5 Vertical one-sided dual

We can also adapt Fusy's algorithm to generate a vertically one-sided dual. We then need top generate a regular edge labeling without red faces that have 3 or more edges on both borders.

We will have an additional requirement on top of the requirement that $\bar{G}$ has no separating triangles. We will also require that $\bar{G}$ has no separating four cycles.

Notational concerns Just as in Section 4 we will use $\mathcal{C}$ to indicate the current sweep line cycle. We will repeatedly only consider the path $\mathcal{C} \setminus \{\mathrm{S}\}$. In that case we will always order it from W to E.

Instead of interior paths we will consider interior walks but we will use similar notation. That is a walk between two distinct vertices of $\mathcal{C}$ of which all vertices except the first and last one are in the interior of $\mathcal{C}$.

We will let $\mathcal{W}$ denote a interior walk. Given such a walk of k vertices we index it's nodes $w_1, \ldots, w_k$ in such a way that w_1 is closer to W then w_k is (and thus that w_k is closer to E then w_1 is).

Then w_1 and w_k indicate the two unique vertices of the walk that are also part of the cycle. We will then let $\mathcal{C}|_{\mathcal{W}}$ denote the part of $\mathcal{C} \setminus \mathrm{S}$ that is between w_1 and w_k (including). $\mathcal{C}_{\mathcal{W}}$ will denote the closed walk formed when we paste $\mathcal{C}|_{W}$ and $\mathcal{W}$.

Since paths are a subclass of walks all of the above notation can also be used for a path $\mathcal{P}$. Note that the closed walk $\mathcal{C}_{\mathcal{P}}$ in this case will actually be a cycle.

We note that the interior of some cycle $\mathrm{Int}(C)$ are all vertices strictly in the interior of this cycle. We will sometimes also take $\mathrm{Int}(C)$ to refer to the induced subgraph of these vertices.

We let $\mathrm{Int}^{+}(C)$ denote the the vertices of C and C's interior vertices. We will also sometimes let it refer to the subgraph of G induced by these vertices.

Non-crossing walks We will call a walk *noncrossing* if at every vertex w in the walk that is visited k times such that $w_{i_1} = w_{i_2} = \ldots = w_{i_k}$ in the walk the clockwise intervals $[w_{i_j-1}, w_{i_j+1}]$ for $j \in \{1, \ldots, k\}$ are disjoint in the rotation at w.

The nice thing about non-crossing walks is that they if they return to their startpoint they allow a notion of interior and exterior. We can see this by applying Jordans curve theorem to a version of this walk that is very slightly perturbed at every vertex visited multiple times. Which we can do due to the disjoint intervals in the rotation.

FiXme: We might add a (small) figure for clarity (i.e. of a crossing and a non-crossing walk)

Hence we can talk about the interior vertices of a closed non-crosing walk.

FiXme: TODO

FiXme: More general for non-crossing paths

5.1 The neighbor walk of a path

During this proof we will frequently use the concept of the left or right neighbor walk of a path. Given a path $P = p_1 \ldots p_k$ in a graph G The *right neighbor walk* W of P will consist of p_1, we will then take the vertices adjacent to p_2 between p_1 and p_3 in the clockwise rotation at p_2, followed by the vertices between p_2 and p_4 in the rotation at p_3 and so further until we add the vertices between

FiXme: Note that the same/similar things hold for the left boundary walk

p_{k-2} and p_k in the rotation around p_{k-1} and finally we finish by adding p_k to W. We then remove all subsequent duplicates from W

Lemma 22. *The right neighbor walk W is a walk.*

Proof. Let w and w' be two subsequent vertices in W. We will show they are connected. We first consider the case $\{w, w'\} \cap \{p_1, p_k\} = \emptyset$. Now there are two cases. Either (a) w and w' are vertices adjacent to some p_i an thus subsequent in the rotation at p_i or (b) w was the last vertex adjacent to some p_i and thus w' is the first vertex adjacent to p_{i+1}.

The following two situations can also be seen in Figure 8.

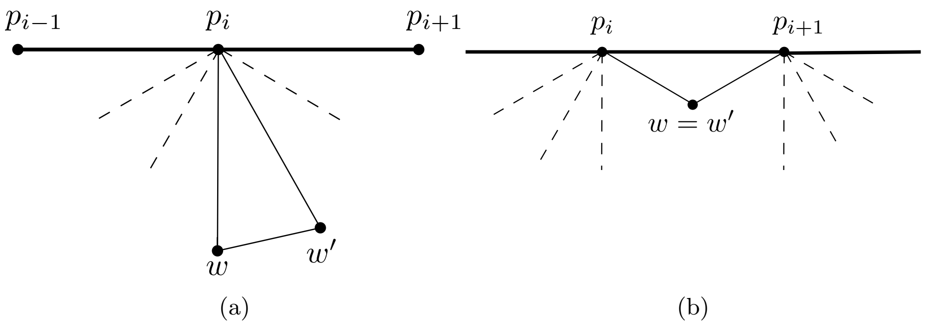


Figure 8: The two main cases of the proof showing that W is a walk

In case (a) we note that since w and w' are subsequent in the rotation at p_i ww' is an edge by Lemma 3.

In case (b) we note that $p_i w$ and $p_i p_{i+1}$ are edges subsequent in clockwise order, hence wp_{i+1} is also an edge. Hence w is the first vertex adjacent to p_{i+1} subsequent to v_i in the clockwise rotation. Thus $w = w'$. They are duplicates and one of them must have been removed.

Now for the edge cases: Let x be the first vertex adjacent to p_{i+1} and let y be the last vertex adjacent to p_{j-1}. p_i and x are vertices adjacent to p_{i+1} subsequent in the clockwise rotation, and hence connected by Lemma 3. In the same way y and v_j are subsequent vertices in the rotation at v_n and hence connected.

Hence $\mathcal{W}$ is a walk. □

Lemma 23. *The right neighbor walk W is a non-crosssing walk.*

Proof. Suppose that the right neighbor walk is crossing at a vertex $w = w_i = w_j$. Then one of w_{j-1} and w_{j+1} is in the clockwise interval $[w_{i-1}, w_{i+1}]$ at the rotation at w. We will denote this vertex by w'. It is clear that w' cannot be on the path unless w' is p_1 or p_k. In this case however we see that w_{i-1} or w_{i+1} respectively couldn't have been part of the path.

