Short Topological Decompositions of Non-Orientable Surfaces

Niloufar Fuladi
LIGM, CNRS, Univ. Gustave Eiffel, ESIEE Paris, F-77454 Marne-la-Vallée, France

Alfredo Hubard
LIGM, CNRS, Univ. Gustave Eiffel, ESIEE Paris, F-77454 Marne-la-Vallée, France

Arnaud de Mesmay
LIGM, CNRS, Univ. Gustave Eiffel, ESIEE Paris, F-77454 Marne-la-Vallée, France

Abstract

We investigate short topological decompositions of non-orientable surfaces and provide algorithms to compute them. Our main result is a polynomial-time algorithm that for any graph embedded in a non-orientable surface computes a canonical non-orientable system of loops so that any loop from the canonical system intersects any edge of the graph in at most 30 points. The existence of such short canonical systems of loops was well known in the orientable case and an open problem in the non-orientable case. Our proof techniques combine recent work of Schaefer-Štefankovič with ideas coming from computational biology, specifically from the signed reversal distance algorithm of Hannenhalli-Pevzner. This result confirms a special case of a conjecture of Negami on the joint crossing number of two embeddable graphs. We also provide a correction for an argument of Negami bounding the joint crossing number of two non-orientable graph embeddings.

2012 ACM Subject Classification Mathematics of computing → Geometric topology; Mathematics of computing → Graphs and surfaces

Keywords and phrases Computational topology, embedded graph, non-orientable surface, joint crossing number, canonical system of loop, surface decomposition

Digital Object Identifier 10.4230/LIPIcs.SoCG.2022.41

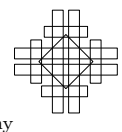
Related Version *Full Version*: `https://arxiv.org/abs/2203.06659` [9]

Funding This work was partially supported by the ANR project SoS (ANR-17-CE40-0033).

Acknowledgements We are grateful to Marcus Schaefer and Daniel Štefankovič for providing us the full version of [25], to Francis Lazarus for insightful discussions, and to the anonymous reviewers for very helpful comments.

1 Introduction

Decomposing a surface along a graph or a curve is a standard way to simplify its topology. The classification of surfaces and classical tools to compute both homology groups and fundamental groups typically rely on such topological decompositions, which are also important in meshing and 3D-modeling (see for example [27]). Surfaces often come with extra structure which can be modeled by an embedded graph. Decomposing such a surface efficiently then means finding a graph that intersects the original graph transversely and that does not intersect the embedded graph too much but nevertheless carries the topological complexity of the surface, as is done for example in [18] or [8]. Such decompositions also appear in algorithm design: often, to generalize results on planar graph to graphs embedded on surfaces, it is enough to find a decomposition that cuts open the surface into a disk, then solve the resulting planar instance and stitch back the solution, see, e.g., [4, 7, 18]. Sometimes it is important that the cut graph is canonical in some way, e.g. in order to compute a homeomorphism between two surfaces, one cuts them into disks, puts these disks in correspondence, and glues back the surfaces. This works only if the cut graphs have the same combinatorial structure.




Lazarus, Pocchiola, Vegter and Verroust [18] (see also [17]) were the first to design an algorithm that finds, for any graph G embedded in a closed orientable surface S a *canonical system of loops* H such that no edge of H intersects[1] any edge of G more than a constant number of times. Here, by a canonical system of loops we mean a one-vertex and one-face embedded graph in which the cyclic ordering of the edges around the vertex is $a_1b_1a_1^{-1}b_1^{-1}\dots a_gb_ga_g^{-1}b_g^{-1}$. Such a decomposition is an instance of the joint crossing number problem. More precisely, consider a pair of graphs G_1 and G_2 embedded on a surface S of genus g and define the *joint crossing number* as the minimal number of crossings between $h(G_1)$ and G_2 over all the homeomorphisms $h : S \to S$. This quantity was introduced by Negami [22] who made the following conjecture:

▶ **Conjecture 1.** *There exists a universal constant C such that for any pair of graphs G_1 and G_2 embedded on a surface S, the joint crossing number is at most $C|E(G_1)||E(G_2)|$.*

Furthermore, he proved an upper bound of $Cg|E(G_1)||E(G_2)|$. His conjecture has been investigated further [1, 15, 23] and variants of this problem have appeared in works with applications as diverse as finding explicit bounds for graph minors [10] or designing an algorithm for the embeddability of simplicial complexes into $\mathbb{R}^3$ [19].

In the non-orientable case, no instance of Negami's conjecture seem to be known. Even if G_1 is the *non-orientable canonical system of loops*, that is, a system of one-sided loops with the cyclic ordering $a_1a_1a_2a_2\dots a_ga_g$ around the vertex, the best known bound is $O(g|E(G_2)|)$ crossings for each loop (see [17]), which matches Negami's bound. Non-orientable surfaces have been often somehow neglected in computational topology, but there are many reasons to want to correct this: a random surface that arises from pasting a set of polygons along their edges is non-orientable with overwhelming probability, they appear as configuration spaces in diverse contexts [11, 26], and insights garnered from non-orientable surfaces can sometimes also be applied to the orientable ones; see for example [24]. Furthermore, the orientable genus of a graph can be arbitrarily larger than its non-orientable genus, while the reverse does not happen (see Lemma 7).

Our results. In this article, we initiate a study of short topological decompositions on non-orientable surfaces. We first show that the proof of the aforementioned result of Negami [22] has a minor flaw in the non-orientable case, we provide a counter-example to the proof technique and an alternative proof based on different techniques.

▶ **Theorem 2.** *Let S be a non-orientable surface of genus $g \geq 1$ and G_1 and G_2 be two graphs embedded on S. Then there exists a homeomorphism h such that any edge of $h(G_1)$ crosses each edge of G_2 at most $O(g)$ times. In particular, the total number of crossings between $h(G_1)$ and G_2 is $O(g|E(G_1)||E(G_2)|)$.*

Then our main result is the following theorem providing, to the best of our knowledge, the first known case of a short topological decomposition into a disk for non-orientable surfaces.

▶ **Theorem 3.** *There exists a polynomial time algorithm that given a graph cellularly embedded on a non-orientable surface computes a non-orientable canonical system of loops such that each loop in the system intersects any edge of the graph in at most* 30 *points.*

[1] Throughout the article, we decompose surface-embedded graphs by cutting them along embedded graphs which are transverse to the original graph, and count the number of intersections. This is equivalent to the primal setting studied in, e.g., Lazarus, Pocchiola, Vegter and Verroust [18] via graph duality.

