Technische Universiteit Eindhoven University of Technology
Department of Mathematics and Computer Science
Applied Geometric Algorithms Research Group

k-Sided Rectangular Duals

Sander Beekhuis

Supervisors:
prof.dr. B. Speckman
dr. K.A.B. Verbeek
dr J. Nederlof

Version Draft 1.5

Eindhoven, Feb 2017

Abstract

A *rectangular layout* (or simply *layout*) $\mathcal{L}$ is a partition of an axis-parallel rectangle into a finite set of interior-disjoint axis-parallel rectangles. Hence, the interior of this rectangle contains vertical and horizontal line segments. We will call any such line segment that is not extended any farther on either side a *maximal segment*. Such a layout is *one-sided* if every maximal segment has only one rectangle on one of its sides and *k-sided* if every maximal segment has at most k rectangle on one of its sides.

All graphs in this thesis will be *triangulations of the k-gon*. They have an outer face of degree k and interior faces of degree 3. Vertices bordering the outer face are *outer vertices* while all other vertices are *interior vertices*.

Two vertices are *adjacent* when they are connected by an edge. Two rectangles are *adjacent* when their boundaries overlap. A *rectangular dual* of G is a rectangular layout whose adjacencies are the same as those of G for a bijection between vertices and rectangles.

A *corner assignment* $\bar{G}$ of G is an augmentation of G with 4 vertices (which we call its *poles*). Such that every interior face has degree 3, the exterior face has degree 4 and all poles are incident to the outer face

It is known that a triangulation of the k-gon $\mathcal{G}$ has a rectangular dual if and only if it has a corner assignment without separating triangles $\bar{\mathcal{G}}$ A graph G can have multiple rectangular duals. G can even have duals that are not equivalent.

A graph is k-sided when it has a k-sided rectangular dual. This thesis presents two results on k-sided graphs.

1. There is a family of graphs G_k that for any $k \in \mathbb{N}$ has members that are not k-sided (Theorem 5).

2. Graphs G that have a corner assignment without separating 4-cycles are $d-1$-sided, where d is the maximal degree of the vertices of G. (Theorem 8)

Contents

1 Introduction

Rectangular layout. A *rectangular layout* (or simply *layout*) $\mathcal{L}$ is a partition of an axis-parallel rectangle into a finite set of interior-disjoint axis-parallel rectangles. Hence, the interior of this rectangle contains vertical and horizontal line segments. We will call any such line segment that is not extended any farther on either side a *maximal segment*. Such a layout is *one-sided* if every maximal segment has only one rectangle on one of its sides and *k-sided* if every maximal segment has at most k rectangles on one of its sides.

We say two layouts are *combinatorially equivalent* or simply *equivalent* when their rectangles have the same adjacencies with the same orientation(horizontal or vertical) between their rectangles.

Consider for example Figure 1a, the three highlighted lines are all line segments. However, only the red and blue segment are maximal segments. The red segment is one-sided and the blue segment is 2-sided and the whole layout is 2-sided. Furthermore, the layout in Figure 1b is 4-sided.

Graphs. A *graph* G is an abstraction of a network. The objects are represented by a set of *vertices*. Connections between objects are represented by a set of *edges*; Each edge connects two vertices. In all graphs every pair of vertices is connected by at most one edge and there are no edges starting and ending at the same vertex. That is, all graphs in this thesis are *simple*. An edge is *incident* to a vertex v if that edge connects v to another vertex. The *degree* of a vertex is the number of edges incident to this vertex. All graphs in this thesis are *planar*. That is, they can be embedded in the plane without their edges crossing. A *face* is connected component of the maximal subset of the plane that is disjoint from the embedded graph. The *degree* of a face is the number of vertices on its boundary. A face of degree 3 is a *triangular* face. The *outer face* is the one and only unbounded face. A vertex is *incident* to a face when it lies on its boundary.

All graphs in this thesis will be *triangulations of the k-gon*. A triangulation of the k-gon has an outer face of degree k and interior faces of degree 3. Vertices bordering the outer face are *outer vertices* while all other vertices are *interior vertices*. Triangulations of the k-gon are called *(plane) triangulated graphs* by some other authors.

Rectangular duals. Two vertices are *adjacent* when they are connected by an edge. Two rectangles are *adjacent* when their boundaries overlap. A *rectangular dual* of G is a rectangular layout whose adjacencies are the same as those of G for a bijection between vertices and rectangles.

Corner assignments. If we want to determine which graphs do have a

(a) A rectangular layout. (b) Another rectangular layout.

Figure 1: Rectangular layouts

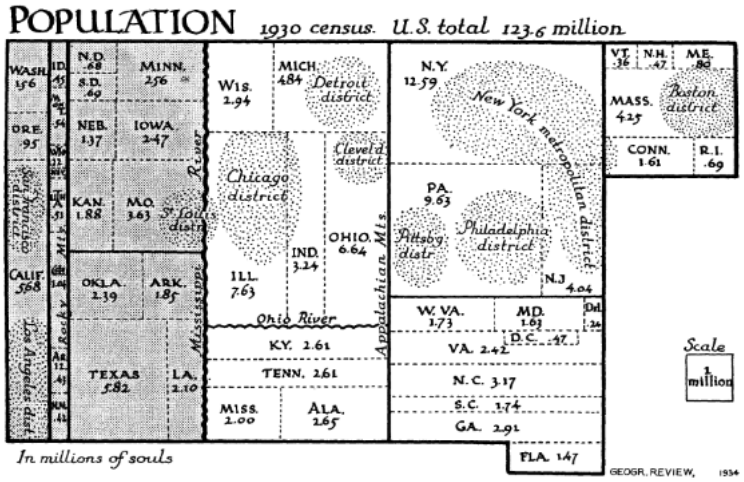


Figure 2: A cartogram by Raisz made in 1934.

rectangular dual, then we need to introduce the notion of a *corner assignment*. A corner assignment $\bar{G}$ of G is an augmentation of G with 4 vertices (which we call its *poles*). Such that every interior face has degree 3, the exterior face has degree 4 and all poles are incident to the outer face

A corner assignment of G only exists if G is a triangulation of the k-gon for some k. Otherwise, there is no way of adding poles that makes all the interior faces of degree 3. Because of this, we only consider triangulations of the k-gon in this thesis. A corner assignment $\bar{G}$ of G is an example of a triangulation of the 4-gon. Each corner assignment fixes which rectangles are in the corners of the rectangular dual $\mathcal{L}$, which explains the terminology.

Existence and uniqueness. Now we can state which graphs admit a rectangular dual. The following result was shown independently by Kozminski and Kinnen [6] and Ungar [8]

Theorem 1 (Existence of a rectangular dual)**.** *A triangulation of the k-gon $\mathcal{G}$ has a rectangular dual if and only if it has a corner assignment without separating triangles $\bar{\mathcal{G}}$*

A graph G can have multiple rectangular duals. G can even have duals that are not equivalent. An example is given by the two non-equivalent duals of the same graph given in Figure 1.

Rectangular cartograms.

In for example atlases *rectangular cartograms* are used to display information. A rectangular cartogram is a map where the regions are replaced by rectangles while keeping their adjacencies. An example of such a cartogram for population of the state of the United States of America is given in Figure 2. The size of each region can change according to the variable or moment in time displayed in the cartogram. A rectangular cartogram is the rectangular dual of the adjacency graph of the map G. If the areas change it might be that a certain rectangular layout can not fulfill its adjacencies anymore and we have to switch to another non-equivalent rectangular dual of G.

We would like to find a rectangular dual that has adjacencies that hold

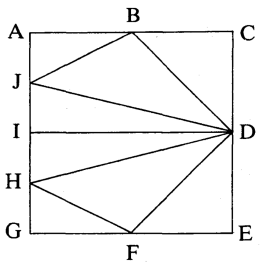


Figure 3: The graph by Rinsma [7] that is not one-sided.

regardless of the area sizes we assign to each rectangle. We say such a dual is *area-universal*. Eppstein et al. have shown that rectangular duals are area-universal exactly when they are one-sided. [1] Unfortunately not all graphs admit a one-sided dual. One such graph, displayed in Figure 3, is given by Rinsma. [7]

Results. Unfortunately k-sided layouts for $k > 1$ are not area-universal but we suspect that for small k they are more robust to changes in the areas of their rectangles. This thesis has two results on k-sidedness.

1. There is a family of graphs G_k that for any $k \in \mathbb{N}$ has members that are not k-sided (Theorem 5).

2. Triangulations of the k-gon G that have a corner assignment without separating 4-cycles are $d-1$-sided, where d is the maximal degree of the vertices of G. (Theorem 8)

Paths and Cycles. In the rest of the thesis we will frequently need paths and cycles, hence we will definite them here.

A path $\mathcal{P}$ is a sequence of vertices such that every two consecutive vertices are connected by an edge. The first and last vertex of the path are its *extreme* vertices while the rest are *internal* vertices of this path. The *length* of a path is the number of edges used to connect the vertices. That, is one less than the number of vertices. In this thesis all paths are *simple*, that is, no vertex occurs twice in the path except possibly the extreme vertices.

A cycle is a path whose extreme vertices coincide. Because a cycle is a path the start and end vertex are the only vertices that occurs more than once. We call a cycle of length k a *k-cycle*. A *triangle* is cycle of length 3 (i.e. a 3-cycle). By Jordan's curve theorem a cycle splits the plane into two parts, one bounded and one unbounded. We will call the bounded part the *interior* of this cycle and the unbounded part the *exterior* of this cycle. Furthermore, the cycle of all vertices bordering the outer face is the *outer cycle*. We call a cycle *separating* if there are vertices in both its interior and exterior. An *interior edge* of a cycle is then an edge contained in the interior of the cycle. An *interior path* is a path connecting two distinct vertices off the cycle and whose edges are interior edges.

Overview. In Section 2 we will introduce the notion of regular edge labellings which we will use in the rest of the thesis. In Section 3 we show Theorem 5 and in Section 4 we show Theorem 8.

2 Regular edge labellings

Given a layout $\mathcal{L}$ we can easily find its adjacency graph and thus for which graph G it is a rectangular dual. However, finding a rectangular dual of a graph G is more involved. Due to the algorithm by He [4] it is sufficient to find a *regular edge labeling* on a corner assignment of G. In this section we will introduce regular edge labellings.

Adjacency graphs of layouts. The *adjacency graph* $\mathcal{G}(\mathcal{L})$ of a layout $\mathcal{L}$ each rectangle is represented by a vertex and we connect two vertices by an edge exactly when their rectangles are adjacent. In the *extended adjacency graph* $\mathcal{G}_{\mathcal{E}}(\mathcal{L})$ we also add 4 vertices $\mathrm{N}, \mathrm{E}, \mathrm{S}, \mathrm{W}$ (so-called *poles*) in the outer face, one associated to the north, east, south, west boundary segment of the outer rectangle, respectively. Two vertices are then connected if their rectangles or boundary segments intersect. If we take the *extended adjacency graph* of a layout and remove the 4 vertices corresponding to the outer face we end up with the regular *adjacency graph* of that layout. In this setting a layout $\mathcal{L}$ is a *rectangular dual* of a graph G if we have that $G = \mathcal{G}(\mathcal{L})$.

Regular edge labellings. Regular edge labellings were first introduced by Kant and He [5] and were also used in [1]. Fusy also studied these structures [3, 2] under the name of *transversal structures*. A *regular edge labeling* is a coloring and orientation of the edges of the extended adjacency graph $\mathcal{G}_{\mathcal{E}}(\mathcal{L})$. This coloring and orientation is given by the following procedure. For every edge vw in $\mathcal{G}_{\mathcal{E}}(\mathcal{L})$ we consider whether the shared boundary of the rectangles is vertical or horizontal we then color the edge blue or red respectively. In the first case we orient the edge from the leftmost rectangle to the rightmost rectangle and in the second case we orient from bottom to top. We neither color nor orient the edges between the poles.