So we continue with w' not on the path. All neighbors of w between w_{i-1} and w_{i+1} in the clockwise rotation are on the path. . So we have a series of triangles by Lemma 3. Now w' must be inside one of these triangles, otherwise we would have a crossing edge (and thus a non-planar graph.) Now the triangle containing w is a separating triangle.

FiXme: TODO make this a lemma

We conclude that W must be a non-crossing walk. □

FiXme: Actullly W rev(P)

Lemma 24. *The closed non-crosing walk WP has no interior vertex.*

Proof. The interior of WP consists of only triangles with all vertices in WP. We can see this from the construction of the neighbor walk. Both cases in Figure 8 add a triangle to the interior with all vertices in WP.

Suppose there is a interior vertex. Then the triangle containing this vertex is a separating triangle. □

Lemma 25. *The left of the of a right neighbor walk and the right of the left neighbor walk are chordfree.*

Proof. Suppose that the right neighbor walk $W = w_1 \dots w_k$ has a chord on the left, say between w_i and w_j with $i < j-1$. There is a vertex $p_\ell \in P$ on the path such that w_{i+1} is a neighbor of p_ℓ to the left of p_ℓ Consider now the following non-crossing closed walk $Pw_k \dots w_{j+1}w_jw_iw_{i-1} \dots w_1$ (Thick in Figure 9)this walk has w_{i+1} in its exterior. But then $p_\ell w_{i+1}$ is a crossing edge. Which is forbidden.

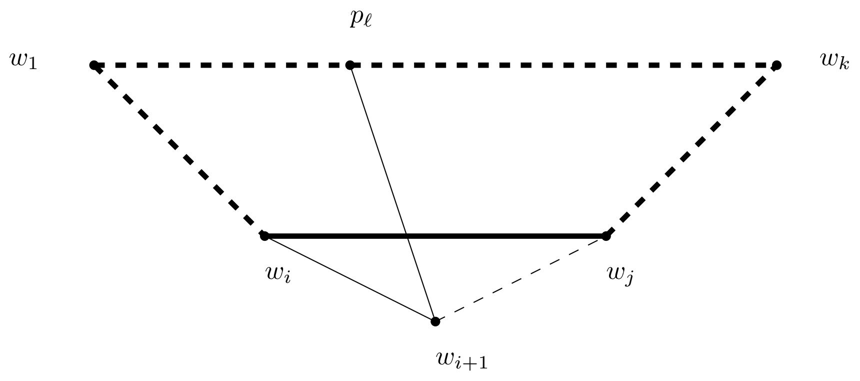


Figure 9: The construction in the proof of Lemma 25

□

FiXme: Alternate proof based on ($\mathcal{W}$ being oriented from W to E), since if it would lie on the left of $\mathcal{W}$ the vertices $w_{i+1}, \dots, w_{j-1}$ would not have been chosen in the construction of the prefence.

5.2 Outline

We will assume we have no separating 3 or 4 cycles.

To describe the algorithm two more definitions are necessary

Definition (Prefence)**.** A prefence $\mathcal{W}$ is a interior walk of $\mathcal{C}$ starting at $v_i \in \mathcal{C}$ and ending at $v_j \in \mathcal{C}$ a both adjacent to S

(P1) $\mathcal{C}_\mathcal{W}$ Has no interior vertex

(P2) $\mathcal{W}$ has no chords on the left

(P3) $\mathcal{C}|_\mathcal{W}$ has no chords on the right

We enforce these conditions because they imply (E3) as we will show in Lemma 27.

For a walk however the interior is not clearly defined.

Definition (Fence)**.** A fence is a valid path starting and ending at a vertex adjacent to S

The algorithm will receive as input a extended graph $\bar{G}$ and will return a regular edge labeling such that all red faces are $(1-\infty)$ using a sweep-cycle approach inspired by Fusy [1].

FiXme: expand on naming/reasons of fence

We will start by creating a prefence W. This may not be a valid path, it doesn't even have to be a path. During the algorithm we will make a number of moves that will turn this prefence into a fence. In each move we shrink C by employing a valid paths and change the prefence.

5.3 Finding a initial prefence

Let v_i denote all the vertices of $\mathcal{C}\backslash\{\mathrm{S}\}$ in the following order $\mathrm{W} = v_1\, v_2\, \ldots v_{n-1}\, v_n =$ E. Some intervals of these vertices will be adjacent to S. However, they can't be all adjacent to S since then the sweepcycle would be non-separating since we can't have separating triangles. We denote by v_i the last vertex of fist interval of vertices adjacent to S and by v_j the first vertex of the second interval. As candidate walk we will take the right neighbor path of $C\backslash\{S, v_1, \ldots v_{i-1}, v_{j+1}, \ldots v_n\}$.

FiXme: beter notation

Lemma 26. *The collection W described above is a prefence.*

Proof. W is a walk by lemma 22.
We note (P1) holds due to Lemma 24
We note (P2) holds due to invariant 17 (I2).
We note (P3) holds due to Lemma 25

□

We then orient $\mathcal{W}$ from v_i (the vertex closest to W)to v_j (the vertex closest to E) and denote it's vertices by $w_1 \ldots w_k$.

5.4 Irregularities

We will distinguish two kinds of *irregularities* in a prefence.

1. The candidate walk is non-simple in a certain vertex. That is, if we traverse the sequence of vertices in $\mathcal{W}$ we see that $w_i = w_j$ for some $i < j$.
2. The candidate walk has a chord on the right. That is, there is an edge $w_i w_j$ on the right of $\mathcal{W}$ with $i < j$ and i and j not subsequent (i.e. $i < j-1$).

Note that we can't have a chord can on the left of $\mathcal{W}$ by Lemma 25.

Lemma 27. *If a prefence has no irregularities it is a fence.*

Proof. We will show that all the requirements of being a valid path are met.

Path Let us begin by noting that since there are no non-simple points we have a path and not just a walk.

(E1) It is clear that both w_1 and w_k are not S by the construction of the candidate walk.

(E2) For $\mathcal{W}$ or $\mathcal{C}|_{\mathcal{W}}$ to have only one edge we need to have that $v_i v_j$ is an edge. However, $v_i v_j$ can not be an edge in $\mathcal{C}$ since v_i and v_j are from different intervals of vertices adjacent to S. It can also not be an edge in $\bar{G} \setminus \mathcal{C}$ since that would be a chord of the cycle and these don't exist by Invariant 17 (I2)

(E3) This holds because every interior edge that does not lead to an interior vertex, nor is a chord of $\mathcal{W}$ or $\mathcal{C}|_{\mathcal{W}}$ is of the required type. These three requirements are forces by (P1), (P3) and (P2) respectively.

