Discrete & Computational Geometry
https://doi.org/10.1007/s00454-023-00580-3

Short Topological Decompositions of Non-orientable Surfaces

Niloufar Fuladi[1] · Alfredo Hubard[1] · Arnaud de Mesmay[1]

Received: 31 May 2022 / Revised: 6 July 2023 / Accepted: 14 July 2023

Abstract
In this article, 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 on 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. The existence of short canonical non-orientable systems of loops 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. Finally, we provide a generalization of $O(g)$-universal shortest path metrics to non-orientable surfaces.

Keywords Non-orientable surface · Cutgraph · System of loops · Negami's conjecture · Joint crossing number

Mathematics Subject Classification 05C10 · 05C35 · 57N05

This article is dedicated to the memory of Eli Goodman, whose allowable sequences inspired some of this work.

Editor in Charge: Kenneth Clarkson

This work was partially supported by the ANR project SoS (ANR-17-CE40-0033). A. Hubard was also supported by the ANR projet CAAPS (ANR-17-CE40-0018).

Alfredo Hubard
alfredo.hubard@univ-eiffel.fr

[1] Univ Gustave Eiffel, CNRS, LIGM, 77454 Marne-la-Vallée, France

1 Introduction

1.1 Topological Decompositions and Joint Crossing Numbers

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 [29]). Surfaces often come with extra structure which can be modeled by an embedded graph. Decomposing such a surface efficiently corresponds to finding another cellularly embedded graph that has few (transverse) intersections[1] with the original graph (see for example [20] or [10]). Such decompositions also appear in algorithm design: often, to generalize results on planar graph to graphs embedded on surfaces, its 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, 9, 20].

In many applications, it is important that the graph along which we cut is canonical in some sense. For example, in order to compute a homeomorphism between two surfaces, a common approach is to cut them into disks, put these disks in correspondence, and glue back the surfaces so as to obtain a homeomorphism. However, this only works if the cut graphs have the same combinatorial structure. A seminal result on topological decompositions was pionereed by Lazarus, Pocchiola, Vegter and Verroust [20] (see also [19]) who designed an algorithm that finds, for any graph G embedded on a closed orientable surface S a *canonical system of loops* H such that no edge of H intersects 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 $a_1b_1a_1^{-1}b_1^{-1}\dots a_gb_ga_g^{-1}b_g^{-1}$. The polygon obtained after cutting along such a system of loops, with the data of the boundary identifications, is called a *canonical polygonal scheme*.

Such a decomposition is an instance of the problem of finding simultaneous embeddings for two graphs on a surface such that the number of crossings between the two graphs is minimized. More precisely, consider a pair of graphs G_1 and G_2 embedded on a closed surface S of genus g and define the *joint crossing number* as the minimal number of crossing points between $h(G_1)$ and G_2 over all the homeomorphisms $h : S \to S$. This quantity was initially introduced by Negami [24] who proved that any two graphs G_1 and G_2 embedded on a closed surface of genus g, have joint crossing number $O(g|E(G_1)||E(G_2)|)$, where $|E(G)|$ is the number of edges in G. Furthermore, he made the following conjecture, which is still open.

Conjecture 1.1 *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 $C|E(G_1)||E(G_2)|$.*

This conjecture has been investigated further [1, 16, 25] and variants of this problem have appeared in various works with applications as diverse as finding explicit

[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 et al. [20] via graph duality.

bounds for graph minors [11] or designing an algorithm for the embeddability of simplicial complexes into $\mathbb{R}^3$ [21]. From the perspective of topological decompositions, Negami's conjecture posits that *short* decompositions of any fixed shape exist, in the sense that one can always decompose an embedded graph along a chosen topological decomposition (modeled by a second, cellularly embedded graph), in such a way that each edge of the decomposition crosses each edge of the graph $O(1)$ times.[2] Such a bound is known for only very few shapes. Beyond orientable canonical systems of loops, Colin De Verdière and Erickson [6] proved the existence of a short *octagonal decomposition* for orientable surfaces and provided an algorithm to compute it. We do not know of any other construction of short decompositions than those two and variants thereof.

In particular, no short decomposition at all seems to be known for non-orientable surfaces. Even if G_1 is a *non-orientable canonical system of loops*, that is, a system of one-sided loops with the cyclic ordering $a_1a_1a_2a_2\ldots a_ga_g$ around the vertex, the best known bound for this system is $O(g|E(G_2)|)$ crossings for each edge of the decomposition (see [19]), which matches the bound claimed by Negami. Due to their extra difficulty, non-orientable surfaces have been often somehow neglected in computational topology, but there are many reasons to want to correct this: natural models of random surfaces yield non-orientable surfaces with overwhelming probability, they appear naturally as configuration spaces in diverse contexts [12, 28], and insights garnered from non-orientable surfaces can sometimes also be applied in a subtle way to the orientable ones; see for example [26]. Furthermore the orientable genus of a graph can be arbitrarily larger than its non-orientable genus, while the reverse does not happen (see Lemma 2.5).

1.2 Our results

In this article, we initiate a thorough study of short topological decompositions on non-orientable surfaces. As outlined above, one of the only results known on topological decompositions of non-orientable surfaces is a theorem of Negami [24]. We first show that the proof of this result has a minor flaw and exhibit a specific counter-example to the proof technique. Then we provide an alternative proof based on different techniques.

Theorem 1.2 *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 $O(g|E(G_1)||E(G_2)|)$.*

In order to prove this theorem we take advantage of a technique in [21] to compute a short *orienting curve*, i.e., a curve such that cutting along it yields an orientable surface. This versatile technique is also primordial in our main result, which 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.

[2] This statement is slightly stronger than Conjecture 1.1 since it enforces a control on the number of crossings between each pair of edges instead of the total number of crossings, but it is equally open.

Theorem 1.3 *There exists a polynomial time algorithm that given a graph cellularly embedded on the non-orientable surface N 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.*

Finally, one more application of cutting along an orienting loop is the following theorem, generalizing results on universal shortest path metrics obtained in [18] to non-orientable surfaces.

Theorem 1.4 *For $g \geq 3$, there exists a hyperbolic metric m on the non-orientable surface N of genus g such that any graph embeddable on N can be embedded so that every edge is a concatenation of $O(g)$ shortest paths.*

The proof of Theorem 1.4 relies, after cutting along an orienting loop, on techniques fairly identical to those in [18].

1.3 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 graphs embedded on a non-orientable surface. The combinatorics of a one-vertex embedded graph are completely described by an embedding scheme, i.e., by the circular order of the edges around the vertex, and a signature for each loop indicating whether it is one-sided or two-sided. Such an embedding scheme will be the basic object with which we work.

A simple but important object that we rely on extensively is an orienting curve. It was shown by Matoušek, Sedgwick, Tancer and Wagner [21] that given a graph embedded on a non-orientable surface, one can compute an orienting curve that crosses each edge of the graph at most a constant number of times. This lemma allows us, at the cost of slightly increasing the constants in our results, to assume that our embedding schemes always have an orienting curve. With this tool at hand, we provide a corrected proof of Theorem 1.2. Furthermore, the existence of this orienting curve will significantly simplify our work to prove Theorem 1.3.

For the proof of Theorem 1.3, we first point out that the techniques used to prove the orientable version in [20] do not readily apply, as they rely on a fine control of the cut-and-pasting operations used in the proof of the classification of surfaces, and in the non-orientable case there is an additional step in these operations which incurs an overhead of $O(g)$ in the number of crossings of the resulting curves (see [19, Thm. 4.3.9]). Instead, our proof of Theorem 1.3 builds on important recent work of Schaefer and Štefankovič [27]. The foundational idea behind this work, which takes its roots in an article of Mohar [22] on the degenerate crossing number, is to represent a graph embedded on a non-orientable surface of genus g as a planar drawing, with g *cross-caps*, which are points where multiple edges are allowed to cross in a way that reverses the permutation, as pictured in the second picture of Fig. 1. Using an intricate argument inducting on the loops of an embedding scheme, Schaefer and Štefankovič showed that any graph embedded on a non-orientable surface can be represented by such a cross-cap drawing so that each edge uses each cross-cap at most twice. Our

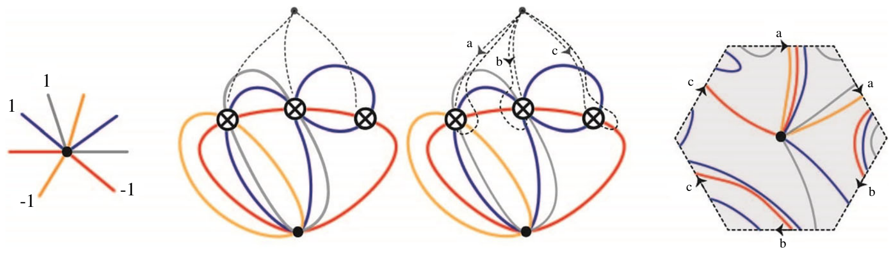

Fig. 1 From left to right: (1) the combinatorial information of a one-vertex graph. (2) A cross-cap drawing of this graph, with cross-caps connected to a basepoint. (3) A joint drawing of the graph and a canonical system of loops. (4) A different representation: decomposing the graph with a canonical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

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 2.6), so that each loop intersects each edge of the one-vertex graph in at most 30 points (see Fig. 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. But 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 of Schaefer and Štefankovič so as to obtain a finer control on the resulting drawing.

The order in which we choose loops comes from a seemingly unrelated problem in computational biology, and more precisely genome rearrangements. Given a permutation w on a set of distinct letters 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* 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 [15]). A cornerstone of the theory is the breakthrough of Hannenhalli and Pevzner [13] 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 Fig. 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 [3, 17]). Surprisingly the algorithms of Hannenhalli and Pevzner on one side and of Schaefer and Štefankovič on the other also have similarities. The proof of Theorem 1.3 leverages the literature on the signed reversal distance problem, and in particular the structure we impose on the cross-caps drawings of non-orientable graphs is inspired on ideas from the aforementioned genome rearrangements algorithms. We hope that further interpollination between computational genomics and computational topology will lead to new surprises.

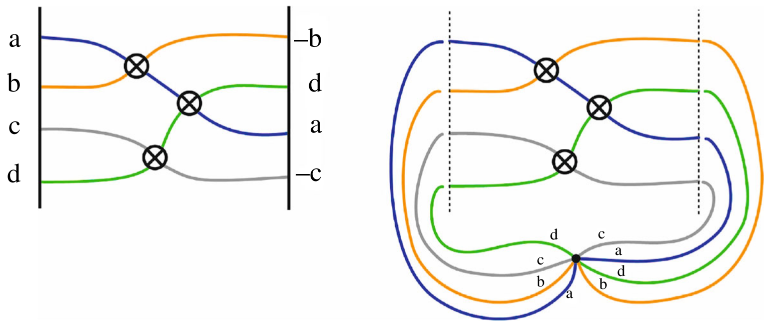

Fig. 2 Left: a pictorial representation of three signed reversals bringing the signed permutation on the left to the signed permutation on the right. Right: Attaching the two permutations to a common basepoint yields a one-vertex graph with an embedding scheme, and the signed reversals provide a cross-cap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

Outline After introducing the preliminary definitions and results in Sect. 2, we will prove Theorems 1.2, 1.3 and 1.4 in Sections 3, 4 and 5 respectively.

2 Preliminaries

While this paper strives to be mostly self-contained, we refer the reader to standard references such as Hatcher [14] and Stillwell [30] for more topological background, the book of Mohar and Thomassen for an extensive overview of graphs on surface [23] and the survey of Colin de Verdiere [5] on topological algorithms for embedded graphs.

2.1 Surfaces

A *surface* S is a topological Hausdorff space where each point has a neighborhood homeomorphic to either the plane or the closed half-plane. The points without a neighborhood homeomorphic to the plane comprise the boundary of M. Compact surfaces without boundaries are called *closed surfaces*. A surface is called *orientable* if it does not contains a subspace homeomorphic to a Möbius band; otherwise, it is called *non-orientable*. Throughout this work, we denote orientable surfaces and non-orientable surfaces by M and N respectively, and by S whenever orientability does not make a difference.

The classification theorem for closed surfaces states that any orientable surface M of genus $g \geq 0$ is homeomorphic to a sphere with g handles,[3] g is called the *orientable genus* of M; and any non-orientable surface N of genus $g \geq 1$ is homeomorphic to a sphere with g cross-caps,[4] in this case, g is the *non-orientable genus* of N (see Fig. 3).

[3] A *handle* is obtained by removing a small disk and gluing a punctured torus along its boundary to the boundary circle of the resulting hole (see the left picture in Fig. 3).

[4] A *cross-cap* is obtained by removing a small disk from the sphere and gluing in a Möbius band along its boundary to the boundary circle of the resulting hole (see the right picture in Fig. 3).

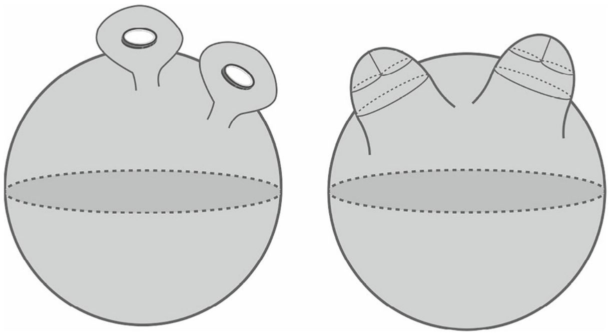

Fig. 3 Depiction of compact surfaces without boundaries obtained by attaching handles (orientable surfaces, left picture) or cross-caps (non-orientable surfaces, right picture)

Throughout this work, by the *genus* of a surface, we mean the orientable genus for orientable surfaces and the non-orientable genus for non-orientable surfaces. If S is a surface with k boundary components, the classification theorem still applies, except that we need to replace the initial sphere with a sphere with k holes. Thus a surface is uniquely determined by its genus, by its number of boundary components, and by its orientability.

2.2 Curves and Embedded Graphs

A *closed curve* or a *cycle* on a surface S is a continuous map $\theta : S^1 \to S$. A map $\theta : S^1 \to S$ is called a *constant cycle* if it is a constant map. By a *path* from x to y on a surface, we mean a continuous map $\theta : [0,1] \to S$ where $\theta(0) = x$ and $\theta(1) = y$. A path with two ends on the boundary of a surface is called an *arc*. These are called simple if the maps are injective.

We call a closed curve on a surface *two-sided* if a small closed neighborhood of it is homeomorphic to the annulus. Otherwise, it is called *one-sided* and it has a closed 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 called *non-separating* if the surface we obtain by cutting along δ is connected; otherwise δ is separating. An *orienting curve*[5] 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 [21].

Lemma 2.1 [21, Lem. 5.3] *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 $h_\gamma = h + 2$.*

[5] This terminology is slightly non-standard, we follow Schaefer and Štefankovič [27] who attribute it to B. Mohar.

Two paths p and q with the same endpoints a and b on a surface S, are *homotopic* if there is a continuous map $H : [0,1] \times [0,1] \to S$ such that $H(0,.) = p$, $H(1,.) = q$, $H(.,0) = a$, and $H(.,1) = b$. Two cycles γ and γ' are (freely) homotopic if there is a continuous map $H : [0,1] \times S^1 \to S$ such that $H(0,t) = \gamma(t)$ and $H(1,t) = \gamma'(t)$ for all t. A cycle is *contractible* if it is homotopic to a constant cycle; an arc is *contractible* if it is homotopic to a path on the boundary.