From the nature of the adjacencies in a rectangular layout we can deduce the following two rules for a regular edge labeling.

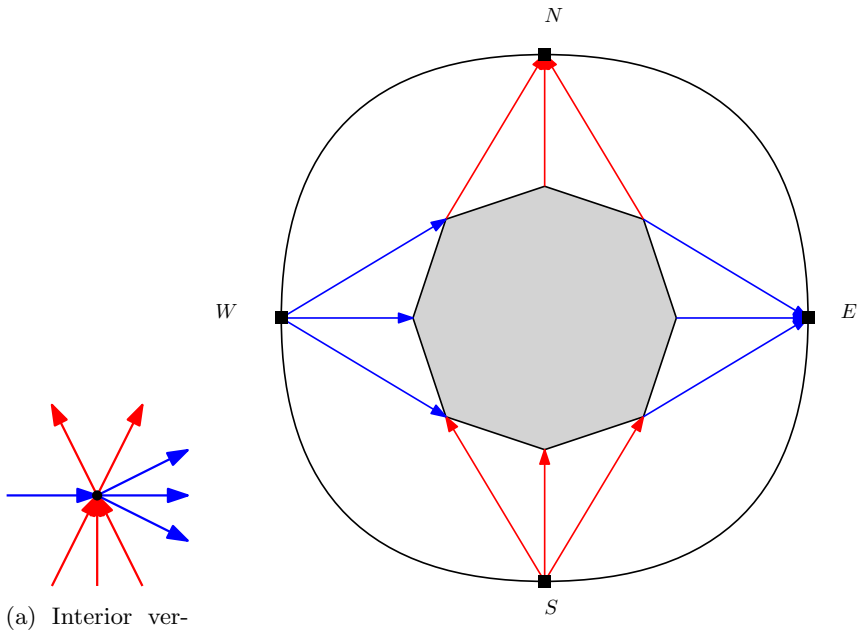
1. (Interior vertex) In the rotation around every interior vertex we have the following subsequent non-empty sets: Incoming red edges, incoming blue edges, outgoing red edges and outgoing blue edges and only these sets.

2. (Pole) N has only incoming red edges, E has only incoming blue edges, S has only outgoing red edges and W has only outgoing blue edges ,except for the uncolored edges between the poles.

Both rules are illustrated in 4.

In [4] He showed that given a regular edge labeling of a corner assignment we can reconstruct a rectangular layout represented by this regular edge labeling. A regular edge labeling of $\bar{G}$ corresponds to an equivalence class of rectangular layouts $\mathcal{L}$ that are a rectangular dual of G.

Properties Since we use regular edge labellings a lot in this thesis we will show some properties for them. We show that a regular edge labeling has no mono-colored triangles and that the red and blue subgraph of a regular edge labeling both form a *st*-planar graph. Before we can show this, we need to introduce the rotation at a vertex. For a fixed embedding for G the *rotation* at a vertex v is the clockwise order of the edges incident to v. We will identify these



(a) Interior vertex condition

(b) Pole condition

Figure 4: Regular edge labeling conditions

edges with their other endpoints. Two vertices x, y are said to be *consecutive* in the rotation at v when the edges vx and vy are consecutive in the rotation.

Lemma 2. *A regular edge labeling has no monochromatic triangles even when ignoring orientation*

Proof. Suppose we have a monochromatic triangle. Without loss of generality we suppose this triangle is blue. Then at least one of the vertices has an incoming blue edge followed directly by an outgoing blue edge or an outgoing blue edge followed directly by an incoming blue edge in its rotation. Thus, this vertex has either an empty set of outgoing or incoming red edges and hence violates the interior vertex condition of a regular edge labeling. □

st-planar graphs. In this thesis we repeatedly use regular edge labellings, so it is a good idea to investigate their structure. Kant and He [5] show that a regular edge labeling is closely linked to a pair of *st*-planar graphs. We repeat this in a different form in Lemma 3 below. The difference is that they orient the edges between poles while we remove the non-relevant poles altogether.

An *st*-planar graph is an oriented planar graph with one source (in-degree 0) s and one sink (out-degree 0) t. Both s and t lie on the outer face. Moreover, such an *st*-planar graph has no directed cycles.

Lemma 3. *The blue edges of* $G \setminus \{\mathrm{N}, \mathrm{S}\}$ *form an st-planar graph with* $s = \mathrm{W}$ *and* $t = \mathrm{E}$*. Moreover, the red edges of* $G \setminus \{\mathrm{W}, \mathrm{E}\}$ *form an st-planar graph with* $s = \mathrm{S}$ *and* $t = \mathrm{N}$*.*

Proof. Note that we have no monochromatic directed cycles because such a cycle would for example correspond to a group of adjacent rectangles that have

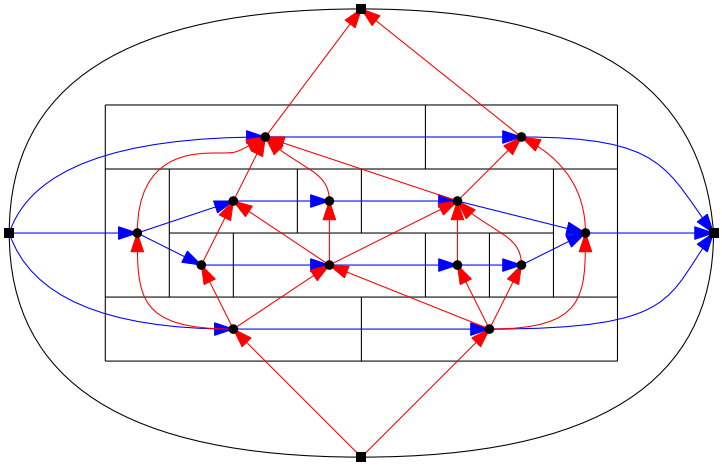


Figure 5: An example regular edge labeling with corresponding rectangular dual.

no leftmost or topmost one. By the interior vertex condition interior vertices can not be sources or sinks, this leaves the non-removed poles to be the sources and sinks, as required. □

We refer to these st-planar graphs as the *blue graph* and *red graph* of some regular edge labeling and we refer to their faces as *blue faces* and *red faces*. An example of such a colored extended adjacency graph with the blue and red graph can be found in Figure 5.

Every face F in an st-planar graph has the same structure. The boundary of F consists of two directed paths, so-called *boundary paths*, with common start vertex v and end vertex w. We say v is the *split* vertex of F and w is the *merge* vertex of F. These boundary paths are subsequent in the clockwise rotation at v. We say the first path in the rotation, starting at the beginning of the adjacent pair, is the *top boundary path* (blue graph) or *left boundary path* (red graph) of F and the second one is the *bottom boundary path* (blue graph) or *right boundary path* (red graph).

Maximal segments. Due to the way we color a regular edge labeling of $\mathcal{G}_{\mathcal{E}}(\mathcal{L})$ given a layout $\mathcal{L}$ a horizontal maximal segment corresponds to a blue face and a vertical maximal segment corresponds to a red face; as one can see in Figure 5.

Consider a face corresponding to a maximal segment. The number of internal vertices of both boundary paths without counting the split and merge vertices is the number of rectangles on the respective sides of the maximal segment. Hence, a one-sided maximal segment corresponds to a face with one boundary path of length 2 and a k-sided maximal segment to face with a shortest boundary path of length at most $k + 1$. We can't have faces with a boundary path of length 1 since such a face can't enclose a valid segment.

Lemma 4. *No face can have a boundary path of length* 1

Proof. A boundary path can not have length 1 since by construction of a regular edge labeling a red or blue face encloses a maximal segment and thus has to go

trough at least one intermediate rectangle/vertex. □

3 Family of graphs not k-sided for any k

In this section we will show the following theorem.

Theorem 5. *There is a family of graphs G_k that, for any $k \in \mathbb{N}$, has members that are not k-sided.*

All members of this family contain separating 4-cycles. If there is no k-sided rectangular dual for a certain corner assignment $\bar{G}$ of G. There may still be another corner assignment of G that admits a k-sided dual. However, if we view $\bar{G}_k = H_k$ as a graph in its own right then G_k is the interior of a separating 4-cycle of H_k. We show in Lemma 7 this implies that G_k, as induced subgraph, has to be colored in accordance with the corner assignment $\bar{G}_k$. We first have to prove some lemmas before we can prove Theorem 5.

Lemma 6. *Let $\mathcal{C}$ be a separating 4-cycle of a corner assignment $\bar{G}$. Then, in any rectangular dual the region enclosed by the rectangles dual to the vertices in $\mathcal{C}$ is a rectangle.*

Proof. The interior I of $\mathcal{C}$ will be represented by some rectilinear shape in every rectangular dual $\mathcal{L}$ of $\bar{G}$. Such a shape must have at least 4 clockwise right turns if we travel along its boundary in a clockwise direction otherwise the shape is not closed.

Yet, such a clockwise turn can not occur due to a single rectangle. Instead, such a turn can only occur when two rectangles are adjacent to each other. Because a 4-cyle represents only 4 pairs of adjacent rectangles, the representation of I in $\mathcal{L}$ can only have 4 clockwise turns. Hence, it must be a rectangle, the only rectilinear shape with just 4 clockwise turns. □

Lemma 7. *Let $\mathcal{C}$ be a separating 4-cycle of $\bar{G}$ with interior I. We can label the vertices of I by a, b, c and d in clockwise order such that all interior edges incident to a, b, c and d are incoming red, incoming blue, outgoing red and outgoing blue, respectively.*

Proof. By Lemma 6 the interior I of $\mathcal{C}$ will be represented by a rectangle $\mathcal{I}$ in any rectangular dual. Since two disjoint rectangles can only be adjacent to each other at one side, $\mathcal{I}$ has four sides that need to be covered and $\mathcal{I}$ is adjacent to only four rectangles we know that every side of the rectangle $\mathcal{I}$ is adjacent to a single rectangle. We then denote by a the vertex corresponding to the rectangle above $\mathcal{I}$, b the rectangle left of $\mathcal{I}$, c the rectangle below $\mathcal{I}$ and d the rectangle right of $\mathcal{I}$.

Then the required coloring follows from how a regular edge labeling is linked to an equivalence class of layouts.

□

Hence, if we know the color and orientation of one interior edge incident to a vertex of a separating 4-cycle $\mathcal{C}$ we know the color and orientation of all interior edges of $\mathcal{C}$ incident to $\mathcal{C}$.

Lemma 7 is useful because it allows us to consider a single corner assignment $\bar{G}$ of G by building a *scaffold*. Suppose we want to investigate

NE	N	E	SE	NW
SE	S	E	NE	SW
SW	S	W	SE	NW
NW	N	W	NE	SW

Table 1: The neighbors of the new poles.