(E4) The cycle $\mathcal{C}'$ only changes between v_i and v_j. There can be no chord with one vertex from cycle $\mathcal{C} \setminus \mathcal{C}|_{\mathcal{W}}$ and one from $\mathcal{W}$ since such a chord would cross Sv_i or Sv_j. There is no chord with two vertices in $\mathcal{W}$ since that would be a irregularity and there is no chord with two vertices from $\mathcal{C} \setminus \mathcal{C}|_{\mathcal{W}}$ by Invariant 17 (I2).

Hence, if $\mathcal{W}$ has no irregularities it is a valid path.

Furthermore, $\mathcal{W}$ is a path starting and ending at a vertex adjacent to S because it is prefence. And thus it is a fence. □

Definition (Range of a irregularity)**.** For a non-simple point $w_i = w_j$ with $i < j$ has *range* $\{i, \ldots, j\} \subset \mathbb{N}$. A chord $w_i w_j$ with $i < j - 1$ has *range* $\{i, \ldots, j\} \subset \mathbb{N}$.

FiXme: Is it better to call this a non-simple point or a non-simple vertex?

Note that a chord can't have the same range as a non-simple point since then $w_i w_j$ will be a loop and we are considering simple graphs. Furthermore two chords have different ranges because we otherwise have a multiedge. Two nonsimple points with the same range are, in fact, the same. This leads us to the following remark.

Remark 28. *Distinct irregularities have distinct ranges.*

Definition (Maximal irregularity)**.** A irregularity is maximal if it's range is not contained[8] in the range of any other irregularity.

Lemma 29. *Maximal irregularities have ranges whose overlap is at most one integer.*

Proof. We let I and J denote two distinct maximal irregularities with ranges $\{i_1, \ldots i_2\}$ and $\{j_1, \ldots, j2\}$. Let us for the moment suppose that I and J have ranges that overlap more then one integer. Since I and J are both maximal their ranges can not be contained in each other.

Without loss of generality we thus have $i_1 < j_1 < i_2 < j_2$.

Now two chords to the right of $\mathcal{W}$ would cross each other but we have a planar graph so this can't be the case.

Now let us without loss of generality suppose that I is a non-simple point. A non-simple point $w_{i_1} = w_{i_2}$ is adjacent to two ranges of vertices in $\mathcal{C} \setminus \{\text{S}\}$. $v_a \ldots v_b$ and $v_c \ldots v_d$ then $\tilde{C} = w_{i_1} v_b \ldots v_c$ is a cycle. And because of the rotation at $w_{i_1} = w_{i_2}$ we have that $w_{i_1+1}, \ldots, w_{i_2-1}$ are inside this cycle while $w_1 \ldots w_{i_1-1}$ and $w_{i_2+1} \ldots w_k$ are outside the cycle. See Figure.

Now if J is a chord we have $\tilde{C}$, which can't be. If J is also a nonsimple point this would imply that the vertex $w_{j_i} = w_{j_2}$ is at the same time inside and outside $\tilde{C}$ which is clearly impossible. □

FiXme: We could add figure to clarify.

[8]Because of Remark 28 being contained is the same as being strictly contained

5.5 Moves

The algorithm will remove these irregularities by recursing on a subgraph for each maximal irregularity. We shrink the cycle $\mathcal{C}$ with every valid path that is found in the recurrence, in the order they are found. Afterwards we update the prefence by removing $w_{i+1}, \ldots, w_{j-1}$. In subsection 5.5.3 we will show that the updated prefence is a prefence for the updated cycle $\mathcal{C}$.

We will first show how to remove these maximal irregularities in Subsections 5.5.1 and 5.5.2. That is, we show which *recursion subgraph* H we recurse upon for both kinds of irregularity. Furthermore we show that these subgraphs suffice the requirements of the algorithm.

Afterwards, in subsection 5.5.3 we will make sure that the subgraphs we recurse upon are edge-disjoint. That is, they only overlap in border vertices.

It is worth noting that other irregularities contained in such a maximal irregularity are solved in the recurrence.

5.5.1 Chords

If we encounter a chord we will extract a subgraph and recurse on this subgraph. A chord $w_i w_j$ has a triangular face on the left and on the right (like every edge). The third vertex in the face to the left will be called x. x is not necessarily distinct from w_{i+1} and/or w_{j-1} but this is also not necessary for the rest of the argument.

The vertex v_a on the cycle is uniquely determined as the vertices adjacent to both w_i and $wi + 1$. In the same way v_b is the unique neighbor of w_{j-1} and w_j.

FiXme: We might also work these out in a Figure.

We will describe a walk $\mathcal{U}$ running from v_a to v_b. This path consists of all vertices adjacent to w_i in clockwise order from v_a (inclusive) to x(inclusive) and subsequently all vertices adjacent to w_j in clockwise order from x (exclusive) to v_b (inclusive). This path is given in bold in Figure 10.

Note that $\mathcal{U}$ is the left neighbor walk of $v_a w_i w_j v_b$.

Lemma 30. *$\mathcal{U}$ is a chordfree path*

Proof. We note that $\mathcal{U}$ is a walk due to Lemma 22.

$\mathcal{U}$ cant have a non-simple point x' since it would have to be connected to at least two vertices. However a vertex x' that is distinct from x and is connected to both w_i and w_j will induce a separating triangle $w_i x' w_j$. $\mathcal{U}$ also can't be nonsimple at x since x is the the third vertex of the triangular face $w_i w_j x$. Hence $\mathcal{U}$ is a path.

$\mathcal{U}$ can't have chords $u_i u_j$ since they would either induce a separating 3- or 4-cycle either $w_i u_i u_j$ or $w_j u_i u_j$ or $w_i u_i u_j w_j$ depending on the vertex adjacent to u_i and u_j. □

FiXme: We use that we have no 4-cycles here

We take $\mathrm{Int}^+(\mathcal{C}_\mathcal{U})$ as the subgraph H. We then take the tight extension at v_a and v_b. We will then recurse on this graph $\bar{H}_t$. See also Figure 10. Since $\mathcal{C}$ is chordfree by invariant 17 (I2) so is $\mathcal{C}|_\mathcal{U}$. We have also just shown that $\mathcal{U}$ is chordfree. So $\bar{H}_t$ is indeed defined. Furthermore, since H is a induced subgraph of G, $\bar{H}_t$ contains no separating 4-cycles not involving the poles.