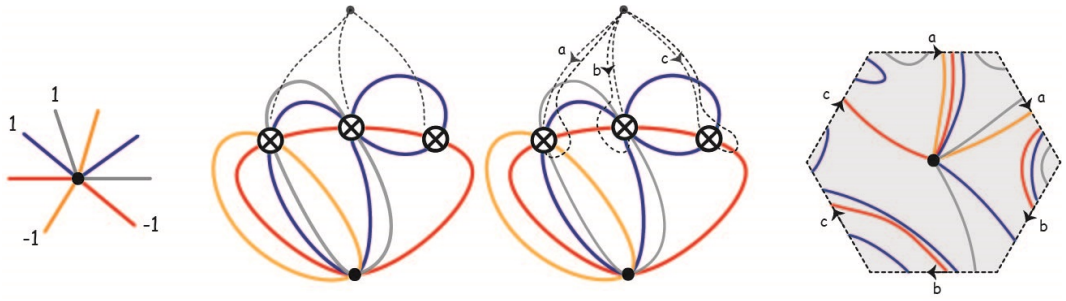


Figure 1 From left to right: 1) The combinatorial information of a one-vertex graph. 2) A cross-cap drawing of this graph, with cross-caps connected to a base-point. 3) A joint drawing of the graph and a canonical system of loops. 4) A different representation: decomposing the graph along the canonical system of loops.

Main ideas and proof techniques. As in many similar works, the first step in most of our results is to contract a spanning tree of the underlying graph, reducing the problems to the setting of one-vertex graph embedded on a non-orientable surface. The combinatorics of a one-vertex embedded graph are completely described by a rotation system, or embedding scheme.This will be the basic object with which we work.

Then, a simple but important object that we rely on extensively is an *orienting curve*, i.e., a closed curve on a non-orientable surface such that cutting along it produces an orientable surface. It was shown by Matoušek, Sedgwick, Tancer and Wagner [19] that given a graph embedded on a non-orientable surface, one can compute such an orienting loop that crosses each edge of the graph at most twice. This tool will be crucial in our corrected proof of Theorem 2 and in our work to prove Theorem 3.

For the proof of Theorem 3, we first point out that the techniques used to prove the orientable version in [18] incur an overhead of $O(g)$ in the number of crossings of the resulting curves (see [17, Theorem 4.3.9]). Instead, our proof of Theorem 3 builds on important recent work of Schaefer and Štefankovič [25]. The foundational idea behind this work, which takes its roots in an article of Mohar [20] on the degenerate crossing number, is to represent a graph embedded on a non-orientable surface as a planar drawing, with a finite set of *cross-caps*, as pictured in the second image of Figure 1. Schaefer and Štefankovič showed that any graph embedded on a non-orientable surface can be represented with a cross-cap drawing so that each edge uses each cross-cap at most twice. Our main technical contribution is to upgrade their construction so that the cross-caps can be connected to each other so as to yield a non-orientable canonical system of loops (Lemma 8), so that each loop intersects each edge of the one-vertex graph in at most 30 points (see Figure 1).

The complexity of the drawings provided by the proof of Schaefer and Štefankovič increases too fast to directly obtain a good bound by just connecting the cross-caps. Therefore, we modify their algorithm. First, by the aforementioned techniques, we can assume that we always have an orienting loop, which simplifies some of the steps and provides additional structure to the inductive argument. More importantly, we show that one can impose a certain order in which we choose the one-sided loops, as well as the separating loops, in the inductive argument to obtain a finer control on the resulting drawing.

The order in which we choose the loops comes from a seemingly unrelated problem in computational biology, and more precisely genome rearrangements. Given a permutation with signatures (a bit assigned to each letter), a signed reversal consists in choosing a subword in w, and reversing it as well as the signatures of all its letters. The *signed reversal distance*

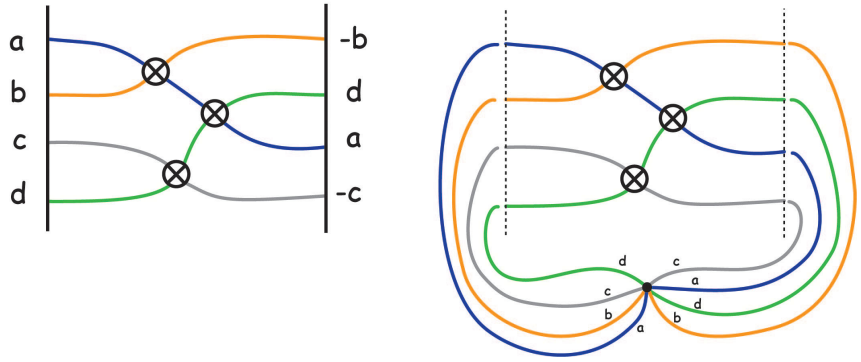


Figure 2 Left: a pictorial representation of three signed reversals bringing the signed permutation on the left to the signed permutation of the right. Right: Attaching the two permutations to a common basepoint yields a one-vertex graph with an embedding scheme, and the signed reversals provide a cross-cap drawing of that scheme where each loop enters each cross-cap at most once.

between two signed permutations is the minimum number of signed reversals needed to go from one permutation to the other one. This distance, and in particular algorithms to compute it has been intensively studied in the computational biology literature due to its relevance for phylogenetic reconstruction (see for example [14]). A cornerstone of the theory is the breakthrough of Hannenhalli and Pevzner [12] who provided an algorithm to compute the signed reversal distance between two signed permutations in polynomial time (see also the reformulation by Bergeron [2]). Now, as we illustrate in Figure 2, there is a very strong similarity between computing the signed reversal distance between two permutations and embedding a one-vertex graph built from these two permutations with a minimum number of cross-caps (see also [3, 16]). Surprisingly the algorithms of Hannenhalli and Pevzner on one side and of Schaefer and Štefankovič on the other also have strong similarities, which we leverage for example in Lemma 22. We hope that further cross-pollination between computational genomics and computational topology will lead to new surprises.

Due to space limitations, many proofs are omitted but can be found in the full version [9].

2 Preliminaries

We refer the reader to standard references such as Hatcher [13] and Stillwell [28] for topological background, the book of Mohar and Thomassen for graphs on surfaces [21] and the survey of Colin de Verdière [5] for topological algorithms for embedded graphs.

Surfaces, curves and embedded graphs. We assume working knowledge with the classification of surfaces. By the *genus* of a surface, we mean the orientable genus for an orientable surface, which we denote by M, and the non-orientable genus for a non-orientable surface, which we denote by N. The Euler genus is twice the orientable genus for orientable surfaces $eg(M) = 2g(M)$, and equals the non-orientable genus for non-orientable ones $eg(N) = g(N)$.

We call a closed curve on a surface *two-sided* if it has a neighborhood of it homeomorphic to the annulus. Otherwise, it is called *one-sided* and it has a neighborhood homeomorphic to the Möbius band. Given a closed curve ν on a surface S, cutting S along ν gives a (possibly disconnected) surface with one or two boundary components depending on whether ν is one-sided or two-sided. A curve δ on a surface S is *non-separating* if the surface we obtain by cutting along δ is connected; otherwise δ is separating. An *orienting curve* on a non-orientable surface N is a curve γ such that by cutting along γ, we get a connected orientable surface. We recall the following lemma from [19, Lemma 5.3].

