of the Cycle Drawing Algorithm

The Cycle Drawing Algorithm can easily be extended to the case in which the direction d_{in} of only one edge incident to a vertex v_{in} of C is given: Calling this edge e_{in}, we first select an arbitrary vertex $v_{\text{out}} \in V(C) \setminus \{v_{\text{in}}\}$ and then apply the Cycle Drawing Algorithm. Note that although there are two possible choices for d_{out}, they both result in the same number of bends, since it is $v_{\text{in}} \neq v_{\text{out}}$ (cf. Corollary 5.10).

A second variant deals with drawing bend-minimal non-essential cycles. In this case the exact directions of the edges do not matter. We only have to ensure that in the drawing of a cycle C all edges that belong to another simple cycle C' are placed on the same side of C. Otherwise, C and C' would have to cross. We proceed as in the regular Cycle Drawing Algorithm except that we need to have at least 4 turn vertices, even if there are no vertices that force turns. Moreover, all cycles have the shape that is described by $00(10)^k00$, which we already used in the case that $v_{\text{in}} = v_{\text{out}}$ and $d_{\text{in}} \neq d_{\text{out}}$ (cf. Figure 5.10(c)).

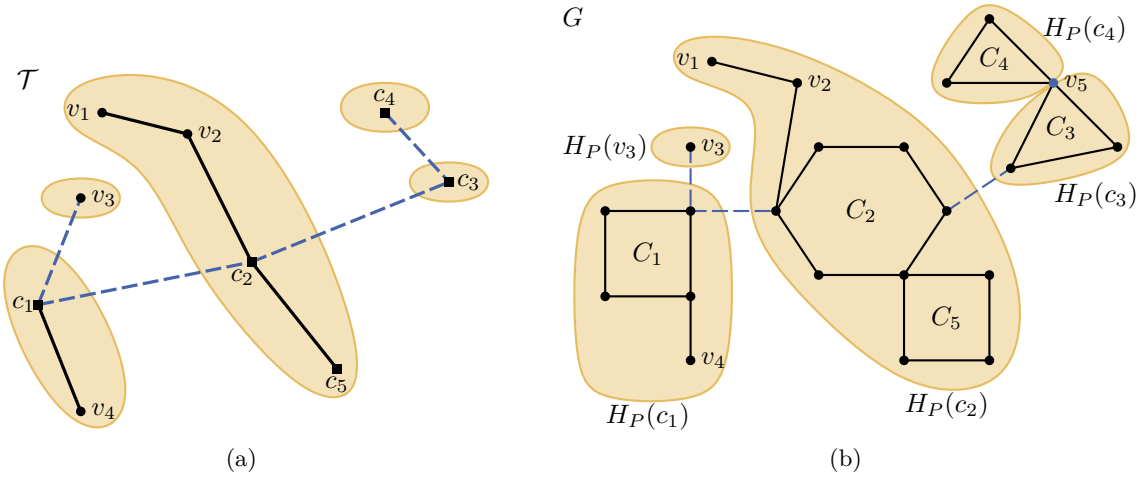


Figure 5.11: (a) A path P in the cycle tree $\mathcal{T}$. The edges of P are indicated by dashed blue lines. (b) The subgraphs of the cactus graph that correspond to the vertices on P.

5.3 P-Drawings and Spines

In an ortho-radial drawing of the whole cactus graph G cycles may be drawn either as essential or as non-essential cycles. In order to produce a bend-minimal drawing of a 4-planar cactus graph G, we need to decide which cycles are essential. But the choices for the cycles are not independent, as all essential cycles must lie on one path in the cycle tree $\mathcal{T}$ by Lemma 5.5.

For a directed path P in $\mathcal{T}$ we say an ortho-radial drawing of G is a *P-drawing* if all cycle vertices of essential cycles lie on P. Note that by Lemma 5.5 any drawing of G is a P-drawing for a suitable path P. Recall that each directed edge e of P corresponds to a directed edge of G (or a vertex if e is virtual). We call these edges the *spine edges* of P. We furthermore associate the direction of each spine edge with the corresponding edge of P. We then say a P-drawing is *downward* if no segment of a spine edge points upwards, i.e., all segments point either left, right or downwards.

Our goal is to give an algorithm that produces bend-minimal P-drawings for a given path P as input. To this end, we first prove in the following section that there always is a bend-minimal P-drawing that is downward and that has no bends on spine edges. The algorithm that computes the bend-minimal P-drawings is then described in Section 5.3.2.

5.3.1 Properties of P-Drawings

Removing all edges of P from $\mathcal{T}$ decomposes $\mathcal{T}$ into several connected components, each of which contains exactly one vertex of P as shown in Figure 5.11. Such a component corresponds to a subgraph of G, which we denote by $H_P(a)$, where $a \in V(P)$ is the vertex of P lying in the component. Note that $H_P(a)$ and $H_P(b)$ are disjoint unless $ab \in E(P)$ and ab is virtual. In this case $H_P(a)$ and $H_P(b)$ have a common vertex, namely the vertex corresponding to ab, but still no common edge. For instance, in Figure 5.11 the subgraphs $H_P(c_3)$ and $H_P(c_4)$ share v_5.

Lemma 5.12. *Let P be a path in $\mathcal{T}$. There exists a downward P-drawing of G with the minimum number of bends among all P-drawings.*

Proof. Let Γ be a P-drawing of G with the minimum number of bends. We assume without loss of generality that the spine edges have no bends, which is achieved by adding dummy vertices at bends of the spine edges.