FiXme: ALSO WITH POLES IS STILL BAD

We update the prefence by removing $w_{i+1}, \ldots, w_{j-1}$.

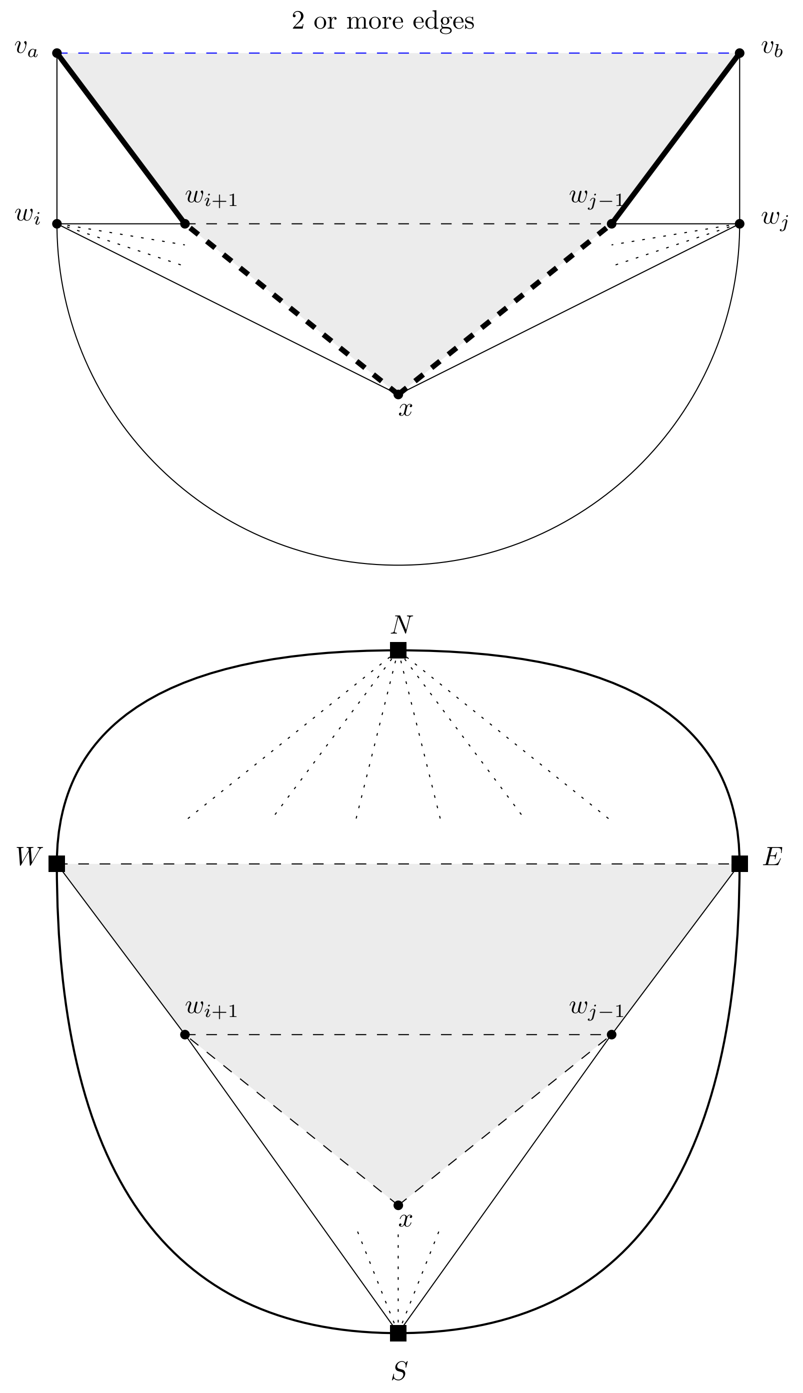


Figure 10: Removing a chord

5.5.2 Nonsimple points

Removing a non-simple point is done is a similar manner.

The vertex v_a on $\mathcal{C}$ is uniquely determined as the vertices adjacent to both $w_i = w_j$ and $wi + 1$. In the same way v_b is the unique neighbor of w_{j-1} and $w_j = w_i$. Note that it may be that $w_{i+1} = wj - 1$ this does not matter for the rest of the argument.

We will describe a walk $\mathcal{U}$ running from v_a to v_b. This path consists of all vertices in the rotation at $w_i = w_j$ from v_b (inclusive) to v_a(inclusive). This path is given in bold in Figure 11.

FiXme: We may show this in a figure.

Note that $\mathcal{U}$ is the left neighbor walk of $v_a w_i v_b$.

Lemma 31. *$\mathcal{U}$ is a chordfree path.*

Proof. We note that $\mathcal{U}$ is a walk due to Lemma 22.

If we orient $\mathcal{U}$ from v_a to v_b we see that $\mathcal{U}$ can't have a non-simple point on the left by construction and by the right since such a point would have edges to at least two vertices on the right. However every vertex can only be connected to $w_i = w_j$. Hence $\mathcal{U}$ is a path.

$\mathcal{U}$ can't have chords on the right of the path by the way we construct $\mathcal{U}$. Furthermore $\mathcal{U}$ can't have chords $u_i u_j$ on the left since they would either induce a separating 3-cycle $w_i u_i u_j$. □

We then take $\text{Int}^+(\mathcal{C}_{\mathcal{U}})$ as the subgraph H.

We then take the tight extension of H at v_a and v_b to recurse on. See also Figure 11. Since $\mathcal{C}$ is chordfree by Invariant 17 (I2) so is $\mathcal{C}|_{\mathcal{U}}$. We have also just shown that $\mathcal{U}$ is chordfree. So $\bar{H}_t$ is indeed defined. Furthermore, since H is a induced subgraph of G, $\bar{H}_t$ contains no separating 4-cycles not involving the poles.

We update the prefence by removing $w_{i+1}, \ldots, w_{j-1}$ and we also recognize that $w_i = w_j$ is now a duplicate subsequent occurrence of the same vertex. So we also remove w_j.