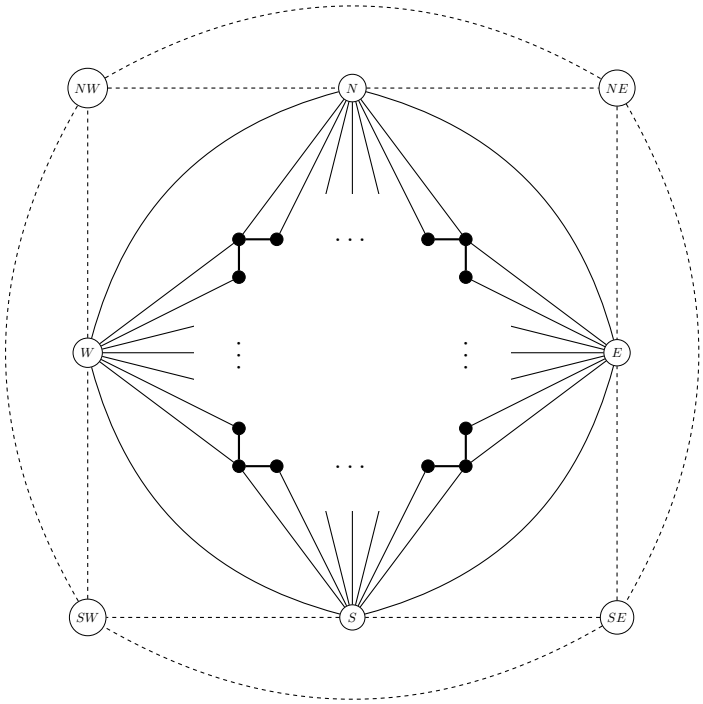


Figure 6: The construction of a scaffold.

some specific corner assignment $\bar{G}$ of G with poles N, E, S and W then we can consider the graph $\bar{G} = H$ as a graph in its own right. H admits a corner assignment $\bar{H}$ without separating triangles by connecting the new poles as indicated in Table 1. $\bar{H}$ is shown in Figure 6. G is displayed in thick lines and with closed vertices. An arbitrary corner assignment $\bar{G} = H$ is then drawn with thin lines and open vertices. A corner assignment of H is then drawn with dashed edges and open vertices.

The graph H can have more than one corner assignment but they all contain the separating 4-cycle $\mathcal{C} = NESW$. Thus, by Lemma 7 we see that, without loss of generality, the interior edges of $\mathcal{C}$ incident to N are colored incoming red, those incident to E are colored incoming blue, those incident to s are colored outgoing red and those incident to W are colored outgoing blue.

Proof of Theorem 5. Now all the preparations are done we can consider the family of graphs G_k with the corner assignment $\bar{G}_k$ given in Figure 7. We know we only have to look at this corner assignment since we can force it using a scaffold and Lemma 7. In G_1 the dots are replaced by a single vertex, in G_2 the dots are replaced by two vertices and so on. Each member has 2 maximal separating 4-cycles. These are both marked by thick lines in Figure 7. Many of the edges in this graph have only one possible color and orientation that does not violate the constraints of a regular edge labeling. Firstly, we can color the edges incident with the poles in accordance with the exterior vertex condition. Subsequently, we can use Lemma 7 on both maximal separating 4-

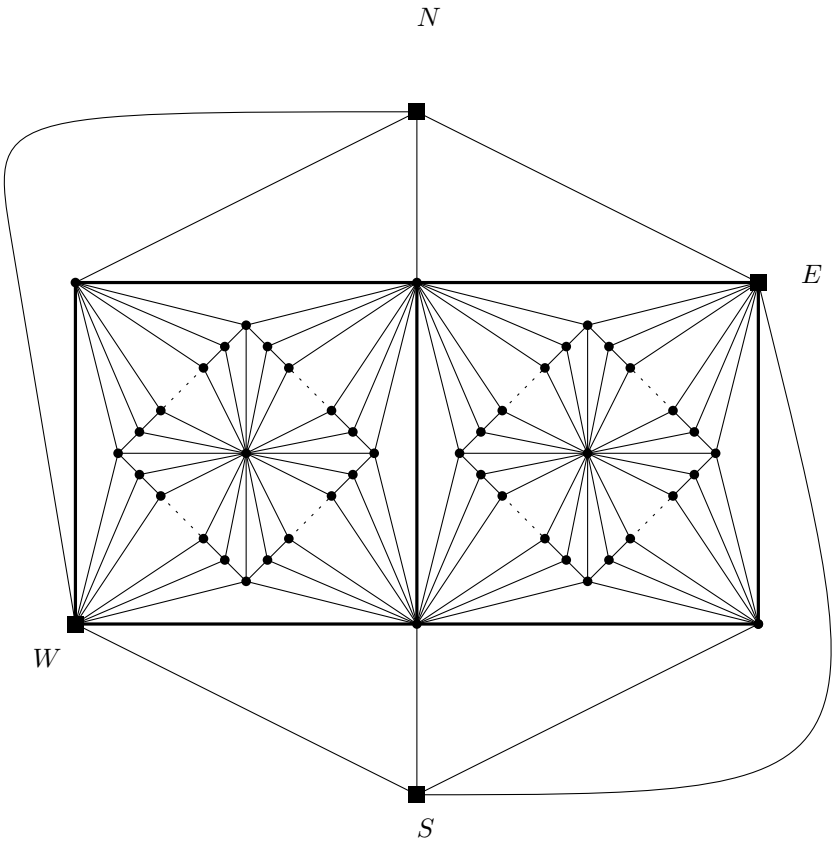
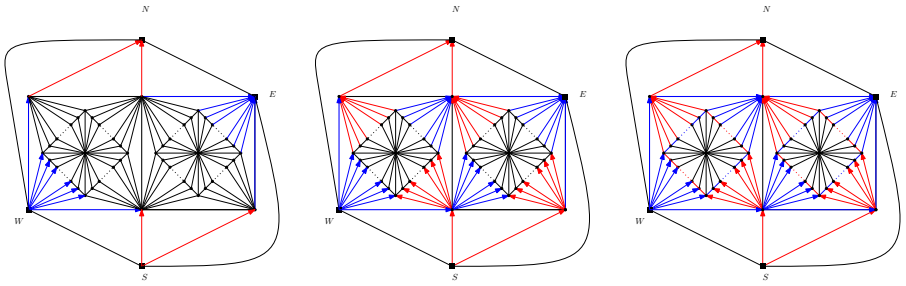


Figure 7: A family of graphs not k-sided for any k

cyles in accordance to color even more edges and finally we can color the edges in triangles of which the other two edges have the same color using Lemma 2. These forced colorings are performed in Figure 8.

The result is then the graph displayed in Figure 9. The black edges in this figure are edges that do not have a forced coloring by the above argument (Although most of them can be forced by Lemma 7). We focus on the centered black edge e, e is an interior edges of both the red and blue faces drawn with dashed edges in Figure 9. Both boundary paths of these faces are of length larger than k. Hence, e has to be colored both red and blue to prevent that face corresponding to a k-sided segment occurs in the regular edge labeling. An edge can not be colored red and blue at the same time and hence G_k is not k-sided.

Since this proof does not depend on the value of k, the family G_k has graphs that are not k-sided for any k. □



(a) Coloring the edges adjacent to the poles.

(b) Propagate trough the 4-cycle.

(c) Color such that there are no monochromatic triangles.

Figure 8: Coloring the graph.

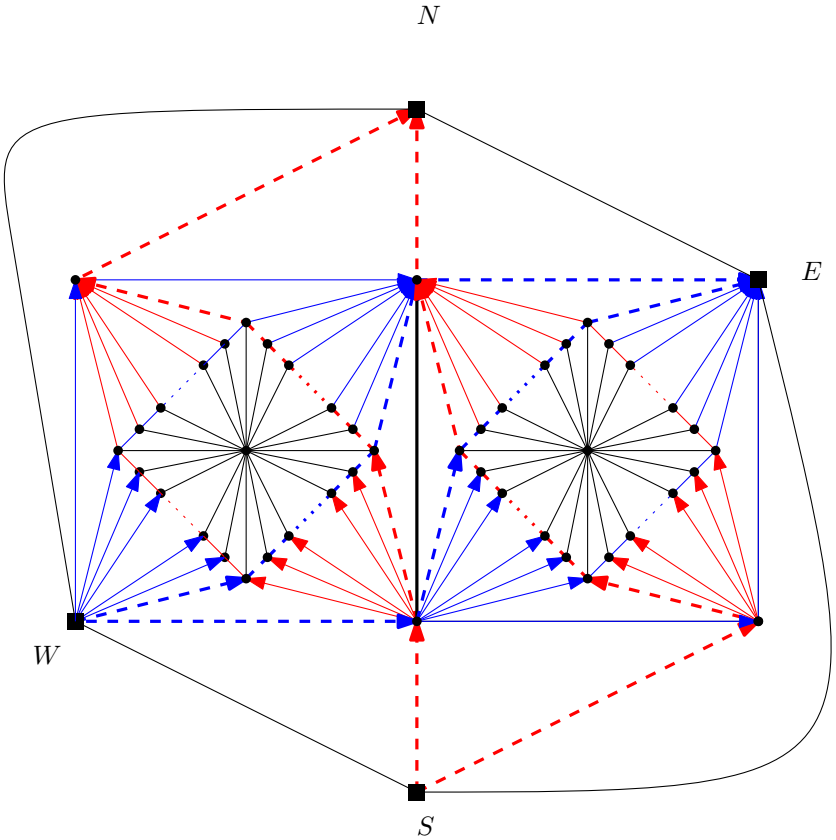


Figure 9: The graph after all coloring steps

4 Algorithm

Algorithms for regular edge labellings. Kant and He [5] were the first to design algorithms that determine a regular edge labeling given a graph G. Fusy [3] recently developed a different algorithm computing a specific regular edge labeling using a method which shrinks a cycle while coloring the exterior of this cycle in accordance with a regular edge labeling. He refers to such a cycle as a *sweepcycle*. Unfortunately his proof of this algorithm is rather concise.

In this algorithm Fusy starts by denoting the outer cycle of $\bar{G} \setminus \{\mathrm{N}\}$ as the sweepcycle $\mathcal{C}$. He then shrinks this sweepcycle by updating it with interior paths of $\mathcal{C}$. During this update he maintains invariants on the structure of both the cycle and its exterior. After doing a finite amount of updates, the sweepcycle has no more interior vertices. At this point he completes the algorithm and obtains a regular edge labeling.

Our algorithm. In this section we will present an algorithm providing a d-sided rectangular dual for any corner assignment $\bar{G}$ without separating 4-cycles, where d is the highest degree among vertices of G in $\bar{G}$. Hence, we will prove the following theorem.

Theorem 8. *Triangulations of the k-gon G that have a corner assignment $\bar{G}$ without separating 4-cycles are $d-1$-sided, where d is the maximal degree of the vertices of G in $\bar{G}$.*

To describe this algorithm we introduce the notion of right neighbor paths in Section 4.1. We use these right neighbor paths to find a regular edge labeling with some nice properties in Section 4.2. Afterwards we unfortunately still have to do three post-processing steps. In Section 4.3 we make sure our regular edge labeling is *vertically one-sided*, that is, that all vertical segments are one-sided. In Section 4.4 we will remove most occurrences of something called topfans. Then, in Section 4.5 we can finally subdivide all blue faces that are too large without creating red faces that are too large. This finishes the algorithm.

4.1 The right neighbor path of a path

In the sweepcycle step of the algorithm (Section 4.2) we will use the *right neighbor path* of a path. In this section we show that for any path $P = p_1 \dots p_k$ with no internal vertices incident to the outer face and without chords or separating 2-chords on the right of the path, the right side neighbors of P are a path Q (Lemma 11). We even show some additional properties hold for Q (Lemma 12 and 14). Similar things also hold for the left neighbors of P but we will not need this for our algorithm.

The right side of a path is not yet defined. To do so we start this section by expanding on the notion of rotations. During the proofs in this section we also need various types of chords so we subsequently introduce these. Afterwards, we can finally discuss right neighbor paths.

Rotations. We assume a fixed embedding for G. Recall that the *rotation* at a vertex v is the clockwise order of the edges incident to v and that two vertices x, y are said to be *consecutive* in the rotation at v when the edges vx and vy are consecutive in the rotation. Sometimes, we want to denote number of subsequent vertices, which we call an *interval*, in the rotation. We let $[x, y]$ denote all the vertices in the rotation of v from x to y and we let the *exclusive interval* (x, y) denote the same vertices without x and y. In Figure 10 for example the interval $[v, p_{i+1}]$ consists of the vertices v, w, p_{i+1} and (w, u) consists of the vertices p_{i+1}, x.

