Recognizing Weak Embeddings of Graphs

Hugo A. Akitaya†  Radoslav Fulek‡  Csaba D. Tóth†§

September 28, 2017

Abstract

We present an efficient algorithm for a problem in the interface between clustering and graph embeddings. An embedding \( \varphi : G \to M \) of a graph \( G \) into a 2-manifold \( M \) maps the vertices in \( V(G) \) to distinct points and the edges in \( E(G) \) to interior-disjoint Jordan arcs between the corresponding vertices. In applications in clustering, cartography, and visualization, nearby vertices and edges are often bundled to a common node or arc, due to data compression or low resolution. This raises the computational problem of deciding whether a given map \( \varphi : G \to M \) comes from an embedding. A map \( \varphi : G \to M \) is a weak embedding if it can be perturbed into an embedding \( \psi : G \to M \) with \( \| \varphi - \psi \| < \varepsilon \) for every \( \varepsilon > 0 \).

A polynomial-time algorithm for recognizing weak embeddings was recently found by Fulek and Kynčl \[14\], which reduces to solving a system of linear equations over \( \mathbb{Z}_2 \). It runs in \( O(n^{2\omega}) \leq O(n^{4.75}) \) time, where \( \omega \approx 2.373 \) is the matrix multiplication exponent and \( n \) is the number of vertices and edges of \( G \). We improve the running time to \( O(n \log n) \). Our algorithm is also conceptually simpler than \[14\]: We perform a sequence of local operations that gradually “untangles” the image \( \varphi(G) \) into an embedding \( \psi(G) \), or reports that \( \varphi \) is not a weak embedding.

It generalizes a recent technique developed for the case that \( G \) is a cycle and the embedding is a simple polygon \[1\], and combines local constraints on the orientation of subgraphs directly, thereby eliminating the need for solving large systems of linear equations.

1 Introduction

Given a graph \( G \) and a 2-dimensional manifold \( M \), one can decide in linear time whether \( G \) can be embedded into \( M \) \[18\], although finding the smallest genus of a surface into which \( G \) embeds is NP-hard \[23\]. An embedding \( \psi : G \to M \) is a continuous piecewise linear injective map where the graph \( G \) is considered as a 1-dimensional simplicial complex. Equivalently, an embedding maps the vertices into distinct points and the edges into interior-disjoint Jordan arcs between the corresponding vertices. We would like to decide whether a given map \( \varphi : G \to M \) can be “perturbed” into an embedding \( \psi : G \to M \). Let \( M \) be a 2-dimensional manifold equipped with a metric. A continuous piecewise linear map \( \varphi : G \to M \) is a weak embedding if, for every \( \varepsilon > 0 \), there is an embedding \( \psi_\varepsilon : G \to M \) with \( \| \varphi - \psi_\varepsilon \| < \varepsilon \), where \( \|\cdot\| \) is the uniform norm (i.e., \( \sup \) norm).

*Research supported in part by the NSF awards CCF-1422311 and CCF-1423615, and the Science Without Borders program. The second author gratefully acknowledges support from Austrian Science Fund (FWF): M2281-N35.
†Department of Computer Science, Tufts University, Medford, MA, USA.
‡Institute of Science and Technology, Klosterneuburg, Austria.
§Department of Mathematics, California State University Northridge, Los Angeles, CA, USA.
In some cases, it is easy to tell whether \( \varphi : G \to M \) is a weak embedding: Every embedding is a weak embedding; and if \( \varphi \) maps two edges into Jordan arcs that cross transversely, then \( \varphi \) is not a weak embedding. The problem becomes challenging when \( \varphi \) maps several vertices (edges) into the same point (Jordan arc), although no two edges cross transversely. This scenario arises in applications in clustering, cartography, and visualization, where nearby vertices and edges are often bundled to a common node or arc, due to data compression, graph semantics, or low resolution. A cluster in this context is a subgraph of \( G \) mapped by \( \varphi \) to the single point in \( M \).