▶ **Lemma 4.** *Let N be a non-orientable surface of genus g with h boundary components and let γ be an orienting closed curve. Let g_γ be the (orientable) genus and h_γ be the number of boundary components in N after cutting along γ.*

- *If g is odd, then γ is one-sided, $g_\gamma = \frac{g-1}{2}$, and $h_\gamma = h + 1$*
- *If g is even, then γ is two-sided, $g_\gamma = \frac{g-2}{2}$, and $h_\gamma = h + 2$.*

On a surface with boundary, by an *essential* proper arc, we mean an arc with endpoints on a boundary component that does not cut off a disk from the surface.

An *embedding* of a graph G on a surface S is a continuous injective map from G into S. A graph embedding is called *cellular* if its faces are homeomorphic to open disks.

Discrete Metrics on Surfaces. A cellularly embedded graph G on a surface S induces a discrete metric in two different ways. In the *combinatorial model*, the metric is defined on walks in G, and the length of a curve C is the number of edges of G traversed by C. In the *cross-metric model* [6], the metric is defined on curves *transverse* to G (i.e., that intersect G only at edges and in a non-tangent way), and their lengths is the number of crossings with G. Both models are naturally connected via graph duality. We mostly work in the cross-metric model and we refer to the embedded graph G of a cross metric surface S as the primal graph on S. The *multiplicity* of a curve (or a system of curves) at some edge e of G is the number of times e is crossed by the curve (curves). The multiplicity of a curve (or a system of curves) is the maximal multiplicity of the curve (curves) at any edge e of G.

Embedding schemes. For v a vertex of an embedded graph G, by a *rotation* ρ_v at v, we mean the cyclic permutation of the ends of edges incident to v. A *rotation system*, ρ, of a graph assigns a rotation to each vertex. and a signature to each edge, which is a number from $\{1, -1\}$. A rotation system ρ and a signature λ for the edges determine a cellular embedding for the graph up to homeomorphism i.e. we can compute the faces of the embedding purely combinatorially (see [21, Section 3.2] for further details). The pair (ρ, λ) is called an *embedding scheme* for the graph G, we simply use *scheme* throughout this work. Since a first step in all of our arguments is to contract a spanning tree, almost all the schemes considered in this article will have a single vertex. A cycle in a scheme is one-sided if the signature of its edges multiply to -1 and it is two-sided otherwise. The Euler genus of a scheme is the Euler genus of the underlying surface. A scheme is *orientable* if all its cycles are two-sided, and *non-orientable* otherwise. A loop e in the scheme divides the half-edges around the vertex into two parts; each part is called a *wedge* of e. When a loop γ has exactly one end in each wedge of e, we say that the ends of γ *alternate* with those of e; otherwise both ends of γ is in one wedge of e and we say that the ends of e *enclose* the ends of γ.

Following Schaefer and Štefankovič [25], we use the following model with localized cross-caps to represent non-orientable embedded graphs. A *planarizing system of disjoint one-sided curves* on a non-orientable surface, abbreviated *PD1S*, is a system of g disjoint one-sided curves such that by cutting along them, we obtain a sphere with g holes (this was first introduced by Mohar [20]). Therefore, from any graph embedded on a non-orientable surface, we obtain a planar representation by cutting along such a system. The non-orientable surface is recovered by gluing a Möbius band on each boundary component, which we depict using $\otimes$ and call a *cross-cap*. It is easily checked that a family of edges entering a cross-cap emerges on the other side with a reversed order, and that the sidedness of a loop is determined by the number of cross-caps that it crosses.

The planar drawing that we obtain by this cross-cap localization is called a *cross-cap drawing*, see Figure 3 for examples. In this model, we say that a drawing realizes an embedding scheme (G, ρ, λ) if the rotation at each vertex is as prescribed by ρ, and if

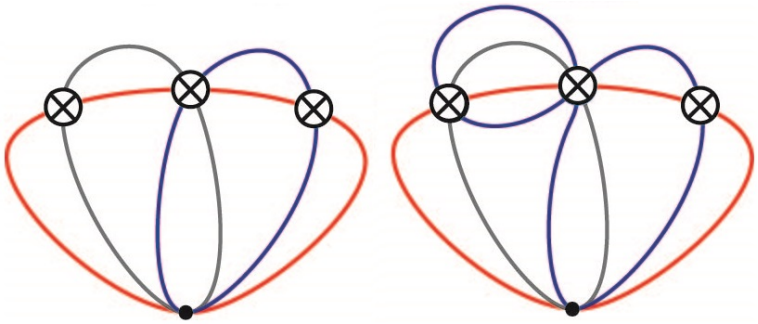


Figure 3 Different localization for the same embedding scheme gives different cross-cap drawings.

whenever a closed curve in the drawing passes through an odd (resp. even) number of cross-caps, the multiplication of the signatures of the edges it follows is -1 (resp. 1). Note that such a realization might not correspond to a cellularly embedded graph, and that while a cross-cap drawing uniquely describes an embedded graph, the converse is not true, see Figure 3. Throughout this article, by a cross-cap drawing for a graph G with an embedding scheme (ρ, λ), we mean the planar graph with cross-caps treated as extra vertices and edges being the sub-edges in G.

The following lemmas establish basic properties of orienting, separating loops and cross-cap drawings.

▶ **Lemma 5.** *A loop o in a cellularly embedded one-vertex graph G with a non-orientable embedding scheme is orienting if and only if its ends enclose the ends of any two-sided loop and alternate with the ends of any one-sided loop in the scheme.*

▶ **Lemma 6.** *In any cross-cap drawing of a scheme, a separating loop passes through each cross-cap an even number of times; and an orienting loop passes through each cross-cap an odd number of times.*

▶ **Lemma 7.** *Let G be an orientable scheme corresponding to a cellular embedding on a surface M with a minimum number of $g > 0$ handles. The minimum number of cross-caps needed in a cross-cap drawing realizing G is $2g + 1$ and this can always be achieved.*

Canonical System of Loops. For a non-orientable surface of genus g, a *non-orientable canonical system of loops* is a family of one-sided loops with the cyclic ordering $a_1a_1a_2a_2 \dots a_ga_g$ around the base point such that cutting M along this family yields a topological disk. The following lemma, illustrated in Figure 1 underpins our strategy to prove Theorem 3.

▶ **Lemma 8.** *Let H be a cross-cap drawing for a graph of non-orientable genus g and let b be a point in one face of the drawing. Let $\{p_i\}$ be a family of paths in the dual graph of this drawing from each cross-cap to b. Introduce a loop c_i by starting from b, passing along the path p_i, entering the corresponding cross-cap, going around the cross-cap and passing along p_i to return to b. The system of loops $\{c_i\}$ is a non-orientable canonical system of loops.*

Short Orienting Curves. The following lemma is a restatement of [19, Proposition 5.5].

▶ **Lemma 9.** *Let N be a non-orientable surface without boundary and with genus g and G be a graph embedded on N. Then there exists an orienting curve of multiplicity at most 2.*

By a little modification of the building process in the proof of this lemma, we can get an orienting arc in the case with boundaries instead of an orienting cycle, see the full version.