Given a path P and an internal vertex $p_i \in P$. A neighbor $v \notin \mathcal{P}$ of p_i lies on the *left* of P if it lies in the interval (p_{i-1}, p_{i+1}) in the rotation of p_i. Otherwise, v lies in the interval (p_{i+1}, p_{i-1}) in the rotation of p_i. In this case v lies on the *right* of P. We will use the same notion of left and right for edges. That is, an edge $e \notin P$ adjacent to p_i lies to left or right if its other end point lies to the left or right, respectively. In Figure 10 v and p_iv lie on the left of P and u and p_iu lie on the right of P.

Path manipulations. With $\overleftarrow{P}$ we denote the *reversed path* $p_k \dots p_1$. We use $\oplus$ to denote the *concatenation* of paths. That is, given a second path Q with vertices $q_1 \dots q_\ell$ and $p_k = q_1$ the path $P \oplus Q$ consists of $p_1 \dots p_{k-1} q_1 q_2 \dots q_\ell$. Recall that a cycle is simply a path starting and ending at the same vertex. Hence, if we have two internally disjoint paths P, Q from s to t then $P \oplus \overleftarrow{Q}$ is a cycle. Furthermore, we use a vertical bar to denote the *restriction* of a path to a certain set of vertices. So $P|_{p_i,p_j}$ with $i < j$ is the subpath of P with vertices $p_i \dots p_j$.

Chords. A *chord* of a path is an edge that connects two non-subsequent

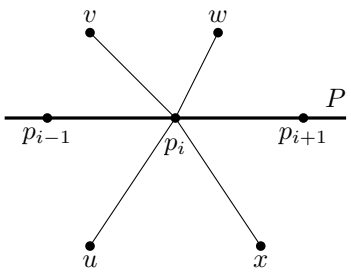


Figure 10: Rotation at p_i.

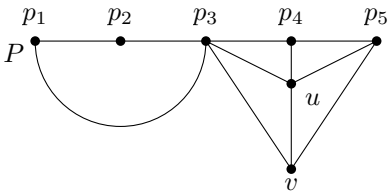


Figure 11: A path with a chord and a separating 2-chord.

vertices. A path without chords is *chordfree*. The path P in Figure 11 has the chord p_1p_3. A *k-chord* is a path Q of length k that connects two non-subsequent vertices p_i, p_j of P such that $P \cap Q = \{p_i, p_j\}$. Note that $P|_{p_i,p_j} \oplus \bar{Q}$ is a cycle. A (k-)chord Q is *separating* if this cycle is separating. In Figure 11 there are two 2-chords, p_3up_5 and p_3vp_5, but only one of them is separating, namely p_3vp_5.

Right neighbor paths. We already mentioned that in the sweepcycle step we use the right neighbor path of a path P. Recall that we assumed P has no internal vertices incident to the outer face and no chords or separating 2-chords on the right. We first show that every vertex has right neighbors. Then we give the procedure for finding the right neighbor path. Afterwards we show that the right neighbor path is indeed a path (Lemma 11) and some additional properties in Lemmas 12 and 14.

Lemma 9. *Every internal vertex of P has at least one neighboring vertex on the right.*

Proof. Suppose that an internal vertex p_i has no neighbor on the right of the path. Then $\dots p_{i-1}p_ip_{i+1}\dots$ is a partial face border. Since p_i is not incident to the outer face, p_i must be incident to a face of degree 3. Thus, $p_{i-1}p_ip_{i+1}$ is a face. However, this would imply a chord on the right of P as can be seen in Figure 12. Hence, by contradiction p_i must have a neighbor on the right. □

The right neighbors of the internal vertices of P will form the *right neighbor path* Q of P. Let us first define a larger list of vertices Q' from which we later remove vertices to get Q. Q' will consist of p_1 and those vertices adjacent to p_2 that are in the interval (p_1, p_3) of the clockwise rotation at p_2. Followed by the vertices in the interval (p_2, p_4) of the rotation at p_3. We continue this up to the vertices in the interval (p_{k-2}, p_k) of the rotation at p_{k-1} and finally p_k. We get Q from Q' by removing all subsequent duplicates from Q. In Figure 13 an example of a right neighbor path is given.

To break the proof that Q is a path into two parts we define a *walk* as a path without the constraint that the same vertex does not occur twice. Hence,

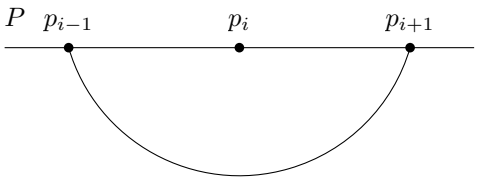


Figure 12: The hypothetical situation that p_i has no right neighbor.

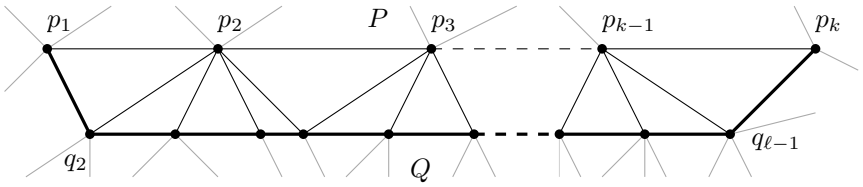


Figure 13: A right neighbor path.

a walk is a sequence of vertices that are connected to each other but vertices may repeatedly occur.

Lemma 10. *The right neighbor path Q is a walk.*

Proof. Let us denote the vertices of Q by $q_1 q_2 \dots q_\ell$. Let q_i and q_{i+1} be two subsequent vertices of Q'. We will show they are either connected or the same vertex. We first consider the case where $1 < i < \ell - 1$. Now there are two sub-cases. Either, (a), q_i and q_{i+1} are vertices adjacent to the same vertex p_j and thus subsequent in the rotation at p_j or, (b), q_i was the last vertex adjacent to p_j and thus q_{i+1} is the first vertex adjacent to p_{j+1} since by Lemma 9 every internal vertex of P has right neighbors. Both cases are depicted in Figure 14.

In case (a) we note that since q_i and q_{i+1} are subsequent in the rotation at p_j $q_i q_{i+1}$ is an edge since p_j is not incident to the outer face and every interior face of G is a triangle.

In case (b) we note that $p_i q_i$ and $p_i p_{i+1}$ are edges subsequent in clockwise order, hence $q_i p_{i+1}$ is also an edge. Hence, q_i is the first vertex adjacent to p_{i+1} subsequent to v_i in the clockwise rotation. Thus, $q_i = q_{i+1}$, that is q_i and q_{i+1} are duplicates.

Now for the cases $i = 1$ and $i = k - 1$. q_1 and q_2 are vertices adjacent to p_2 subsequent in the clockwise rotation of p_2 and hence connected since every interior face is a triangle. In the same way q_{k-1} and q_k are subsequent vertices in the rotation at q_{k-1} and hence connected. This can also be seen in Figure 13.

Since all pairs of subsequent vertices in Q' are connected or duplicates the step removing all duplicates from Q' ensures Q is a walk.

□

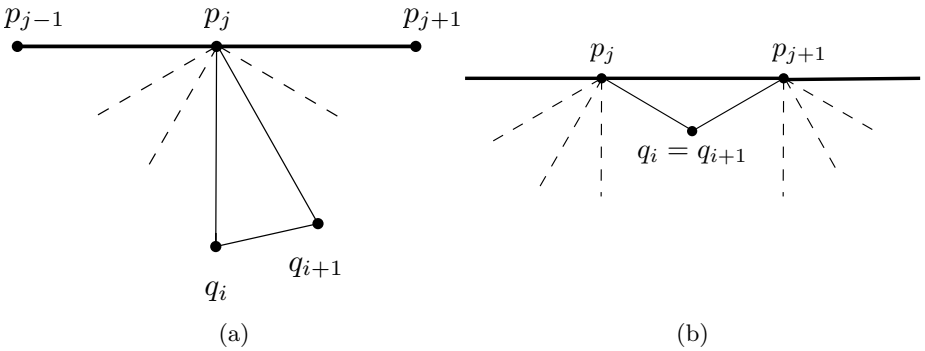


Figure 14: The two main cases of the proof showing that W is a walk.

Lemma 11. *The right neighbor path Q is a path*

Proof. We already know Q is a path by Lemma 10. Hence, we only have to show that Q contains no duplicate vertices.

Suppose that Q has a duplicate vertex $q_i = q_j$ with $i < j$. Then this vertex must have been a neighbor to two different vertices in P, since it can not have been added twice while being connected to only one vertex. We denote these vertices p_ℓ, p_k with $\ell < k$. This gives us the situation depicted in Figure 15.

Due to the order in which we added vertices to Q', which is preserved by the removal of vertices when we go to Q, we know that any vertices in between q_i and q_j in Q must be one of the following:

1. Adjacent to p_ℓ and in the interval $(q_i, p_{\ell+1})$ in p_ℓ's rotation.
2. Adjacent to one of $p_{\ell+1}, p_{\ell+2}, \ldots, p_{k-1}$ and to the right of P.
3. Adjacent to p_k and in the interval (p_{k-1}, q_j) in p_k's rotation.

All three cases describe a vertex that lies in the interior of the cycle $q_i p_i p_{i+1} \ldots p_j$. However, since P has no separating 2-chords on the right this cycle must be empty. Therefore, there are no vertices in between q_i and q_j and they must be subsequent duplicates. However, Q is a walk and G is simple so Q has no subsequent duplicates. Hence, Q contains no duplicates at all and is thus a path. □

Lemma 12. *The cycle $P \oplus \overleftarrow{Q}$ has no interior vertices.*

Proof. In the construction of the right neighbor path both cases in Figure 14 add a triangle to the interior with all vertices of the triangle in $P \oplus \overleftarrow{Q}$. Hence, the interior of $P \oplus \overleftarrow{Q}$ can be subdivided in a number triangles. Suppose there is an interior vertex in the cycle $P \oplus \overleftarrow{Q}$. Then the triangle containing this vertex is a separating triangle. Hence, $P \oplus \overleftarrow{Q}$ has no interior vertices. □

Lemma 13. *Every internal vertex of a right neighbor path Q has a left neighbor.*

Proof. Let q be an interior vertex of Q. q Was added as right neighbor of some internal vertex p of P. Since Q does not start or end at p, p must also be a left neighbor of q in Q. □

Lemma 14. *The left of a right neighbor path is chordfree.*

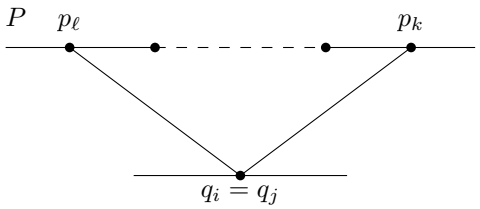


Figure 15: A hypothetical duplicate vertex.

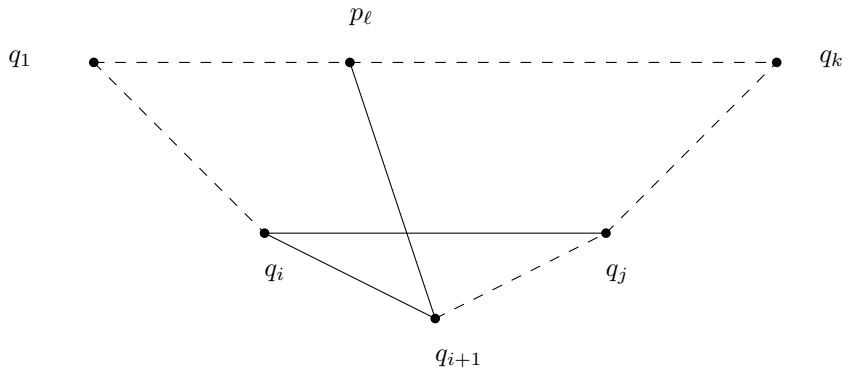


Figure 16: The construction in the proof of Lemma 14.