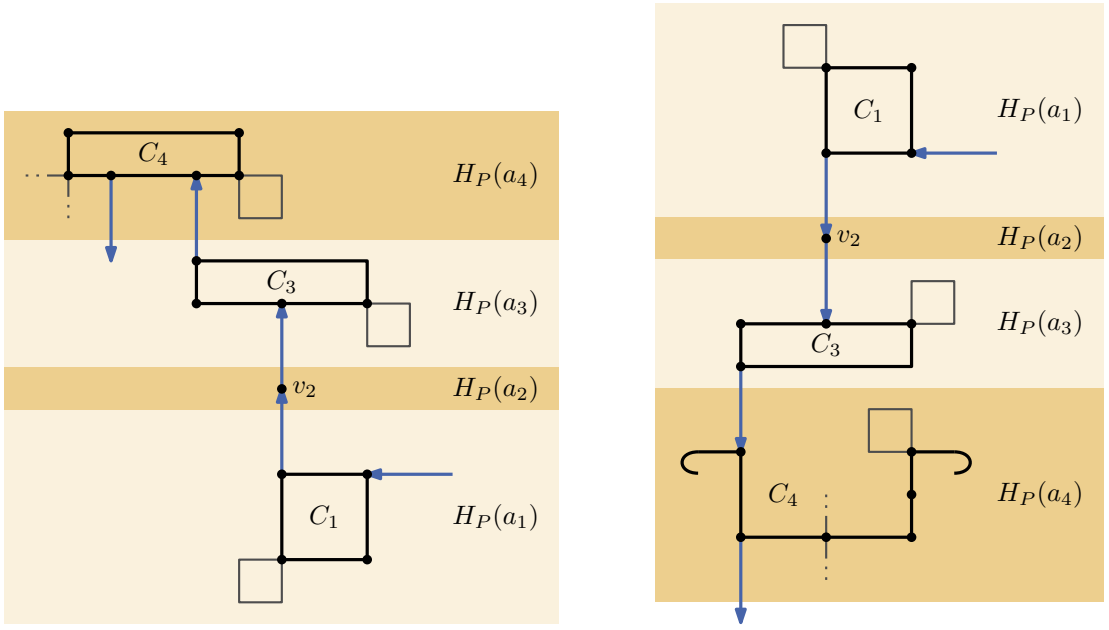


Figure 5.12: An example of redrawing some parts of the cactus graph to avoid spine edges that point upwards as described in the proof of Lemma 5.12. (a) An excerpt of the cactus graph, in which the spine edges (drawn as blue arrows) point upwards. (b) Another drawing, in which no spine edge points upwards.

If Γ does not contain any spine edges that point upwards, Γ has the desired form. Otherwise, we find a maximal subpath $Q = a_1 \dots a_k$ of P which is completely directed upwards ($k \geq 2$). That is, all edges of G corresponding to the edges of Q point upwards in Γ. For instance, in Figure 5.12(a) the spine edges between C_1, v_2, C_3 and C_4 point upwards. We show in the following how to draw the part of G corresponding to Q such that the spine edges in this part are drawn downwards instead: For $i = 2, \dots, k-1$ we flip $H_P(a_i)$ horizontally. Thus, the spine edges between these subgraphs must point downwards. For instance, in Figure 5.12 this the subgraphs $H_P(a_2)$ and $H_P(a_3)$ are flipped.

For the first part $H_P(a_1)$ we distinguish between two cases: the incoming spine edge e_{in} (i.e., the one not on Q) points left or right, or it points downwards. In the first case we flip $H_P(a_1)$ horizontally (as shown in Figure 5.12). Flipping horizontally does not change the direction of edges pointing left or right. Hence, e_{in} can still be connected to $H_P(a_1)$ as before.

If e_{in} points downwards, we denote the endpoint of e_{in} on $H_P(a_1)$ by v_{in} and the starting point of the following spine edge e_{out} on $H_P(a_1)$ by v_{out}. We have $v_{\text{in}} \neq v_{\text{out}}$, since the two spine edges e_{in} and e_{out} both are incident to the top of v_{in} and v_{out}, respectively. In particular, a_1 is not regular but a cycle vertex for a cycle C. The Cycle Drawing Algorithm from Section 5.2 produces a bend-minimal drawing of C such that both e_{in} and e_{out} point downwards (cf. Corollary 5.10). We then replace the drawing of C in Γ with this drawing and rearrange the remaining parts of $H_P(a_1)$ accordingly.

The last part $H_P(a_k)$ is handled similarly: If the outgoing spine edge points left or right, the part is flipped horizontally; otherwise, it is replaced by a drawing by the Cycle Drawing Algorithm. In the graph of Figure 5.12 the cycle C_4 was redrawn by the Cycle Algorithm. Applying this procedure to all subpaths of P that point upwards yields a drawing of G, in which no spine edge points downwards. □

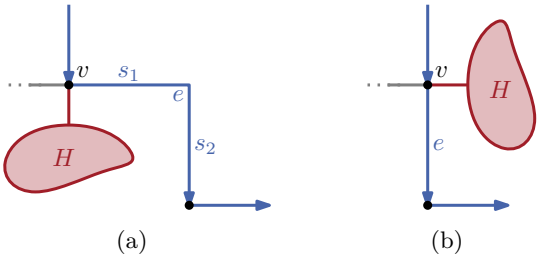


Figure 5.13: Removal of an edge bend in the case that v does not lie on a cycle. (a) The original drawing with a bend on e. (b) The drawing without a bend, which is obtained by exchanging the positions of e and H at v.

Furthermore, we do not need bends on the spine edges:

Lemma 5.13. *For any path P in $\mathcal{T}$ there is a downward P-drawing of G with the minimum number of bends such that no spine edge has a bend.*

Proof. Let Γ be a downward P-drawing of G with the minimum number of bends. Such a drawing exists by Lemma 5.12. If no spine edge of Γ has a bend, Γ is the desired P-drawing. Otherwise, let e be a spine edge with a bend. Without loss of generality we may assume that e has exactly one bend by temporarily replacing all but one bend with a dummy vertex. The bend then divides e into two segments. Exactly one of these segments points down and the other is horizontal. Let v be the endpoint of the horizontal segment. Then, v lies either on exactly one simple cycle or no simple cycle at all. Note that it is impossible that v belongs to two simple cycles as the degree of v would be at least 5.

If v does not lie on any cycle, i.e., v appears as a regular vertex in the cycle tree, we draw e without bends pointing downwards. If e starts at v, we move the subgraph H that is connected to the bottom of v to the old position of e as shown in Figure 5.13. Note that the incoming spine edge of v does not belong to H, since the drawing is downward. If e ends at v, the bend on e can be removed similarly. Hence, H does not contain any essential cycles, and therefore, moving H to the old position of e is possible.

It remains the case that v lies on a simple cycle C. In this case we apply the Cycle Drawing Algorithm and obtain a drawing of C, in which e points downwards and the other spine edge e' incident to C keeps its direction. If e' is not incident to v, the new drawing of C has at most as many bends as the original one by Corollary 5.10. As e can now be drawn without bends, the total number of bends decreases by 1, which contradicts the assumption that the original drawing of G was optimal.