The recognition of weak embeddings turns out to be a purely combinatorial problem independent of the global topology of the manifold \( M \) and the neighborhood of \( \varphi(G) \) (as noted in [5]). The key observation here is that we are looking for an embedding in a small neighborhood of the image \( \varphi(G) \), which can be considered as the embedding of some graph \( H \). As such, we can replace \( M \) with a neighborhood of an embedded graph in the formulation of the problem.

![Figure 1](image.png)

**Figure 1:** (a) An embedding of \( H = K_3 \) in the torus. (b) Strip system \( \mathcal{H} \) of the embedding of \( H \). (c) A weak embedding where \( G \) is disconnected and \( H = C_4 \). (d) A negative instance where \( G = C_8 \) and \( H = C_4 \).

**Problem Statement and Results.** An embedded graph \( H \) in an orientable 2-manifold \( M \) is an abstract graph together with a rotation system that specifies, for each vertex of \( H \), the cyclic order of incident edges\(^1\). The strip system \( \mathcal{H} \) of \( H \) (a.k.a. thickening of \( H \)) is a 2-manifold with boundary constructed as follows (Fig. 1(a)–(b)): For every \( u \in V(H) \), create a topological disk \( D_u \), and for every edge \( uv \in E(H) \), create a rectangle \( R_{uv} \). For every \( D_u \) and \( R_{uv} \), fix an arbitrary orientation of \( \partial D_u \) and \( \partial R_{uv} \), respectively. Partition the boundary of \( \partial D_u \) into \( \text{deg}(u) \) arcs, and label them by \( A_{u,v} \), for all \( uv \in E(H) \), in the cyclic order around \( \partial D_u \) determined by the rotation of \( u \) in the embedding of \( H \). Finally, the manifold \( \mathcal{H} \) is obtained by identifying two opposite sides of every rectangle \( R_{uv} \) with \( A_{u,v} \) and \( A_{v,u} \) via an orientation preserving homeomorphism (i.e., consistently with the chosen orientations of \( \partial R_{uv} \), \( \partial D_u \) and \( \partial D_v \)). See Fig. 1(a)–(b).

We formulate a problem instance as a function \( \varphi : G \to H \) (for short, \( \varphi \)), where \( G \) is an abstract graph, \( H \) is an embedded graph, and \( \varphi : G \to H \) is a simplicial map that maps the vertices of \( G \) to vertices of \( H \), and the edges of \( G \) to edges or vertices of \( H \), such that incidences are preserved. The simplicial map \( \varphi : G \to H \) is a weak embedding if there is an embedding \( \psi_\varphi : G \to \mathcal{H} \) that maps each vertex \( v \in V(G) \) to a point in \( D_{\varphi(v)} \), and each edge \( uv \in E(G) \) to a Jordan arc in \( D_{\varphi(u)} \cup R_{\varphi(u)\varphi(v)} \cup D_{\varphi(v)} \) that has a connected intersection with each of \( D_{\varphi(u)} \), \( R_{\varphi(u)\varphi(v)} \), and \( D_{\varphi(v)} \), and \( R_{\varphi(u)\varphi(v)} = \emptyset \) if \( u = v \). We say that the embedding \( \psi_\varphi \) approximates \( \varphi \). Our main results is the following.

\(^1\)Our methods extend to nonorientable surfaces with minor changes in the combinatorial representations, using a signature \( \lambda : E(H) \to \{ -1, 1 \} \) to indicate whether the edge \( u \) (and \( R_{uv} \)) is orientation-preserving or -reversing [5]; see Section 5.
**Theorem 1.** (i) Given an abstract graph $G$ with $m$ edges, an embedded graph $H$, and a simplicial map $\varphi : G \to H$, we can decide in $O(m \log m)$ time whether $\varphi$ is a weak embedding.

(ii) If $\varphi : G \to H$ is a weak embedding, then for every $\varepsilon > 0$ we can also find an embedding $\psi_\varepsilon : G \to M$ with $\| \varphi - \psi_\varepsilon \| < \varepsilon$ in $O(m \log m)$ time.