Proof. Suppose that the right neighbor path $Q = q_1 \dots q_k$ has a chord on the left, say between q_i and q_j with $i < j - 1$. Then $q_i q_j$ is an interior edge of $P \oplus \overleftarrow{Q}$. There is a vertex $p_\ell \in P$ such that q_{i+1} is a right neighbor of p_ℓ, hence $p_\ell q_{i+1}$ is an interior edge of $P \oplus \overleftarrow{Q}$. But now we have a crossing between $q_i q_j$ and $p_\ell q_{i+1}$ since both edges run in the interior of $P \oplus \overleftarrow{Q}$ and their endpoints occur alternately in $P \oplus \overleftarrow{Q}$. See figure 16. Since we assume G is planar, there can be no chords on the left of the right neighbor path.

□

4.2 Sweepcycle algorithm

The first step of our algorithm is executing a sweepcycle algorithm inspired by the sweepcycle algorithm by Fusy [3]. We use $\mathcal{C}$ to indicate the current sweep cycle. We shrink $\mathcal{C}$ by updating it with interior paths. The algorithm finishes when $\mathcal{C}$ has no more interior vertices. When the algorithm finishes, it has produced a regular edge labeling. One of the nicest things about the regular edge labeling is Lemma 20. This lemma states that we can not have the fan handle of a large topfan after a split vertex on a so-called bottom path.

FiXme: Q: These are all words the reader does not know yet. How to explain? Should I explain here?

During the algorithm we maintain several invariants on $\mathcal{C}$. The first four are equivalent to those imposed by Fusy. The final invariant is new and allows us to prove Lemma 20.

Invariants 15

(I1) The cycle $\mathcal{C}$ contains the two edges SW and SE.
(I2) $\mathcal{C} \setminus \{\mathrm{S}\}$ has no chords.
(I3) The inner vertex condition holds for all vertices in the exterior of C.
(I4) Every non-pole vertex on the sweepcycle has a red outgoing edge.
(I5) $\mathcal{C} \setminus \{\mathrm{S}\}$ has no separating 2-chords that do not use S.

We initialize the sweepcycle $\mathcal{C}$ with the outer cycle of $\bar{G}$. We denote the vertices of the sweepcycle $\mathcal{C}$ by $\mathrm{S}, v_1 = \mathrm{W}, v_2, \ldots v_{n-1}, v_n = \mathrm{E}, \mathrm{S}$. We repeatedly consider the path $\mathcal{C} \setminus \{\mathrm{S}\}$. In which case we will always order it from W to E. That the edges SW and SE are always in $\mathcal{C}$ is a result of Invariant 15 (I1).

Each update of the sweepcycle consists of the following three steps

1. Take the right neighbor walk of a subpath of $\mathcal{C} \setminus \{\mathrm{S}\}$ to get the *candidate path* P.
2. Evade chords and separating 2-chords on P to get the *updating path* P'.
3. Update the sweepcycle with P'.

We repeat these steps until the sweepcycle does not contain anymore interior vertices. At which point we can terminate the algorithm by coloring the edges of the cycle $\mathcal{C}$ blue and its interior edges red.

4.2.1 Find the right neighbor path

Recall we denote all the vertices of $\mathcal{C} \setminus \{\mathrm{S}\}$ by $\mathrm{W} = v_1 v_2 \ldots v_{n-1} v_n = \mathrm{E}$.

Suppose they are all adjacent to S, then any vertex still in the interior of $\mathcal{C}$ would lie in a separating triangle of G. So we have no interior vertices and hence we can terminate the algorithm as described in Section 4.2.4. In the remainder we assume some vertices from $\mathcal{C} \setminus \{\mathrm{S}\}$ will not be adjacent to S.

Since $\mathcal{C} \setminus \{\mathrm{S}\}$ has some vertices incident to S (W, E) and some that are not, we can consider maximal subpaths of $\mathcal{C} \setminus \{\mathrm{S}\}$ consisting of vertices adjacent to S. We denote by v_i the last vertex of first maximal subpath of vertices adjacent to S and by v_j the first vertex of the second maximal subpath. As candidate path P we take the right neighbor path of $\mathcal{C} \setminus \{\mathrm{S}\}|_{v_i, v_j}$. This right neighbor path does indeed exist since all internal vertices of $\mathcal{C} \setminus \{\mathrm{S}\}|_{v_i, v_j}$ are interior vertices of G and $\mathcal{C} \setminus \{\mathrm{S}\}$ has no cycles or separating 2-chords by Invariants 15 (I2) and 15 (I5). This situation is depicted in Figure 17.

4.2.2 Evade chords and separating 2-chords

The candidate path P we found can have two structures we want to avoid namely

1. Chords
2. Separating 2-chords

By *irregularities*, we will mean these two classes of structures together and the *middle* vertex of a 2-chord is the only internal vertex of that 2-chord. All irregularities are on the right of the candidate path due to Lemma 14 (no chords on the left) and Lemma 12 (no separating 2-chords on the left).

Before we can show how to evade these structures, we first introduce more notation. We orient P from $v_i \in \mathcal{C}$ (the vertex closest to W)to $v_j \in \mathcal{C}$ (the vertex closest to E) and denote its vertices by $p_1 \dots p_k$. The *index* of a vertex $p_i \in P$ is its position in the path, that is, the index of p_i is i. The *start index* of an irregularity I is the index of the first vertex in P that is also in I. Similarly, the *end index* is the index of the last vertex in P that is also in I. The *range* of an irregularity will be given by its start and end index. Depending on what irregularities we find on the candidate path P we will update the sweepcycle with an *update path* P'. This update is described in Section 4.2.3.

While describing the steps we will also show that the following two lemmas hold for every case.

Lemma 16. *The updating paths has no chords or separating* 2*-chords*

Lemma 17. *There are no chords or separating* 2*-chords not containing* S *with one endpoint on the old sweepcycle and one endpoint on the updating path* P'*.*

We have no irregularity. When there are no irregularities, we update the sweepcycle with the entire candidate path P. In this case the update path P' is equal to P.

P' has no irregularity by the definition of this case. Moreover, there are no irregularities with one opinionated on P' and one endpoint on $\mathcal{C}$ since v_i and v_j are both adjacent to S we can not have any chords and any 2-chords must have S as middle vertex.

We have a chord on P. Note that we can not have a chord incident to one of the exterior vertices of the candidate path P since any such chord would violate Invariant 15 (I5) of $\mathcal{C}$ as can be seen in Figure 18.

We identify the chords by their ranges. Of the chords with the smallest end index j we will consider the one with the largest start index i. We denote this

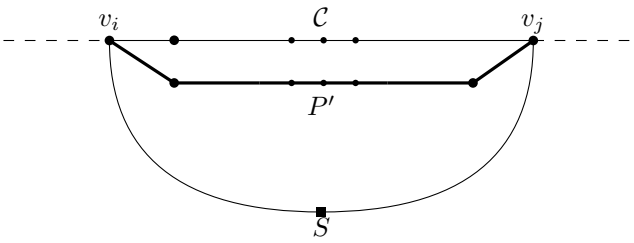


Figure 17: Updating path when P contains no irregularity.

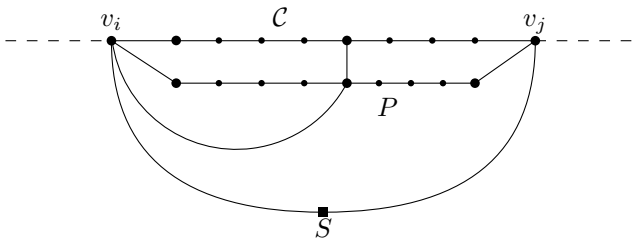


Figure 18: Hypothetical situation where P would have a chord on an exterior vertex.

chord by C. Note that this chord can not contain any other chords since such a chord would have either a large start index or a smaller end index. The way in which we find C is illustrated in Figure 19.

What we do now depends on whether a 2-chord shows up in the interior of the chord concatenated to the candidate path $P|_{i,j} \oplus \overleftarrow{C}$.

No separating 2-chord. If there is no separating 2-chord in the interior of $P|_{i,j} \oplus \overleftarrow{C}$ we let v_k be the shared neighbor in $\mathcal{C}$ of p_i and p_{i+1} and we let v_l the shared neighbor in $\mathcal{C}$ of p_{j-1} and p_j. The updating path P' is the right neighbor path of $\mathcal{C} \setminus \{\mathrm{S}\}|_{v_k,v_l}$. See Figure 20.

P' is entirely inside a chord containing no more chords and thus can not contain a chord. Moreover, there are no separating 2-chords on $P|_{p_i,p_j}$ so P' can not have separating 2-chords. Any irregularity with one endpoint in P' and one in $\mathcal{C}$ will have to cross $v_k p_i p_j v_l$ so we can not have a chord and any 2-chord has p_i or p_j as middle vertex. But, with this restriction such a 2-chord can not be separating.

At least one separating 2-chord. Let j' be end index of the separating 2-chord with the lowest end index in the interior of $P|_{i,j} \oplus \overleftarrow{C}$. And let v_k be the shared neighbor on the sweepcycle of p_i and p_{i+1} and let v_l be the shared neighbor on the sweepcycle of $p_{j'-1}$ and $p_{j'}$. Then the updating path P' is the right neighbor path of $\mathcal{C} \setminus \{\mathrm{S}\}|_{v_k,v_l}$. See Figure 21.

P' is entirely inside a chord containing no more chords and thus can not contain a chord. Moreover, all separating 2-chords are evaded since we evaded the end of the first one ending. Just as in the above case we can not have any chord or separating 2-chord with one endpoint in P' and one in $\mathcal{C}$ any chord or separating 2-chord to connect outside the containing chord $p_i p_j$. That leaves irregularities within the second endpoint inside containing chord $p_i p_j$. Suppose

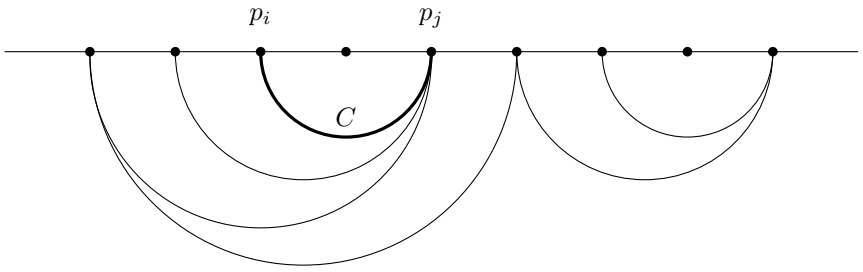


Figure 19: Finding the chord C.

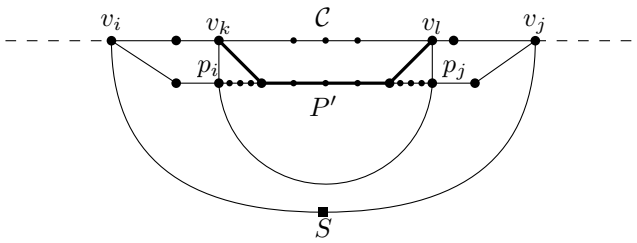


Figure 20: Updating path when P has a chord not containing a separating 2-chord.

that we have a separating 2-chord, then this would have been a chord of P. This is in contradiction with p_ip_j being a minimal chord. We also can not have a chord since this would break $P|_{p_i,p_j} \oplus p_jp_i$.