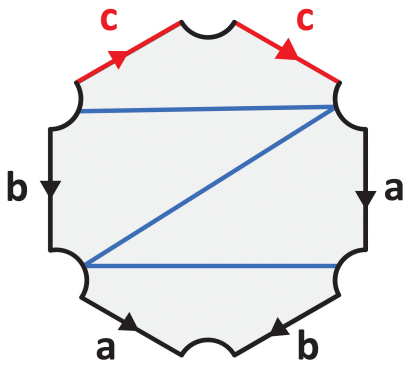


Figure 4 A non-orientable surface of genus 3 with an embedded system of arcs. The dents in the picture indicate the segments of the boundary component.

3 Correcting the proof of Negami

Negami proved in [22, Theorem 1] that if we have two graphs embeddable on a closed surface, we can reembed them simultaneously such that their edges cross few times.

▶ **Theorem 10.** *Let G_1 and G_2 be two connected graphs embeddable on a closed surface of genus g, orientable or non-orientable. We can embed them simultaneously such that they intersect transversely in their edges at most $4g\beta(G_1)\beta(G_2)$ times, where $\beta(G) = |E(G)| - |V(G)| + 1$ is the Betti number for a connected graph G.*

Negami's proof in the non-orientable case is reduced to showing the following two lemmas (with slightly different constants). However, his proof of Lemma 12 is flawed.

▶ **Lemma 11.** *For two orientable surfaces M_i of genus $g \geq 1$, with one boundary component and β_i disjoint essential proper arcs ($i = 1, 2$) where $\beta_i \leq \beta(G_i)$, there are homeomorphisms $\phi_i : M_i \to M$, where M is an orientable surface of genus g and one boundary component, so that the image of the arcs in M_1 and M_2 on M intersect at most $4(g-1)\beta_1\beta_2$ times.*

▶ **Lemma 12.** *For two non-orientable surfaces N_i of genus $g \geq 1$ with one boundary component and β_i disjoint essential proper arcs ($i = 1, 2$) where $\beta_i \leq \beta(G_i)$, there are homeomorphisms $\phi_i : N_i \to N$, where N is a non-orientable surface of genus g and one boundary component, so that the image of the arcs in N_1 and N_2 on N intersect at most $18(g-1)\beta_1\beta_2$ times when g is odd and $72(g-2)\beta_1\beta_2$ when it is even.*

In the proof of Lemma 12, Negami uses induction on the genus of the non-orientable surface. Assuming that the lemma is true for genus $g-1$, to prove it for genus g, he claims that there is an essential proper arc α that runs along the center line of a Möbius band (a one-sided arc). The idea is then to cut along α to get a non-orientable surface of genus $g-1$ to use the induction hypothesis. The problem lies in the fact that cutting along such an arc, we might not end up with a non-orientable surface.

▶ **Lemma 13.** *Consider the non-orientable surface of genus 3 with one boundary component and embedded essential arcs shown in Figure 4. Any one-sided arc disjoint from the embedded arcs cut the surface into an orientable surface.*

A Correction. To prove Theorem 2, we provide a different proof of Lemma 12. Our approach is to cut both surfaces along an orienting arc guaranteed by Lemma 9, this has the effect of multiplying the number of arcs by at most three. Depending on the parity of the genus we obtain an orientable surface with one or two boundary components. In the odd case, we apply Lemma 11 and then carefully modify one of the maps near the boundary controlling the multiplicity so that we can reconstruct the surface. In the even case, we have two boundaries. In that case, we connect them using a path of controlled multiplicity. Then we argue like in the odd case.[2].

4 Non-orientable Canonical System of Loops

4.1 The Schaefer-Štefankovič Algorithm

Schaefer and Štefankovič proved the following theorem [25, Lemma 9].

▶ **Theorem 14.** *If G is a one-vertex non-orientable (respectively orientable) scheme (ρ, λ), then it admits a cross-cap drawing with $eg(G)$ (respectively $eg(G)+1$) cross-caps in which every edge passes through every cross-cap at most twice.*

Schaefer and Štefankovič gave an inductive algorithm computing the cross-cap drawing claimed by this theorem. We first introduce the different moves that they use to deal with different types of loops.

For an embedding scheme around a vertex v, by *flipping* a wedge of a one-sided loop e in a one-vertex scheme, we mean reversing the order of the edges in the wedge and changing the signature of the loops that have exactly one end in the wedge. We call an empty wedge between two consecutive half-edges around v a *root wedge*.

Contractible loop move. Let c be a contractible loop with consecutive ends in the scheme G. Remove c. The new scheme can be drawn using the same number of cross-caps. Having a drawing for the new scheme, we can draw the loop c without passing through any of the cross-caps.

Gluing move. Let s be a non-contractible separating loop in the scheme G. We divide the scheme to G_1 and G_2 by cutting along s and splitting the vertex into two vertices (the embedding schemes of G_1 and G_2 are induced by the embedding scheme of G). Denote by f_o^1 and f_o^2 the root wedges in G_1 and G_2 in which s was formerly placed. Let H_1 and H_2 be drawings for G_1 and G_2 respectively. We glue the drawings by identifying the root wedges f_o^1 and f_o^2 to get the drawing $H^{'}$ for $G \setminus \{s\}$.

Note that removing s does not change the genus and we have $eg(G) = eg(G_1) + eg(G_2)$. If G_1 and G_2 are both non-orientable, then $H^{'}$ can be extended to a cross-cap drawing for G by adding s without using any of the cross-caps; see Figure 5. When at least one of G_1 or G_2 is orientable, say G_2, $H^{'}$ uses one extra cross-cap (G_2 needs $eg(G_2)+1$ cross-caps to be drawn). To deal with this case, we need the following lemma and the dragging move which allows us to reduce one cross-cap from the drawing.

▶ **Lemma 15.** *Let (ρ, λ) be an orientable embedding scheme for the one vertex graph G. Adding a one-sided loop o with consecutive ends to the scheme (anywhere in the rotation around the vertex) increases the Euler genus by 1. Thus, the new scheme needs as many cross-caps as G to embed. Furthermore, the loop o is orienting.*

[2] Our Theorem 2 is stated with respect to the numbers of edges $|E(G_i)|$ for simplicity, but the proof also provides a bound in terms of the Betti numbers $\beta(G_i)$, as in Theorem 10.

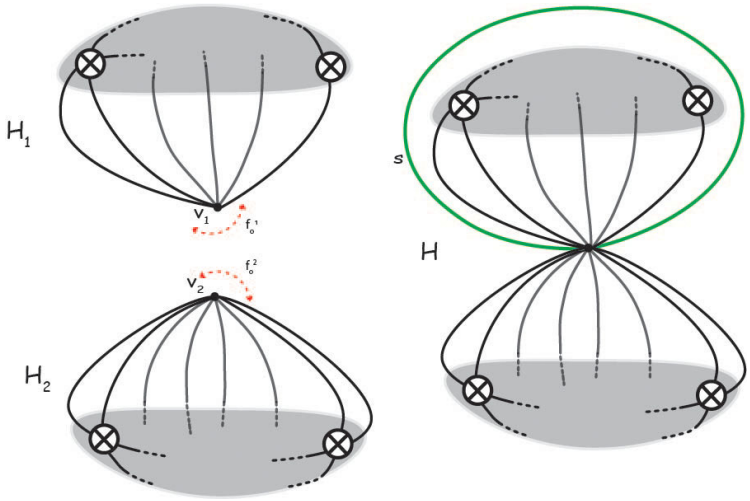


Figure 5 The gluing move on two cross-cap drawings when G_1 and G_2 are both non-orientable.