FiXme: ARRARGH also poles are not allowed

5.5.3 Validity

Lemma 32. *After doing a move the updated prefence $\mathcal{W}'$ is a prefence for the updated cycle $\mathcal{C}'$*

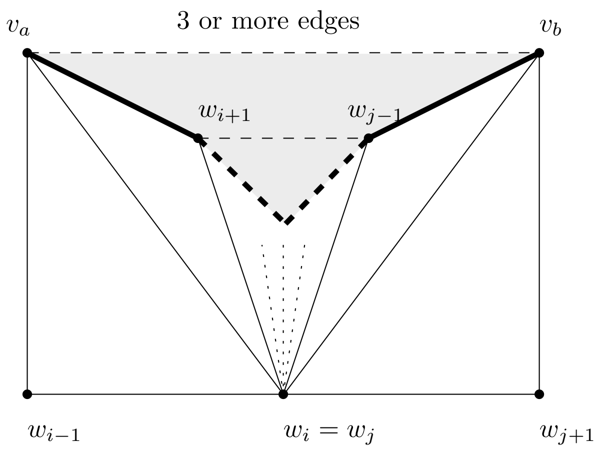
Proof. We want to show that the three properties of a prefence again hold for $\mathcal{W}'$.

We will first show (P1) still holds. We note we can cut the interior of $\mathcal{C}'_{\mathcal{W}'}$ into three parts: $w_1 \ldots w_i v_a v_{a-1} \ldots v_{i+1} w_1, \mathcal{U} w_j w_i$ (or $\mathcal{U} w_i$) and $w_j \ldots v_j \ldots v_{b+1} v_b w_j$. Note that the first and last part are free from interior vertices because they already were before the move by (P1). As for the middle part since $\mathcal{U}$ is the left neighbor path of $v_a w_i w_j v_b$ (or $v_a w_i v_b$) this is also without interior vertices by Lemma 24.

Since in the recursion the cycle $\mathcal{C}$ is only updated with valid paths it remains a valid cycle. Thus Invariant 17 (I2) still holds for $\mathcal{C}'$ and thus implies (P2).

Furthermore we can easily see (P3) holds. Suppose we have a chord on the left of $\mathcal{W}'$ then this would already have been a chord of $\mathcal{W}$ however $\mathcal{W}$ has no chords due to (P3). □

Lemma 33. *Let H_I and H_J be two recursion subgraphs for different maximal irregularities I and J. Then H_I and H_J are edge disjoint.*



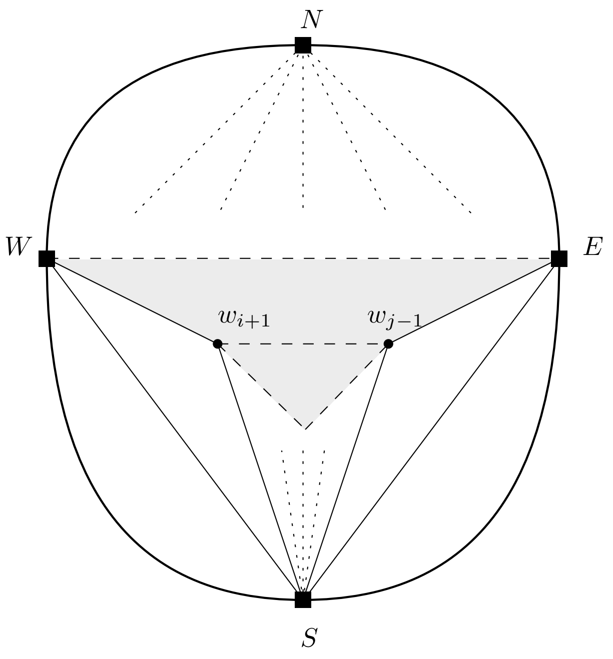


Figure 11: Removing a non-simple point

Proof. Maybe the best proof is that since maximal ranges of irregularities overlap at most 1 then the cycles $v_{a_I} \dots v_{b_I} w_{j_I} w_{i_I}$ and $v_{a_J} \dots v_{b_J} w_{j_J} w_{i_J}$ with v_{a_I} being the unique neighbor on the cycle of w_{i_I} and w_{i_I+1} and .... don't cross

Then the left neighbor paths U_I and U_J are edge disjoint □

FiXme: TODO fix this lemma

5.6 Correctness

As long as the interior of $\mathcal{C}$ is nonempty we can find a prefence. And thus we find valid paths. Since we continuously shrink the cycle with valid paths we end up with a regular edge labeling. See core algorithm

The algorithm finishes because it keeps on recursing and shrinking until no graph is left.

5.6.1 The red faces

Let us then argue that the red faces are all $(1-\infty)$ faces, corresponding to one-sided vertical segments. As is shown in Lemma 10 it is sufficient to show no two vertices subsequent on a blue path are first a merge and then a split or vice versa.

We will show the following

Lemma 34. *A split or merge always happens on a vertex that is adjacent to S for some recursion.*

Proof. Every valid path we shrink the cycle by is found as a fence on some recursion level. In this recursion level both w_1 and w_k are adjacent to S. □

Lemma 35. *A path starting at a certain recursion level will stay at that recursion level. It may share vertices with the north boundary of a lower recursion level but never with the south boundary.*

Proof. A valid path can never leave the subgraph H in which its start- and end-vertex are located. Because it is found as a fence in this subgraph. It can also never run trough a graph H' on a lower recursion level (except for the north boundary path) because in every move the vertices of the prefence in H' are deleted. □

Recall that all our valid paths are oriented from a start vertex to end vertex.

Lemma 36. *A split can't directly be followed by a merge along any valid path during the algorithm.*

Proof. One of paths after the split is no longer on the south boundary of this subgraph H, nor on the south boundary of any other subgraph by Lemma 35. This path hence can't contain a merge.

The other path still potentially follow the south boundary. However merging from the southward side of the path is impossible by Lemma 35 from the northward side is equally impossible since the split and merge have to be neighboring vertices in the rotations of these vertices and thus the path $\mathcal{P}$ that merged must also join again.

But then it is not a valid path. □

However for a blue Z to occur there has to be a valid path that first has a split and then has a merge. Since this can't be all red faces must have only 2 edges on at least one side. Hence the regular edge labeling this algorithm produces corresponds to a vertically one-sided rectangular dual.

FiXme: Show how the algorithm works with some cool examples: For example: The multiple non-simple point $v_i = v_j = v_k$; Example of page $F1$; Example with lots of layered chords

6 Unified Algoritmh

FiXme: For now this chapter assumes we did not need to make moves when creating fences

FiXme: For now this chapter assumes we did not need to make moves when creating fences

Now we have presented sweep-line based algorithms that preserver vertical and horizontal onesidedness respectively. We will now introduce an algorithm that gives a (k, ∞)-sided rectangular dual for any graph without separating 4-cycles. We will show an algorithm exists with $k = 17$.

6.1 Fans

In order to effectively describe the algorithm and its proof we will have to introduce some more concepts.