Hence, both e and e' are incident to v. Switching the direction of e from horizontal to vertical therefore changes how C must be drawn at v (cf. Figure 5.14): If there was a turn at v before (i.e., e' pointed down), there cannot be a turn at v now and vice versa. Let T be the set of vertices on C at which there are turns in the original drawing Γ. Bends are represented by dummy vertices. Since the segment of e incident to C is horizontal, T cannot be empty. Hence, Observation 5.8 implies that $|T|$ is even and it is $|T| \geq 4$. By removing v from T or adding v to T (depending on whether T contains v or not) and adding a new dummy vertex w on an arbitrary edge of C we obtain another set of vertices T'. We have $|T'| \geq |T| \geq 4$ and $|T'|$ is even. Therefore, we can use T' as the set of turn vertices in the Cycle Drawing Algorithm, which produces a drawing of C with exactly one bend more than C has in Γ. Since e can now be drawn completely pointing downwards without bends (i.e., one bend less than before), the new drawing of C can be combined to give a drawing of G with the same number of bends as before, but in which the spine edge e does not have a bend. □

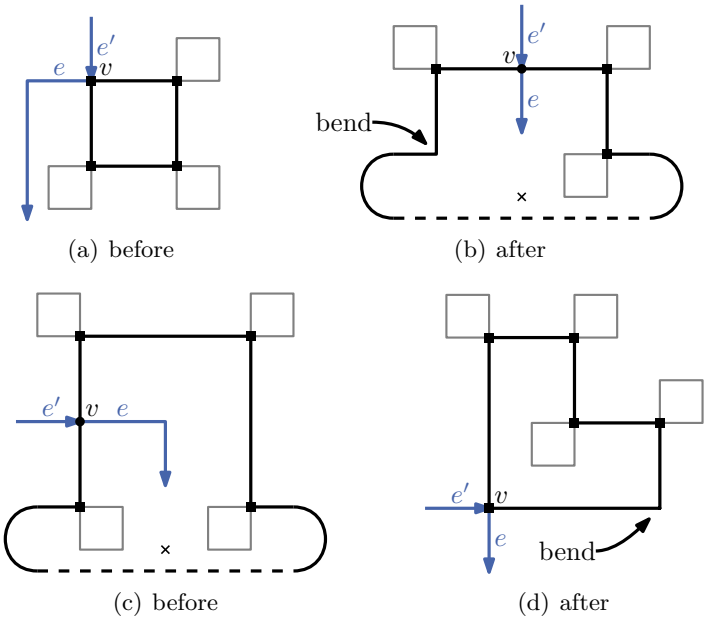


Figure 5.14: Removal of bend on spine edge if v lies on a cycle and is incident to two spine edges e and e'. The bend is moved to an edge of the cycle. (a)–(b) The edge e' points downwards. (c)–(d) The edge e' points left or right.

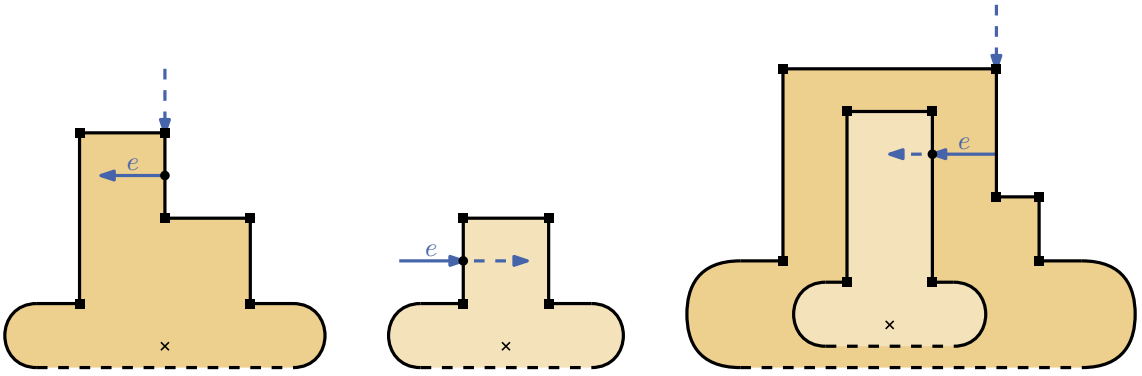
5.3.2 An Algorithm for Bend-Minimal Drawings with a Fixed Spine

Given a downward P-drawing of G with no bends on spine edges, we label the non-virtual edges of P with H if the corresponding spine edge points left or right, or with V if the corresponding edge points downwards. Furthermore, virtual edges are labeled with H. This fits Observation 5.6, which states that if the label is H, the drawings by the Cycle Drawing Algorithm of the two cycles whose cycle vertices are connected by the virtual edge fit together. We call such a path P in $\mathcal{T}$ together with the labeling ℓ of its edges the *spine* (P, ℓ). In the following we describe how a bend-minimal drawing of G can be constructed for a given spine (P, ℓ) with $P = a_1 \dots a_k$. That is, the result must be a P-drawing in which the directions of the spine edges adhere to the labels given by ℓ.

We first draw all parts $H_P(a_i)$ individually: If a_i is a regular vertex, it is simply drawn as one point. Otherwise, a_i is the cycle vertex of a simple cycle C and we use the Cycle Drawing Algorithm. For now, we assume that $i \notin \{1, k\}$, i.e., a_i has two incident edges on P. If $a_{i-1}a_i$ is non-virtual, we set e_{in} to the edge corresponding to $a_{i-1}a_i$ and $d_{\text{in}} = \ell(a_{i-1}a_i)$. Otherwise, let $v \in V(G)$ be the vertex corresponding to $a_{i-1}a_i$ and we pick any edge incident to v that does not lie on C as e_{in}. In this case we set $d_{\text{in}} = H$. Similarly, we define e_{out} and d_{out} by considering a_ia_{i+1}. Applying the Cycle Drawing Algorithm to C, e_{in}, e_{out}, d_{in} and d_{out} then gives an extensible drawing of C. If a_i is the first or the last vertex of P, we use the variant of the Cycle Drawing Algorithm in which the direction of only one edge incident to the cycle is fixed.

In the parts of the $H_P(a_i)$ that do not correspond to a_i we draw all cycles with the variant of the Cycle Drawing Algorithm that produces only non-essential cycles. Vertices that do not belong to a simple cycle are drawn as points. Finally, we compose all parts of $H_P(a_i)$ by adding the remaining edges and possibly rotating some of the parts (but not the one represented by a_i).

Having drawn the subgraphs $H_P(a_i)$ for all $a_i \in V(P)$, we combine them iteratively: The graph $H_P(a_1)$ is drawn as is. If $H_P(a_1)$ to $H_P(a_i)$ have already been combined to a



(a) Γ_i (b) $H_P(a_{i+1})$ (c) Γ_{i+1}

Figure 5.15: An example how the drawings of Γ_i and $H_P(a_{i+1})$ are combined. Only the cycles corresponding to a_i and a_{i+1} are shown and the remainder of Γ_i and $H_P(a_{i+1})$ is omitted. In the combined drawing Γ_{i+1}, the subgraph $H_P(a_{i+1})$ is flipped.

drawing Γ_i, we add $H_P(a_{i+1})$ as follows: If a_ia_{i+1} is non-virtual, we add the edge $e \in E(G)$ corresponding to a_ia_{i+1} to Γ'. We direct it according to the label $\ell(a_ia_{i+1})$. That is, if $\ell(a_ia_{i+1}) = V$, the edge e points downwards, and if $\ell(a_ia_{i+1}) = H$, it points left or right. In the latter case we choose that direction of left and right such that e lies inside the central face of Γ_i as shown, for example, in Figure 5.15(a). We then add the drawing of $H_P(a_{i+1})$ to the end of e. Note that we might need to flip $H_P(a_{i+1})$ vertically (exchanging left and right), since e may point right but $H_P(a_{i+1})$ expects e to point left or vice versa. For instance, for the drawings in Figure 5.15 it is necessary to flip the drawing of $H_P(a_{i+1})$ as e points left in Γ_i but right in $H_P(a_{i+1})$.