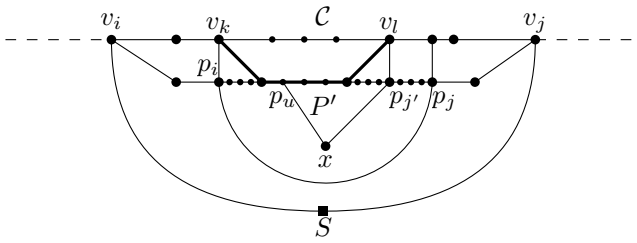


Figure 21: Updating path when P has a chord containing at least one separating 2-chord.

Only separating 2-chords. In this case the candidate path P has no chords since we would otherwise be in the above case.

Any separating 2-chords with v_j as end vertex. We find the smallest separating 2-chord with v_j as end vertex (i.e. the one with the highest start index). Say this separating 2-chord has start index i. Let v_k be the shared neighbor on the sweepcycle of p_i and p_{i+1}. The updating path is the right neighbor path of $\mathcal{C} \setminus \{\mathrm{S}\}|_{v_k,v_j}$. See Figure 22.

P' starts inside the smallest separating 2-chord hence it has no 2-chords, and furthermore has no chords since P already had none.

We have no chords with one endpoint in P' and one in $\mathcal{C}$ since these would have to break $v_jxp_iv_k \oplus \overleftarrow{P'}$ or be adjacent to v_j, which is in violation of Invariant 15 (I5). Any 2-chords with one endpoint in P' and one in $\mathcal{C}$ would have x or p_i as middle vertex. However, the first yields a 2-chord of the candidate path with a higher start range, this is a contradiction. And the second can not yield a separating 2-chord.

Only other separating 2-chords Find the 2-chord with the lowest end index, say that this is j. Let v_l be the shared neighbor on the sweepcycle of p_j and p_{j-1}. The updating path is the right neighbor path of $\mathcal{C} \setminus \{\mathrm{S}\}|_{v_i,v_l}$. See Figure 23.

Any updating path stops before the end of a separating 2-chord and furthermore contains no chords since P already did not.

Any chord or 2-chord with one vertex in the updating path and the other on the old sweepcycle must end to the right of the updating path since the updating

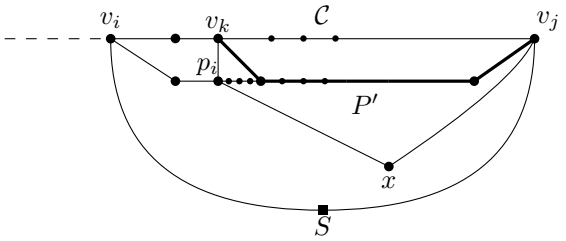


Figure 22: Updating path when P has a separating 2-chord with v_j as end vertex.

path starts at v_i, a vertex adjacent to S. Suppose that we have a separating 2-chord then that would have been a chord of the candidate path. This is in contradiction with the assumption that the candidate path had no chords. We also have no chord since such a chord would violate Invariant 15 (I5) of the old sweepcycle. Furthermore, the second-to-last vertex of the updating path has no chords since it would break $v_l p_j x p_u$. (see Figure 23).

4.2.3 Updating

Once we found the updating path P', we can update the sweepcycle with this path. Let p_a and p_b indicate the two unique vertices of P' that are also part of $\mathcal{C}$. We will then let $\mathcal{C} \setminus \{\mathrm{S}\}|_{P'}$ denote the path $\mathcal{C} \setminus \{\mathrm{S}\}|_{p_a,p_b}$. In this section we describe how to update the sweepcycle with an updating path and we show that the update maintains all sweepcycle invariants (Lemma 18). To execute the update we color all interior edge of $\mathcal{C} \setminus \{\mathrm{S}\}|_{P'} \oplus \overleftarrow{P'}$ red and orient them towards $\mathcal{C} \setminus \{\mathrm{S}\}|_{P'}$. We also color all edges of $\mathcal{C} \setminus \{\mathrm{S}\}|_{P'}$ blue and orient them from lower to higher induces. We then update the sweepcycle to $\mathcal{C}'$ by replacing $\mathcal{C} \setminus \{\mathrm{S}\}|_{P'}$ by P' in $\mathcal{C}$. An example of the whole update for an updating path P' can be seen in Figure 24.

Lemma 18. *Updating with a path P' maintains all sweepcycle invariants*

Proof. We will prove that the new sweepcycle $\mathcal{C}'$ obtained by updating $\mathcal{C}$ is again a valid sweepcycle. Invariant 15 (I1) remains true. Invariant 15 (I3) holds due to the way we colored the edges around the new interior vertices as can be seen in Figure 24. Furthermore, Invariant 15 (I4) holds because every internal vertex of P' has a left neighbor by Lemma 13.

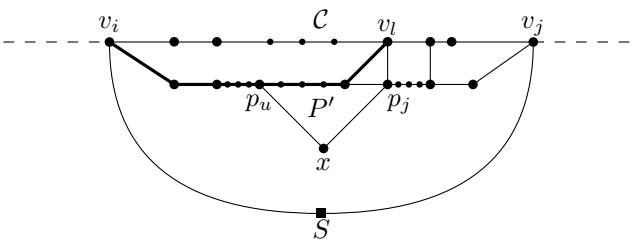
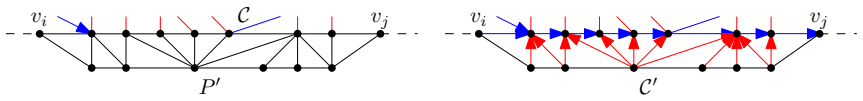


Figure 23: Updating path when P has separating 2-chords none of which have v_j as end vertex.



(a) Before. (b) After.

Figure 24: The update.

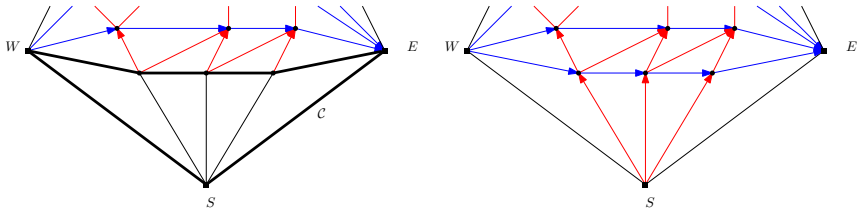
To see that Invariants 15 (I2) and 15 (I5) hold, note that there can be no chords or 2-chords with both endpoints in the overlap of the old and new sweepcycle $\mathcal{C} \cap \mathcal{C}'$ by Invariants 15 (I2) and 15 (I5). Since the updating path itself also has no irregularities (Lemma 16), we know any chord or separating 2-chord C has to have one vertex on $\mathcal{P}$ and one vertex on the unchanged part of old sweepcycle $\mathcal{C} \cap \mathcal{C}'$. However, these potential chords can not exist by Lemma 17. Hence, $\mathcal{C}'$ is a valid new sweepcycle. □

If after the update the new sweepcycle $\mathcal{C}'$ has no interior vertices we terminate the algorithm as will be described in Section 4.2.4. Otherwise, we start the update loop again by finding a new candidate path.

4.2.4 Terminating the algorithm

When the sweepcycle has no more interior vertices, we can not update it anymore. However, at this point it is easy to color the remainder of the graph. All vertices in $\mathcal{C} \setminus \{S\}$ must be adjacent to S since SW and SE are part of $\mathcal{C}$ by Invariant 15 (I1), $\mathcal{C} \setminus \{S\}$ has no chords by Invariant 15 (I2) and $\mathcal{C}$ does not contain any interior vertices. All sweepcycle interior edges are adjacent to S, since otherwise we would have a chord in $\mathcal{C} \setminus \{S\}$ (violating Invariant 15 (I2)).

We color all interior edges of $\mathcal{C}$ red and orient them towards $\mathcal{C} \setminus \{S\}$ and the edges in $\mathcal{C} \setminus \{S\}$ are colored blue and oriented towards E. The termination step can be seen in Figure 25. This last move completes the interior vertex condition for vertices in $\mathcal{C} \setminus \{W, S, E\}$ and also correctly completes the exterior vertex condition.



(a) Before. (b) After.

Figure 25: The termination step.

Lemma 19. *The resulting structure is a regular edge labeling*

Proof. After running the whole algorithm the interior vertex condition holds for all vertices in the graph by Invariant 15 (I3). Furthermore, the poles are also colored correctly due to Invariant 15 (I1). □

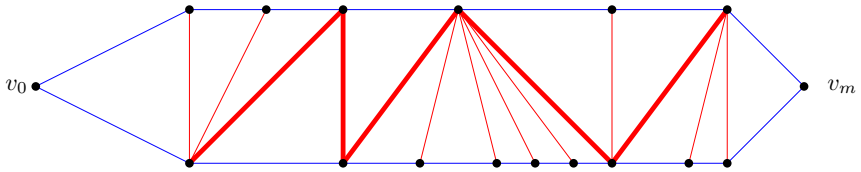


Figure 26: An example face with fans.

4.2.5 A useful property of this regular edge labelling

There still is a useful property of this regular edge labeling left to discuss (Lemma 20). Before we can state this property, we first need to introduce fans.

Fans. We want to better describe the interior of blue (or red) faces. Every interior edge of such face goes from one boundary path to the other (otherwise its start or end vertex would violate the interior vertex condition or create a face with a boundary path of length one violating Lemma 4). We will describe the edges from the split to the merge vertex of F.

Let $u_0, u_1, \ldots u_n$ be the vertices of the top boundary path of F and $v_0, v_1, \ldots, v_m$ the vertices of the bottom boundary path. That is $u_0 = v_0$ is the split vertex and $u_n = v_m$ is the merge vertex. Since our graph is a triangulation, u_1v_1 must be an edge. For the second edge in the face we have two options, either u_1v_2 or u_2v_1, otherwise this edge and the previous one would not form a triangle. This argument holds for every subsequent edge, we can either increase the index of the top boundary path or the index of the bottom boundary path. Hence, this face is, for the readers that know this term, a *triangle strip*.

We call a maximal sequence of at least two edges increasing the index on the top boundary path (and thus keeping the index on the upper path fixed) a *bottomfan* or simply *B-fan* and a maximal sequence of at least two edges increasing the index on the bottom boundary path is called a *topfan* or just *T-fan*. The *size* of such a fan is the number of edges contained in the sequence. By the definition of a fan it has size of at least 2. We use *fans* to refer to both these *types* of fans (i.e. T- and B-fans). We call a fan of size 3 or larger a *large fan* and a fan of size 2 a *small fan*.

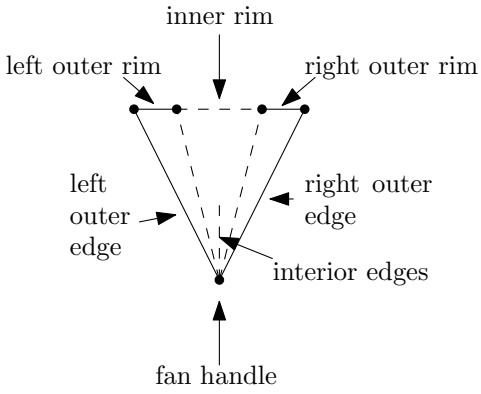


Figure 27: A number of fan-related terms.

In a face we alternately encounter B- and T-fans. If we would have two adjacent fans of the same type, we would just have a single larger fan of that type. In Figure 26 we see a blue face consisting of subsequently a B-fan of size 3, a T-fan of size 2, a B-fan of size 2, a T-fan of size 6, a B-fan of size 3 and a T-fan of size 3.