2.3 Graph Embeddings

In a drawing of a graph on a surface, every vertex is mapped to a point on the surface and edges are realized as curves connecting their endpoints. Informally, a drawing of the graph G on a surface S with no crossing on the edges is called an *embedding* of G on S. More precisely, an embedding of G is a continuous, one-to-one map from G into S. In this work, we generally identify G with its embedding on S. A graph embedding is called *cellular* if its faces are homeomorphic to open disks. For a graph G cellularly embedded on a surface S, *Euler's formula* states that $v - e + f = \chi(S)$, where v, e and f represent the number of vertices, edges and faces of G and $\chi(S)$ is the *Euler characteristic*, a topological invariant that depends only on the surface and not on the cellular embedding. The *Euler genus* of a closed surface is defined by $eg(S) = 2 - \chi(S)$. It is easy to see that the Euler genus is twice the orientable genus for orientable surfaces, and equal to the non-orientable genus for non-orientable ones. We take these identities as definition for surfaces with boundary, then the Euler characteristic of a surface with boundary is $2 - eg(S) - k$ where k is the number of the boundary components of the surface. Cutting a surface along an arc α increases the Euler characteristic of the surface by 1 and cutting along a closed curve does not change the Euler characteristic. These relations allow us to relate the genus and the number of boundary components of a surface after a cutting.

Lemma 2.2 *Let t be a separating (closed) curve on a closed surface S and let S_1 and S_2 be the surfaces we obtain by cutting along t. We have $eg(S) = eg(S_1) + eg(S_2)$.*

Proof By cutting along t, we get two boundary components, one on each of S_1 and S_2 and since t is a closed curve, we have $2 - eg(S) = (2 - eg(S_1) - 1) + (2 - eg(S_2) - 1)$ which proves the claim. □

2.4 Discrete Metric on Surfaces

We briefly define the notions of combinatorial and cross-metric surfaces, which define a discrete metric on a surface and are essentially the same, up to duality (see [6] for more details). For a graph G cellularly embedded on a surface without boundary, the *dual graph* G^* is defined as follows: to each face of G we associate a vertex and for each edge e in G that is in common between the faces f_1 and f_2, we connect the vertices associated to f_1 and f_2. In this construction, two vertices can be connected with multiple edges if they have more than one common edge. A dual graph embedding is also cellular and $G^{**} = G$.

A *combinatorial surface* is a surface S together with a graph G which is cellularly embedded on S. In the case that S has boundary, G is embedded so that the boundary is the union of some edges of G. In this model, the only allowed curves are walks in G, and the length of a curve C is the number of edges of G traversed by C. A *cross-metric surface* is a surface S together with a graph G which is cellularly embedded on S. In the case that S has boundary, G is embedded so that the boundary is the union of some edges of G. The curves allowed in this definition, are those that cross G only transversely (i.e., that intersect G only at edges and in a non-tangent way), and the length of a curve C is the number of edges of G that C crosses.

In this work, we mostly work in the cross-metric model and we refer to the embedded graph G of the cross-metric surface S, as the primal graph on S. The *multiplicity* of a curve, respectively of a system of curves, at some edge e of G is the number of times e is crossed by the curve, respectively the sum of all the intersections of e with the curves of the system. The multiplicity of a curve (or a system of curves) is the maximal multiplicity of the curve (curves) at any edge e of G.

2.5 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. We assign to each edge a *signature* 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 [23] for further details). The pair (ρ, λ) is called an *embedding scheme* for the graph G, we sometimes use *scheme* instead of embedding scheme in this work. Since a first step in all of our arguments is to contract a spanning tree, almost all the embedding schemes considered in this article will have a single vertex. We sometimes allow ourselves to denote an embedding scheme with a single letter, e.g., $G := (\rho, \lambda)$.

A cycle in an embedding scheme is one-sided if the signature of its edges multiply to -1 and it is two-sided otherwise. An embedding scheme is *orientable* if all its cycles are two-sided, and *non-orientable* otherwise. A loop e in the embedding scheme divides the half-edges around the vertex into two parts; each part is called a *wedge* of e. When a loop g has exactly one end in each wedge of e, we say that the ends of g *alternate* with those of e; otherwise both ends of g is in one wedge of e and we say that the ends of e *enclose* the ends of g. Finally, we slightly depart from the convention in topological graph theory: we do **not** assume that an embedding scheme describes a cellularly embedded graph, since we will sometimes consider orientable schemes over non-orientable surfaces. However, we will always consider embedded graphs (and their embedding schemes) on surfaces of minimal non-orientable genus, so while these graphs are sometimes not cellularly embedded, they have at most one non-cellular face with non-orientable genus one (see Lemma 2.5).

Following Schaefer and Štefankovič [27], we will 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

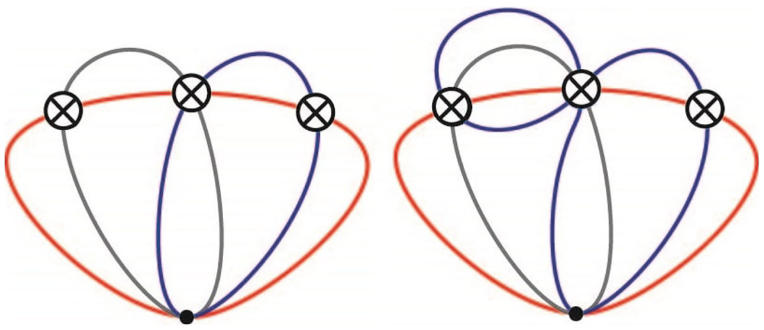

Fig. 4 Different localization for the same embedding scheme gives different cross-cap drawings

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 [22]). By cutting along such a system, from any graph embedded on a non-orientable surface, we obtain a planar representation. 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 can be easily checked that a family of edges entering a cross-cap emerge on the other side with a reversed order, and that the sidedness of a loop is determined by the number of times it enters the cross-caps.

The planar drawing that we obtain by this cross-cap localization is called a *cross-cap drawing*, see Fig. 4 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 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). While a cross-cap drawing uniquely describes an embedded graph, the converse is not true, see Fig. 4. 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 lemma helps us recognize an orienting loop in a one-vertex scheme.

Lemma 2.3 *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 embedding scheme.*

Proof For the forward implication, let o be an orienting loop and consider a cross-cap drawing of G with a minimal number of cross-caps. By the classification of surfaces and Lemma 2.1, there is, up to homeomorphism, a single orienting loop, and therefore we can assume that in this cross-cap drawing the loop o goes exactly once through each cross-cap. The loop o separates the plane into two regions, and since two-sided loops cross an even number of cross-caps, they start and end in the same wedge formed by o. Likewise, since one-sided loops cross an odd number of cross-caps, they start and end in a different wedge, and thus alternate with o.

For the reverse implication, we investigate the surface and the graph obtained after cutting along o. If o is one-sided, the resulting surface has one boundary, and the vertex is split into two vertices on the boundary. If o is two-sided the resulting surface has two boundaries, and the vertex is split into two vertices, one on each boundary. In both cases, loops which were alternating with o are now arcs connecting two distinct vertices, while loops which were not alternating with o has both endpoints on one vertex. After the cutting, edges keep their signature, and thus we see that edges connecting the two vertices have signature -1 and any cycle in the resulting graph has signature 1. Thus the resulting embedding scheme is orientable, and since the graph embedding was cellular before cutting, it still is after cutting, and the resulting surface is orientable. Therefore, o is orienting. □

The following lemma helps us identify separating and orienting loops in cross-cap drawings. This result also appears in [27, Lems. 3 and 4]. We include a proof for completeness.

Lemma 2.4 *In any cross-cap drawing of an embedding 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.*

Proof Actually this statement does not depend on the whole embedding scheme, only on the loops. It is enough to show that:

- A cross-cap drawing of a separating closed curve passes through each cross-cap an even number of times.
- A cross-cap drawing of an orienting closed curve passes through each cross-cap an odd number of times.

Observe that if γ is one-sided it cannot be separating, moreover γ can separate the surface into at most two connected components, one for each side of γ and if γ is separating every time we cross γ we should change connected component. Now consider a cross-cap drawing of γ, and assume to reach a contradiction that some cross cap is crossed an odd number of times by γ. Choose any wedge around that cross cap and notice that if we go around the cross-cap with a curve γ', to reach the opposite wedge defined by γ, then γ' and γ intersect an odd number of times, so they should be in different connected components of the complement of γ. This is a contradiction since opposite wedges are clearly in the same connected component.

Let γ be an orienting curve, and denote by $\alpha_1, \dots, \alpha_g$ the PD1S underlying the cross-cap drawing. By the classification of surfaces, γ is unique up to homeomorphism, i.e., there exists (another) cross-cap drawing where γ goes exactly once through every cross-cap. The image of a curve α_i under this homeomorphism is a simple closed curve on the same surface, which in this new cross-cap drawing intersects γ and the new cross-caps. Furthermore, it intersects the new cross-caps an odd number of times since it is one-sided. Now it is immediate that in this new representation, γ partitions the plane into two regions, and any curve crossing the cross-caps an odd number of times must cross γ an odd number of times. This proves the needed property. □

Lemma 2.5 *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.*

Proof The embedding scheme has Euler genus $2g$, hence at least $2g$ cross-caps are required. Lemma 6 in [27] states that any cross-cap drawing of an orientable scheme of genus $g \neq 0$ requires an odd number of cross-caps, so at least $2g + 1$ are required. To see that this always suffices we begin with the embedding on the orientable surface and add a cross-cap in one of the faces of G on M; this gives us an embedding of G on a non-orientable surface with one cross-cap and g handles that is homeomorphic to the non-orientable surface N with $2g + 1$ cross-caps. □

2.6 Canonical System of Loops

A *system of loops* is a family of loops cutting a surface into a topological disk. For an orientable surface M of genus g, we define the *orientable canonical system of loops* to be a family of two-sided loops with the cyclic ordering $a_1 b_1 a_2 b_2 \ldots a_g b_g a_g b_g$ around the basepoint, such that cutting M along this family yields a topological disk. For a non-orientable surface N of genus g, the *non-orientable canonical system of loops* is a family of one-sided loops with the cyclic ordering $a_1 a_1 a_2 a_2 \ldots a_g a_g$ around the basepoint such that cutting N along this family yields a topological disk. Specifying a system of loops and a cyclic ordering of the edges around the basepoint is the same as specifying a polygon and data indicating how to glue the edges so as to recover the surface: such a polygon is called a *polygonal scheme*; it is a *canonical polygonal scheme* if the system of loops is canonical.

The following lemma underpins our strategy to prove Theorem 1.3: in order to find a non-orientable canonical system of loops, first find a cross-cap drawing and connect the cross-caps to a root using short paths (see Fig. 1).

Lemma 2.6 *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 to 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.*

Proof It is easy to check that each c_i has consecutive ends around b. Each curve c_i is homotopic to a concatenation of a curve in a planarizing system of disjoint one-sided curves and two copies of a path p_i, therefore it is one-sided. Cutting along these system of curves corresponds to cutting along a planarizing disjoint one-sided loops which gives us a sphere with g boundary components and then cutting along the paths $\{p_i\}$ which connect these boundary components and cut them to a single boundary. Therefore, cutting along $\{c_i\}$, cuts the surface into a disk, and we obtain a non-orientable canonical system of loops. This is illustrated in Fig. 1. □

2.7 Short Orienting Curves

Throughout this paper, to deal with non-orientable surfaces, we turn them to an orientable surface by cutting along an orienting curve. For this approach to work in our problems, we need to ensure that we can find an orienting curve that crosses our primal

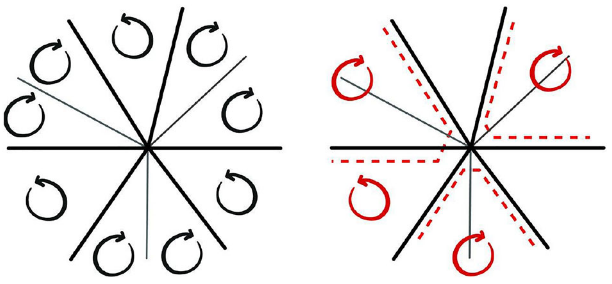

Fig. 5 The thick lines in the left picture, depict the edges adjacent to a vertex that inherit the same orientation from different faces. The dashed red lines in the right picture are shortened and shifted copies of these edges joined according to the pairing. The arrows indicate the refinement of the local orientation

graph not too many times. The following lemma is a restatement of in [21, Prop. 5.5]. We provide a sketch of the proof, explaining how to extend it to arbitrary embedded graphs and how to modify it to get orienting arcs in the presence of boundaries.

Lemma 2.7 *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.

Sketch of the proof We begin by adding edges to the embedded graph G in order to get cellular faces. We assign a local orientation to each face, that is a cyclic order to the vertices of each face along its boundary. Two adjacent faces are said to have *incoherent* orientations if they induce the same orientation on the edge they have in common. Cutting the surface along all incoherent edges gives us an orientable surface. For any vertex v there is an even number of edges with incoherent orientations adjacent to v. We pair these edges around each vertex, shift them slightly so that they cross the original graph only transversely and we add segments to join two paired edges. We can modify our local orientation slightly to show that cutting along our new system of edges gives us an orientable surface; see Fig. 5 which is almost identical to one in the proof of [21, Prop. 5.5].

This collection of edges forms a disjoint union of closed curves (possibly a single curve). These curves can only cross an edge of the graph near the vertices and therefore the whole system of curves crosses each edge of G at most twice. If we have one curve, then this curve is the desired orienting curve. In the case we have more than one curve, we can find short paths (paths that do not intersect edges that are already crossed by the curves) that join a pair of these curves at each step. By slightly changing the orientations around these paths, we can merge these curves and paths to a single curve. For the complete proof see [21]. $\square$

Remark 2.8 If N is a non-orientable surface with a boundary component, by a little modification of the building process in the proof of the lemma, we can get an orienting arc instead of an orienting cycle. If the surface has boundary, the local orientation around each boundary is similar to that of a vertex. We choose one boundary component. We shift incoherent edges around this boundary and join all these edges by pairs except for one pair of edges. This way we get an arc and a system of closed curves and we can proceed as explained in the proof to join these components and get one orienting arc.

3 Correcting Negami's Proof

Negami proved in [24] that if we have two graphs embeddable on a closed surface, we can reembed them simultaneously such that their edges cross few times. The Betti number of a connected graph G is $\beta(G) = |E(G)| - |V(G)| + 1$.

Theorem 3.1 [24, Thm. 1] *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.*

The statement in Theorem 3.1 is correct in the case of orientable surfaces. In the non-orientable case, Negami's proof is incorrect, but the statement remains correct (with a slightly worse constant factor).

Negami reduces the proof of Theorem 3.1 to the following lemmas (with slightly different constants). The proof of Lemma 3.2 is correct, but the proof technique behind Lemma 3.3 is not. An arc is called an *essential* proper arc if it does not cut off a disk from the surface.

Lemma 3.2 *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 exist an orientable surface M of genus g with one boundary component and homeomorphic embeddings of M_1 and M_2 in M so that the images of the arcs in M_1 and M_2 intersect at most $4(g-1)\beta_1\beta_2$ times.*

Lemma 3.3 *For two non-orientable surfaces N_i of genus $g \geq 1$ with one boundary component and β_i disjoint essential proper arcs ($i = 1, 2$), there exist a non-orientable surface N of genus g with one boundary component and homeomorphic embeddings of N_1 and N_2 in N so that the images of the arcs in N_1 and N_2 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 3.3, Negami uses induction on the genus of the non-orientable surface. Assuming that the claim 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). This arc can be either included in the system of arcs or be disjoint from it. 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 such an arc might be orienting (cutting along an orienting arc leaves us with an orientable surface; this interferes with the induction). This is illustrated in the following lemma.