If a_ia_{i+1} is virtual, it corresponds to a vertex v that is drawn at a turn in both Γ_i and the drawing of $H_P(a_i)$. Therefore, we can combine these drawings (possibly flipping $H_P(a_{i+1})$ vertically) such that v is drawn at the same position and no edges of Γ_i and $H_P(a_{i+1})$ overlap.

Lemma 5.14. *Given the spine (P, ℓ) of a cactus graph G with maximum degree 4, the algorithm described above draws G respecting the spine such that the drawing has the minimum number of bends among all drawings of G with that spine.*

Proof. Let Γ be the drawing of G produced by the algorithm. By construction, Γ is a P-drawing and all spine edges are directed according to ℓ. Since the Cycle Drawing Algorithm and its variants produce optimal drawings of the cycles (cf. Theorem 5.9) and the edges to combine these drawing have no bends, the resulting drawing Γ also has the minimum number of bends. □

5.4 Finding an Optimal Spine

We have seen how to create drawings with the minimum number of bends for a given spine in the previous section and Lemma 5.13 states that there is a spine for the drawing with the minimum number of bends among all drawings. Hence, it remains to actually find an optimal spine.

5.4.1 Properties of Spines

Studying the properties of spines helps us to develop an efficient algorithm for computing an optimal spine. We denote the subgraph of G that corresponds to a vertex $a \in V(G)$

by H_a, i.e., if a is a cycle vertex for a cycle C, then it is $H_a = C$, and if a is regular, it is $H_a = a$. First, we recall that the number of bends needed by H_a only depends on whether it may be drawn as an essential cycle and—if this is the case—the labels of the incident edges on the spine and whether these edges are incident to the same vertex (cf. Observation 5.11). In particular, the number of bends is independent of the actual spine edges. Therefore, we can regard the number of bends that H_a needs as a function

$$\begin{aligned} b : V(\mathcal{T}) \times \{H, V\}^2 \times \{0, 1\} &\to \mathbb{N}_0, \\ (a, d_{\text{in}}, d_{\text{out}}, \delta) &\mapsto b(a, d_{\text{in}}, d_{\text{out}}, \delta), \end{aligned}$$

where the $\delta = 1$ if and only if the spine edges are incident to the same vertex of H_a. Note that if a is a regular vertex, H_a consists of one vertex and therefore it is $b(a, \cdot, \cdot, \cdot) = 0$.

If H_a may only be drawn as a non-essential cycle, the number of bends $b'(a)$ only depends on a and is described by the function

$$\begin{aligned} b' : V(\mathcal{T}) &\to \mathbb{N}_0, \\ a &\mapsto b'(a). \end{aligned}$$

Furthermore, we define the *weight* of the vertex as

$$w(a, d_{\text{in}}, d_{\text{out}}, \delta) = b'(a) - b(a, d_{\text{in}}, d_{\text{out}}, \delta).$$

Intuitively, the weight of a vertex a describes by how much the number of bends required by H_a can be reduced if a lies on the spine and the two incident spine edges are labeled with d_{in} and d_{out}, respectively. Hence, we want the sum of the weights of the vertices on the spine to be as large as possible.

Consider any edge e between two subgraphs H_a and $H_{a'}$. As G is a cactus graph, e is a bridge, i.e., the removal of e decomposes G in two components. Hence, e does not have bends in a bend-minimal orthogonal drawing. Since orthogonal drawings are equivalent to ortho-radial drawings without essential cycles, we make the following observation:

Observation 5.15. *The number of bends required to draw G without essential cycles is*

$$B' = \sum_{a \in V(T)} b'(a).$$

To calculate the number of bends required by drawings with a given spine, it suffices to know B' and the weight function w only for vertices on the spine:

Lemma 5.16. *Let (P, ℓ) be a spine with $P = a_1 \dots a_k$. For $i = 1, \dots, k-1$ define $\ell_i = \ell(a_i a_{i+1})$ and pick $\ell_0, \ell_k \in \{H, V\}$ arbitrarily. Then, the number of bends required by the spine is*

$$B(P, \ell) = B' - \sum_{i=1}^{k} w(a_i, \ell_{i-1}, \ell_i, \delta_i),$$

where δ_i indicates whether $a_{i-1}a_i$ and $a_i a_{i+1}$ are non-virtual and their corresponding edges in G are incident to the same vertex ($\delta_i = 1$) or not ($\delta_i = 0$), and where $\delta_0 = \delta_k = 0$.

Proof. The number of bends needed by the subgraph of G corresponding to a_i is given by $b(a_i, \ell_{i-1}, \ell_i, \delta_i)$. Note that as a_1 and a_k have only one incident edge on P, the number of bends required to draw their corresponding subgraphs is independent of ℓ_0 and ℓ_k. By the definition of spines all bends are on edges that are part of simple cycles. Hence, the total number of bends $B(P, \ell)$ is the sum of the number of bends of all subgraphs H_a

corresponding to vertices $a \in V(\mathcal{T})$. We can furthermore distinguish between vertices on P and those not on P.

$$\begin{aligned} B(P,\ell) &= \sum_{i=1}^{k} b\left(a_i, \ell_{i-1}, \ell_i, \delta_i\right) + \sum_{a \in V(\mathcal{T}) \setminus V(P)} b'(a) \\ &= \sum_{i=1}^{k} \left(b(a_i, \ell_{i-1}, \ell_i, \delta_i) - b'(a_i)\right) + \sum_{i=1}^{k} b'(a_i) + \sum_{a \in V(\mathcal{T}) \setminus V(P)} b'(a) \\ &= -\sum_{i=1}^{k} w(a_i, \ell_{i-1}, \ell_i, \delta_i) + \sum_{a \in V(\mathcal{T})} b'(a) \end{aligned} \tag{5.8}$$

By Observation 5.15 it is $\sum_{a \in V(\mathcal{T})} b'(a) = B'$ and therefore we obtain

$$B(P,\ell) = B' - \sum_{i=1}^{k} w(a_i, \ell_{i-1}, \ell_i, \delta_i). \tag{5.9}$$

□

Hence, finding the spine that requires the fewest bends is equivalent to finding a spine with the largest weight. In our algorithm to compute this spine we build spines by combining two *partial spines*, which have already been constructed. Formally, a partial spine is a pair (P, ℓ) where $P = c_1 \dots c_k$ with $k \geq 2$ and a labeling ℓ of the edges of P. The difference to ordinary spines is that the first vertex c_1 is not considered to belong to the partial spine, but at the actual end c_2 the labels of two incident edges are fixed. We call the edge c_1c_2 the *connector* of the partial spine. The weight of a partial spine is given by the sum of the weights of its vertices except c_1, i.e., the weights of $c_2, \dots, c_k$. The subgraphs of G corresponding to the connector c_1c_2 and the vertex c_1 intersect at exactly one vertex, which we call the *port* of the partial spine. Partial spines with the same port are called *partners*.