Consider a blue face F in between two fences, we will call such a face a *strip*. Every interior edge of this face goes from one fence to the other (due to property (E3)). To better understand the structure of such a strip we will describe the edges from split(F) to merge(F) .

Let $u_0, u_1, \ldots u_n$ be the vertices of the upper boundary path of F and $v_0, v_1, \ldots, v_m$ the vertices of the bottom boundary path. That is $u_0 = v_0 =$ split(F) and $u_n = v_m =$ merge(F). Since our graph is a triangulation u_1v_1 must be an edge. For the second edge in the face we have two options u_1v_2 or u_2v_1, otherwise this edge and the previous one would not form a triangle. This principle holds for every subsequent edge, we can either increase a the index of the upper boundary path or the index of the bottom boundary path.[9]

We will call a maximal sequence of at least two edges increasing the index on the bottom boundary path (and thus keeping the index on the upper path fixed) a *Bottom-fan* or simply *B-fan* and a maximal sequence of at lest two edges increasing the index on the upper boundary path will be called a *Top-fan* 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 will simply use *fans* to refer to both these *types* of fans (i.e. T- and B-fans).

FiXme: Or series of edges? It is not a sequence in the usual meaning.

We will call a fan of size 5 or larger a *large fan*. Then it is only natural that we call fans of size 2, 3 or 4 *small fans*. This distinction may seem arbitrary but turns out to be very useful in the proof.

FiXme: Emphize fan in *large fan* or not?

In a strip we alternately encounter B- and T-fans. Since if we would have two adjacent fans of the same type we would just have a single larger fan of that type. In Figure 12 we see a strip 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.

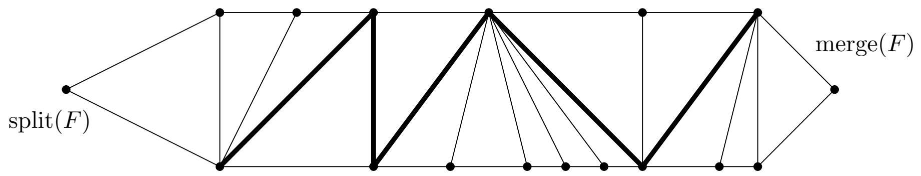


Figure 12

[9]For the benefit of the readers that know this concept; Our strip is a *triangle strip*.

We will sometimes describe a strip using a compact notation. In this notation we will denote a fan by $\text{Type}_{\text{Size}}$ so B_3 will be B-fan of size 3 and T_2 a T-fan of size 2. The strip in Figure 12 can be characterized by $B_2T_2B_2T_5B_3T_3$ in this notation.

We will expand this notation with special characters $(+, -, ?, \infty)$ to denote a fan without knowing it's exact size. We will let T_- or B_- imply a small fan, T_+ or B_+ a large fan, $T_?$ or $B_?$ a fan of any size and T_∞ or B_∞ an arbitrarily large fan. [10]

We introduce some more terminology for fans: *outer edges*, *fan handles* and the *rim*

FiXme: TODO

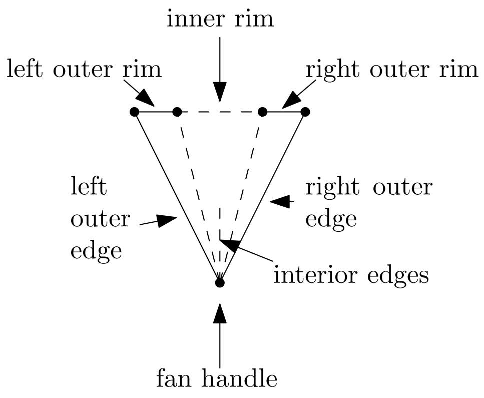


Figure 13

Note that the only edges that are flippable are those that are in both a T-fan and a B-fan. This are exactly the outer edges of any fan.

FiXme: We might want to show this

6.2 Blue Z's and loaded edges

During the algorithm we will often recolor a red edge into a blue edge. When we do this we create a structure that we will refer to as a *blue* Z. A blue Z is a path of three blue edges all in the same red face with it's first and last edge on a fence. When we recolor a red edge in a strip into we create at least on such a blue Z with it's *top* and *bottom* edges on the two different fences bordering this strip. Its *middle* edge is the recently recolored edge.

FiXme: TODO adept for moves

We refer to two blue Z's as chained when the bottom edge of one Z is the same as the top edge of the other. Chained blue Z's are something we want to prevent because they indicate a large red face.

FiXme: explain this better?

In order to prevent to many blue Z's from chaining we use the following technique in our algorithm. When we recolor an edge we say that the bottom edges of any blue Z containing this edge become *loaded*. Any edge that is not (yet) loaded is referred to as a *free* edge.

6.2.1 Usefull lemma's

Before we describe the algorithm it is handy that we already show the following lemma's

[10]this is useful in counterexamples.

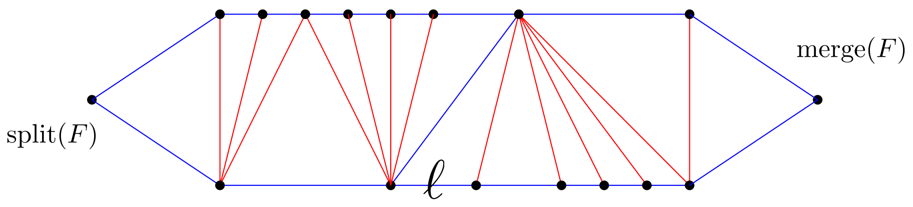


Figure 14: The algorithm has decided to flip an edge. The bottommost edge of the Z is marked as loaded.

Lemma 37. *A graph without overlapping blue Z's is vertically* $(2, \infty)$ *sided.*

This depends how we define overlap, it seems true for vertex overlap

Lemma 38. *Above B-fan we have fans of size* 2

Proof. Otherwise we create a 4-cycle □

6.3 Algorithm

Outline The approach taken by this algorithm will be the following

1. Subdivided the graph by fences as in the red algorithm.
2. We separate big B-fans from the rest of the fans in this strip
3. If blue faces in this strip are still too long we will flip some of their interior edges. We try to prevent the creation of new chained blue Z's.
4. We solve the blue Z's we have created by doing a few flips.

order of strips There is a partial order of strips. we start with the largest element and can always treat such a strip such that all strips larger than that strip are already treated while all strips smaller than that strip are note yet treated.