Lemma 3.4 *Consider the non-orientable surface of genus 3 with one boundary component and embedded essential arcs shown in Fig. 6. Any one-sided arc disjoint from the embedded arcs is orienting.*

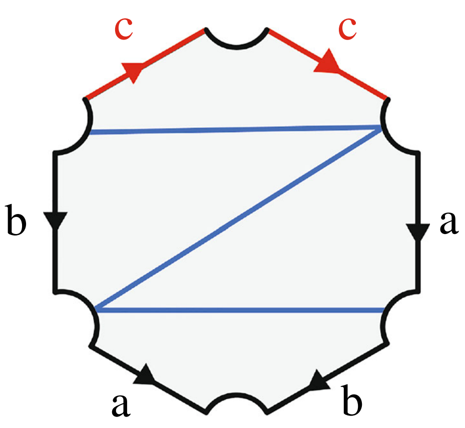

Fig. 6 A non-orientable surface of genus 3 with embedded system of arcs. The dents in the picture indicate the segments of the boundary component

Proof Figure 6 depicts a non-orientable surface of genus 3 (obtained by identifying the boundary edges according to their letters and orientations) and one boundary component, and a family of essential arcs on it (consisting of the boundary edges and the blue arcs). The arc c is a one-sided arc and a and b are two non-homotopic two-sided arcs. The blue arcs are two-sided arcs embedded on the surface. A one-sided arc disjoint from the system of arcs in this polygonal schema, must have one end on the segment of the boundary component between two copies of c and the other end on one of the two other segments adjacent to c. Such an arc is orienting and by cutting along it, we obtain an orientable surface of Euler genus 2 with one boundary. □

3.1 A Correction

To prove Theorem 1.2, we provide a different proof of Lemma 3.3. The idea of the proof is to cut the surface along an orienting curve that does not cross the graph embedded on the surface too many times (this curve exists by Lemma 2.7). By cutting along such a curve, we obtain an orientable surface and we can use Lemma 3.2. Let us first introduce some additional terminology that is tailored to surfaces with boundaries.

Let M be an orientable surface with boundaries that are given with some orientations. We choose an orientation for M, i.e., a consistent choice of clockwise and counter-clockwise for simple contractible curves (see, e.g., Hatcher [14, Sect. 3.3] for a formal definition); such an orientation induces a (possibly different) orientation for each boundary, which we call its natural orientation. We say that the orientations of the boundaries are mutually *compatible* if they either all match the natural orientation or are all oriented oppositely to the natural orientation. Figure 7 shows a surface with non-compatible boundary orientations. For an arc with both ends on the same boundary component, we say the arc is two-sided (resp. one-sided) if the closed curve obtained by connecting the two ends of the arc along one of the boundary segments is two-sided (resp. one-sided). An orienting closed curve γ is either two-sided or one-sided depending on the genus, as described in Lemma 2.1. In the same spirit of this

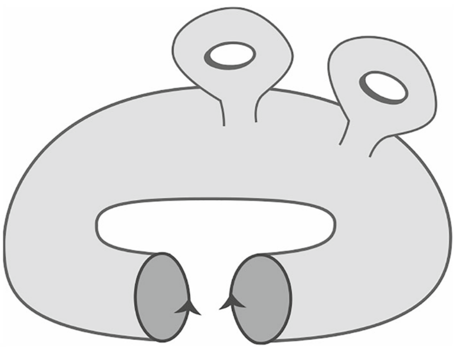

Fig. 7 An orientable surface with non-compatible boundary orientations

lemma, one can characterize orienting arcs instead of orienting closed curves, the only difference is that when g is odd, $h_\gamma = h$ and when g is even, $h_\gamma = h + 1$. This is because cutting along a two-sided arc increases the number of boundary components by one and cutting along a one-sided arc does not change the number of boundary components.

Lemma 3.5 *Let N be a non-orientable surface of even genus and M be the orientable surface obtained from cutting N along an orienting curve. The orientations on the two boundaries of M induced by the orienting curve are compatible.*

Proof If the orientations on the boundaries are not compatible, identifying the boundaries along their orientation introduces a handle which contradicts the fact that N was non-orientable. Figure 7 illustrates this. ◻

Proof of Lemma 3.3 For $i = 1, 2$, let N_i be a non-orientable surface of genus g and with one boundary component, with Γ_i, a system of β_i disjoint essential proper arcs. We distinguish the cases where g is odd from even.

g is odd. Let γ_i be an orienting arc in N_i which has $c_i \leq 2\beta_i$ intersections with Γ_i, whose existence is guaranteed by Lemma 2.7 and Remark 2.8. Cut N_i along γ_i. Let M_i be the resulting surface. From Lemma 2.1 we know that M_i is an orientable surface of orientable genus $\frac{g-1}{2}$ and with a boundary component. Each arc in Γ_i is cut into at most 3 arcs by γ_i. We denote by Γ'_i the system of disjoint essential arcs in M_i and thus we have $|\Gamma'_i| \leq 3\beta_i$. By Lemma 3.2, we know that there exist a surface M' with genus $\frac{g-1}{2}$ and one boundary component and homeomorphisms ϕ_1 and ϕ_2 which map M_i to M' such that $\phi_1(\Gamma'_1)$ and $\phi_2(\Gamma'_2)$ have at most $4^{\frac{g-3}{2}}|\Gamma'_1||\Gamma'_2| \leq 36^{\frac{g-3}{2}}\beta_1\beta_2$ crossings.

We modify ϕ_2 in a small neighborhood of the boundary so that the copies of γ_2 are aligned with those of γ_1. This will allow us to glue back the surface along γ_1 (or γ_2). In order to do so, we slide out the ends of the arcs in Γ'_2 which are not on γ_2, containing $2\beta_2$ ends, into the two segments of the boundary of N_1. Each one of the ends we are sliding might intersect each end of the arcs in Γ'_1 which lies in the two segments of the primary boundary component which contains $2\beta_1$ ends. This modification introduces at most $4\beta_1\beta_2$ intersections. Each copy of γ_i contains c_i ends of the arcs Γ'_i. Next we

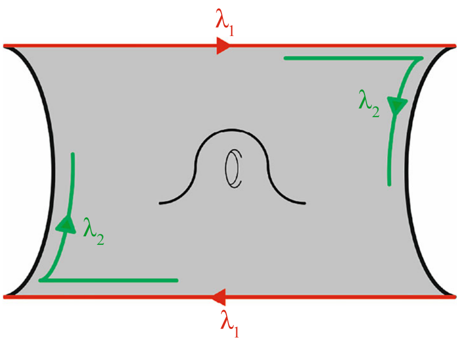

Fig. 8 The non-orientable surface cut along an orienting curve

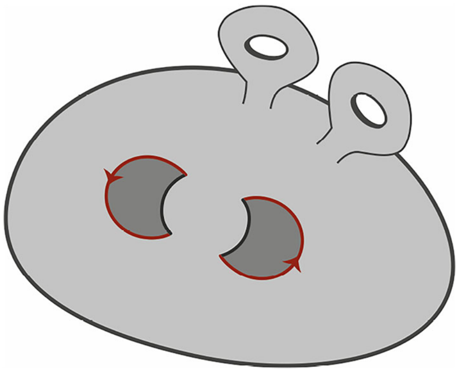

Fig. 9 The surface M_i, the red arcs are the copies of γ_i

align copies of γ_1 with γ_2 and this introduces at most c_1c_2 new intersections on each copy. Therefore, we have $4\beta_1\beta_2 + 2c_1c_2 \le 12\beta_1\beta_2$ new intersections.

After this modification, we are able to glue back $M^{'}$ along γ_1 (or γ_2) to get back to a non-orientable surface N with genus g and one boundary component. Now ϕ_1 and ϕ_2 introduce two homeomorphisms from N_1 and N_2 into N such that $\phi_1(\Gamma_1)$ and $\phi_2(\Gamma_2)$ intersect at most $12\beta_1\beta_2 + 36\frac{g-3}{2}\beta_1\beta_2 \le 18(g-1)\beta_1\beta_2$ times.

g is even. Let γ_i be an orienting arc in N_i which has $c_i \le 2\beta_i$ intersections with Γ_i whose existence is guaranteed by Lemma 2.7 and Remark 2.8. Let M_i be the surface obtained by cutting N_i along γ_i. Each arc in Γ_i is cut into at most 3 arcs by γ_i. We denote by Γ_i' the system of disjoint essential arcs in M_i and we have $|\Gamma_i'| \le 3\beta_i$. By Lemma 2.1, we know that M_i is an orientable surface of genus $\frac{g-2}{2}$ and two boundary components with $3\beta_i$ disjoint essential arcs. Fixing an orientation for γ_i will let us see how the two boundary components were pasted and according to Lemma 3.5, we know that these orientations are compatible (see Fig. 9).

We claim that there is a curve ν_i with an end on each of the boundary components and with at most $3\beta_i$ intersections with the system of arcs in M_i. We know that cutting along an arc increases the Euler characteristic of the surface by 1 and we choose ν_i such that it reduces the number of boundary components by 1. From the relation $\chi = 2-2g-h$ where h is the number of boundary components, we will see that after

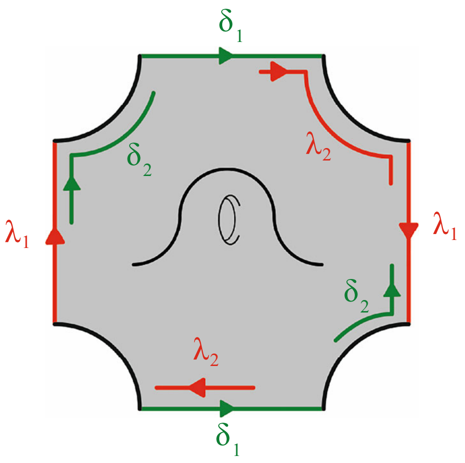

Fig. 10 The orientable surface $M^{'}$ with one boundary component

the cut, the genus is unchanged. Therefore, by cutting along ν_i, we get the orientable surface $M_i^{'}$ with genus $\frac{g-2}{2}$ and one boundary component with $6\beta_i$ disjoint essential arcs.

We choose ν_i such that its ends lie on the segments of the boundary components which belonged to the primary boundary component of N_i and so that it connects these two boundaries. The problem reduces to finding such an arc so that it has at most $3\beta_i$ intersections with the system of arcs in M_i. Consider the graph H such that the vertices are the endpoints of the arcs and the edges are the arcs Γ_i' which we denote by E_1 together with the segments on the boundary components denoted by E_2. We choose a face ρ which has one edge in E_2 such that a part of this edge lies on the segment of the primary boundary component. We choose the face ρ' analogously to ρ but on the other boundary component. The shortest path between vertices associated to ρ and ρ' in the dual graph of H induces a curve with ends on ρ and ρ' that passes through the inner faces at most once. We connect the ends of this curve to the edges on the boundary components in both ρ and ρ'. This gives us an arc ν_i that intersects each edge of E_1 at most once. Thus ν_i is the desired arc.

Again, by Lemma 3.2, we know that there exist ϕ_1 and ϕ_2 mapping $M_1^{'}$ and $M_2^{'}$ to $M^{'}$ with $4\frac{g-4}{2}\cdot 6\beta_1\cdot 6\beta_2$ intersections on the arcs (see Fig. 10). Similar to the case where the genus was odd, we can modify ϕ_1 and ϕ_2 by introducing at most $4\cdot 6\beta_1\cdot 6\beta_2$ intersections such that we can align copies of γ_1 with γ_2 and ν_1 with ν_2 and glue back $M^{'}$ to obtain N and at the end we have at most $72(g-2)\beta_1\beta_2$ intersections. This finishes the proof. □

Sketch of the proof of Theorem 1.2 Once we are equipped with Lemma 3.3, the proof of Theorem 1.2 follows the same strategy as that of Negami [24]. For completeness, we provide the following sketch. For each $i = 1, 2$, we contract a spanning tree in G_i, reducing it to a one-vertex embedding scheme, and remove contractible arcs, yielding an embedding scheme E_i. We puncture the surface at the single vertex of E_i, yielding a non-orientable surface N_i with one boundary component and $O(|E(G_i)|)$ disjoint essential arcs. We are now in the situation to apply Lemma 3.3, which gives us a pair of homeomorphisms sending N_1 and N_2 to N such that the number of crossings between the arcs is $O(g|E(G_1)||E(G_2)|)$. We glue back a disk on the puncture and connect all the incoming arcs of E_i to a basepoint b_i, so that the two basepoints b_1 and b_2 are slightly off. This induces an additional $O(|E(G_1)||E(G_2))$ crossings. Finally, we add back the contractible arcs and uncontract the spanning trees in small neighborhoods of b_1 and b_2, which does not change the number of crossings, concluding the proof. □

4 Non-orientable Canonical System of Loops

The objective of this section is to prove the following theorem.

Theorem 1.3 *There exists a polynomial time algorithm that given a graph cellularly embedded on the non-orientable surface N 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.*

In order to prove this theorem, we heavily rely on an algorithm introduced by Schaefer and Štefankovič in [27] for drawing graphs on non-orientable surfaces. Compared to that algorithm, our approach enforces more structure on this construction by imposing specific orders in the way we deal with the loops in the induction. In Sect. 4.1, we first explain this algorithm and then in Sect. 4.2, we introduce and explain our modifications, ultimately leading to a proof of the theorem in Sect. 4.3.

4.1 The Schaefer–Štefankovic Algorithm

Throughout this section, we will be working with a one-vertex graph G endowed with an embedding scheme (ρ, λ), which for simplicity we denote by G. Schaefer and Štefankovič proved the following theorem.

Theorem 4.1 ([27, Lem. 9]) *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.*

The proof is by induction on the number of loops and is readily algorithmic, computing an actual cross-cap drawing with the required bound on the number of intersections with the cross-caps. Before explaining the algorithm, let us introduce the different moves and techniques we use to deal with different types of loops.

First, let us introduce the following terminology 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 the empty wedge between two consecutive half-edges around v a *root wedge*. A face incident to v in a cross-cap drawing may correspond to more than one root wedge. We refer to such a face as a *root face*.

4.1.1 Contractible Loop Move

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

4.1.2 Gluing Move

Let s be a non-contractible separating loop in the embedding scheme G. We divide the embedding 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 formerly was 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 (Lemma 2.2) 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 Fig. 11. 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). In order to get a drawing with minimum number of cross-caps, we need to eliminate one cross-cap from the drawing. 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 4.2 *Let G be a one-vertex orientable embedding scheme. Adding a one-sided loop o with consecutive ends to the embedding scheme (anywhere in the rotation around the vertex) increases the Euler genus by 1. Thus, the new embedding scheme needs as many cross-caps as G to embed. Furthermore, the loop o is orienting.*

Proof Adding a one-sided curve with consecutive ends to an embedding scheme does not change the number of faces. By the Euler formula we know that the Euler genus of the new embedding scheme is increased by 1. By Lemma 2.5, G needs $eg(G) + 1$ cross-caps to embed. Therefore, the new embedding scheme needs as many cross-caps as G to embed.