Dragging move. Let us assume that G_2 is orientable. By Lemma 15, we can add a one-sided loop o with consecutive ends in the root wedge f_o^2 without increasing the number of cross-caps that we need to draw G_2. The loop o is orienting and the new scheme needs $eg(G_2)+1$ cross-caps to be drawn. Having a drawing for the new scheme, we can draw the loop s in the drawing for $G_2+\{o\}$ as follows: we start the loop from one of the root wedges between o and another loop of G_2, we draw s by following o through all the cross-caps, except that after coming out of the last cross-cap, we go back to the first one entered, and traverse all of the cross-caps again. At the end, we follow o back to the vertex; see Figure 6, left. We denote this drawing of $G_2+\{o\}+\{s\}$ by $H_2^{'}$.

By gluing H_1 to $H_2^{'}$, we get a drawing $H^{'}$ for $G+\{o\}+\{s\}$ but the drawing is not using the minimum number of cross-caps. We eliminate one of the cross-caps in $H^{'}$ as follows.

Let i_1 be the rightmost half-edge in G_1 that follows immediately the separating loop in G. Denote by $\mathfrak{c}$ the first cross-cap that i_1 passes through. Let us assume that there are $2k$ half-edges passing through $\mathfrak{c}$. Let us denote by $(i_1, f_1, \ldots, i_{2k}, f_o^1)$ the alternating sequence of half-edges and faces adjacent to $\mathfrak{c}$ in the cross-cap drawing by moving clockwise around it. Now, we disconnect the edges that enter $\mathfrak{c}$ and remove the cross-cap $\mathfrak{c}$. We drag $i_1, \ldots, i_k$ through all the cross-caps in G_2 along the loop o. After exiting the last cross-cap in G_2, we remove o and we attach the half edges to their other ends $(i_{k+1}, \ldots, i_{2k})$. Since G_2 uses an odd number of cross-caps (Lemma 7), the half edges will have the correct orientability and order to get attached to their other ends; see Figure 6. If only one of G_1 and G_2 is orientable, the drawing we get uses $eg(G)$ cross-caps and if both are orientable, we get a drawing with $eg(G)+1$ cross-cap, that is, the minimum cross-cap needed to draw the scheme in this case.

One-sided loop move. Let r be a one-sided loop in the scheme G. We remove r and flip one of its wedges. One can check that the new scheme $G^{'}$ has Euler genus $eg(G)-1$. Let us assume that $H^{'}$ is a drawing for $G^{'}$. We add r to this drawing by adding a cross-cap near the vertex and the flipped wedge and dragging r and every edge in the flipped wedge in it; see Figure 7. Note that flipping different wedges of r leads to two different cross-cap drawings. This freedom in choosing the wedge will be used later.

If we apply a one-sided loop move on an orienting loop, the drawing we get does not use the minimum amount of cross-caps, hence we need the following different move.

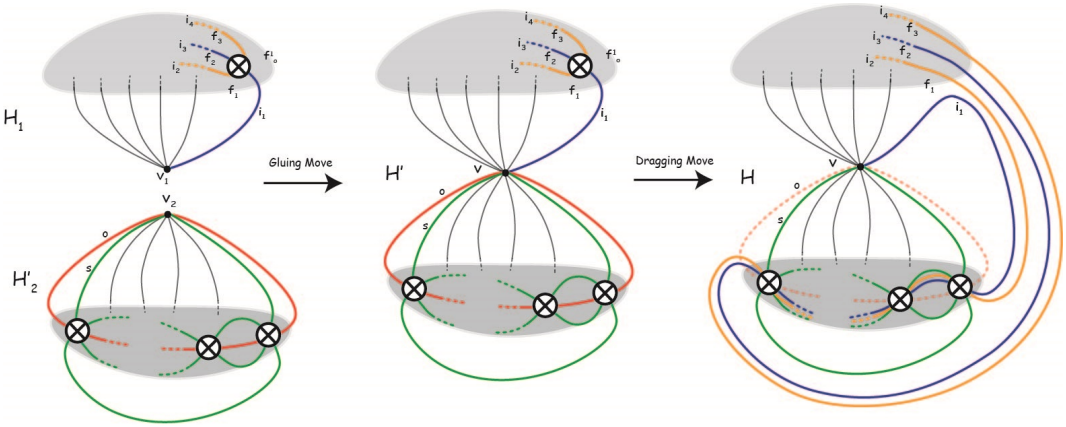


Figure 6 Left: the gluing move. Right: the dragging move when G_2 is orientable: the top right crosscap is removed and the corresponding curves are dragged through the bottom component.

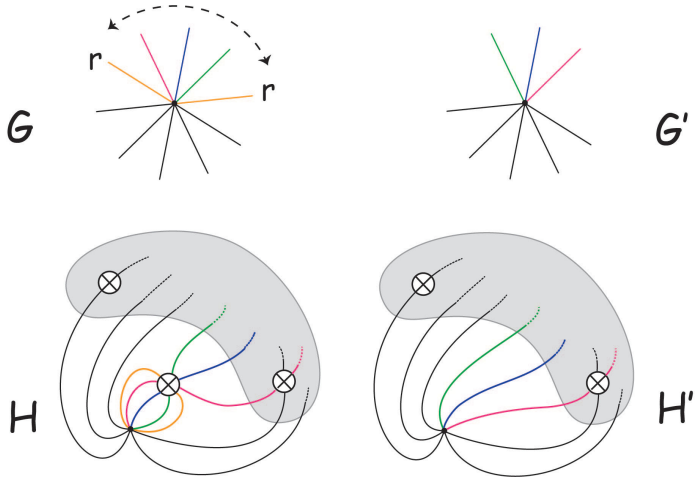


Figure 7 The one-sided loop move on the loop r.

Concatenation move. Let o be an orienting loop in the scheme G such that one of its ends is immediately followed by an end of a two-sided non-separating loop t in the rotation. By Lemma 5, since t is non-separating, the concatenation of o and t which we denote by o', is not orienting. Denote by G' the scheme in which we replace o by o' (we need $eg(G)$ cross-caps to draw both G and G'). If H' is a drawing for G', one can obtain from H' a drawing of G by replacing the drawing o' by its concatenation with t. Depending on the wedge of o' that we choose to flip, we slide o' along t in the drawing:

If we flipped the wedge that does not encompass the loop t, we detach the end of o' next to t and slide it along t and we attach it to the vertex. This way, it ends up where the end of o was placed originally; see Figure 8.