FiXme: TODO Does this order exists, find ref in literature

6.3.1 Phase 1: Finding fences

6.3.2 Phase 2: Removing large B-fans

While deciding which edges to flip on each strip we will maintain the following invariants

Invariants 39

(I1) The edges on a fence obey the following:

- There are no more then 2 subsequent loaded edges
- 1 loaded edge followed by at least 1 unloaded edge
- 2 loaded edges followed by at least two unloaded edges

Lemma 40. *We can create blue* $(9, \infty)$*-faces containing all the large B-fans while obeying Invariant 39.*

Proof. We will begin scanning at the split of the strip when we encounter a large B-fan (B_+) we will enter a multistage case distinction ending with a small enough blue face and and edges on the bottom fence of the strip satisfying Invariant 39.

Note that in all cases we flip the left outer edge of the first B_+ we encounter when scanning. Also note that we only ever flip edges bordering a large B-fan.

Let us first inspect the 3 T-fans following our B_+. If any of them is size 5 or larger we flip the right outer edge of the last large B-fan before this T-fan and continue our scan from the fan onward. In this case the largest face we create is $(7, \infty)$.

FiXme: TODO insert figure of situstion

Otherwise all three T-fans following our B_+ are small, we have the following sequence $B_+T_-B_?T_-B_?T_-B_?$. If the rightmost B-fan is large we continue our scan at this B-fan, creating a $(9, \infty)$- face. Otherwise, if the second-right most B-fan is large we continue our scan there creating a $(6, \infty)$-face.

Otherwise we check whether the third-rightmost B-fan is large if this is the case we flip it's **right** outer edge and continue our scan after this B-fan. Otherwise we also flip the **right** outer edge of the original B_+. We continue the scan right after this flipped edge. In both cases we can do this without offending Invariant 39 since the next two/three B-fans are small so the next one/two edges will not be loaded even when continuing the scan immediately after. These last two steps yield faces of size at most $(4, \infty)$.

One can also see the order in which we try to place edges in Figure 15

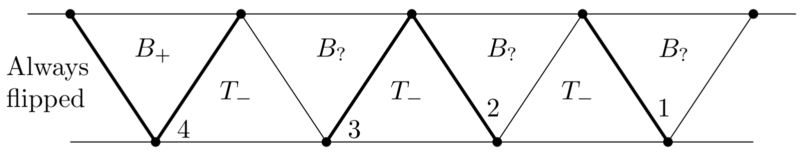


Figure 15: Order in which we try to flip edges. We only flip an edge if it neighbors a large B-fan.

Trough this case distinction we've seen that we indeed can create blue $(9, \infty)$-faces containing all the large B-fans while obeying Invariant 39.

□

Note that we only ever flipped outer edges of a B_+ in this proof

6.3.3 Phase 3: The rest of the strip

Any further blue faces that are still to large do not contain any large B-fans. We will be careful to flip only if we satisfy the following invariants in addition to Invariants 39

Invariants 41

(I1) We don't flip an edge creating a blue Z containing a loaded edge.

(I2) We don't flip an edge creating a blue Z containing a pad of a large B-fan in the next strip.

We consider two cases for our still to large blue face F, either this blue face is above the entirety of a large B-fan in the next strip or it isn't. In the first case we can find a flippable edge satisfying all the invariants and we are done.

FiXme: TODO give these a proper ref and environment (beore starting real proofs)

Lemma 42. *We can flip at least one edge above in any blue face that lies above the entirety of a large B-fan in the next strip.*

FiXme: This lemma should probably change when we start to allow moves

FiXme: I could/should make this a lemma somewhere

Proof. In the strip above a large B-fan and directly above such a large B-fan any T-fan can only be of size 2 since otherwise we would have a separating 4-cycle . Furthermore there can't be large B-fan in the face we are treating since all large B-fans are in $(9, \infty)$ faces. We are thus in the situation depicted in Figure 16.

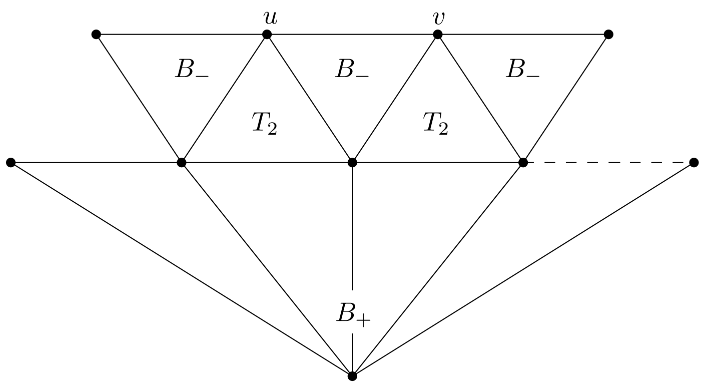


Figure 16

Now suppose one of the edges neighboring u is not loaded then we can flip here. Now let us assume both edges incident to u on the fence are loaded then at least one of the edges of the fence incident to v is unloadned because the next two edges are unloaded due to Invariant 39.

Hence we can alway flip an edge above a large B-fan. □

In the second case we don't have to consider more then one right pad and more the one left pad otherwise we could execute Lemma 42

Lemma 43. *In any blue face that is not a* $(31, \infty)$ *blue face we can flip a edge.*

Proof. The face under consideration does not lie above the enirty of any B-fan becuase of Lemma 42.

Such a face that is not $(31, \infty)$ has at least 32 vertices on the top fence. This means it has at least 31 edges lying in at least 11 B-fans. These B-fans have 10 inbetween T-fans. The two T-fans closest to the both border of the face will remain unloaded in order to preserve Invariant 39. This leaves 6 fans. One pad or two loaded edges can possibly block an entire T-fan. We can't block two subsequent T-fans with loaded edges due to the fact that only small B-fans. We

FiXme: TODO fix this on right side only need one edge of seperation only have two pads. So one of the fans has to remain free and we can thus flip an edge satisfying both Invariants 39 and 41. □

6.3.4 Phase 4: Flips

After we have treated all strips the only chained blue Z's are those that have their middle edge (the one crossing a strip) as outer edge of a large B-fan. This because the rest of the edges has been placed obeying the properties above.

We then have essentially two cases which occur around both the left and right outer rim of the large B-fan. Giving us a total of at most 4 flip operations above a large B-fan.

FiXme: TODO argue this

They are depicted in Figures 17 and 19 and we treat these as depicted in Figures 18 and 20.