The loop o is the only one-sided loop in the embedding scheme and every other loop is in the same wedge of o. Lemma 2.3 implies that o is orienting. □

4.1.3 Dragging Move

Let us assume that G_2 is orientable. By Lemma 4.2, 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 embedding scheme needs $eg(G_2) + 1$ cross-caps to be drawn. Having a drawing for the new embedding 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

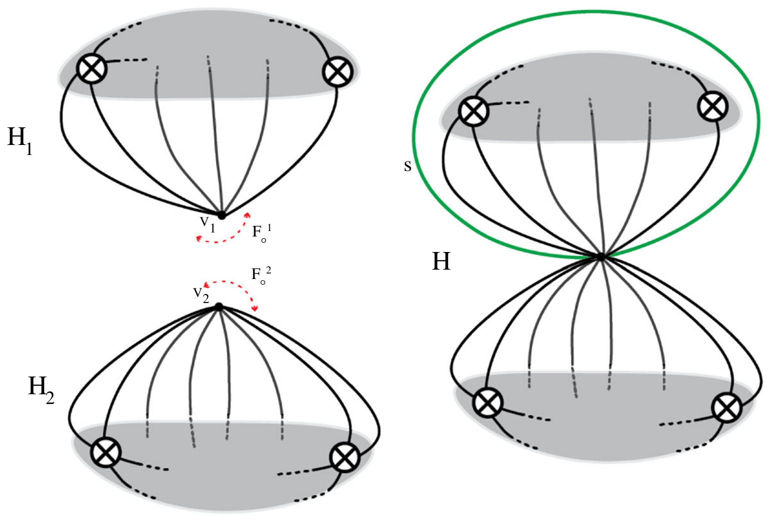

Fig. 11 The gluing move on two cross-cap drawings when G_1 and G_2 are both non-orientable

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 Fig. 12. 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 $(i_1, F_1, \dots, i_{2k}, F_o^1)$ 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 $i_1, \dots, i_k$ 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 $(i_{k+1}, \dots, i_{2k})$. Since G_2 uses an odd number of cross-caps (Lemma 2.5), the half-edges will have the correct orientability and order to get attached to their other ends; see Fig. 12. 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 which is the minimum number of cross-caps needed to draw the embedding scheme in this case.

4.1.4 One-Sided Loop Move

Let r be a one-sided loop in the embedding scheme G. We remove r and flip one of its wedges. One can check that the new embedding 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 Fig. 13. Note that flipping different wedges of

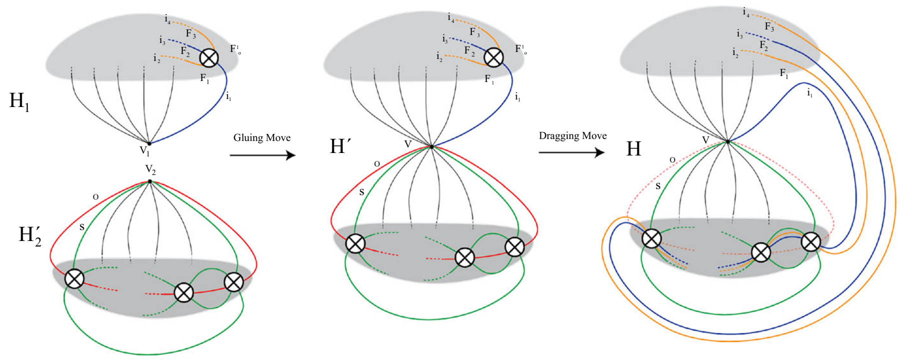

Fig. 12 Left: the gluing move. Right: the dragging move when G_2 is orientable: the top right cross-cap is removed and the corresponding curves are dragged through the bottom component

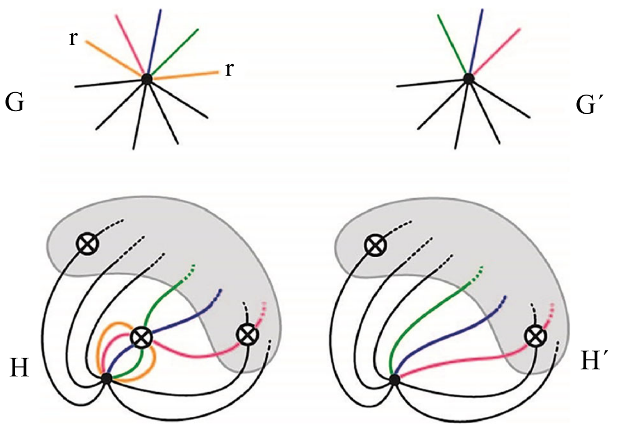

Fig. 13 The one-sided loop move on the loop r

r leads to two different cross-cap drawings. This freedom in choosing the wedge is important for us and we use this later in the paper.

If r is not orienting, the drawing we get at the end uses $eg(G)$ cross-caps. But if r is orienting, then G' is orientable and any drawing for G' needs an extra cross-cap (Lemma 2.5). This means that if we apply a one-sided loop move to an orienting loop, the drawing we get does not use the minimum number of cross-caps (the embedding is not cellular).

We use the following move to deal with orienting loops.

4.1.5 Concatenation Move

Let o be an orienting loop in the embedding 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 2.3, since t is non-separating, the concatenation of o and t which we denote by o', is not orienting. Denote by G' the embedding 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

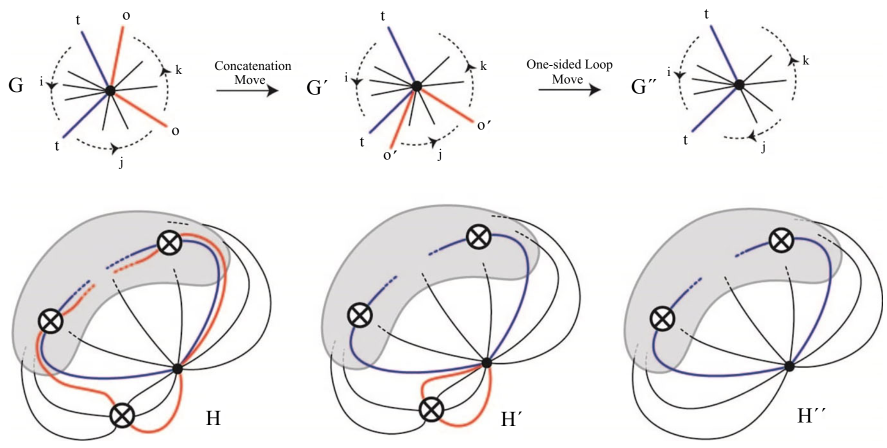

Fig. 14 The concatenation move on the loops o and t, when in applying the one-sided loop move we flip the wedge of $o^{'}$ that does not encompass the ends of the loop t

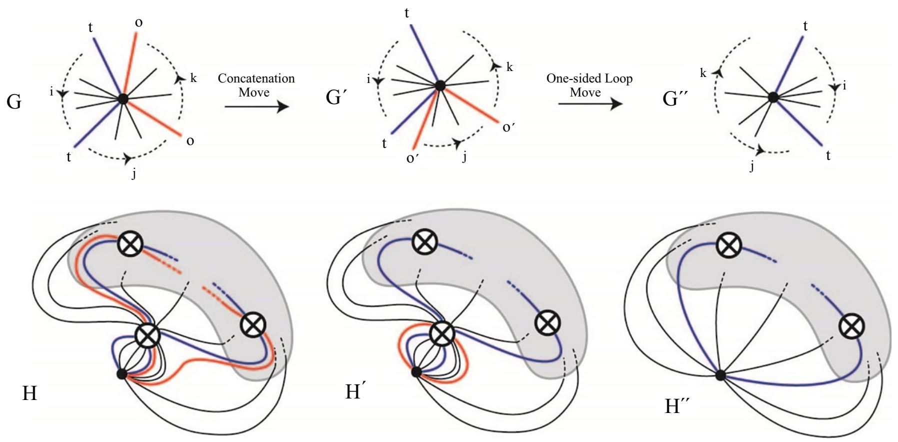

Fig. 15 The concatenation move on the loops o and t, when in applying the one-sided loop move we flip the wedge of $o^{'}$ that encompasses the ends of the loop t

can obtain from $H^{'}$ a drawing for 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 Fig. 14.

If we flipped the wedge that encompasses the loop t, we do as follows: the loop $o^{'}$ passes through 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 Fig. 15.

4.1.6 Exchange Move

Let G be an embedding scheme that has only two-sided loops and no separating loop. We exchange two consecutive half-edges and change the signatures of the corresponding edges. One can prove that the new embedding scheme can be drawn using the same number of cross-caps as the initial embedding scheme. Having a drawing of the new embedding scheme, we obtain a drawing of the original embedding scheme by adding a cross-cap near the reversed half-edges. This drawing uses $eg(G) + 1$ cross-caps and this is the minimum number of cross-caps we need to draw an orientable scheme (Lemma 2.5).

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. Therefore, we can use the moves in an inductive algorithm as follows. The input is an embedding scheme G.

The Schaefer-Štefankovič algorithm:

- **Step 1: If there exists a contractible loop.** We recurse on the embedding scheme without the loop and apply the contractible loop move.
- **Step 2. If there exists a separating (non-contractible) loop s.** We divide the embedding scheme into two sub-schemes on each side of s. We have the following cases:
  - **Step 2.1: Both sub-schemes are non-orientable.** We recurse on the two sub-schemes and apply the gluing move.
  - **Step 2.2: At least one of the sub-schemes is orientable.** We recurse on the two sub-schemes and apply the gluing move followed by the dragging move.
- **Step 3.1: If there is a one-sided non-orienting loop r.** We recurse on the embedding scheme without the loop r and the flipped wedge, and apply the one-sided loop move to the loop r.
- **Step 3.2: If all one-sided loops are orienting.** If there exists no two-sided loop, by Lemma 2.3, all pairs of loops are interleaving and we can draw the embedding scheme with one cross-cap so that each loop enters it once. If there exists a two-sided loop in the embedding scheme, we can find a place in the embedding scheme where an orienting loop o is followed immediately by a two-sided loop t. We recurse on the embedding scheme H' described in the concatenation move, and apply the concatenation move to the curve o.
- **Step 3.3: If every loop in the embedding scheme is two-sided.** We apply the exchange move to two consecutive half-edges and recurse on the new embedding scheme.

The proof of Schaefer and Štefankovič of Theorem 4.1 proceeds by analyzing this algorithm and proving that (1) the resulting cross-cap drawing has the correct number of cross-caps and (2) each loop passes through every cross-cap at most twice.

Remark 4.3 By Lemma 2.4, in a drawing that is obtained by this algorithm, 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.

There is some leeway in this algorithm: while the steps have to be applied in this specific order, in each step a loop of the given type is chosen arbitrarily. Our modification of the algorithm follows the exact same blueprint but enforces specific orders in which we choose separating and one-sided non-orienting loops. These specific orders provide more structure to the resulting drawing, make it lend itself more to connecting the cross-caps, in order to form the desired non-orientable canonical system of loops of low multiplicity.

4.2 Our Modification to the Schaefer–Štefankovic Algorithm

4.2.1 Preprocessing

Our algorithm first starts with some preprocessing.

Lemma 4.4 *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 such that each loop crosses each edge of $\hat{G}$ at most k times, then G has a non-orientable canonical system of loops such that each loop crosses each edge of G at most $3k$ times.*

Proof By Lemma 2.7, there exists an orienting curve γ embedded on the surface N that has multiplicity two. Denote by $G^{'}$ the overlay of G and γ, that is, the graph obtained by superimposing G and γ and adding dummy vertices of degree four at the crossings. Each edge in G is divided into at most three sub-edges in $G^{'}$. If γ crosses G, n times totally, then it corresponds to n edges in $G^{'}$. We choose a spanning tree T in $G^{'}$ that contains $n-1$ of these edges. Without loss of generality, we can assume that all the signatures of the edges of T are $+1$ (this is because one can apply local changes to the embedding scheme without changing its homeomorphism class, see, e.g. [23, Sect. 4.1]). We contract the edges of T into a single point to get a one-vertex graph $\hat{G}$ and merge the rotations in the embedding scheme at the vertices to obtain an embedding scheme for the one-vertex graph $\hat{G}$. The non-contracted sub-edge of γ is an orienting loop in $\hat{G}$.

Let us assume that $\hat{G}$ has a non-orientable canonical system of loops such that each loop crosses each edge of $\hat{G}$ at most k times. We can uncontract the spanning tree T close to the vertex so that the uncontracted edges do not cross the canonical system of loops. This gives us a canonical system of loops for $G^{'}$. To get a drawing for G, we remove the orienting curve γ. Since any edge of G is formed by at most three edges of $G^{'}$, the canonical system of loops for $G^{'}$ crosses each edge of G at most $3k$ times. □

Remark 4.5 The curve γ divides each edge of G into at most three sub-edges. Therefore, each edge in G is obtained as the concatenation of at most 3 edges of $\hat{G}$.

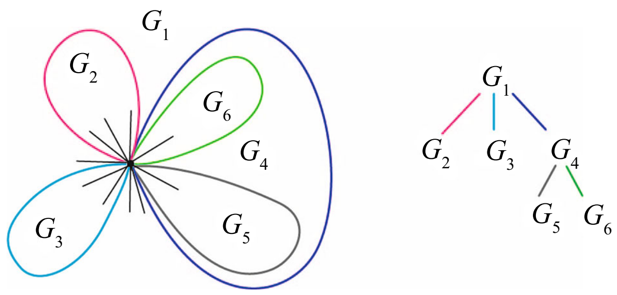

Fig. 16 Left: a saturated one-vertex scheme in which the drawn loops are the separating loops; right: the component tree of the left embedding scheme. The component G_4 is an empty sub-scheme

This lemma lets us assume, at the cost of a slightly bigger multiplicity, that (1) the input graph is a one-vertex graph endowed with an embedding scheme, and (2) the embedding scheme that we work with contains an orienting loop. This orienting loop will in turn allow us to avoid some steps in the algorithm.

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 [13]. 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, see Fig. 17 for an example. 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. We call a component with only one-vertex a *trivial* component, and *non-trivial* otherwise. Separating loops (contractible or non-contractible) correspond to isolated two-sided vertices, i.e., trivial orientable components, in the interleaving graph.

Our second preprocessing step aims at subdividing G into sub-schemes G_i such that each I_{G_i} has only one non-trivial component. In order to do this, we *saturate* the embedding 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.

Remark 4.6 If $\bar{G}$ is a saturated extension of G, then $eg(G) = eg(\bar{G})$ by Lemma 2.2. Thus a minimum genus drawing of $\bar{G}$ contains a minimum genus drawing of G.

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 Fig. 16). The *component tree* of G has a vertex for every such subgraph G_i, and two vertices are connected if their corresponding components are separated by a separating loop. See Fig. 16 for an example of a component tree. We shall quickly see that in the context of our algorithm, there will actually be exactly one non-orientable component. We root the tree at the vertex corresponding to the non-orientable component.

In a saturated embedding scheme, each separating loop relates two subgraphs G_i that exist on its different sides. If we have k non-trivial components or empty subgraphs in total in the interleaving graph, in order to separate all of them from each other, we need exactly $k - 1$ separating loops.

4.2.2 A Cross-cap Drawing Algorithm

We are now ready to describe our modified algorithm. Our modification consists of forcing the presence of an orienting curve on the graph using Lemma 4.4 and putting more structure on the order we deal with different separating loops and different one-sided non-orienting loops.

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 4.4.
- **Step B.** If G is not saturated by separating curves, we saturate it.

Main loop:
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.

- **Step 1: If there is a contractible loop.** We recurse on the embedding 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 sub-schemes 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 embedding 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 to 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 to 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 orienting loop added in step A.

The numbering in this algorithm comes from the Schaefer–Štefankovič algorithm. A main difference with our modified version is that it is not clear at first sight that we cover all cases as we do not have the steps 2.1 and 3.3 in the Schaefer and Štefankovič algorithm. Yet we do cover all the cases: this follows from the presence of an orienting loop and proving it will be the main purpose of the subsections 4.2.3 and 4.2.4. After that we will be ready to prove in Lemma 4.15 that our algorithm terminates and outputs a cross-cap drawing where a loop does not enter each cross-cap more than 6 times.

4.2.3 Completeness of the Case Analysis

The following lemma explains why we do not need to consider the case in which all loops are two-sided.

Lemma 4.7 *An embedding scheme with an orienting loop is non-orientable.*

Proof Let us assume that G is an orientable scheme, hence the orienting loop o is two-sided. By Lemma 2.5, G needs $eg(G) + 1$ cross-caps to embed and we know that $eg(G)$ is twice the orientable genus of G. Therefore, G needs an odd number of cross-caps to embed and the loop o passes through each of them an odd number of times. Thus, o is one-sided which is a contradiction. □

Lemma 4.7 guarantees that we do not need to consider the case where all loops are two-sided when the algorithm starts, but this case might a priori still happen during recursive calls to the algorithms. Fortunately, this will actually not be the case, as we will prove in Corollary 4.13 that there is always an orienting loop in each of the recursive calls.