We introduce some more terminology for fans: *outer edges*, *fan handles* and the *rim* as can be seen in Figure 27. The *fan handle* v is the vertex shared by all edges in the fan. Let H be the induced subgraph of vertices incident to the edges in the fan. H contains no edges not belonging to F since these would lead to separating 3-cycles. The *rim* is the path given by $F \setminus \{v\}$. The *outer rim* are the two extreme edges of this path and the *outer edges* are the edges between the fan handles and the extreme vertices of the *rim*.

FiXme: Q is this definition of Rim more clear or still unclear?

A similar discussion can be given for red faces. However then we have *right fans* and *left fans* instead of bottom and top fans

The property.

Before finally discussing the property, we will introduce two more definitions. A *splitvertex* is a vertex with more than one outgoing blue edge. Given a splitvertex v we will by *bottom* path mean the path that comes in trough the first edge in the interval of incoming blue edges in the rotation at v and leaves through the last edge in the interval of outgoing blue edge in the rotation at v. See Figure 28.

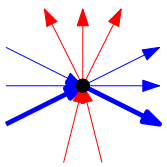


Figure 28: The bottom path of this splitvertex is given in bold.

Lemma 20. *Let v be any splitvertex. Then the subsequent vertex on the bottom path w can not be the handle of a large topfan.*

Proof. There are two possible causes of v being a splitvertex, v is either adjacent to S or v is a splitvertex due to a chord.

If v is a splitvertex because it is adjacent to S then since w is on the bottom path it also has to be adjacent to S by the definition of the bottom path. Hence, w is not the handle of a large topfan.

If v is a splitvertex due to a chord $vabx$ we can continue the bottom path past w as a bottom path that will eventually go to x since every chord is evaded by a single path from v_k to v_ℓ in the algorithm. We will denote this extended bottom path by $\mathcal{P}$. The situation is depicted in Figure 29.

The interior of $vabx \oplus \overline{\mathcal{P}}$ has no vertices. Suppose there would be such a vertex . Then, since $\mathcal{P}$ is a bottom path the blue path going trough this vertex

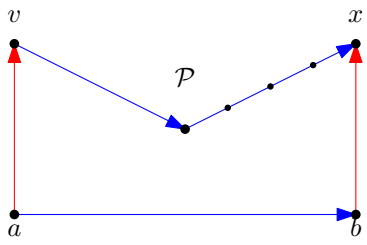


Figure 29: The situation in the proof of Lemma 20

has to start at a and end at b. But this gives a blue face with only one edge on its bottom boundary path violating Lemma 4. Since our graph is a regular edge labeling, $vabx \oplus \overleftarrow{\mathcal{P}}$ has no interior vertices.

This also implies all interior edges are red (by the definition of bottom path) and thus that ab is blue otherwise we would get a monochromatic triangle.

Now w can not be connected to any vertex in $\mathcal{P}$ since that would again give a face with a boundary path of length 1 (Lemma 4). So w can only be connected to a and b and is thus a topfan of size at most 2. (If it is a topfan at all, since we do not consider topfans of size 1 as topfans.) □

4.3 Flipping Blue Z's

The regular edge labeling provided by the sweepcycle algorithm of Section 4.2 is often vertically one-sided but I have not succeeded in proving that this is always the case. We would prefer to get a vertically one-sided regular edge labeling since if we then recolor edges to subdivide large blue faces it is harder to accidentally create many-sided vertical segments. In this section we modify the current regular edge labeling to make it vertically one-sided while maintaining the property of Lemma 20.

A *blue* Z is a path of three blue edges all in the same red face. A Z has a *middle* edge, this is the second edge in this path. If the current regular edge labeling is not one-sided there must be a blue Z as is shown in Lemma 21.

Lemma 21. *A regular edge labeling is not one sided if and only if it contains a blue Z*

Proof. Consider a regular edge labeling that is not one-sided, then it contains a red face of which both boundary paths are of length at least 3. However, since the interior faces of G are triangles there must then be a blue Z in this face.

If a face contains a blue Z, it can not be one-sided. Since path of length 3 inside a red face must have at least 2 vertices on both boundary paths. □

As long as the regular edge labeling is not vertically one-sided we find such a blue Z and recolor its middle edge as in Figure 31. We call this a *flip* and we will say that this edge is *flipped*. Note that both flips transfer a valid regular edge labeling to another valid regular edge labeling. If the interior vertex condition was fulfilled in Figure 30 then it is also fulfilled in Figure 31.

We repeat these flips until the regular edge labeling is vertically one-sided. Since every flip reduces the number of blue edges by one, this is a finite procedure.

Lemma 22. *The result is a vertically one-sided rectangular edge labeling*

Proof. We still have a regular edge labeling since the flips maintain a regular edge labeling. By construction, we flip all blue Z's. If we do not have anymore Z's then the remaining regular edge labeling is vertically one-sided by Lemma 21. □

Lemma 23. *Let v be any splitvertex. Then the subsequent vertex on the bottom path w can not be the handle of a large topfan.*

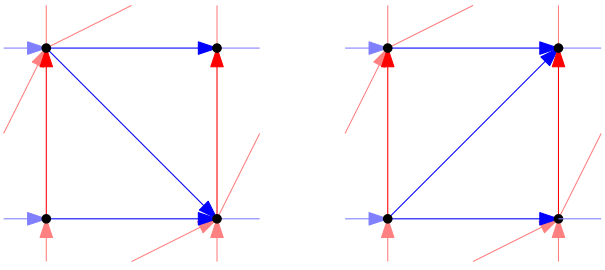


Figure 30: The two possible blue Z's.

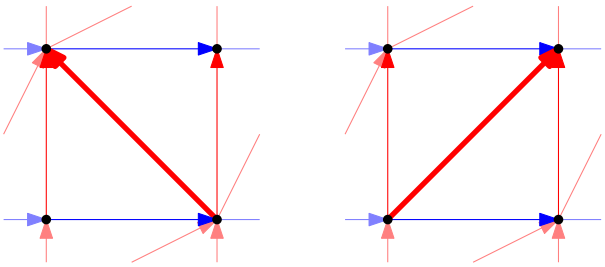


Figure 31: The flip.

Proof. Note that this is the same statement as in Lemma 20. We will show that the operation of flipping Z's does not compromise the validity of this statement.

Note that the flips can only reduce the number of split vertices. Hence it suffices, to show that the statement still holds for all previously existing split vertices.

For a split vertex v adjacent to S we can note that the edge vw will not be flipped because it can not be a middle edge. Hence, w is still on the bottom path and still not the handle of a big topfan.

If v is a split vertex due to a chord, let us note the following. The edges of $\mathcal{P}$ and ab in Figure 29 can not have been flipped since then we would find a monochromatic red triangle while a flip leads to another valid regular edge labeling. Hence, w is still on the bottom path trough v and still can not be the handle of a large topfan. □

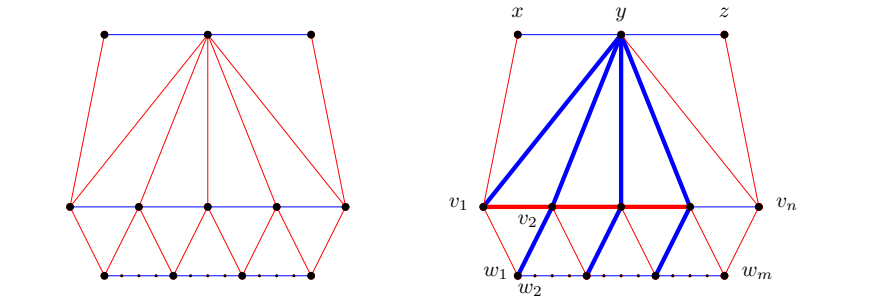
4.4 Topfan flips

In Section 4.3, we obtained a vertically one-sided regular edge labeling (Lemma 22), moreover, this regular edge labeling never has a split next to a topfan handle along a bottom path (Lemma 23). Using local recoloring (*flips*) on the topfans we will maintain a one-sided regular edge labeling (Lemma 24) and make sure that large topfans only occur in very specific situations (Lemma 25). It will turn out that we can deal with these specific situations in the final step of the algorithm described in Section 4.5. Our flips differ depending on whether we encounter a split and or merge in the bottom boundary path. Refer to Figure 32 for a first glance at the different kinds of topfan flips.

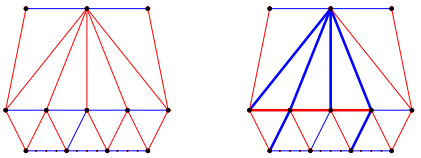
In what order do we flip topfans. We consider all faces in reverse order from last created face to first created face. Since a topfan flip only affects the current face and faces below it we never have to flip in a face that is affected by the results of a topfan flip. We do not flip topfans whose fanhandle is adjacent to the merge of the face.

How to flip a topfan. A topfan is above a number of edges of the bottom boundary path of the blue face containing the topfan. These edges are the rim of the fan. We will call a vertex S-*adjacent* if it adjacent to S.

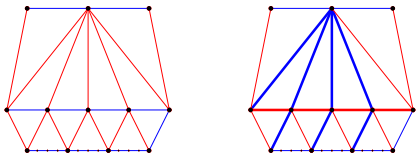
We flip the edges of the topfan along the rim starting at the first vertex and ending at the vertex **before** the first splitvertex or S-adjacent vertex or the vertex **before** the last vertex. This can imply that we do not flip any edges (in



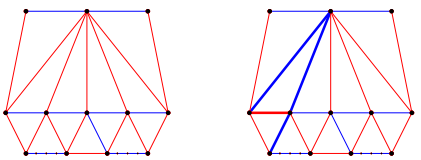
(a) The regular topfan flip.



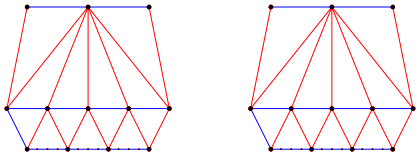
(b) Topfan flip above a merge.



(c) Topfan flip next to merge. Note the additional red edge.



(d) Before a split we stop.



(e) If the split is on the first vertex we do not flip at all.

Figure 32: Topfan Flips.

the case that the first vertex is a split vertex or S-adjacent).

We will use the notation introduced in figure 32a. For the first vertex v_1 we recolor the adjacent outer edge of the topfan. For subsequent vertices v_i we recolor the rim edge between this vertex and the previous vertex v_{i-1} red and we recolor the both edges directly adjacent to this edge in the rotation at v_i blue (if they were not already blue). If we stop flipping before a merge vertex v_{i+1} we flip an additional edge $v_i v_{i+1}$ along the rim, in order to prevent a blue Z from forming.

Examples. Let us show a few examples of this procedure to improve clarity. If the rim has no merges or splits, we execute the topfan flip depicted in Figure 32a. We color all but the rightmost fan edge blue, color all but the rightmost rim edge red and color the left outer edges of all topfans below this topfan in the face below the current face blue.

If the rim consists of only merges, we easily adept a topfan flip to this situation. We simply do not flip the edge merging in as depicted in Figure 32b.

A special case is given by a merge on the last vertex on the bottom edges of the topfan. In that case we flip all rim edges (even the last one) to prevent a blue Z from forming. See figure 32c.

Splits are more difficult to handle. We are unfortunately unable to keep flipping once we hit a split hence we stop before we get that far. See Figure 32d. It this happens on the first vertex we do not flip at all, see Figure 32e.

The result. Before the topfanflips, we had a vertically one-sided regular edge labeling. Afterwards we still have a vertically one-sided regular edge labeling, as we will prove in Lemma 24. Moreover, we have no large top-fans except for some controlled cases (Lemma 25).

Lemma 24. *The regular edge labeling is still vertically one-sided after a topfan flip*

Proof. We take another look at Figure 32. Note that due to Lemma 21 a regular edge labeling is vertically one-sided as long as there is no blue Z. Since the graph was one-sided we can assume that we had no blue Z's. Let us first consider the regular case. Since the edge $v_n w_m$ is red, (otherwise we would have a merge) this change does not produce any blue Z's.