Throughout the paper we assume that $G$ has $n$ vertices and $m$ edges. In the plane (i.e., $M = \mathbb{R}^2$), only planar graphs admit weak embeddings hence $m = O(n)$, but our techniques work for 2-manifolds of arbitrary genus, and $G$ may be a dense graph. Our result improves the running time of the previous algorithm [11] from $O(m^{2\omega}) \leq O(m^{4.75})$ to $O(m \log m)$. It also improves the running times of several recent polynomial-time algorithms in special cases, e.g., when the embedding of $G$ is restricted to a given isotopy class [13], and $H$ is a path [2].

**Nonsimplicial Maps.** If $\varphi : G \to H$ is a continuous map (not necessarily simplicial) that is injective on the edges (each edge is a Jordan arc), we may assume that $\varphi(V(G)) \subseteq V(H)$ by subdividing the pipes of $H$ with at most $n = |V(G)|$ additional clusters if necessary. Then $\varphi$ maps every edge $e \in G$ to a path of length $O(n)$ in $H$. By subdividing the edges $e \in E(G)$ at all clusters along $\varphi(e)$, we reduce the recognition problem to the regime of simplicial maps (Theorem 1). The total number of vertices may increase to $O(mn)$ and the running time to $O(mn \log(mn))$.

**Corollary 1.** Given an abstract graph $G$ with $m$ edges, an embedded graph $H$ with $n$ vertices, and a piecewise linear continuous map $\varphi : G \to H$ that is injective on the interior of every edge in $E(G)$, we can decide in $O(mn \log(mn))$ time whether $\varphi$ is a weak embedding.

For example, this applies to straight-line drawings in $\mathbb{R}^2$ if the edges may pass through vertices.

**Corollary 2.** Given an abstract graph $G$ with $n$ vertices and a map $\varphi : G \to \mathbb{R}^2$ where every edge is mapped to a straight-line segment, we can decide in $O(n^2 \log n)$ time whether $\varphi$ is a weak embedding.

**Related Previous Work.** The study of weak embeddings lies at the interface of several independent lines of research in mathematics and computing science. In topology, the study of weak embeddings and its higher dimensional analogs were initiated by Sieklucki [20] in the 1960s. One of his main results [20, Theorem 2.1] implies the following. Given a graph $G$ and an embedded path $H$, every simplicial map $\varphi : G \to H$ is a weak embedding if and only if every connected component of $G$ is a subcubic graph with at most one vertex of degree three. It is easy to see that an analogous statement is false when $G$ has maximum degree 2 (i.e., $G$ is a union of vertex disjoint paths and cycles). In this case, a series of recent papers on weakly simple polygons [1, 6, 7] show that weak embeddings can be recognized in $O(n \log n)$ time (the same time suffices when $\varphi : C_n \to H$ is a simplicial map [6] or a continuous map [1]).

Finding efficient algorithms for the recognition of weak embeddings $\varphi : G \to H$, where $G$ is an arbitrary graph, was posed as an open problem in [1, 6, 7]. The first polynomial-time solution for the general version follows from a recent variant [14] of the Hanani-Tutte theorem [15, 24], which was conjectured by M. Skopenkov [22] in 2003 and in a slightly weaker form already by Repovš and A. Skopenkov [19] in 1998. However, this algorithm reduces the problem to a system of $O(m)$ linear equations over $\mathbb{Z}_2$, where $m = |E(G)|$. The running time is dominated by solving this system in $O(m^{2\omega}) \leq O(m^{4.75})$ time, where $\omega$ is the matrix multiplication exponent.

Weak embeddings of graphs also generalize various graph visualization models such as the recently introduced *strip planarity* [2] and *level planarity* [17]; and can be seen as a special case [4]...
of the notoriously difficult cluster-planarity (for short, c-planarity) \[11\] \[12\], whose tractability remains elusive despite many attempts by leading researchers.

**Outline.** Our results rely on ideas from \[2\] \[6\] \[7\] and \[14\]. To distinguish the graphs \(G\) and \(H\), we use the convention that \(G\) has vertices and edges, and \(H\) has clusters and pipes. A cluster \(u \in V(H)\) corresponds to a subgraph \(\varphi^{-1}[u]\) of \(G\).