The following lemma explains why there is only one case that can happen in step 2 of our algorithm.

Lemma 4.8 *Let G be an embedding scheme with an orienting loop o and a non-contractible separating loop s. The loop s separates the graph into an orientable and a non-orientable subgraph.*

Proof By Lemma 4.7, G is non-orientable and therefore it has at least one one-sided loop. Let us assume that s separates the embedding scheme into G_1 which contains the loop o, and G_2. We show that G_1 is non-orientable and G_2 is orientable.

By Lemma 2.3, any one-sided loop has exactly one end in the wedge of the orienting loop in G_1 and has to have both ends in the same side of the separating loop, therefore no one-sided loop can exist in G_2 so G_1 is non-orientable and G_2 is orientable. The case where the only one-sided loop is the orienting loop is trivial. □

4.2.4 The Order on the One-Sided Non-orienting Loops

In this section, we explain an order on one-sided loops when we apply the one-sided loop move in step 3.1 of the algorithm. The reason for imposing this restriction is to avoid creating separating loops in the induction that did not exist in the embedding scheme initially.

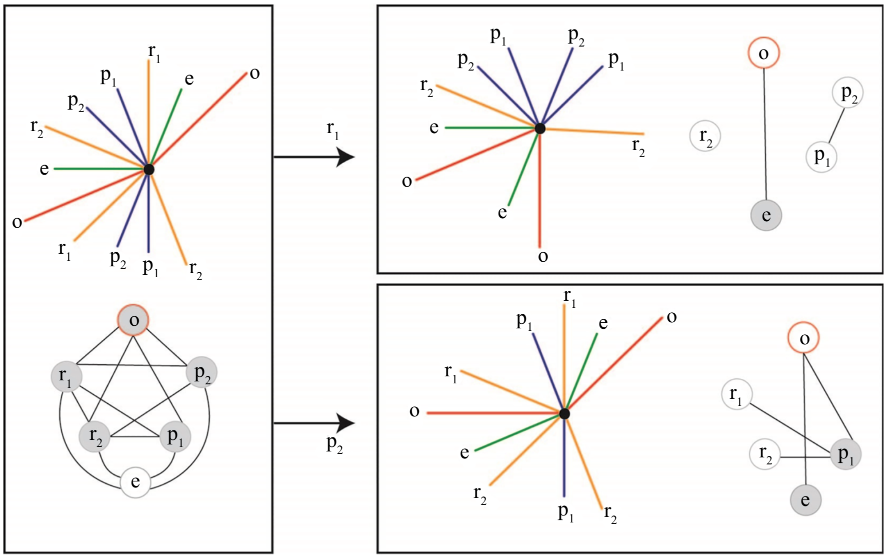

Fig. 17 Each box represents an embedding scheme with its interleaving graph. In all these embedding schemes the loop o is orienting; right: the impact of applying the one-sided loop to r_1 (top) and p_2 (bottom)

Lemma 4.9 *If there exists an orienting loop o (two-sided or one-sided) in the embedding scheme G, the connected component that has the vertex o is the only non-orientable component in I_G.*

Proof By Lemma 2.3, the ends of every one-sided loop interleave with the ends of the orienting loop. Therefore, the vertex o is connected to every one-sided vertex in I_G and this finishes the proof; see Fig. 17. □

The following lemma is analogous to a similar result in signed reversal distance theory [2, Fact 2].

Lemma 4.10 *Applying a one-sided loop move to a loop r corresponds to removing the vertex r in the interleaving graph and complementing the subgraph induced by its neighbors.*

Proof In a one-sided loop move, we remove a one-sided loop r and flip one of its wedges. When we flip, we change the signature of each loop that has exactly one end in each wedge of r. Thus, we are changing the sidedness of everything that was connected to the vertex r in the interleaving graph. Also, due to the flip, every two vertices adjacent to r, that were connected to each other before the flip, are now disconnected and vice versa. The situation of the loops that are not interleaving with r in the embedding scheme is unchanged; see Fig. 17. □

Such a move can change the number of connected components in the interleaving graph, and might increase the number of orientable components, as is the case when we apply the one-sided loop move to o in Fig. 17.

Lemma 4.11 *Let G be an embedding scheme with orienting loop o and a one-sided non-orienting loop r. Let $G^{'}$ be the graph we obtain after removing r and flipping its wedge. The loop o is orienting in $G^{'}$.*

Proof We need to show that after removing r, the loop o is connected to all the one-sided vertices in the interleaving graph and it is not connected to any two-sided vertex (Lemma 2.3). We know that at the start o is connected to r. Any one-sided loop that r used to be adjacent to is now two-sided and since o used to be connected to these loops, by complementing the subgraph induced by the vertices adjacent to r (Lemma 4.10), o is no longer connected to them. Similarly, we can see that if r used to be connected to two-sided loops, after flipping the wedge, they become one-sided and since the loop o was not connected to any two-sided vertex before, now it gets connected to them. The situation for the loops to which o was connected and r was not, remain unchanged. □

Lemma 4.12 *If G is an embedding scheme that contains only orienting and non-separating two-sided loops, and has at least one of each, then there exists an orienting loop o followed by a non-separating two-sided loop t. Denote by $o^{'}$ a loop homotopic to the concatenation of o and t. If $G^{'}$ is the embedding scheme obtained by replacing o by o', and $G^{''}$ is the embedding scheme obtained by applying a one-sided loop move to $o^{'}$ in G', then t is orienting in $G^{''}$.*

Proof Since there exist only orienting and two-sided non separating loops, and there is at least one of each, there exists an orienting loop o and a non-separating two-sided loop t such that an end of o is consecutive to an end of t in the embedding scheme.

First we show that t and o belong to different components in I_G. Since o is orienting and there is no one-sided non-orienting loop in the embedding scheme, the component of o is a complete graph with only orienting loops. Therefore, t does not belong to this component and everything in the component of t is two-sided. Replacing o by $o^{'}$ corresponds to replacing the vertex $o^{'}$ with o in I_G and connecting it to the neighbors of t, since $o^{'}$ now interleaves with every loop that t interleaves with. Now applying the one-sided loop move to $o^{'}$ makes every neighbor of t one-sided and all the orienting vertices (that were formerly adjacent to o) two-sided. Therefore, the only one-sided loops in the new embedding scheme are the neighbors of t and t is not adjacent to any two-sided vertices since everything in its component used to be two-sided. By Lemma 2.3, t is orienting in $G^{''}$. □

Since the contractible loop move clearly preserves orienting loops, Lemmas 4.8 (the dragging move first adds an orienting loop to the orientable part), 4.11 and 4.12 imply the following Corollary 4.13.

Corollary 4.13 *Let G be a one-vertex scheme with an orienting loop. Let G' be the graph on which the modified algorithm recurses when applying one of the following moves:*

- *A contractible loop move.*
- *A one-sided loop move on a one-sided non-orienting loop.*
- *The concatenation move on an orienting loop.*

Then G' has an orienting loop. Likewise, when the modified algorithm applies a gluing and dragging move to a separating loop s, the two subgraphs G_1 and G_2 on which it recurses have an orienting loop.

The next lemma explains our choice of rule in Step 3.1:

Lemma 4.14 *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. The embedding scheme G can be drawn by applying a sequence of contractible loop moves, one-sided loop moves and concatenation moves. Specifically, we do not use the gluing move, the dragging move and the exchange move.*

This ensures that in Step 3.1 no non-contractible separating loop is created during the process, hence we can avoid increasing the number of orientable components. This proof mirrors results in the signed reversal distance theory (see Bergeron [2, Thm. 1]) in which similar claims are proved in the context of applying reversals to permutations.

Proof In order to have a non-contractible separating loop, it is necessary to have either two non-trivial components, or an orientable non-trivial component and a trivial non-orientable component (an isolated one-sided vertex).

Since there exists an orienting loop in the embedding scheme and the ends of the orienting loop interleave with those of every one-sided loop, all one-sided loops belong to the same component in the interleaving graph and thus there exists only one non-orientable component. By Corollary 4.13, in applying these moves, there is always an orienting loop in every step. This, together with Lemma 4.9, implies that the number of non-orientable components remains 1 through the algorithm. Therefore, it suffices to prove that we can draw G such that in each step we do not increase the number of non-trivial orientable components.

The Non-trivial Component is Non-orientable If there exists no one-sided non-orienting loop, then every loop in the embedding scheme is orienting or contractible. In this case, the non-orientable genus is one and the result is trivial. Let us assume that there exists at least a one-sided non-orienting loop. We claim that if we choose the one-sided non-orienting loop such that flipping its wedge maximizes the number of one-sided loops, then we do not increase the number of orientable components. In Fig. 17, it is shown that applying a one-sided loop move to p_2 maximizes the number of one-sided loops but applying it to r_1, increases the number of orientable components and turns r_2 into a separating loop for the new embedding scheme.

Let us assume that applying a one-sided loop move to the one-sided loop r maximizes the number of one-sided loops and increases the number of non-trivial orientable components. Let i be a vertex that was adjacent to r, belonging to a component U that got disconnected when complementing the subgraph induced by the neighbors of r. The vertex i was one-sided and we claim that taking i instead of r would have created more one-sided vertices which is a contradiction.
Denote by $\#_1(r)$ (resp. $\#_2(r)$) the number of one-sided (resp. two-sided) loops adjacent to r. Removing a one-sided loop is equivalent to removing the vertex in the interleaving graph and complementing the subgraph induced by its adjacent vertices. Therefore,

removing r increases the number of one-sided loops in the embedding scheme by $\#_2(r) - \#_1(r)$. All two-sided vertices adjacent to r are one-sided after removing r and therefore they are not in U, meaning that they were formerly connected to i, so we have $\#_2(i) \geq \#_2(r)$.
Similarly, r has to be connected to every one-sided vertices that were formerly connected to i, so we have $\#_1(r) \geq \#_1(i)$.
If $\#_2(i) = \#_2(r)$ and $\#_1(r) = \#_1(i)$, then they have the same neighbors and removing r will isolate i which contradicts the fact that the connected component U is not trivial. Therefore, applying a one-sided loop move to the loop i creates more one-sided loops than applying a one-sided loop move to the loop r, which is a contradiction. Thus, removing r cannot add to the number of non-trivial orientable components.

The Non-trivial Component is Orientable In this case, there exists only one one-sided loop o which is orienting. We replace o with its concatenation with one of the two-sided loops that is immediately next to it in the rotation. Denote this two-sided loop by t and the concatenated loop by $o^{'}$. This corresponds to replacing the vertex o by $o^{'}$ that is connected to all the neighbors of t in the interleaving graph and therefore reduces the number of components by 1. Applying the one-sided loop move to $o^{'}$, isolates t and makes it orienting for the new embedding scheme (Lemma 4.12). The resulting graph falls in the last case where the non-trivial component is non-orientable and therefore we can draw it by applying only contractible loop moves and one-sided loop moves. This completes the proof. □

4.2.5 Correctness of the Modified Algorithm

Lemma 4.15 *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.*

Proof By Lemma 4.4, Step A in the algorithm reduces the graph G to a one-vertex scheme $\hat{G}$ that has an orienting loop such that a drawing $\hat{G}$ leads to a drawing for G. Let $\bar{G}$ be the embedding scheme that we obtain after step B on $\hat{G}$. This step does not change the Euler genus of the embedding scheme.

Thus, by Remark 4.5, it is sufficient to prove that there is a cross-cap drawing for $\bar{G}$ with $eg(G) = eg(\bar{G})$ cross-caps in which each edge passes through each cross-cap at most twice. We prove this claim for any one-vertex scheme G with an orienting loop.

In order to prove this claim, we follow the recursive steps of the main loop of the modified algorithm, using induction on $eg(G) + |E(G)|$.

Step 1. We apply the contractible loop move to every contractible loop. By the induction hypothesis, we can obtain a drawing with $eg(G)$ cross-caps for the resulting embedding scheme in which every loop passes through each cross-cap at most two times and the contractible loop does not use any of them.

Step 2. If there exists separating (non-contractible) loops, we deal with them in the prescribed order. Take the separating loop s and divide the embedding scheme. By

Lemma 4.8, we know that one of these subgraphs is orientable and the other one is non-orientable, without loss of generality let us assume that G_1 is non-orientable. We apply a combination of the gluing move and the dragging move to these subgraphs: we add an auxiliary orienting loop o to G_2. By the induction hypothesis, there are cross-cap drawings H_1 with $eg(G_1)$ cross-caps and a drawing H_2 with $eg(G_2)+1$ cross-caps for G_1 and $G_2+\{o\}$ so that each edge of G_1 and G_2 passes through each cross-cap at most twice. Let $H_2^{'}$ be the drawing for $G_2+\{o\}+\{s\}$ that we obtain as described in the dragging move. By Remark 4.3 and by the induction hypothesis, we know that in H_2, the loop o passes through each cross-cap exactly once. The loop s follows o twice and therefore it passes through each cross-cap in H_2 exactly twice.

The gluing move does not interact with the number of entrances for any loop. In the dragging move, every loop that is being dragged from H_1 to H_2 follows the auxiliary orienting loop o in G_2. Since each edge in H_1 passes through each cross-cap at most twice, at most two of its sub-edges are being dragged along o and therefore they pass through the cross-caps in H_2 at most twice.

Since our graph is non-orientable, and we have only dealt with two-sided loops so far, not all of the edges can be orientable at this point.

Step 3.1. If G has one-sided non-orienting loops, we apply a one-sided loop move to the one that respects our prescribed order; we denote this loop by r. Since r is non-orienting, the embedding scheme $G^{'}$ obtained after removing r and flipping its wedge is still non-orientable and by the induction hypothesis, there is a drawing $H^{'}$ with $eg(G)-1$ cross-caps for $G^{'}$ in which every loop passes through each cross-cap at most twice. After drawing r and the new cross-cap, we can see that each loop in the flipped wedge passes through this cross-cap at most twice (the exact value depends on the number of its ends within the flipped wedge).

Step 3.2.a We can find an orienting loop o that is followed immediately by a two-sided loop t. We apply the concatenation move to o and replace it with the non-orienting loop $o^{'}$ (denote by $G^{'}$, the embedding scheme we obtain at this point; we have $eg(G^{'})=eg(G)$). We apply the one-sided loop move to $o^{'}$. By the induction hypothesis, there is a drawing $H^{'}$ for $G^{'}$ with $eg(G)$ cross-caps in which each loop passes through each cross-cap at most twice. Since we first apply a one-sided loop move to $o^{'}$, we can see that $o^{'}$ uses only one cross-cap. Depending on our choice in flipping a wedge of $o^{'}$, the loop t uses this cross-cap either twice or not at all (the loop t uses every cross-cap except this cross-cap exactly once, since after removing $o^{'}$, t gets orienting, see Lemma 4.12). One can check that at the end for both choices of wedge, the loop o passes through each cross-cap exactly once.

Step 3.2.b If all one-sided loops in the embedding scheme are orienting and there is no two-sided loop in the embedding scheme, all of the orienting loops in the embedding scheme are homotopic and we can draw them using one cross-cap with all of the loops passing though the cross-cap exactly once.

By Corollary 4.13, the graph has an orienting loop at each step of the algorithm and therefore by Lemma 4.8, we never have a graph in which every loop is two-sided throughout the algorithm. This completes the proof of the claim and we conclude. □

This proof is independent of the orders we defined for the loops in Step 2 and Step 3.1. These orders are shown to be useful in the next section.