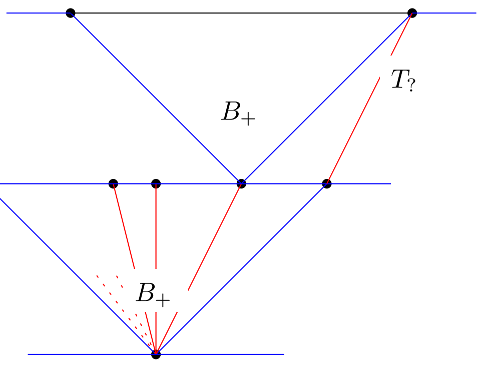


Figure 17: Case a) before the flip

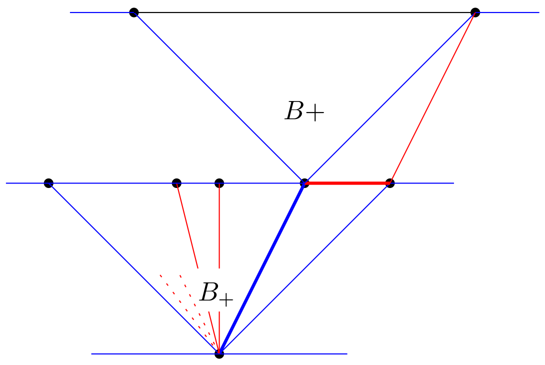


Figure 18: Case a) after the flip

We will now also describe how each flip is executed and that each flip individually maintains a regular edge labeling .

Flip type a)

FiXme: TODO

Flip type b) It can be that a flip of case a) necessitates doing a flip of type b). However then we are done.

FiXme: TODO

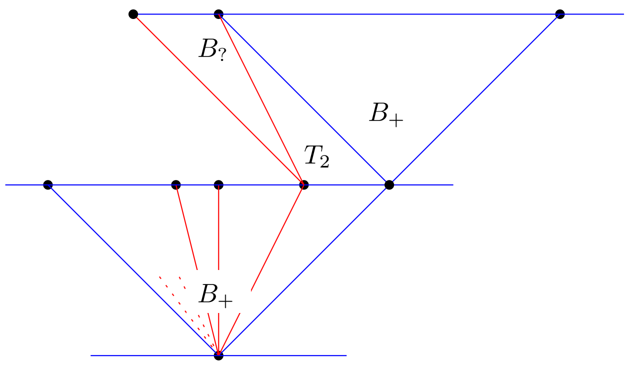


Figure 19: Case b) before the flip

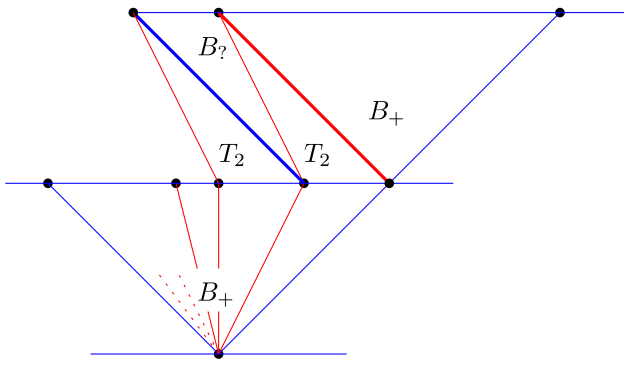


Figure 20: Case b) after the flip

Above a B-fan of size 5 or 6 we would not produces a legal regular edge labeling if we do all 4 flips. Since the B-fan would become only size 3 after the two *a*) flips and we need a B-fan of size 5 to do both *b*) flips without them interfering.

Lucky enough for us this is not a problem as the Algorithm proposed in Lemma 40 does only color all 4 edges necessitating a flip blue above B-fans of size 7 or more which we will show in Lemma below.

Lemma 44. *Doing these flips we remove all chained blue Z's in a finite amount of flips*

Proof.

1. Every chain of Z's only occurs if both middle edges are adjacent to a large B-fan.
2. We can break such a chain in the second Z for every chain doing only at most 4 flips for every large B-fan.
3. Doing any or all of these flips on a B-fan of size 7 or more yields a valid regular edge labeling
4. On a B-fan of size 5 or 6 at most three flips are available and necessary. We can do any or all of them and yield a valid regular edge labeling
5. After doing all available flips for a B-fan there is no longer any chain of blue Z's using the edges of that B-fan.

6. Doing these flips only increases the length of any face by 2. (what is the length of a face)

Every chain of Z's only occurs if both middle edges are adjacent to a large B-fan A chain of Z's can only start if a first Z is followed by a second Z. By Invariant 41 (I1) this can only happen when placing edges around a large B-fan. The middle edge of the second Z is thus adjacent to a large B-fan. The middle edge of the first Z is also adjacent to large B-fan since it also can't have been placed in the second phase of the algorithm due to Invariant 41 (I2).

We can break such a chain in the second Z for every chain doing only at most 4 flips for every large B-fan Given any chain of blue Z's we will try to do a flip in the disrupt the chain. Given any large B-fan there are at most two edges of this B-fan colored blue (the outer edges). Only these two edges can be middle edges for some blue Z. The edges on the strip above this B-fan can only be above this Z in one way (either type a) or b)) without creating a monocolored triangle (which is forbidden). However doing a flip of type a) may then necessitate a flip of type b).

Hence we might need 4 flips in total.

FiXme: Figure?

Doing any or all of these flips on a B-fan of size 7 or more yields a valid regular edge labeling It is clear that every flip on it's own changes a valid regular edge labeling into another valid regular edge labeling . When we inspect the descriptions of the flips we see that they do not interfere with each other.

FiXme: Expand ?

On a B-fan of size 5 or 6 at most three flips are available and necessary. We can do any or all of them and yield a valid regular edge labeling Suppose we force these 4 flips to happen on B-fan of size 5 or 6 then the fence above this B-fan offends Invariant 39

FiXme: Figure?

After doing all available flips for a B-fan there is no longer any chain of blue Z's using the edges of that B-fan. It is clear that each flip break the original chain Argue that flips don't introduce (too long) chains

Doing these flips only increases the length of any face by 2. It is clear every flip only increase face length with 1.

Argue that every face is in at most 2 flips.

FiXme: TODO Make this precise for red/blue faces

□

Then after doing these flips we don't have any subsequent blue Z's Except for after a flip of type b) While we sometimes have enlarged faces with 1. Hence the final coloring has $(3, \infty)$ red faces and $(33, \infty)$ blue faces.

FiXme: I feel the order of treatment of strips is important and i've not yet used it.

References

[1] 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.