If we flipped the wedge that encompasses the loop t, we draw o as follows: note that o' passes trough only one cross-cap. We draw o next to the end of o' that is not slid along t, but instead of following o' into the cross-cap, we follow t. We can do this because the loop o' is next to the loop t in the rotation around this cross-cap; see Figure 9.

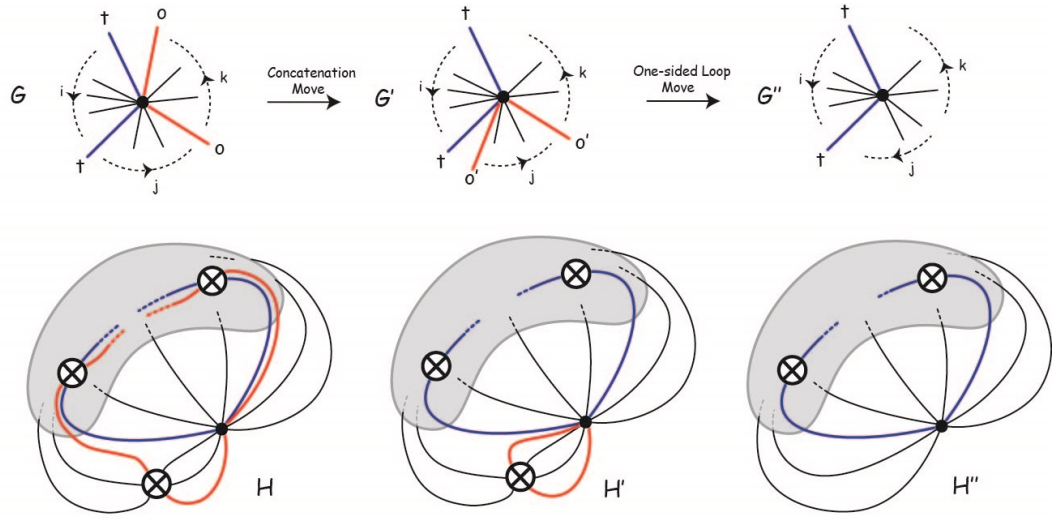


Figure 8 The concatenation move when the flipped wedge does not encompass the ends of t.

The Schaefer-Štefankovič algorithm also uses an additional move which we will not need, so we do not introduce it here. Each of these moves provides a way to draw a loop assuming that some simpler one-vertex graph without that loop has already been drawn. The algorithm behind Theorem 14 works by applying these moves in a specific order, we refer to the full version for details. Note that by Lemma 6, in a drawing that is obtained by Theorem 14, every orienting loop passes through each cross-cap exactly once and if a separating loop enters a cross-cap, it passes through that cross-cap exactly twice.

4.2 Our Modification to the Schaefer-Štefankovič algorithm

Our modification to the Schaefer-Štefankovič is to add two preprocessing steps, and then enforce more specific rules as to how to apply the moves described in the previous section. The first preprocessing is to add an orienting curve (via Lemma 9) and contract a spanning tree. In doing so, each edge in G is subdivided in at most 3 edges. This is summarized in the following lemma.

▶ **Lemma 16.** *Given a graph G embedded on a non-orientable surface N, there exists a one-vertex scheme $\hat{G}$ such that $\hat{G}$ has an orienting loop, and if $\hat{G}$ has a non-orientable canonical system of loops of multiplicity at most k, then G has a non-orientable canonical system of loops of multiplicity at most $3k$.*

For the second prepossessing move we need a definition that is inspired by a similar notion from the literature on sorting signed permutations by reversals [12]. Given an embedding scheme G, the *interleaving graph* I_G has as vertex set the set of loops of G, and two vertices are connected if their corresponding loops have interleaving ends. When we talk about the sidedness of a vertex, we mean the sidedness of the loop it is associated to. A connected component in the interleaving graph is called *non-orientable* if it has a one-sided vertex, and *orientable* otherwise. A component with only one vertex is a *trivial* component, and *non-trivial* otherwise. Separating loops correspond to trivial orientable components.

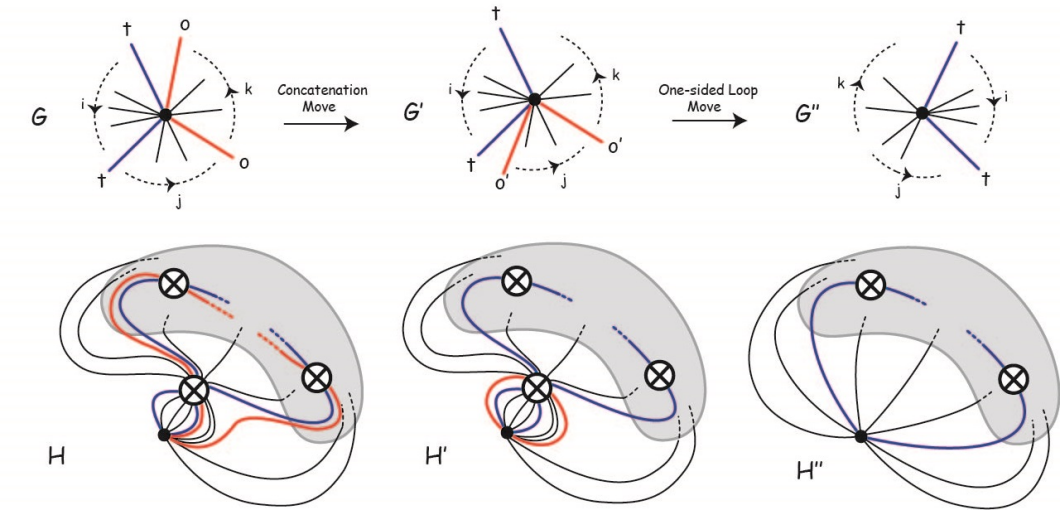


Figure 9 The concatenation move when the flipped wedge encompasses the ends of t.

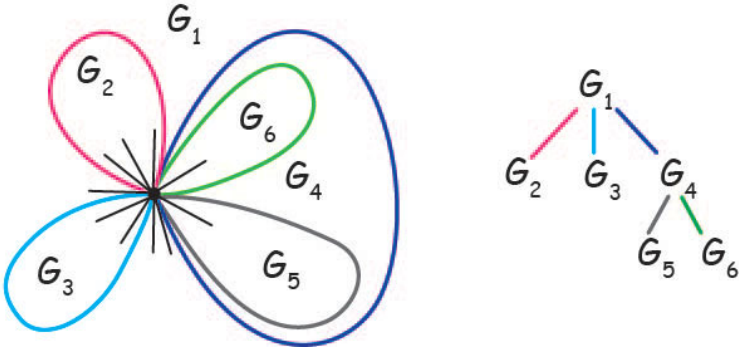


Figure 10 Left: a saturated one-vertex scheme in which the drawn loops are the separating loops; Right: the component tree of the left scheme. Note that the component G_4 is an empty sub-scheme.