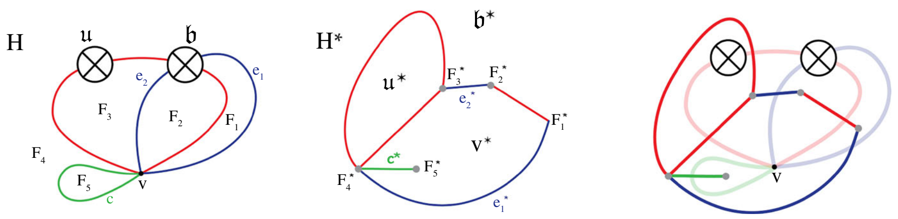

Fig. 18 H is a cross-cap drawing for an embedding scheme G and H^* is the dual graph to H. The rightmost picture shows the overlay of H and H^*. The red loop in H is orienting and the red edges in H^* correspond to segments of this orienting loop. The faces $\mathfrak{b}^*$, $\mathfrak{u}^*$ and v^* in the dual, each has exactly two edges corresponding to the orienting loop

In addition to the fact that our proof does not cover all the steps that happen in the original case (the case that there might not exist an orienting loop in the embedding scheme), another difference between this proof and the proof of the Schaefer and Štefankovič algorithm is in Step 3.2. In this Step, the original algorithm flips the wedge of o' that does not encompass the loop t. We prove the step for both choices of wedge because we favour the freedom to choose a wedge that we want to flip for our further purposes.

4.3 The Non-orientable Canonical System of Loops

The modified algorithm that we described in the previous sections provides us with a cross-cap drawing of any embedded graph G where each edge of the graph enters each cross-cap at most six times, as per Lemma 4.15. Furthermore, 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. When the algorithm draws an embedding scheme with a single non-trivial component, it only relies on contractible, one-sided and concatenation moves. Second, due to the order in which we choose the loops in Steps 2 and 3.1, 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.

4.3.1 The Dual Graph of the Cross-cap Drawing

In this rather tedious but straightforward section, we first investigate the effect of every move involved in the modified algorithm on the dual graph of the cross-cap drawing (viewed as a planar graph). Every edge e in a cross-cap drawing H, corresponds to an edge e^* and every face F corresponds to a vertex F^* in the dual graph. The vertex v corresponds to the face v^* and the cross-caps correspond to the other faces in the dual graph. See Fig. 18 for an example of a cross-cap drawing and its dual. In this section, cross-caps are denoted by $\mathfrak{u}$, $\mathfrak{b}$, $\mathfrak{c}$, $\mathfrak{k}$ and $\mathfrak{g}$ and the letter F is reserved for faces in a cross-cap drawing.

By Remark 4.3, in the drawing obtained via the modified algorithm, the orienting loop passes through each cross-cap exactly once. Thus if G is drawn using k cross-caps, an orienting loop in G, corresponds to a set of $k+1$ edges in the dual graph. Furthermore, there are exactly 2 dual edges corresponding to the orienting loop in each face of the dual, see Fig. 18.

The effect of a contractible loop move is as follows:

Lemma 4.16 *Drawing a contractible loop in a face F of the cross-cap drawing corresponds to adding a vertex with degree one attached to the vertex F^*.*

Proof The proof follows directly from the definition of graph duality. Figure 18 depicts the edge c^*, dual to the contractible loop c. □

We can see that applying a contractible loop move does not change the situation of any of the dual edges corresponding to the orienting loop.

Let us assume there is a loop s that separates G into G_1 and G_2, where G_1 is non-orientable and G_2 is orientable. We glue the drawings H_1 and H_2 for G_1 and G_2 and we apply a dragging move to this case. The following lemma explains the effect of this move on the dual graph. In this lemma we use the notation introduced in the description of the dragging move. We denote the vertex associated to the root face F_o^i in H_i by F_o^{i*} for $i \in \{1,2\}$. We use the notation $(i_1, F_1, \dots, i_{2K}, F_o^1)$ for the sequence of edges and faces around the eliminated cross-cap in the dragging move, and finally we denote by o the auxiliary orienting loop drawn in H_2.

Lemma 4.17 *Let s be a loop that separates the embedding scheme G into the non-orientable subgraph G_1 and the orientable subgraph G_2. In this case, the gluing move, the dragging move and drawing back the separating loop corresponds to:*

- *splitting F_o^{1*} into two vertices F_o^{11*} and F_o^{12*} such that F_o^{11*} inherits only i_1^* and F_o^{12*} inherits the rest of the edges incident to F_o^{1*},*
- *removing the two dual edges corresponding to the loop o incident to the vertex F_o^{2*} in v_2^* (o_1^* and o_2^* in Fig. 19),*
- *connecting F_o^{11*} and F_o^{12*} to the adjacent vertices to o in the correct order by adding an edge for each one (this edges correspond to the segments of the separating loop that are attached to the vertex),*
- *connecting F_o^{2*} to F_k^* by adding an edge,*
- *replacing the dual edges corresponding to the segments of o in H_2 by $k+2$ edges.*

The operations performed in Lemma 4.17 are pictured in Fig. 19.

Proof Splitting F_o^{1*} and connecting it to two vertices formerly incident to F_o^{2*} corresponds to the gluing move between the drawings. We can see in Fig. 19 that these steps merge the faces v_1^* and v_2^* to the face v^*. The edges that we add to connect these vertices correspond to the sub-edges of the separating loop s that are incident to the vertex. On the other hand, connecting F_o^{2*} to F_k^* and replacing the dual edges in H_2 by a path corresponds to the dragging move; see Fig. 19. □

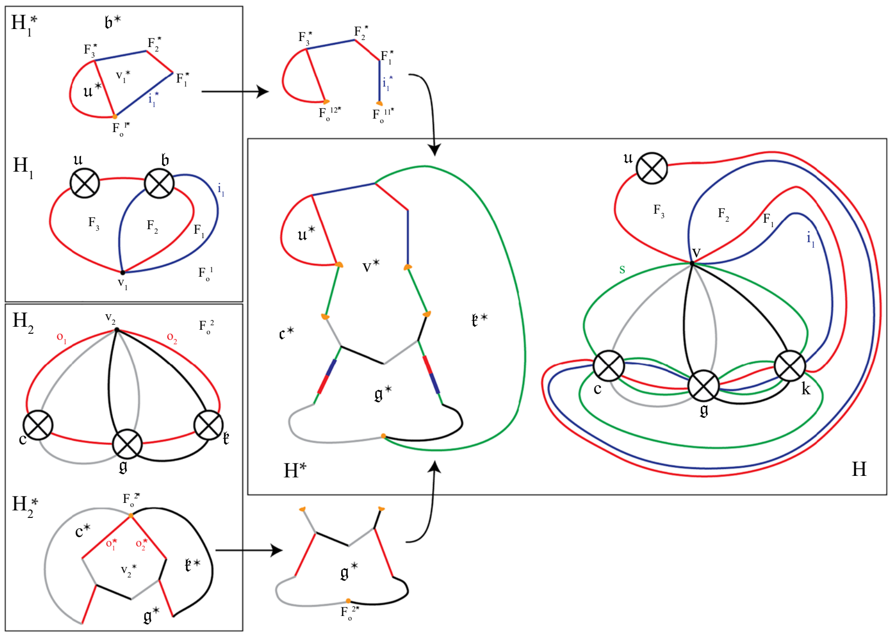

Fig. 19 The impact of the gluing move and dragging move on the dual graph

Remark 4.18 Let us denote by H the drawing of G that we obtain by applying the three moves in Lemma 4.17. The only modification done on the subgraph induced by the vertices that come from H_1^* in H^* is that we split the vertex F_o^{1*} to F_o^{11*} and F_o^{12*}. Both F_o^{11} and F_o^{12} are root faces in G.

The modifications in the subgraph induced by the vertices that come from H_2^* is that we replace some edges by paths and we disconnect the two root faces adjacent to F_o^{2*} by removing the incident edges. Also F_o^{2*} gets connected to a face in H_1^* and it is not necessarily a root face anymore.

The effect of a one-sided loop move is as follows:

Lemma 4.19 *Adding a one-sided non-orienting loop r with ends in root faces F_1 and F_2 in the drawing (F_1 and F_2 are possibly identical), corresponds to subdividing the face v^* into two faces and adding a path of length $k+2$ from F_1^* to F_2^* where k is the length of one of the paths from F_1^* to F_2^* in the face v^*.*

Proof Figure 20 depicts that the addition of the new cross-cap in the one-sided loop move corresponds to adding a duplicate of the set of edges and vertices between F_1^* and F_2^* in the face v^*. Choosing between the two different sequences from F_1^* to F_2^* in the face v^* corresponds to choosing different wedges of r to flip. □

Finally, we analyze the effect of the concatenation move. Let G be an embedding scheme with no separating loop in which every one-sided loop is orienting and it has at least one two-sided loop. Let o' be the one-sided non-orienting loop obtained by concatenating the orienting loop o and the two-sided loop t which has an end

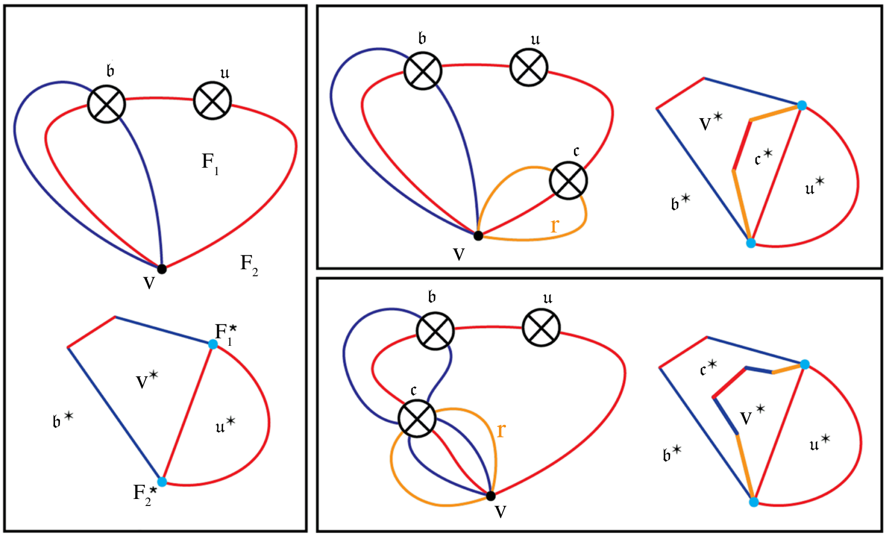

Fig. 20 The impact of the one-sided loop move on the dual graph. Left: the drawing for the embedding scheme before adding the one-sided loop r and its dual; right: the drawings and the dual graphs after drawing r and the impact of flipping different wedges of r is depicted

immediately next to an end of o in the rotation (step 3.2 of the algorithm). Denote by $G^{'}$ the embedding scheme in which o is replaced by $o^{'}$ and let $H^{'}$ be a drawing for $G^{'}$. After applying a one-sided loop move to $o^{'}$, t gets orienting (by Lemma 4.12) and it goes through $eg(G)-1$ cross-caps. The loop $o^{'}$ is the last loop that is drawn in the algorithm and therefore it passes through only one cross-cap. Let us denote this cross-cap by $\mathfrak{c}$. We denote by t_i, $1 \le i \le eg(G)$, the dual edges to sub-edges of t and by $o^{'}_1$ and $o^{'}_2$ the sub-edges of $o^{'}$, where t_1 corresponds to the sub-edge next to $o^{'}_1$.

Depending on the wedge of $o^{'}$ that we choose to reverse, we proceed with concatenating $o^{'}$ with t to get back the loop o as discussed in the description of the concatenation move. The following lemmas describe the effect of applying the concatenation move on the dual graph of the cross-cap drawing.

Lemmas 4.20 and 4.21 explain the effect of the concatenation move in the dual graph:

Lemma 4.20 *When we reverse the wedge of $o^{'}$ that does not encompass t, concatenating the loop $o^{'}$ along t corresponds to:*

- *subdividing the edge adjacent to t_1^* and $o_1^{'*}$ that is in the cyclic rotation of $\mathfrak{c}^*$*
- *contracting the edge $o_1^{'*}$*
- *subdividing the edges t_i^* for $i \ge 2$*

The added edges together with $o_2^{'*}$, correspond to the segments of o in H.

Proof By sliding $o^{'}$ along t, we remove $o^{'}_1$, see Fig. 21. Removing an edge corresponds to contracting its dual edge. Following t into the cross-caps adds parallel edges in the cross-cap drawing that corresponds to subdividing the dual edges t_i^*. □

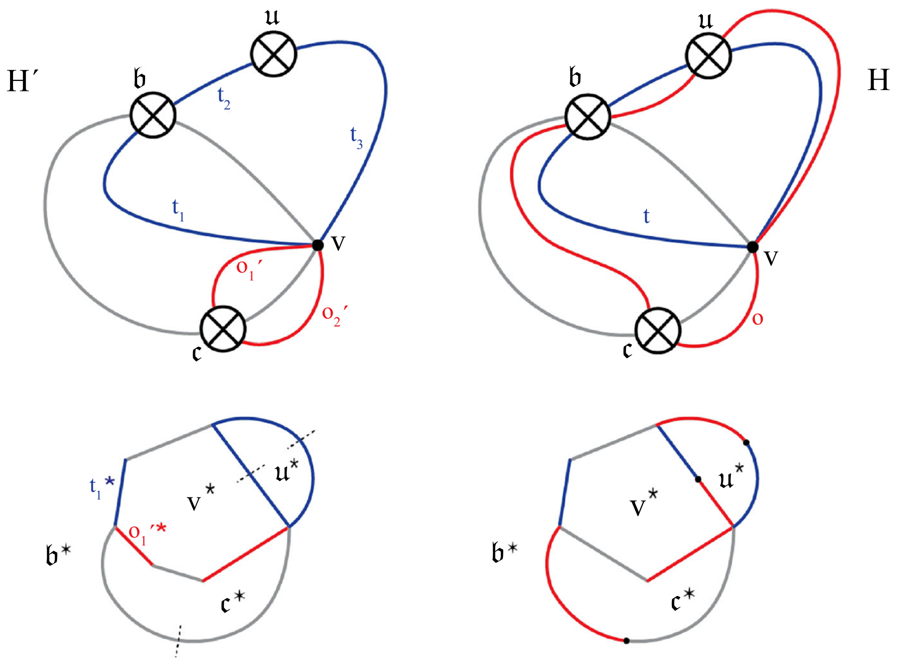

Fig. 21 The impact of the concatenation move in the dual graph. The bottom graphs are dual to the top cross-cap drawings

Lemma 4.21 *When we reverse the wedge of o' that encompasses the ends of t, concatenating the loop o' with t corresponds to:*

- *subdividing every t_i^* except for $i = 1, 2$*
- *subdividing the edge adjacent to t_2^* and $o_1'^*$ in the face v^**
- *contracting the edge $o_1'^*$ and $o_2'^*$*

The added edges together with $o_2'^*$, correspond to the segments of o in H.

Proof The proof is similar to the proof of Lemma 4.20. ◻

4.3.2 Short Paths from Each Cross-cap to the Vertex

Recall that a root face in a cross-cap drawing of a one-vertex graph is a face of the drawing (seen as a planar graph) adjacent to the vertex. The aim of this section is to show that there is a cross-cap drawing output by the modified algorithm, in which the cross-caps are not too far from the vertex (at distance $O(|E(G)|)$). To show this, we find paths in the dual graph of the cross-cap drawing from a vertex adjacent to the face dual to each cross-cap to the vertex corresponding to a root face.

We first show this claim for an embedding scheme with an orienting loop that has exactly one non-trivial component in its interleaving graph. Additionally, we claim that we can find a cross-cap drawing which allows us to force the paths to arrive in the same vertex in the dual graph that is chosen arbitrarily. Furthermore, we show that we can find these paths such that they do not use the dual edges corresponding to the orienting loop.