The merge case, due to the clever recoloring, also does not lead to a blue Z. It is clear the split cases do not produce a blue Z either. Since any S-adjacent fan is treated like a split fan, we also do not create Z's in these cases. □

Lemma 25. *In the remaining faces every large topfan is in one of the following two situations.*

1. *This topfan is at the start of the face*
2. *The left outer rim vertex is a splitvertex.*

Proof. All topfans are manipulated in such a way that they start a new face, or are colored blue entirely, unless the left outer rim vertex is a split. Since in that case we do not flip at all, but then the left outer rim vertex is indeed a split. □

4.5 Blue face subdivision

At this point we have a vertically one-sided graph (due to Lemma 24) without large topfans except for the locations provided in Lemma 25. In this section we are going to recolor edges in blue faces to make all of them $d-1$-sided. While at the same time not recoloring so many edges above each other that we create a large red face.

We would like to start at the bottom most face F. We can unfortunately no longer use the creation order since the topfan flips may have changed which faces lie above which other faces, the topfan flips may even have introduced new faces. Hence, we will have to show that there always is a face F that lies below all untreated faces. That is, the whole top boundary path of F borders faces that are not treated while no part of the bottom boundary path borders such faces.

Lemma 26. *There is a face F such that the whole top boundary path of F borders faces that are not treated while the bottom boundary path does not border such faces.*

Proof. Let us first remove all treated faces from $\bar{G}$ by removing their blue edges and connecting their red edges with S.

Unless there are no faces remaining, and in that case we are finished, there is at least one *splitvertex* (vertex with at least two outgoing blue edges) and one *mergevertex* (vertex with at least two blue incoming edge) along the directed path formed by the vertices adjacent to S. There can be no merges before the first split, nor splits after the last merge. Hence, somewhere along the path there is a split followed by a merge. This face has a bottom boundary path that is entirely adjacent to S after removing treated faces. The top boundary path borders untreated faces. □

We will recolor some of F's edges if it is too large. We then mark the edges on the top boundary path of this face above the recolored edges as *loaded*. This means that we will try to not flip above these edges in future iterations of the algorithm. Then we continue with the next face in the creation order.

Loads. As is mentioned above we will mark some edges with so-called *loads*, we will in the rest of the section refer to these edges as *loaded*. The exact use of these loaded edges will become clear in the rest of this section.

It is important to note that if we load any blue edge we regard any other blue edge sharing at least one vertex with this edge to be loaded as well. The occurrence of this phenomenon will be called *putting trough a load*. An example can be seen in Figure 33 where we flipped the thick edge. Hence, we mark uv as loaded and because of putting trough load uw also becomes loaded.

Step requirements. We flip edges in each face, taking into account loads on the bottom boundary path. Such that

1. We never load the two edges next to a split or merge vertex on the top boundary path.

2. We never load two adjacent edges on the top boundary path.

If we flip edges in line with the step requirements for every face then the following lemma holds for the bottom boundary path for yet untreated faces.

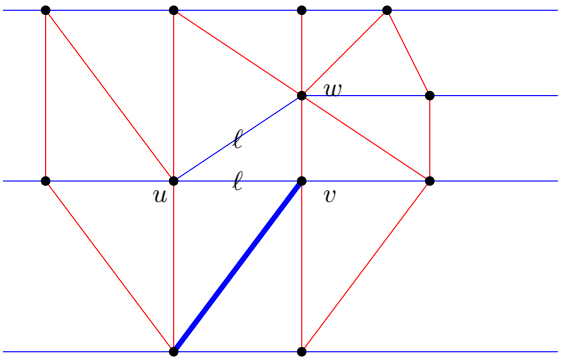


Figure 33: Putting trough load.

Lemma 27. *On the bottom boundary path of a face we never find two subsequent loaded edges. Even when we put trough loads on splits and merges.*

Proof. A single face would never load two subsequent edges. Hence, the only way to get two subsequent loaded edges is using different faces and thus splits and merges. However due to never flipping the two edges next to a split or merge we neither get subsequent loaded edges in such a case. □

4.5.1 Faces without large topfans in the middle

Lemma 28. *We can subdivide any blue face without large topfans into $d-1$-sided chunks while obeying the load rules above.*

Proof. A worst case example is given in Figure 34.

Note that we can flip to the right above each edge in the bottom boundary path except if we would end up next to the merge.

We will look at the vertex on the bottom fence that is incident to the freshly flipped edge, or if we have not flipped an edge yet the vertex next to the split (and we denote it by v). The following are then the rules for flipping above the edges following v.

1. We do not flip above the edges of the first topfan.
2. We flip above the second edge if it is unloaded.
3. Otherwise, we flip above the third edge.
4. We never flip next to the merge.

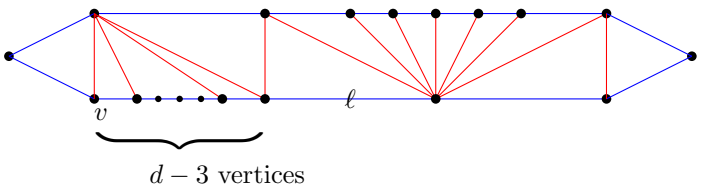


Figure 34: A worst case blue face. We do not flip any edge in this face.

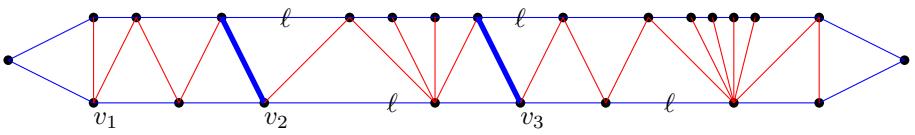


Figure 35: Sample execution of the algorithm.

When flipping above an edge, we always flip the right edge above that edge. Unless we are on the edge next to the merge, because then we flip the left edge above that edge.

The first edge give us the required separation of loaded edges along the top boundary path. The other items make sure we obey the other rules.

The worst case is given by a large topfan at the start and a combination of the last two items. We would in that case want to flip above the second-to-last edge of the bottom boundary path. But we do not because the next edge is incident to the merge vertex. This gives at worst a topfan and two more edges along the bottom boundary path and hence a $d-3+2=d-1$-sided face. □

See Figure 35 for a sample execution of the algorithm described in Lemma 28.

4.5.2 Face encountering a larger topfan

If we have a large topfan in the middle of the face then above the left outer edge of this topfan we can not, as we will see, have another topfan that failed to flip its left outer edges by Lemma 20. This means we can use the following rule: we flip the first edge of a topfan even above a loaded edge. We call such an edge a *forced* flip.

We can not have two such forced flips above each other because that would give a situation as in Figure 36. However, that would mean the fan with fan-handle u must be the handle of a topfan that failed its flip and hence v must have been a split vertex. But then by lemma 23 w can not be the handle of a large topfan.

Since we do not allow a flip above a load (whether it was forced or not) this means that the worst thing that can happen is an ordinary flip followed by a *forced* flip. These two flips can not be followed by any other flip. Hence, the worst case only makes chains of at most 2 blue Z's, that is, a blue path of length 5.

4.5.3 Conclusion

This concludes the last step in the algorithm. Now all the steps in the algorithm are done all that is left is to show that we indeed generated a $d-1$-sided regular edge labeling.

Lemma 29. *Two chained Z's give at worst a red $d-1$-sided face*

Proof. The two chained Z's give a blue path $\mathcal{P}$ of length 5 inside a red face.

Before creating the Z's in this section, the regular edge labeling was vertically one-sided. That is, before recoloring the two edges in this section there were no paths of length 3 inside the face. This also implies that any Z we now create can

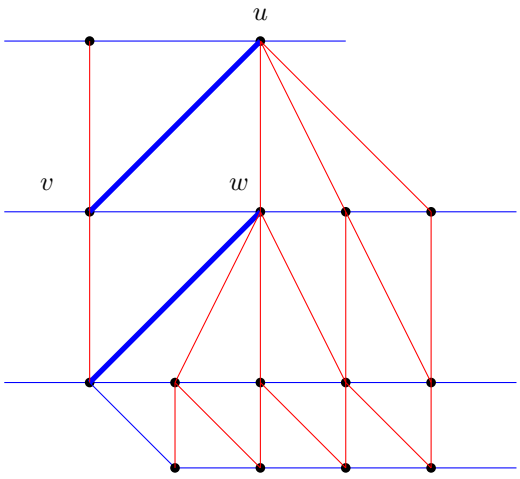


Figure 36: Two forced flips above each other.

have at most one blue fan on the top and one blue fan on the bottom, otherwise we would already have had a 3-path.

So for two Z's we have at most three blue fans. Hence, on one side we have at most one of these. Then the boundary path at this side of the face has at most $d-3+1+1 = d-1$ vertices not counting the split and merge vertex of the red face. □

Then we can now prove Theorem 8.

of Theorem 8 . By construction all blue faces are $d-1$-sided. We have chained at most two $Z's$ so all red face contain at most two blue Z. So red faces are $d-1$-sided by Lemma 29. Hence, we have a $d-1$-sided rectangular edge labeling of $\bar{G}$ corresponding to a d-sided rectangular dual of $\bar{G}$ □

5 Conclusions and future work

In this thesis we proved the following two theorems

1. There is a family of graphs G_k that for any $k \in \mathbb{N}$ has members that are not k-sided (Theorem 5).
2. Triangulations of the k-gon G that have a corner assignment without separating 4-cycles are $d-1$-sided, where d is the maximal degree of the vertices of G. (Theorem 8)

There are still many open questions in this research area. For example, one might hope show that all corner assignments without separating 4-cycles are actually 2-sided. There also might be an algorithm that gives d-sided layouts for graphs with a separating 4-cycle as this is not precluded by Theorem 5.

References

[1] D. Eppstein, E. Mumford, B. Speckmann, and K. Verbeek. "Area-Universal and Constrained Rectangular Layouts". In: *SIAM Journal on Computing* 41.3 (2012), pp. 537–564. DOI: 10.1137/110834032.

[2] É. Fusy. "Transversal structures on triangulations: A combinatorial study and straight-line drawings". In: *Discrete Mathematics* 309.7 (2009), pp. 1870–1894. DOI: 10.1016/j.disc.2007.12.093.

[3] É. Fusy. "Transversal Structures on Triangulations, with Application to Straight-Line Drawing". In: *Graph Drawing*. Springer Science+Business Media, 2006, pp. 177–188. DOI: 10.1007/11618058_17.

[4] X. He. "On Finding the Rectangular Duals of Planar Triangular Graphs". In: *SIAM Journal on Computing* 22.6 (1993), pp. 1218–1226. DOI: 10.1137/0222072.

[5] G. Kant and X. He. "Regular edge labeling of 4-connected plane graphs and its applications in graph drawing problems". In: *Theoretical Computer Science* 172.1-2 (1997), pp. 175–193. DOI: 10.1016/s0304-3975(95)00257-x.

[6] K. Kozminski and E. Kinnen. "An Algorithm for Finding a Rectangular Dual of a Planar Graph for Use in Area Planning for VLSI Integrated Circuits". In: *21st Design Automation Conference Proceedings*. Institute of Electrical and Electronics Engineers (IEEE), 1984. DOI: 10.1109/dac.1984.1585872.

[7] I. Rinsma. "Nonexistence of a certain rectangular floorplan with specified areas and adjacency". In: *Environment and Planning B: Planning and Design* 14.2 (1987), pp. 163–166. DOI: 10.1068/b140163.

[8] P. Ungar. "On Diagrams Representing Maps". In: *Journal of the London Mathematical Society* s1-28.3 (1953), pp. 336–342. DOI: 10.1112/jlms/s1-28.3.336.