Our second preprocessing step aims at subdividing G into subschemes G_i such that each I_{G_i} has only one non-trivial component. To achieve this, we *saturate* the scheme with auxiliary separating loops, i.e., we add a separating loop for any non-trivial component that is not divided from the rest of the graph by some separating loops. Since adding a separating loop does not interfere with the genus, then $eg(G) = eg(\bar{G})$ and we can later remove the added separating loops. Given a non-orientable scheme G saturated with separating loops and any cellular embedding of G on a surface N, cutting G along the separating loops yields subsurfaces N_i of N, each containing (possibly empty) components of G, which we denote by G_i (see Figure 10). The *component tree* of G has a vertex for every such sub-graph G_i, and two vertices are connected if their corresponding components are separated by a separating loop. See Figure 10 for an example of a component tree.

We now describe our algorithm. Throughout the main loop of our algorithm, if we have homotopic loops we remove all of them except one and after drawing this one, we re-introduce them parallel to the one drawn.

A difference of our modified version with the original one is that it is not clear at first sight that we cover all cases. This is justified by the following lemmas.

▶ **Lemma 17.** *A scheme with an orienting loop is non-orientable.*

Algorithm 1 The modified algorithm.

Pre-processing steps:

- **Step A:** If there is no orienting loop, we add an orienting loop and contract a spanning tree using Lemma 16.
- **Step B:** If G is not saturated by separating curves, we saturate it.

Main loop:

- **Step 1: If there is a contractible loop.** We recurse on the scheme without the loop and apply the contractible loop move.
- **Step 2: If there exists a separating (non-contractible) loop.** We pick a separating loop that separates a non-root leaf from the component tree, recurse on the subschemes and apply a gluing and a dragging move.
- **Step 3.1: If there exists a one-sided non-orienting loop.** We pick a one-sided non-orienting loop such that the scheme $G^{'}$ that we obtain when removing it and flipping its wedge maximizes the number of one-sided loops. We recurse on G' and apply the one-sided loop move on this loop.
- **Step 3.2.a: If all one-sided loops are orienting and there are two-sided loops.** We pick an orienting loop adjacent to a two-sided loop, recurse on the drawing H' described in the concatenation move and apply the concatenation move on these loops.
- **Step 3.2.b: If all one-sided loops are orienting and there are no two-sided loops.** In this case one cross-cap is sufficient to draw all the loops.

Post-processing steps:

- **Step B':** Erase the extra separating loops added in step B.
- **Step A':** Uncontract the spanning tree and remove the loop added in step A.

As a corollary, a scheme G that has an orienting loop needs $eg(G)$ cross-caps to be drawn.

▶ **Lemma 18.** *Let G be a scheme with an orienting loop o and a non-contractible separating loop s. Then s separates the graph into an orientable and a non-orientable sub-graph.*

Then the following lemma shows that there is always an orienting curve throughout the recursive calls of the algorithm. We state it as a corollary since it is a direct corollary of three technical lemmas that are featured in the full version.

▶ **Corollary 19.** *Let G be a one-vertex scheme with an orienting loop. Let G' be the graph on which the modified algorithm recurses when applying a contractible loop move, a one-sided loop move or a concatenation move. Then G' has an orienting loop. Likewise, when the modified algorithm applies a gluing and dragging move on a separating loop s, the two subgraphs G_1 and G_2 on which it recurses have an orienting loop.*

We now analyze the cross-cap drawing provided by the modified algorithm.

▶ **Lemma 20.** *Let G be a graph cellularly embedded on a non-orientable surface. If G has an orienting loop, applying the modified algorithm, we obtain a cross-cap drawing of G with $eg(G)$ cross-caps such that each loop of G enters each cross-cap at most twice. Otherwise, we obtain a cross-cap drawing of G with $eg(G)$ cross-caps such that each loop of G enters each cross-cap at most 6 times.*

Sketch of proof. After the preprocessing steps, we obtain a saturated scheme with an orienting loop and by Lemma 16 we can work with this scheme to prove the lemma. We follow the recursive steps of the modified algorithm, and thus provide a proof by induction on $eg(G) + |E(G)|$. By Corollary 19, there is an orienting loop throughout the recursive calls of the algorithm. Based on the description of each move, it can be seen that at each step we obtain a drawing with a minimum number of cross-caps. It is easy to see that the induction carries over after Step 1. Then, by Lemma 18, we know that a separating loop divides the scheme into a non-orientable and an orientable subscheme. This case is handled by Step 2 of the algorithm. One can see that in the dragging move and drawing the separating loop, each edge follows the auxiliary orienting loop o added to G_2 at most twice. Since o passes through every crosscap in the drawing of G_2 exactly once, we get the right multiplicity for every edge. For Step 3.1, let G' be the scheme that we recurse on. By the recursion of the algorithm we obtain a drawing for G' in which each edge uses each cross-cap at most twice. In adding the last cross-cap in the one-sided loop move, only the half-edges in the flipped wedge enter the last cross-cap and therefore each edge enters this cross-cap at most twice. For Step 3.2.a, let o' be the concatenation of the loops o and t. The loop o' is the only one-sided non-orienting loop in G'. By applying a one-sided loop move on o' and then recursing, we obtain a drawing for G' in which o' and t pass through a disjoint set of cross-caps and at most twice through each. Then it can be shown that sliding o' back along t, o' enters every cross-cap at most twice. Step 3.2.b is a base case for the induction satisfying the requirements of the lemma. Finally, since there is always an orienting loop, by Lemma 17 there is always at least one one-sided loop, and thus we are never in a case not covered by the previous steps. This concludes this sketch. ◀

The following two lemmas ensure further properties guaranteed by our algorithm and explain our choice for the rule in Step 3.1. The proof of the second lemma mirrors proofs in the signed reversal distance theory [2, Theorem 1].

▶ **Lemma 21.** *If there exists an orienting loop o in the embedding scheme G, the connected component that has the vertex o is the only non-orientable component in I_G.*

▶ **Lemma 22.** *Let G be a one-vertex scheme with an orienting loop and no non-contractible separating loop such that I_G has only one non-trivial component. Then G can be drawn by exclusively applying a sequence of contractible, concatenation one-sided loop moves.*

4.3 The Non-orientable Canonical System of Loops

Our algorithm has the following key advantage compared to the algorithm of Schaefer and Štefankovič: due to the order in which we choose the loops in Steps 2 and 3.1, we know that dragging moves and the other moves do not intermingle during the recursive calls of the algorithm. Indeed, first, by Lemma 22, when it draws a scheme with a single non-trivial component, it only relies on contractible, concatenation and one-sided loop moves. Second, due to the order in which we choose the loops in Steps 2 and 3.1 and Lemma 21, we know that whenever a dragging move is applied, the orientable sub-scheme on which we recurse has only one non-trivial component. In this section, we leverage these two key advantages to find a non-orientable canonical system of loops of small multiplicity.

A *root face* in a cross-cap drawing is a face adjacent to the vertex. The strategy to prove Theorem 3 is to show that the modified algorithm outputs a cross-cap drawing where cross-caps are not too far from the vertex.