The main tool in our algorithm is a local operation, called “cluster expansion,” which generalizes similar operations introduced previously for the case that \(G\) is a cycle. Given an instance \(\varphi : G \to H\) and a cluster \(u \in V(H)\), it modifies \(u\) and its neighborhood (by replacing \(u\) with several new clusters and pipes) and it is “reversible” in the sense that the resulting new instance \(\varphi' : G' \to H'\) is a weak embedding if and only if \(\varphi : G \to H\) is a weak embedding. Our operation increases the number of clusters and pipes, but it decreases the number of “ambiguous” edges (i.e., multiple edges in the same pipe). The proof of termination and the running time analysis use potential functions.

In a preprocessing phase, we perform a cluster expansion operation at each cluster \(u \in V(H)\). The main loop of the algorithm applies another operation, “pipe expansion,” for two adjacent clusters \(u, v \in V(H)\) under certain conditions. It merges the clusters \(u\) and \(v\), and the pipe \(uv \in E(H)\) between them, and then invokes cluster expansion. If any of these operations finds a local configuration incompatible with an embedding, then the algorithm halts and reports that \(\varphi\) is not a weak embedding (this always corresponds to nonplanar subconfigurations since the neighborhood of a single cluster or pipe is homeomorphic to a disk). We show that after \(O(m)\) successive operations, we obtain an irreducible instance for which our problem is easily solvable in \(O(m)\) time. Ideally, we end up with \(G = H\) (one vertex per cluster and one edge per pipe), and \(\varphi = \text{id}\) is clearly an embedding. Alternatively, \(G\) and \(H\) may each be a cycle (possibly \(G\) winds around \(H\) multiple times), and we can decide whether \(\varphi\) is a weak embedding in \(O(m)\) time by a simple traversal of \(G\). If \(G\) is disconnected, then each component falls into one of the above two cases (Fig. 1(c)–(d)), i.e., the case when \(\varphi = \text{id}\) or the case when \(\varphi \neq \text{id}\).

The main challenge was to generalize previous reversible local operations (that worked well for cycles \[22\]) to arbitrary graphs. Our expansion operation for a cluster \(u \in V(H)\) simplifies each component of the subgraph \(\varphi^{-1}[u]\) of \(G\) independently. Each component is planar (otherwise it cannot be perturbed into an embedding in a disk \(D_u\)). However, a planar (sub)graph with \(k\) vertices may have \(\varphi^{O(k)}\) combinatorially different embeddings: some of these may or may not be compatible with adjacent clusters. The embedding of a (simplified) component \(C\) of \(\varphi^{-1}[u]\) depends, among other things, on the edges that connect \(C\) to adjacent clusters. The **pipe-degree** of \(C\) is the number of pipes that contain its incident edges. If the pipe-degree of \(C\) is 3 or higher, then the rotation system of \(H\) constrains the embedding of \(C\). If the pipe-degree is 2, however, then the embedding of \(C\) can only be determined up to a reflection, unless \(C\) is connected by two independent edges to a component in \(\varphi^{-1}[v]\) whose orientation is already fixed; see Fig. 2.

We need to maintain the feasible embeddings of the components in all clusters efficiently. In \[14\], this problem was resolved by introducing 0-1 variables for the components, and aggregating the constraints into a system of linear equations over \(\mathbb{Z}_2\), which was eventually resolved in \(O(m^{2.75})\) time. We improve the running time to \(O(m \log m)\) by maintaining the feasible embeddings simultaneously with our local operations.

Another challenge comes from the simplest components in a cluster \(\varphi^{-1}[u]\). Long chains of degree-2 vertices, with one vertex per cluster, are resilient to our local operations. Their length may decrease by only one (and cycles are irreducible). We need additional data structures to handle these “slowly-evolving” components efficiently. We use a dynamic heavy-path decomposition data
structure and a suitable potential function to bound the time spent on such components.