To prove this we use the modified algorithm, but with an additional specification: as we mentioned before, different choices in flipping a wedge when doing a one-sided loop move or a concatenation move yield different cross-cap drawings. The modified algorithm that we described gives us the freedom to choose the wedge whenever a one-sided loop move is applied. Here we use this freedom to build our desired paths.

Lemma 4.22 *For any one-vertex scheme G with an orienting loop o that has exactly one non-trivial component in its interleaving graph I_G, for any choice of root wedge ω, there is a choice of wedges in the modified algorithm which outputs a cross-cap drawing H with $eg(G)$ cross-caps such that for every cross-cap $\mathfrak{c}$, there is a dual path $p_{\mathfrak{c}}$ from a face adjacent to $\mathfrak{c}$ to the root face corresponding to ω with multiplicity two, which does not cross the orienting loop.*

Proof We prove the result by induction on $eg(G) + |E(G)|$, following the recursive steps of the modified algorithm. By Lemma 4.14, we know that the modified algorithm draws such an embedding scheme using only contractible loop moves, one-sided loop moves and concatenation moves. In this proof, we show that these moves can be applied in each step such that they do not increase the multiplicity of the paths that we obtain by the induction hypothesis. Crucially, when applying a one-sided loop move or a concatenation move, this relies on choosing a correct wedge to flip in each step.

We fix an arbitrary root wedge ω around the vertex v. When we remove a loop that affects ω, we update ω to be the wedge that encompasses the former fixed root wedge ω. Similarly, when we re-introduce the edges in the drawing, we subdivide ω and we choose the sub-wedge that is consistent with the first choice of ω.

Contractible Loop Move Denote by $G^{'}$ the embedding scheme we have after removing a contractible loop c. By the induction hypothesis, there exists a drawing $H^{'}$ and a system of paths $\{p_{\mathfrak{c}}\}$ from every cross-cap to the wedge ω in $H^{'*}$ with multiplicity two such that they do no use the dual edges of the orienting loop. When re-introducing c, if c does not sub-divide the wedge ω, then it does not affect the paths $p_{\mathfrak{c}}$. In the case that we need to update ω to be the empty wedge between the ends of c itself, by Lemma 4.16, we can see that in this case, the paths need to use the edge c^* in the dual once. Thus, the multiplicity of the paths remains two. The paths still do not use the dual edges of the orienting loop.

One-sided loop move on a one-sided non-orienting loop r. Denote by $G^{'}$ the graph we have after applying a one-sided loop move to r. By the induction hypothesis, there exists a drawing $H^{'}$ and a system of paths $\{p_{\mathfrak{c}}\}$ from every cross-cap to the wedge ω in $H^{'*}$ with multiplicity at most two such that they do not use the dual edges of the orienting loop. In this case, we flip the wedge of r that does not contain ω. By this choice, we can see that the situation of the vertex dual to the face ω does not change, since by Lemma 4.19, drawing a one-sided loop corresponds to adding a path to the dual graph that does not separate ω from v^*. Therefore, this move does not affect any $p_{\mathfrak{c}}$.

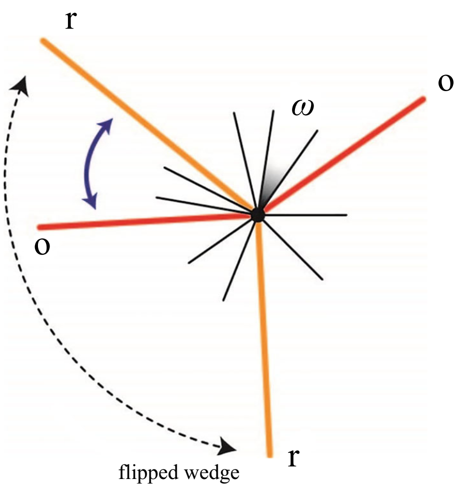

Fig. 22 The choice of the adjacent root face for the new cross-cap: the loop o is orienting and we applied the one-sided loop move on the loop r. The dotted arrow shows the wedge of r that we flipped and the blue arrow shows the adjacent root wedges to the last cross-cap that we can choose from

We know that the orienting loop interleaves with r. For the new cross-cap $\mathfrak{c}_1$, we define the path $p_{\mathfrak{c}_1}$ as follows. We choose a face adjacent to $\mathfrak{c}$ that is a root face, and such that it is both in the flipped wedge of r and in the same wedge of the orienting loop as ω; see Fig. 22. We introduce the path $p_{\mathfrak{c}_1}$ to be the sequence of dual edges and vertices around v between ω and this root face such that it does not use the dual edges corresponding to the orienting loop. Since each edge has exactly two ends around the vertex, the path $p_{\mathfrak{c}_1}$ uses each edge at most twice in the dual graph and its multiplicity is at most two. By construction, all these paths avoid using the dual edges of the orienting loop.

Concatenation move on an orienting loop o **and the two-sided loop** t. We denote by $o^{'}$ the concatenation of o and t and by $G^{'}$ the graph in which we replaced o by $o^{'}$. The modified algorithm applies the one-sided loop move to $o^{'}$, and here again we choose to flip the wedge of $o^{'}$ that does not encompass ω. We denote the new graph after removing $o^{'}$ by $G^{''}$. By the induction hypothesis there exists a drawing $H^{''}$ for $G^{''}$ and a suitable system of dual paths $\{p_{\mathfrak{c}}\}$ in $H^{''*}$ from a face adjacent to each cross-cap to the root wedge ω with multiplicity two. These paths do not use the dual edges corresponding to the loop t since after removing $o^{'}$, t is orienting for the new embedding scheme (this follows from Lemma 4.12). Similar to the case before in which the algorithm applies the one-sided loop move, we can see that after drawing $o^{'}$ and adding a cross-cap, we can re-introduce the paths $\{p_{\mathfrak{c}}\}$ with the same multiplicity so that they do not use the two dual edges corresponding to $o^{'}$. Now, the modified algorithm slides $o^{'}$ along t to get a drawing for the initial embedding scheme. By Lemmas 4.20 and 4.21 (depending on the situation of ω with respect to t and $o^{'}$), we know that sliding $o^{'}$ along t corresponds to sub-dividing the dual edges corresponding to t and since the paths $\{p_i\}$ do not use the dual edges of t, they do not use the dual edges of o either. For the last added cross-cap, we take a root face adjacent to it in the same wedge that ω is placed and introduce a path by going around the vertex. As before, we know that this path has multiplicity at most two and does not use the dual edges of the orienting loop. This finishes the proof. □

Using Lemma 4.22, we prove the claim for the more general case in which the embedding scheme can have more than one non-trivial component. Here, we do not need the paths to arrive in the same root face.

Lemma 4.23 *For any saturated one-vertex scheme G with an orienting loop o, there is a choice of wedges in the modified algorithm which outputs a cross-cap drawing H with $eg(G)$ cross-caps such that there is a path from every cross-cap to a root face (not necessarily fixed) with multiplicity at most two.*

Proof The proof is by induction on the number of separating loops. When there is no separating loop, the graph has only one non-trivial component and it is non-orientable. In this case, the result follows by Lemma 4.22.

Let s_l be the separating loop chosen by the algorithm during Step 2, separating two sub-scheme G_l and $G \setminus G_l$ on which it recurses. Since s_l separates a leaf from the component tree, one of these sub-schemes, say $G \setminus G_l$, is non-orientable and has an orienting loop. Therefore, by the induction hypothesis, there is a drawing $H^{'}$ for $G \setminus G_l$ with $eg(G \setminus G_l)$ cross-caps such that there is a path with multiplicity two from every cross-cap to a root wedge.

Now, G_l is made of exactly one non-trivial component due to our way of choosing s_l. Let ω be a root wedge of G_l different from F_o, the face where the ends of s_l used to exist. We apply Lemma 4.22 to obtain a cross-cap drawing H_l of $G_l + \{o\}$ and a system of dual paths $\{p_\mathfrak{c}\}$ with multiplicity at most two from a face adjacent to every cross-cap to ω, such that none of them use the dual edges corresponding to the orienting loop o. Now, the algorithm glues H_l to $H^{'}$ and proceeds with dragging the loops from $H^{'}$ to H_l. We denote the resulting drawing by H.

By Remark 4.18, we know that the paths connecting cross-caps to root wedges in $H^{'}$ can be re-introduced in H, since dual edges and vertices corresponding to the edges and faces in $H^{'}$ are not changed in H except the vertex that is split into two vertices. Since both of these vertices are root faces in the new embedding scheme, this does not interfere with the multiplicity of these paths and each of them arrives at one of these vertices (recall that we do not require all the paths to arrive at the same root wedge). By the choice of ω and the fact that none of the paths in $\{p_\mathfrak{c}\}$ use the dual edges corresponding to the orienting loop, none of the paths visit the vertex F_o^* (F_o and ω are in different wedges of the orienting loop o). Since the paths $\{p_\mathfrak{c}\}$ do not use o, we can choose the incident face to each cross-cap so that replacing the dual edges of o by a sequence of edges (as explained in Lemma 4.17), does not impact the multiplicity of the paths $\{p_\mathfrak{c}\}$ from each cross-cap to ω. This finishes the proof. □

It is immediate from the proofs of Lemmas 4.22 and 4.23 that the choice of wedges in the modified algorithm in these lemmas is computable in polynomial-time. We call the modified algorithm with the choice of wedges of these lemmas the *refined algorithm*. We finally have all the tools to prove our main result, which we recall for convenience.

Theorem 1.3 *There exists a polynomial time algorithm that given a graph cellularly embedded on the non-orientable surface N 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.*

In the case that the input graph contains an orienting cycle, our proof yields a better bound: each loop in the system intersects any edge of the graph in at most 10 points.

Proof Applying the algorithm to G, we obtain the saturated one-vertex scheme $\bar{G}$ that has an orienting loop after the preprocessing steps. By Lemma 4.4, to prove the theorem, it is sufficient to show that there exists a canonical system of loops for a drawing of $\bar{G}$ such that each loop in the system has multiplicity 10.

The one-vertex scheme $\bar{G}$ has an orienting loop and therefore by Lemma 4.23, the refined algorithm outputs a cross-cap drawing $\bar{H}$ with $eg(N)$ cross-caps such that there are paths $\{p_j\}$ with multiplicity two from a face incident to each cross-cap (denote this face by b_j for each j) to a root face (denote this face by a_j for each j) in this cross-cap drawing.

Fix a root face F in the drawing $\bar{H}$. For each path p_j, build a loop ν_j by going from F to a_j, by going around the vertex in shortest way possible. By doing so, so far the loop has multiplicity at most two. Follow p_j to b_j: this adds at most two to the multiplicity since p_j has multiplicity two. Go into the cross-cap and come back to b_j by going around it (this adds at most two to the multiplicity, since every edge passes through each cross-cap at most twice by Lemma 4.15) Finally, follow p_j back to a_j and go back to F from the same path (these two last steps add at most 4 to the multiplicity). Therefore, each ν_j has multiplicity 10. By Lemma 2.6, we know that the system of loops we obtain, is a non-orientable canonical system of loops. This finishes the proof. □

5 $O(g)$-Universal Shortest Path Metrics on Non-orientable Surfaces

In this section, we explain an analogue of the main result of [18] for non-orientable surfaces. The motivation is to reduce Negami's conjecture (Conjecture 1.1) to a problem in metric geometry. In the following, when we refer to a metric we mean a Riemannian metric (see, e.g., do Carmo and Francis [8] for background on Riemannian geometry). In a nutshell, a Riemannian metric induces a length for every (reasonable) path on the surface. A *geodesic* is a path that locally minimizes the length among all the possible paths, while a *shortest path* is a path that globally minimizes the length among all the possible paths.

If a graph is embedded on a surface equipped with a metric such that every edge is a shortest path, we call it a *shortest path embedding*. If there exists a metric on a surface S such that every graph topologically embeddable on S admits a shortest path embedding, we call this metric a *universal shortest path metric*. Since shortest paths in a Riemannian metric cross at most once, the existence of a universal shortest path metric for all surfaces would imply Negami's conjecture (Conjecture 1.1).

Similarly, a *k-universal shortest path metric* on a surface is a metric that allows every topologically embeddable graph to be embedded such that its edges are concatenations of k shortest paths. We refer to [18] and the references therein for more details and further discussion. One of the main theorems of that paper [18, Thm. 4] is that for any orientable surface of genus $g > 1$, there exists an $O(g)$-universal Riemannian metric of constant curvature -1. Furthermore, in the non-orientable case, Negami's bound on the joint crossing number of two graphs (Theorem 3.2) can be deduced from this theorem (or more precisely from the proof techniques [18, Cor. 20]).

In [18], proving the existence of $O(g)$-*universal shortest path metrics* for non-orientable surfaces was left as an open problem. In this section, we solve this problem by providing a construction that generalizes the one in [18] to the non-orientable case.

Theorem 1.4 *For $g \geq 3$, there exists a hyperbolic metric m on the non-orientable surface N of genus g such that any graph embeddable on N can be embedded so that every edge is a concatenation of $O(g)$ shortest paths.*

The construction in the orientable case is based on a decomposition of orientable genus g surfaces into hexagons. Given a graph embedded on a surface, it is first proved that there exists a *hexagonal decomposition* such that each edge of the graph is cut $O(g)$ times by the graph of the decomposition. Each hexagon is endowed with a hyperbolic metric and the following theorem is applied on each hexagon.

Theorem 5.1 ([18, Thm. 18], see also [7]) *Let G be a graph embedded as a triangulation in a hyperbolic hexagon H endowed with the metric of an equilateral right-angled hyperbolic hexagon. If there are no edges between two non-adjacent vertices on the boundary of H in G, then G can be embedded with geodesics, with the vertices on the boundary of H in the same positions as in the initial embedding.*

A convex hyperbolic hexagon can be isometrically embedded in the hyperbolic plane, and therefore exactly one geodesic connects any pair of points. Thus in this setting, geodesics are shortest paths and this theorem allows us to re-embed the part of G restricted to each face with shortest paths. The metric that we obtain by pasting these hyperbolic hexagons is a $O(g)$-shortest path metric.

It turns out that we can generalize the hexagonal decomposition for non-orientable surfaces, and apply the same strategy as in the orientable case.

Theorem 5.2 *Let N be a non-orientable cross-metric surface, with genus $g \geq 3$ and no boundary. We can decompose N into $2g-4$ hexagons when g is even and into $2g-6$ hexagons and 4 pentagons when g is odd such that the multiplicity of each curve in the decomposition is $O(1)$ except for one closed curve which has multiplicity $O(g)$. Furthermore, the graph of the decomposition is the graph shown in Fig. 23 and its dual graph is the one shown in Fig. 24.*

To prove the theorem, we use the following lemma.

Lemma 5.3 *Let M be an orientable cross-metric surface with orientable genus $g \geq 1$, and two (resp. one) boundary component(s). We can decompose M into $2g$ octagons (resp. 2 hexagons and $2g-2$ octagons) such that each edge of the decomposition has multiplicity $O(1)$.*

The proof of this lemma is completely analogous to that of the octagonal decomposition in Colin de Verdiere and Erickson [6, Thm. 3.1], and thus we only sketch it. The first step in their proof is to cut the surface along a shortest non-separating curve, yielding a surface with two boundaries. Their approach applies equally well if one starts with a surface with two boundaries, as it suffices to skip that first step. Their second step is to connect the two boundaries with an essential arc so as to have a single boundary. Thus, we can deal with the case of a surface with a single boundary by skipping these first two steps.