Lemma 5.17. *Let $a \in V(\mathcal{T})$ be a cycle vertex and S a partial spine with connector ab. Let $\mathcal{C}$ be the set of all connectors of partners of S. Then, it is $|\mathcal{C}| \leq 2$.*

Proof. Let $v \in V(G)$ be the port of S. If ab is virtual, it corresponds to v. As G has a maximum degree of 4, all edges incident to v lie on a simple cycle. Hence, none of them corresponds to an edge of $\mathcal{T}$. Moreover, only ab corresponds to v. Therefore, all partners of S must have ab as a connector.

If ab is non-virtual, it corresponds to an edge $vw \in E(G)$, where v is the port of S. There is at most one other edge incident to v that does not belong to any simple cycle. If such an edge exists, it corresponds to an edge $ac \in E(\mathcal{T})$. Moreover, v only lies on one simple cycle and therefore there is no virtual edge for v in $\mathcal{T}$. Hence, any partner of S can only have ab or ac as connector. □

5.4.2 A Dynamic Program for the Heaviest Spine

The algorithm we use to find a heaviest spine in $\mathcal{T}$ combines a dynamic programing approach with a post-order traversal of $\mathcal{T}$. That is, all children of a vertex are handled before the vertex itself is dealt with. We describe the algorithm and prove that it works correctly in this section. Section 5.4.3 then provides some more implementation details, which are necessary to achieve a linear running time of the algorithm. Figure 5.16 contains an example of a cactus graph and its cycle tree, which we use to illustrate the algorithm.

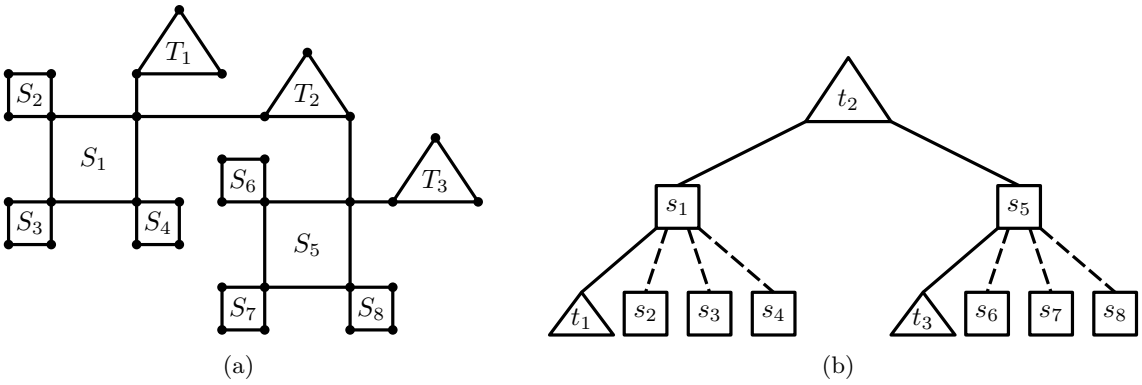


Figure 5.16: (a) An example of a cactus graph. (b) The corresponding cycle tree rooted at t_2. The dashed edges are virtual.

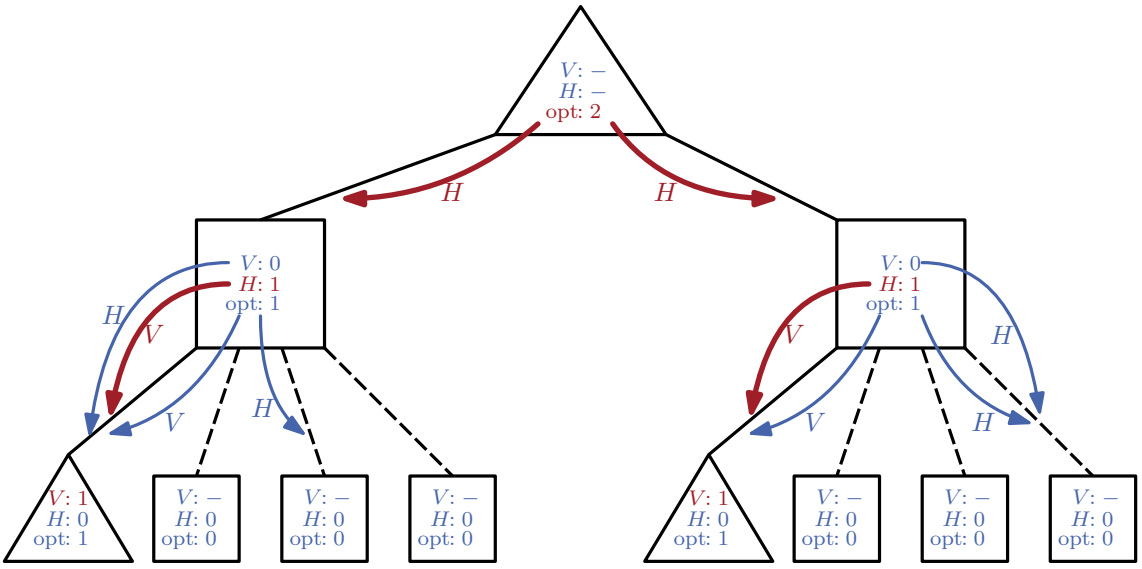


Figure 5.17: The complete result of the dynamic program for the graph in Figure 5.16(a). Inside the vertices the weights of the (partial) spines are shown in the following order: The weight of the partial spine whose connector is labeled with V, the weight of the partial spine whose connector is labeled with H, and the weight of the spine with the vertex as its highest vertex. The arrows point to the next element of the spines. The parts that belong to the optimal spine are shown in red.

For each vertex $a \in V(\mathcal{T})$ we calculate up to three (partial) spines, which we store at a: First, the spine (P_a, ℓ_a) with a as highest vertex and maximum weight is determined, which is the spine we are ultimately interested in. Unless a is the root, we furthermore compute the following two partial spines, where p denotes the parent of a: The partial spine (Q_a^H, ℓ_a^H) is a heaviest partial spine with pa as connector and $\ell_a^H(pa) = H$. Similarly, (Q_a^V, ℓ_a^V) is a heaviest partial spine with pa as connector and $\ell_a^V(pa) = V$. If pa is virtual, only the first partial spine exists since virtual edges must be labeled with H by definition. In this case we ignore the second partial spine.