▶ **Lemma 23.** *For any saturated one-vertex scheme G with an orienting loop o, the cross-cap drawing H output by the modified algorithm has $eg(G)$ cross-caps, and there is a path from every cross-cap to a root face (not necessarily fixed) with multiplicity at most two.*

This lemma is the crux of the paper. We refer to the full version for the proof, and only outline some ideas. The difficulty lies in the fact that long dual paths are added to the cross-cap drawings when doing one-sided loop moves and dragging moves, making it hard to control the diameter of the graph dual to the cross-cap drawings throughout the recursive calls. This is solved by tracking specific paths, whose lengths are controlled using two different strategies for the one-sided loop moves and for the dragging moves. The inductive property that we maintain throughout one-sided loop moves is that there is a short path *that does not cross the orienting curve* from every crosscap to a *fixed* root face. The choice of a fixed root face ensures that, if one chooses carefully the wedge when applying a one-sided loop move, the lengths of the paths stay controlled. The property of not crossing the orienting curve is key because, when we subsequently apply dragging moves, edges dual to the orienting curves might get replaced with long dual paths (see Figure 6). However, dragging moves naturally lead to paths that connect different root faces. Fortunately, at this stage, this is not an issue since all the one-sided moves have already been applied. Therefore, for this proof strategy to succeed, our modifications are crucial, in particular to ensure that one-sided loop moves and dragging moves do not alternate during recursive calls.

Finally, we now have all the tools to prove Theorem 3.

Sketch of proof of Theorem 3. The modified algorithm on G constructs a cross-cap drawing of a saturated scheme with an orienting loop. By Lemma 20, the loops enter each cross-cap at most twice. By Lemma 23, we furthermore have paths $\{p_j\}$ of multiplicity two from a face incident to each cross-cap to a root face in this cross-cap drawing. We follow these paths to construct loops surrounding each of the cross caps. By Lemma 8, the system of loops we obtain is canonical. Adding the different bounds on the multiplicity, we obtain that each loop in the system has multiplicity 10. ◀

References

1 Dan Archdeacon and C Paul Bonnington. Two maps on one surface. *Journal of Graph Theory*, 36(4):198–216, 2001.

2 Anne Bergeron. A very elementary presentation of the Hannenhalli-Pevzner theory. In *Annual Symposium on Combinatorial Pattern Matching*, pages 106–117. Springer, 2001.

3 Andrei C Bura, Ricky XF Chen, and Christian M Reidys. On a lower bound for sorting signed permutations by reversals. *arXiv preprint*, 2016. `arXiv:1602.00778`.

4 Éric Colin de Verdière. Topological algorithms for graphs on surfaces. Habilitation thesis, `http://www.di.ens.fr/~colin/`, 2012.

5 Éric Colin de Verdière. Computational topology of graphs on surfaces. In Jacob E. Goodman, Joseph O'Rourke, and Csaba Toth, editors, *Handbook of Discrete and Computational Geometry*, chapter 23, pages 605–636. CRC Press LLC, third edition, 2018.

6 Éric Colin De Verdière and Jeff Erickson. Tightening nonsimple paths and cycles on surfaces. *SIAM Journal on Computing*, 39(8):3784–3813, 2010.

7 Jeff Erickson and Sariel Har-Peled. Optimally cutting a surface into a disk. *Discrete & Computational Geometry*, 31(1):37–59, 2004.

8 Jeff Erickson and Kim Whittlesey. Greedy optimal homotopy and homology generators. In *SODA*, volume 5, pages 1038–1046, 2005.

9 Niloufar Fuladi, Alfredo Hubard, and Arnaud de Mesmay. Short topological decompositions of non-orientable surfaces, 2022. `arXiv:2203.06659`.

10 Jim Geelen, Tony Huynh, and R Bruce Richter. Explicit bounds for graph minors. *Journal of Combinatorial Theory, Series B*, 132:80–106, 2018.

11 Robert Ghrist. Barcodes: the persistent topology of data. *Bulletin of the American Mathematical Society*, 45(1):61–75, 2008.

12 Sridhar Hannenhalli and Pavel A Pevzner. Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals. *Journal of the ACM (JACM)*, 46(1):1–27, 1999.

13 Allen Hatcher. *Algebraic Topology*. Cambridge University Press, 2002.

14 Brian Hayes. Computing science: Sorting out the genome. *American Scientist*, 95(5):386–391, 2007.

15 Petr Hliněný and Gelasio Salazar. On hardness of the joint crossing number. In *International Symposium on Algorithms and Computation*, pages 603–613. Springer, 2015.

16 Fenix WD Huang and Christian M Reidys. A topological framework for signed permutations. *Discrete Mathematics*, 340(9):2161–2182, 2017.

17 Francis Lazarus. Combinatorial graphs and surfaces from the computational and topological viewpoint followed by some notes on the isometric embedding of the square flat torus. *Mémoire d'HDR*, 2014. Available at `http://www.gipsa-lab.grenoble-inp.fr/~francis.lazarus/Documents/hdr-Lazarus.pdf`.

18 Francis Lazarus, Michel Pocchiola, Gert Vegter, and Anne Verroust. Computing a canonical polygonal schema of an orientable triangulated surface. In *Proceedings of the seventeenth annual symposium on Computational geometry*, pages 80–89, 2001.

19 Jiří Matoušek, Eric Sedgwick, Martin Tancer, and Uli Wagner. Untangling two systems of noncrossing curves. In *International Symposium on Graph Drawing*, pages 472–483. Springer, 2013.

20 Bojan Mohar. The genus crossing number. *ARS Mathematica Contemporanea*, 2(2):157–162, 2009.

21 Bojan Mohar and Carsten Thomassen. *Graphs on surfaces*, volume 10. JHU press, 2001.

22 Seiya Negami. Crossing numbers of graph embedding pairs on closed surfaces. *Journal of Graph Theory*, 36(1):8–23, 2001.

23 R Bruce Richter and Gelasio Salazar. Two maps with large representativity on one surface. *Journal of Graph Theory*, 50(3):234–245, 2005.

24 Marcus Schaefer and Daniel Štefankovič. Block additivity of $\mathbb{Z}_2$-embeddings. In *International Symposium on Graph Drawing*, pages 185–195. Springer, 2013.

25 Marcus Schaefer and Daniel Štefankovič. The degenerate crossing number and higher-genus embeddings. *Journal of Graph Algorithms and Applications*, 26(1):35–58, 2022. `doi:10.7155/jgaa.00580`.

26 James P Sethna. Order parameters, broken symmetry, and topology. In *1991 Lectures in Complex Systems*. Addison-Wesley, 1992.

27 Alla Sheffer, K Hormann, B Levy, M Desbrun, K Zhou, E Praun, and H Hoppe. Mesh parameterization: Theory and practice. *ACM SIGGRAPPH, course notes*, 10(1281500.1281510), 2007.

28 John Stillwell. *Classical topology and combinatorial group theory*, volume 72. Springer Science & Business Media, 1993.