Organization. In Section 2, we review standard notation for SPQR-trees, developed in [9] for the efficient representation of combinatorial embeddings of a graph. We introduce a “normal form” and a combinatorial representation for a weak embedding instance \( \varphi: G \rightarrow H \) that we use in our algorithm. We present the cluster expansion operation and prove that it maintains weak embeddings in Section 3. We use this operation repeatedly in Section 4 to decide whether a simplicial map \( \varphi: G \rightarrow H \) is a weak embedding. Section 5 discusses how to reverse the sequence of operations to perturb a weak embedding into an embedding. The adaptation of our results to nonorientable surfaces \( M \) is discussed in Section 6. We conclude with open problems in Section 7.

2 Preliminaries

Definitions. Two instances \( \varphi: G \rightarrow H \) and \( \varphi': G' \rightarrow H' \) are called equivalent if \( \varphi \) is a weak embedding if and only if \( \varphi' \) is a weak embedding. We call an edge \( e \in E(G) \) a pipe-edge if \( \varphi(e) \in E(H) \), or a cluster-edge if \( \varphi(e) \in V(H) \). For every cluster \( u \in V(H) \), let \( G_u \) be the subgraph of \( G \) induced by \( \varphi^{-1}[u] \). For every \( uv \in E(H) \), \( \varphi^{-1}[uv] \) stands for the set of pipe-edges mapped to \( uv \) by \( \varphi \).

The pipe-degree of a connected component \( C \) of \( G_u \), denoted pipe-deg(\( C \)), is the number of pipes that contain an edge of \( G \) incident to \( C \). A vertex \( v \) of \( G_u \) is called a terminal if it is incident to a pipe-edge. A \( k \)-vertex wheel graph \( W_k \) is a join of a center vertex \( c \) and a cycle of \( k - 1 \) external vertices. Refer to [10] for standard graph theoretic terminology (e.g., cut vertex, 2-cuts, biconnectivity).

Normal form. An instance \( \varphi: G \rightarrow H \) is in normal form if every cluster \( u \in V(H) \) satisfies:

(P1) Every terminal in \( G_u \) is incident to exactly one cluster-edge and one pipe-edge.

(P2) There are no degree-2 vertices in \( G_u \).

We now describe subroutine normalize(\( u \)) that, for a given instance \( \varphi: G \rightarrow H \) and a cluster \( u \in V(H) \), returns an equivalent instance \( \varphi': G' \rightarrow H \) such that \( u \) satisfies [P1][P2] refer to Fig. 3(a)–(b).

normalize(\( u \)). Input: an instance \( \varphi: G \rightarrow H \) and a cluster \( u \in V(H) \).

Subdivide every pipe-edge \( pq \) where \( \varphi(p) = u \) into a path \( (p, p', q) \) such that \( \varphi'(p) = \varphi'(p') = \varphi'(q) = u \). Note that \( p' \) is a terminal in \( G \) and a leaf in \( G_u \) (i.e., \( \deg_{G_u}(p') = 1 \)). Successively
Lemma 1. Given an instance $\varphi : G \to H$ and a cluster $u \in V(H)$, the instance $\varphi' = \text{normalize}(u)$ and $\varphi$ are equivalent and $u$ satisfies (P1)–(P2) in $\varphi'$. The subroutine runs in $O(\sum_{p \in V(G_u)} \text{deg}_G(p))$ time. By successively applying normalize to all clusters in $V(H)$, we obtain an equivalent instance in normal form in $O(|E(G)|)$ time.

Proof. The instances $\varphi$ and $\varphi'$ are clearly equivalent since we can always replace the embedding of an edge by a path and vice-versa. A loop can always be added to a vertex in any embedding. There are $O(\sum_{p \in V(G_u)} \text{deg}_G(p))$ pipe-edges incident to a vertex in $u$. Hence the subroutine performs $O(\sum_{p \in V(G_u)} \text{deg}_G(p))$ subdivisions. There are at most $|E(G_u)| = O(\sum_{p \in V(G_u)} \text{deg}_G(p))$ degree-2 vertices in $G_u$. By construction, the resulting graph $G'_u$ satisfies (P1)–(P2). All changes are local and applying normalize to $u$ does not change properties (P1)–(P2) in other clusters. Then, we can obtain the normal form of $\varphi$ in $O(|E(G)|)$ time. \qed

For every component $C$ of $G_u$ of a cluster $u \in V(H)$ satisfying (P1)–(P2), we define the multigraph $\overrightarrow{C}$ in two steps as follows; refer to Fig. 3(b).