Note that in general there may be several such spines with maximum weight and we can choose any of them. For instance, consider the left child s_1 of the root in Figure 5.17. The heaviest spine with s_1 as highest vertex as indicated by the arrows contains s_1 and two children of s_1: the triangle t_1 and one square s_2. But since s_2, s_3 and s_4 all have the same properties, s_2 could be replaced by any of the other two vertices without changing the weight.

We determine these three spines based on the partial spines stored at the children of a as follows: For two partial spines (Q_1, ℓ_1) and (Q_2, ℓ_2) stored at any two distinct children c_1 and c_2 of a, we define a spine (P'_a, ℓ'_a) by $P'_a = Q_1 + \overline{Q_2}$ and the edges keep their labeling as defined by ℓ_1 and ℓ_2. That is, we simply join two partial spines together to get a spine. We then pick one of these spines that has the maximum weight as (P_a, ℓ_a). If a has only one child, we add a to all partial spines at children of a and choose the heaviest resulting spine as (P_a, ℓ_a). If a is a leaf, the only possible spine contains solely the vertex a and no edges.

Similarly, each of the partial spines at a child c with connector ac can be extended to two partial spines with pa as connector, where p is the parent of a in $\mathcal{T}$: one for each possible label H and V at pa. We choose the heaviest such partial spines as the partial spines stored at a.

After the spines (P_a, ℓ_a) for all vertices of $\mathcal{T}$ have been computed, we select one of these spines that has the maximum weight. If there are several possibilities, we may choose any.

Theorem 5.18. *Any optimal drawing of the cactus graph G with the spine (P, ℓ) computed by the algorithm above contains the minimum number of bends among all ortho-radial drawings of G.*

Proof. Recall that spines are only defined for downward drawings of G that have no bends on the spine. Lemma 5.13 states that there is such a drawing with the minimum number of bends among all ortho-radial drawings of G, and hence, finding an optimal spine is sufficient to obtain a bend-minimal drawing. Moreover, minimizing the number of bends required by the spines is equivalent to finding a spine with the maximum weight by Lemma 5.16. Hence, it remains to show that the spine (P, ℓ) produced by the algorithm has the maximum weight.

We first prove by induction on the structure of $\mathcal{T}$ that for each vertex $a \in V(\mathcal{T})$ the (partial) spines (P_a, ℓ_a), (Q_a^H, ℓ_a^H) and (Q_a^V, ℓ_a^V) are indeed the heaviest such spines. If a is a leaf, there is exactly one choice for each of the three spines, which therefore have the maximum weight.

Let $a \in V(\mathcal{T})$ be a non-leaf of $\mathcal{T}$ and assume as induction hypothesis that the partial spines calculated for its children have the maximum weight. Let (P_a^*, ℓ_a^*) be a heaviest spine with a as highest vertex and maximum length. Then, it is $a \in P_a^*$ and a either lies in the middle of P_a^* or at an end. In the former case splitting (P_a^*, ℓ_a^*) at a yields two partial spines for two children of a. Since the partial spines computed for the children are optimal by induction hypothesis, checking the spines formed by all pairs of partial spines finds a spine with the maximum weight.

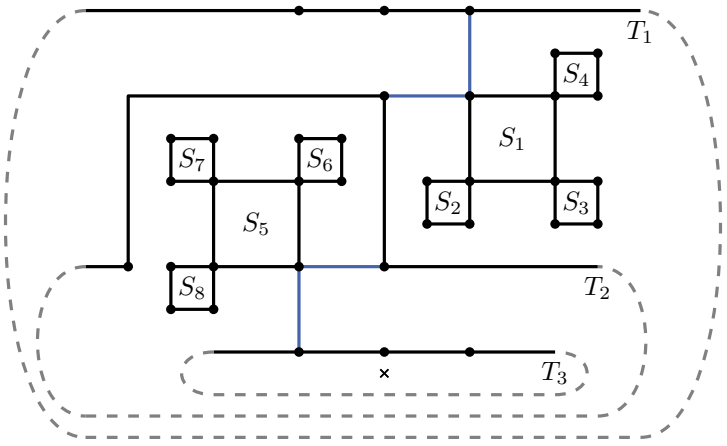


Figure 5.18: The bend-minimal drawing of the graph in Figure 5.16(a) with the spine that was computed by the dynamic program (cf. figure 5.17). The only bend in this drawing lies on T_2.

If a lies at one end of P_a^*, a has only one child, since P_a^* is maximal. Otherwise, P_a^* could be extended by adding another child of a. As all weights are non-negative, the weight of a spine increases or stays the same if it is extended. Clearly, adding a to all partial spines for the children finds an optimal spine. A similar argument shows that the partial spines (Q_a^H, ℓ_a^H) and (Q_a^V, ℓ_a^V) computed by the algorithm are optimal.

Let (P^*, ℓ^*) be a spine with maximum weight and a^* the highest vertex of P^* in $\mathcal{T}$. In particular, we have $w(P^*, \ell^*) \geq w(P, \ell)$, where (P, ℓ) is the spine computed by the algorithm. By the argument above the spine (P_{a^*}, ℓ_{a^*}) stored at a^* has the maximum weight of all spines with a^* as highest vertex. Therefore, it is

$$w(P, \ell) \geq w(P_{a^*}, \ell_{a^*}) \geq w(P^*, \ell^*) \geq w(P, \ell) \tag{5.10}$$

and equality holds. Thus, the algorithm correctly determines a spine with the maximum weight. □

After we found an optimal spine (P, ℓ), the whole graph can be drawn by repeated application of the Cycle Drawing Algorithm and its variants as shown in Section 5.3. Figure 5.18 shows the resulting drawing of the graph in Figure 5.16(a) with the spine computed by the dynamic program.

5.4.3 Implementation in Linear Time

While the algorithm in the previous section correctly produces a spine with maximum weight, it requires some care to implement it in linear time.

Computing and storing the spines explicitly for each vertex a of $\mathcal{T}$ requires time and storage linear to the height of a. As all spines are formed by one or two existing partial spines, it suffices to store pointers to them. Hence, each vertex only needs to store four pointers: two for the spine with a as highest vertex and one for each partial spine. Additionally, we store the weights of the spines. In Figure 5.17 the weights are written inside the vertices: first the weight of the partial spine whose connector is labeled with H, then the one with a connector that is labeled with V, and finally the weight of the spine with a as highest vertex. The pointers are represented by arrows. The label at an arrow from a vertex p to one of its children a determines the label of pa and therefore also which partial spine at a forms the rest of the spine.