Then, the octagonal decomposition is obtained, after cutting along a maximum sequence of non-separating curves (*unzipping*), by going backwards (*zipping*) and adding all the curves one by one (see [6, Sect. 3 and Fig. 3.2(a)]). In the case of two boundaries, we can go back in an identical fashion to obtain an octagonal decomposition. In the case of a single boundary, this backwards step ends one step earlier (since one additional step was skipped at the start), hence we obtain two hexagons instead of octagons. In that case, the Euler characteristic is odd and thus no decomposition made exclusively of octagons can exist.

Proof of Theorem 5.2 Denote by G the primal graph of the surface N. The first curve in the decomposition is the orienting curve λ with multiplicity 2 that exists according to Lemma 2.7. Cutting along λ, we get an orientable surface with boundary which we denote by M and we denote by G' the graph we obtain from G by cutting along λ. Each edge of G is cut into at most three sub-edges. Using Lemma 2.1, we know that the orienting curve is two-sided (resp. one-sided), if the genus of the surface is even (resp. odd). Therefore, if g is odd, M has orientable genus $\frac{g-1}{2}$ and one boundary component and if g is even, M has orientable genus $\frac{g-2}{2}$ and two boundary components.

By Lemma 5.3, we can find a decomposition for M into octagons when g is even, and to both octagons and hexagons when g is odd, such that each curve in the decomposition crosses each edge of G' a constant number of times, see Fig. 23. Since each edge of G is cut into at most three edges in G', we know that the multiplicity of each curve is constant with respect to G.

We add an arc ρ that follows closely the sub-paths of the curves in the decomposition obtained by Lemma 5.3 as depicted in Fig. 23. The multiplicity of this curve is $O(g)$ and divides each octagon into two hexagons (and each hexagon to two pentagons). The segments of this curve that belong to each face have constant multiplicity in G' and therefore in G.

There are two arcs that are intersecting the copies of the orienting curve. At the end, we slide the ends of these curves very close to the boundary so that all of them build a closed curve after gluing the surface back along λ. Since the orienting loop has multiplicity 2, this adds at most 2 to the multiplicity of these arcs. This finishes the proof. □

Theorem 5.1 is also valid for hyperbolic pentagons with the same proof. As in the orientable case we apply Theorem 5.1 to each polygon of the decomposition provided by Theorem 5.2 to obtain the theorem. We rely on the following proposition, showing that this yields an $O(g)$-shortest path embedding.

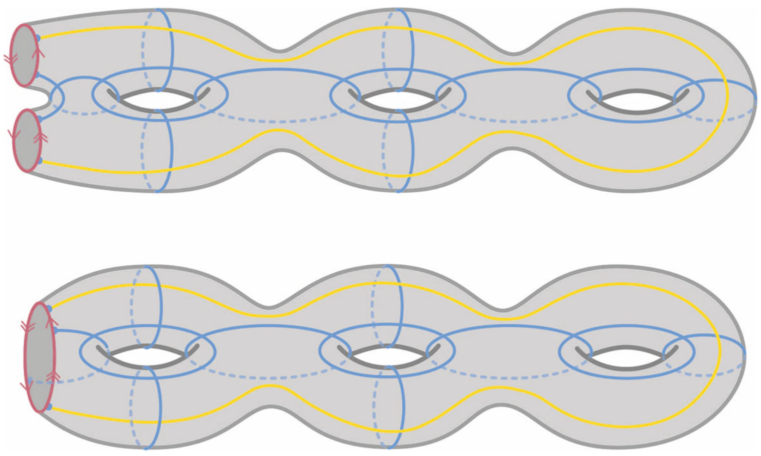

Fig. 23 Our decomposition for non-orientable surfaces with even genus (top picture) and odd genus (bottom picture). The pink curve is the orienting curve and the orientations shows how they are pasting together. These orientations are compatible according to Lemma 3.5. The yellow curve is the arc ρ added as the last step in the proof of Theorem 5.2

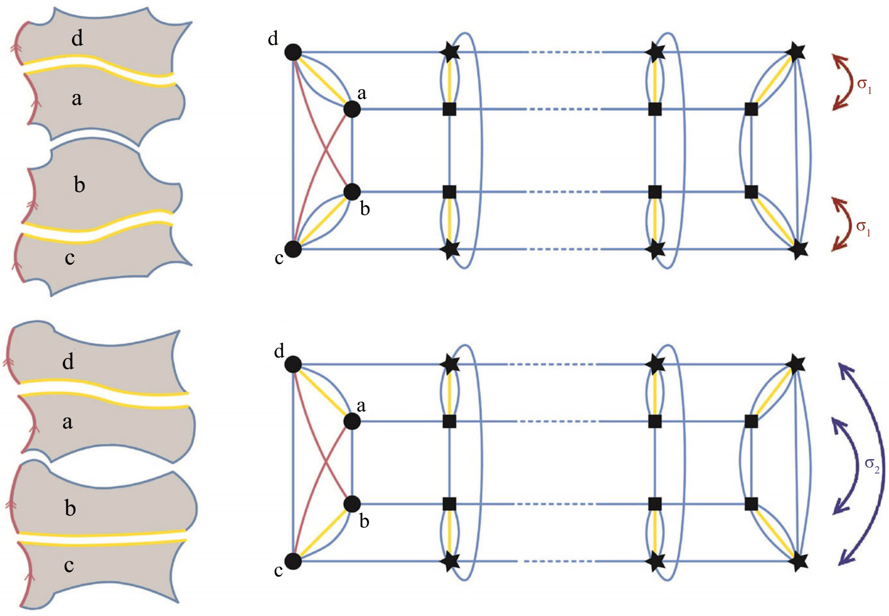

Fig. 24 Right: The dual graph of the decompositions in both cases of even genus (top picture) and odd genus (bottom picture). The circular vertices correspond to the four polygons that are adjacent to the orienting curve and the pink edges are the dual edges to the segments of the orienting curve. Left: the faces a, b, c and d are the faces adjacent to the orienting curve. Note that these four faces are pentagons in the case where the genus is odd

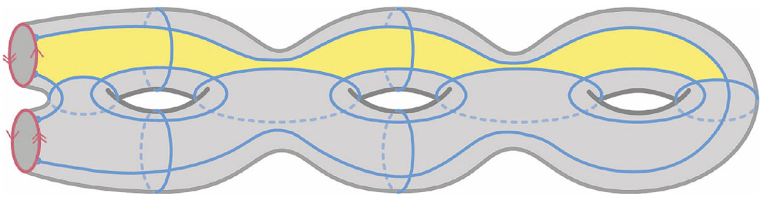

Fig. 25 The surface N cut along the orienting curve γ and the yellow area shows the quadrant Q_1

Proposition 5.4 *For every face F in the decomposition, every path between $x, y \in F$ that is a shortest path in F is also a shortest path in N.*

Coupled with Theorem 5.2, this proposition immediately implies Theorem 1.4. Its proof makes strong use of the symmetries of the decomposition, which we first introduce.

Figure 24 depicts the graphs dual to the decompositions output by Theorem 5.1. It depicts two involutions σ_1 and σ_2 which, since all the hexagons are isometric, and in the case of odd genus, the 4 pentagons are isometric, induce isometric involutions of the whole surface. If we take the square vertices to correspond to the top polygons in the decomposition and the star vertices to be the ones in the bottom, then the involution σ_1 maps the top polygons to the bottom polygons and vice versa and it is identity on the edges that are in common between top and bottom polygons. In particular σ_1 maps the faces a and d to each other and b and c to each other. Similarly σ_2 maps neighboring polygons at the top (and the bottom) on each other, in particular it maps a and b to each other and d and c to each other. We can cut N into four planar quadrants which we denote by Q_i, $1 \leq i \leq 4$. Each of these quadrants are linear concatenation of hexagons and pentagons. Take Q_1 to be the pictured quadrant in Fig. 25. We can see that each of the quadrants Q_2, Q_3 and Q_4 can be obtained by applying one of the σ_1, σ_2 and $\sigma_1\sigma_2$ to Q_1.

Proof of Proposition 17 Take two points x and y in a face F in Q_1 and let θ be a shortest path between them. The path θ may leave the face F, but we will show that in that case there is another shortest path between x and y that remains inside F. If θ leaves Q_1, we reflect the parts of the path that leave Q_1, using one of the maps σ_1, σ_2 or $\sigma_1\sigma_2$ back in Q_1. We need to check that the reflected parts together with the part of θ that is inside Q_1 define a path. The only troublesome case here is when our path leaves Q_1 by crossing the orienting curve γ. In this case, we can see that the path enters c. We use $\sigma_1\sigma_2$ to reflect the sub-path in c back to a. A closer look at $\sigma_1\sigma_2$ shows that it is identity on γ, therefore the sub-path in c gets reflected to a in a way that defines a new path in Q_1. We call the new path $\theta^{'}$. Since the maps are all isometric, $\theta^{'}$ has the same length as θ and therefore it is a shortest path between x and y which remains in Q_1.

We show that $\theta^{'}$ must be contained in F. Let us assume that $\theta^{'}$ leaves F. Since the dual graph of each Q_1 is a line, there is a face $F^{'}$ in Q_1 that $\theta^{'}$ enters and leaves once. We denote by α, the sub-path of $\theta^{'}$ inside $F^{'}$. The endpoints of α are both on the same edge of this face. α could be shortcut by following this edge instead of going inside the face $F^{'}$; see Fig. 26.

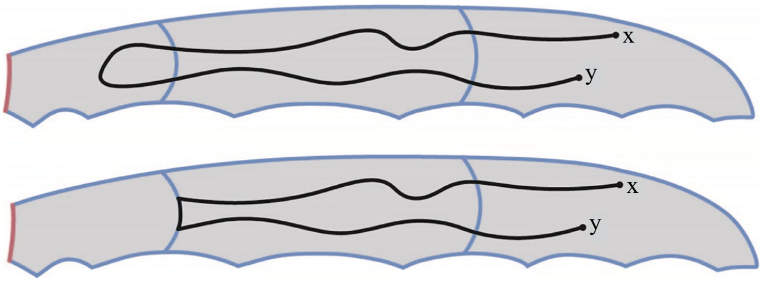

Fig. 26 The figure shows the shortest path θ' in Q_1 and how it can be shortcut

This implies that θ' cannot leave F and it is the shortest path between x and y that we were looking for. This finishes the proof. □

As for the orientable case, our proof techniques also provide an alternative proof of Negami's bound (which we corrected in Theorem 3) on joint crossing numbers on non-orientable surfaces. If two graphs are simultaneously embedded with our $O(g)$-universal shortest path embeddings, then each edge is cut into $O(g)$ shortest paths, but each hexagon/pentagon contains at most $O(1)$ of those. Since two shortest paths cross at most once, it follows that each pair of edges crosses at most $O(1)$ times in each hexagon/pentagon, and thus $O(g)$ times in total.

Acknowledgements We are grateful to Marcus Schaefer and Daniel Štefankovič for providing us the full version of [27], to Francis Lazarus for insightful discussions, and to the anonymous reviewers for very helpful comments.

References

1. Archdeacon, D., Bonnington, C.P.: Two maps on one surface. J. Graph Theory **36**(4), 198–216 (2001)
2. Bergeron, A.: A very elementary presentation of the Hannenhalli–Pevzner theory. In Annual Symposium on Combinatorial Pattern Matching, pp. 106–117. Springer, Berlin (2001)
3. Bura, A.C., Chen, R.X.F., Reidys, C.M.: On a lower bound for sorting signed permutations by reversals. arXiv Preprint (2016). arXiv:1602.00778
4. Colin de Verdière, É.: Topological algorithms for graphs on surfaces. Habilitation thesis (2012). http://www.di.ens.fr/~colin/
5. Colin de Verdière, É.: Computational topology of graphs on surfaces. In: Goodman, J.E., O'Rourke, J., Toth, C. (eds.) Handbook of Discrete and Computational Geometry, 3rd edn., pp. 605–636. CRC Press, Boca Raton (2018)
6. Colin de Verdière, É., Erickson, J.: Tightening nonsimple paths and cycles on surfaces. SIAM J. Comput. **39**(8), 3784–3813 (2010)
7. Colin de Verdière, Y.: Comment rendre géodésique une triangulation d'une surface: L'Enseignement Mathématique **37**, 201–212 (1991)
8. Do Carmo, M.P., Francis, J.F.: Riemannian Geometry, vol. 6. Springer, Berlin (1992)
9. Erickson, J., Har-Peled, S.: Optimally cutting a surface into a disk. Discrete Comput. Geom. **31**(1), 37–59 (2004)
10. Erickson, J., Whittlesey, K.: Greedy optimal homotopy and homology generators. In: SODA, vol. 5, pp. 1038–1046 (2005)
11. Geelen, J., Huynh, T., Richter, R.B.: Explicit bounds for graph minors. J. Combin. Theory Ser. B **132**, 80–106 (2018)
12. Ghrist, R.: Barcodes: the persistent topology of data. Bull. Am. Math. Soc. **45**(1), 61–75 (2008)

13. Hannenhalli, S., Pevzner, P.A.: Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals. J. ACM (JACM) **46**(1), 1–27 (1999)
14. Hatcher, A.: Algebraic Topology. Cambridge University Press, Cambridge (2002)
15. Hayes, B.: Computing science: Sorting Out The Genome. Am. Sci. **95**(5), 386–391 (2007)
16. Hliněnỳ, P., Salazar, G.: On hardness of the joint crossing number. In: International Symposium on Algorithms and Computation, pp. 603–613. Springer, Berlin (2015)
17. Huang, F.W.D., Reidys, C.M.: A topological framework for signed permutations. Discret. Math. **340**(9), 2161–2182 (2017)
18. Hubard, A., Kaluža, V., De Mesmay, A., Tancer, M.: Shortest path embeddings of graphs on surfaces. Discrete Comput. Geom. **58**(4), 921–945 (2017)
19. Lazarus, F.: 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). http://www.gipsa-lab.grenoble-inp.fr/~francis.lazarus/Documents/hdr-Lazarus.pdf
20. Lazarus, F., Pocchiola, M., Vegter, G., Verroust, A.: Computing a canonical polygonal schema of an orientable triangulated surface. In Proceedings of the 17th Annual Symposium on Computational Geometry, pp. 80–89 (2001)
21. Matoušek, J., Sedgwick, E., Tancer, M., Wagner, U.: Untangling two systems of noncrossing curves. In: International Symposium on Graph Drawing, pp. 472–483. Springer, Berlin (2013)
22. Mohar, B.: The genus crossing number. ARS Math. Contemp. **2**(2), 157–162 (2009)
23. Mohar, B., Thomassen, C.: Graphs on Surfaces, vol. 10. JHU Press, Baltimore (2001)
24. Negami, S.: Crossing numbers of graph embedding pairs on closed surfaces. J. Graph Theory **36**(1), 8–23 (2001)
25. Richter, R.B., Salazar, G.: Two maps with large representativity on one surface. J. Graph Theory **50**(3), 234–245 (2005)
26. Schaefer, M., Štefankovič, D.: Block additivity of $\mathbb{Z}_2$-embeddings. In: International Symposium on Graph Drawing, pp. 185–195. Springer, Berlin (2013)
27. Schaefer, M., Štefankovič, D.: The degenerate crossing number and higher-genus embeddings. J. Graph Algorithms Appl. **26**(1), 35–58 (2022). https://doi.org/10.7155/jgaa.00580
28. Sethna, J.P.: Order parameters, broken symmetry, and topology. In: 1991 Lectures in Complex Systems. Addison-Wesley, Reading (1992)
29. Sheffer, A., Hormann, K., Levy, B., Desbrun, M., Zhou, K., Praun, E., Hoppe, H.: Mesh parameterization: theory and practice. In: ACM SIGGRAPPH, Course Notes, 10, 1281500.1281510 (2007)
30. Stillwell, J.: Classical Topology and Combinatorial Group Theory, vol. 72. Springer, New York (1993)