Moreover, checking all pairs of partial spines when computing the heaviest spine at a vertex is both expensive and unnecessary, since it is sufficient to consider only a constant number of pairs. To prove this, we first analyze how the number of bends $b(a, d_{\text{in}}, d_{\text{out}}, \delta)$ required to draw the subgraph H_a of G corresponding to a vertex $a \in V(\mathcal{T})$ depends on the last parameter δ, which specifies whether the two neighboring spine edges incident to the same vertex of H_a. If this is the case, the shape of H_a is fixed at that vertex and therefore there may only be more but not fewer bends than if the two spine edges end at two different vertices of H_a.

Observation 5.19. *For any $a \in V(\mathcal{T})$ and $d_{in}, d_{out} \in \{H, V\}$ it is*

$$b(a, d_{in}, d_{out}, 0) \leq b(a, d_{in}, d_{out}, 1), \text{ and}$$
$$w(a, d_{in}, d_{out}, 0) \geq w(a, d_{in}, d_{out}, 1).$$

With this observation we are prepared to show that only six partial spines must be considered.

Lemma 5.20. *Let $S_1^H, \ldots, S_3^H$ be the three heaviest partial spines for children of a whose connectors are labeled with H. Similarly, let $S_1^V, \ldots, S_3^V$ be the three heaviest partial spines for children of a whose connectors are labeled with V. Then, at least one of the spines formed by two of these partial spines has the maximum weight among all spines with a as highest vertex.*

Proof. Let $\mathcal{S}^H = \{S_1^H, S_2^H, S_3^H\}$, $\mathcal{S}^V = \{S_1^V, S_2^V, S_3^V\}$ and $\mathcal{S} = \mathcal{S}^H \cup \mathcal{S}^V$. We furthermore assume without loss of generality that S_1^H and S_1^V are the heaviest partial spines whose connector is labeled with H and V, respectively.

If a is a regular vertex, it is $w(a, \cdot, \cdot, \cdot) = 0$ and therefore choosing the heaviest partial spine S_1 and the next heaviest partial spine S_2 that is not stored at the same vertex as S_1 gives the heaviest spine with a as highest vertex. Clearly, $S_1, S_2 \in \mathcal{S}$ and the claim holds. Hence, we may assume in the following that a is a cycle vertex.

Consider any heaviest spine S^* with a as the highest vertex. Splitting S^* at a gives two partial spines S_1^* and S_2^*, whose connectors are labeled with d_1 and d_2, respectively. By Theorem 5.18 we may assume that both S_1^* and S_2^* are partial spines that are stored at children of a. We show in the following that there are two partial spines in $\mathcal{S}$ such that merging them yields a spine with the same weight as S^*, which is maximum by the choice of S^*.

If both partial spines belong to $\mathcal{S}$, the claim clearly holds. Otherwise, at least one of the two parts of S^*, say S_2^*, does not belong to $\mathcal{S}$. Let $S_1 = S_1^{d_1}$ be the heaviest spine of $\mathcal{S}^{d_1}$. Let δ be a boolean variable denoting whether S_1^* and S_2^* are partners ($\delta = 1$) or not ($\delta = 0$). By definition, we have $w(S_1) \geq w(S_1^*)$. At least one of the elements of $\mathcal{S}^{d_2}$ is not a partner of S_1, since S_1 has at most two partners in $\mathcal{S}^{d_2}$ by Lemma 5.17. We denote the heaviest such spine in $\mathcal{S}^{d_2}$ by S_2. As $S_2^* \notin \mathcal{S}^{d_2}$, we have $w(S_2) \geq w(S_2^*)$. Combining S_1 and S_2 yields a spine S. Together with $w(a, d_1, d_2, 0) \geq w(a, d_1, d_2, \delta)$ from Observation 5.19 we obtain

$$\begin{aligned} w(S^*) &= w(S_1^*) + w(a, d_1, d_2, \delta) + w(S_2^*) \\ &\leq w(S_1) + w(a, d_1, d_2, 0) + w(S_2) = w(S). \end{aligned} \tag{5.11}$$

Hence, we found a combination S of two partial spines in $\mathcal{S}$ with the maximum weight $w(S) = w(S^*)$. □

With these two modifications the algorithm to compute the heaviest spine from the previous section runs in linear time.

Theorem 5.21. *A spine with maximum weight can be computed in $\mathcal{O}(|G|)$ time and $\mathcal{O}(|\mathcal{T}|)$ space.*

Proof. A vertex $a \in V(\mathcal{T})$ that corresponds to a subgraph H_a of G can be handled in $\mathcal{O}(\deg(v) + |H_a|)$ time: Each value of $b(a, \cdot, \cdot, \cdot)$ and $b'(a)$ can be calculated by applying the Cycle Drawing Algorithm to H_a once for each value. This means nine invocations of the Cycle Drawing Algorithm, each taking $\mathcal{O}(|H_a|)$ time. Hence, this step needs $\mathcal{O}(|H_a|)$ time in total and storing the values needs a constant amount of space.

In order to compute the spine (P_a, ℓ_a) it suffices to find the three heaviest partial spines stored at children of a for each of the labels (cf. Lemma 5.20). These are found by checking all partial spines stored at children of a, which needs $\mathcal{O}(\deg(a))$ time. After that a constant number of combinations is tested. To find the partial spines, a is added to all partial spines at the children. This again needs $\mathcal{O}(\deg(a))$ time. These three (partial) spines are represented by four pointers using constant space.

After all vertices have been handled, the heaviest spine is determined by one traversal of the tree in $O(|\mathcal{T}|)$ time. In total, running time of the algorithm is

$$\mathcal{O}\left(|\mathcal{T}| + \sum_{a \in V(T)} (\deg(a) + |H_a|)\right) = \mathcal{O}\left(|\mathcal{T}| + |G|\right) = \mathcal{O}(|G|),$$

where the last equality follows from $|\mathcal{T}| \in \mathcal{O}(|G|)$. Furthermore, the algorithm needs $\mathcal{O}(1)$ space per vertex of $\mathcal{T}$ and therefore $\mathcal{O}(|\mathcal{T}|)$ space in total. □

6. Conclusion

In this chapter we summarize our results and give an outlook over open questions related to ortho-radial graph drawing.

6.1 Summary

We studied ortho-radial drawings of 4-planar graphs, which are an extension of orthogonal drawings in the plane to the cylinder. As a means to describe the shape of these drawings, we presented ortho-radial representations, which include orthogonal representations as a special case. Furthermore, we characterized those representations of 4-planar graphs that can actually be drawn by a set of conditions and called the representations that satisfy them valid. One important tool for this were labelings of essential cycles. By requiring consistent labelings for all essential cycles such that each labeling attains both positive and negative values (or is identically 0) it is ensured that all essential cycles can be drawn together without any conflicts.

Being able to describe the shape of ortho-radial drawings without actually fixing coordinates or edge lengths is a crucial prerequisite for applying Tamassia's Topology-Shape-Metrics framework to ortho-radial graph drawing. Moreover, our characterization is constructive in the sense that we show a way to produce drawings for valid ortho-radial representations.

We furthermore studied the bend minimization problem for ortho-radial drawings. We proved that in general when no embedding is fixed determining whether a 4-planar graph admits an ortho-radial drawing without bends is $\mathcal{NP}$-complete. Therefore, assuming $\mathcal{P} \neq \mathcal{NP}$, there is no polynomial time algorithm for bend minimization of general 4-planar graphs. For the restricted class of cactus graphs however we presented a linear time algorithm for bend minimization.

6.2 Future Work

The rectangulation algorithm presented in Section 3.5 checks for increasing or decreasing cycles. Clearly, one can compute the labelings of all essential cycles and search for positive and negative labels. However, there may be an exponential number of essential cycles, which makes this approach expensive. Even if the graph initially only has few essential cycles, we insert additional edges during the rectangulation, which may increase the number of essential cycles. Therefore, an efficient test for increasing and decreasing cycles would be desirable.

When devising such an algorithm one needs to keep in mind that the same edge can have different labels for different cycles. Therefore, it is not simply possible to first label all edges and then search for cycles with only non-negative or non-positive labels. Instead, one probably needs to keep track of multiple labels per edge. Furthermore, there may be two essential cycles that differ only at a few edges such that one of these cycles is increasing or decreasing and the other is not. Therefore, cycles cannot be excluded easily because most of the edges have already been part of a cycle that has been checked.

Working towards the application of the Topology-Shape-Metrics framework to ortho-radial graph drawing, we dealt with the questions how to represent the shape and then given a shape how to construct a drawing. But we did not work on the first part of the framework: how to find a good embedding and how to find a shape with the minimum number of bends. In the case of orthogonal drawings in the plane, the number of bends can be minimized by solving a flow network [Tam87]. But in the ortho-radial case the resulting representation must have neither in- nor decreasing cycles. This condition however requires non-local information and therefore it seems to be hard to capture this requirement in a flow network.

Another direction of research is to try to extend the ortho-radial representation to orthogonal drawings on a torus, where the second dimension is cyclical as well. We expect both cyclical dimensions of the torus to behave similar to the cyclical dimension of the cylinder. Locally the drawings on a torus and a cylinder are equal. Hence, all parts of the representation that describe the local shape can be adopted. But there is in general no outer or central face and the rotation of all faces is 4, unless actually an ortho-radial drawing is described. It is however unclear how a condition similar to the last one of Definition 3.3, which requires valid labelings of all essential cycles, can be formulated.

As there are two cyclical dimension on a torus, there are more kinds of essential cycles: We call a cycle an (x, y)-*essential cycle* if it winds itself x times around the first and y times around the second dimension. The only types of cycles that can occur are $(x, 1)$-essential, $(1, y)$-essential and non-essential cycles. We conjecture that only $(0, 1)$- and $(1, 0)$-essential cycles need to be considered when formulating labelings and conditions on them, as all other types of essential cycles seem to be always drawable.

Bibliography

[Bar16] Lukas Barth. Drawing Metro Maps on Concentric Circles. Master's thesis, Fakultät für Informatik, Karlsruher Institut für Technologie (KIT), 2016.

[BBR14] Thomas Bläsius, Guido Brückner, and Ignaz Rutter. *Complexity of Higher-Degree Orthogonal Graph Embedding in the Kandinsky Model*, pages 161–172. Springer Berlin Heidelberg, 2014.

[BK98] Therese Biedl and Goos Kant. A better heuristic for orthogonal graph drawings. *Computational Geometry: Theory and Applications*, 9:159–180, 1998.

[BKRW14] Thomas Bläsius, Marcus Krug, Ignaz Rutter, and Dorothea Wagner. Orthogonal graph drawing with flexibility constraints. *Algorithmica*, 68(4):859–885, 2014.

[BLR16] Thomas Bläsius, Sebastian Lehmann, and Ignaz Rutter. Orthogonal graph drawing with inflexible edges. *Computational Geometry*, 55:26–40, 2016.

[CK12] Sabine Cornelsen and Andreas Karrenbauer. Accelerated bend minimization. *Journal of Graph Algorithms and Applications*, 16(3):635–650, 2012.

[CLRS01] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, 2nd edition, 2001.

[DBETT99] Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. *Graph Drawing - Algorithms for the Visualization of Graphs*. Prentice Hall, 1999.

[dBK12] Mark de Berg and Amirali Khosravi. Optimal binary space partitions for segments in the plane. *International Journal of Computational Geometry and Applications*, 22(03):187–205, 2012.

[DBLV98] Giuseppe Di Battista, Giuseppe Liotta, and Francesco Vargiu. Spirality and optimal orthogonal drawings. *SIAM Journal on Computing*, 27(6):1764–1811, 1998.

[FLW14] Martin Fink, Magnus Lechner, and Alexander Wolff. Concentric metro maps. In *Schematic Mapping Workshop 2014*, 2014.

[GT95] Ashim Garg and Roberto Tamassia. On the Computational Complexity of Upward and Rectilinear Planarity Testing. In *DIAMCS International Workshop*, volume 894 of *Lecture Notes in Computer Science*, pages 286–297. Springer, January 1995.

[HHM09] Mahdie Hasheminezhad, S. Mehdi Hashemi, and Brendan D. McKay. Spherical-rectangular drawings. In *International Workshop on Algorithms and Computation*, pages 345–356. Springer, 2009.

[HHMT10] Mahdie Hasheminezhad, S. Mehdi Hashemi, Brendan D. McKay, and Maryam Tahmabasi. Rectangular-radial drawings of cubic plane graphs. *Computational Geometry*, 43(9):767–780, 2010.

[HHT09] Mahdie Hasheminezhad, S. Mehdi Hashemi, and Maryam Tahmabasi. Ortho-radial drawings of graphs. *Australasian Journal of Combinatorics*, 44:171–182, 2009.

[RNC16] Maxwell J. Roberts, Elizabeth J. Newton, and Maria Canals. Radi (c) al departures: Comparing conventional octolinear versus concentric circles schematic maps for the berlin u-bahn/s-bahn networks using objective and subjective measures of effectiveness. *Information Design Journal*, 2016.

[Tam87] Roberto Tamassia. On Embedding a Graph in the Grid with the Minimum Number of Bends. *SIAM Journal on Computing*, 16(3):421–444, 1987.

[TT91] Roberto Tamassia and Ioannis G. Tollis. Representations of graphs on a cylinder. *SIAM Journal on Discrete Mathematics*, 4(1):139–149, 1991.