1. For every pipe $uv \in E(H)$ incident to $u$, create a new vertex $v'$, called pipe-vertex, and identify all terminal vertices of $C$ incident to some edge in $\varphi^{-1}[uv]$ with $v'$ (this may create multiple edges incident to $v'$).

2. If pipe-deg($C$) = 2, connect the two pipe-vertices with an edge $e$ and let $E_C = \{e\}$. If pipe-deg($C$) $\geq$ 3, connect all pipe-vertices by a cycle in the order determined by the rotation of $u$ and let $E_C$ be the set of edges of this cycle.

Figure 3: Changes in a cluster caused by normalize and simplify. (a) Input, (b) after normalize, (c) after first part of step 1, (d) after step 1, and (e) after step 2 of subroutine simplify. Dashed lines, green dots, green lines, and blue lines represent pipe-edges, pipe-vertices, edges in $E_C$, and virtual edges, respectively.
Let \( \overline{C} = (V(\overline{C}), E(\overline{C})) \), where \( V(\overline{C}) \) consists of nonterminal vertices in \( V(C) \) and pipe-deg(C) pipe-vertices, and \( E(\overline{C}) \) consists of the (multi) edges created in step 1 (each of which corresponds to an edge in \( E(C) \)) and edges in \( \overline{E}_C \). It is clear that every embedding of \( \overline{C} \) can be converted into an embedding of \( C \) such that the rotation of the pipe vertices in \( \overline{C} \) determines the cyclic order of terminals along the facial walk of the outer face of \( C \).

**SPQR-trees** were introduced by Di Battista and Tamassia [9] for an efficient representation of all combinatorial plane embeddings of a graph. Let \( G \) be a biconnected planar graph. The SPQR-tree \( T_G \) of \( G \) represents a recursive decomposition of \( G \) defined by its (vertex) 2-cuts. A deletion of a 2-cut \( \{u,v\} \) disconnects \( G \) into two or more components \( C_1, \ldots, C_i, i \geq 2 \). A split component of \( \{u,v\} \) is either an edge \( uv \) or the subgraphs of \( G \) induced by \( V(C_j) \cup \{u,v\} \) for \( j = 1, \ldots, i \). The tree \( T_G \) captures the recursive decomposition of \( G \) into split components defined by 2-cuts of \( G \). A node \( \mu \) of \( T_G \) is associated with a multigraph called skeleton(\( \mu \)) on a subset of \( V(G) \), and has a type in \( \{S,P,R\} \). If the type of \( \mu \) is \( S \) then skeleton(\( \mu \)) is a cycle of 3 or more vertices. If the type of \( \mu \) is \( P \) then skeleton(\( \mu \)) consists of 3 or more parallel edges between a pair of vertices. If the type of \( \mu \) is \( R \) then skeleton(\( \mu \)) is a 3-connected graph on 4 or more vertices. An edge in skeleton(\( \mu \)) is real if it is an edge in \( G \), or virtual otherwise. A virtual edge connects the two vertices of a 2-cut, \( u \) and \( v \), and represents a subgraph of \( G \) obtained in the recursive decomposition, containing a \( uv \)-path in \( G \) that does not contain any edge in skeleton(\( \mu \)). Two nodes \( \mu_1 \) and \( \mu_2 \) of \( T_G \) are adjacent if skeleton(\( \mu_1 \)) and skeleton(\( \mu_2 \)) share exactly two vertices, \( u \) and \( v \), that form a 2-cut in \( G \). Each virtual edge in skeleton(\( \mu \)) corresponds to a pair of adjacent nodes in \( T_G \). No two S nodes (resp., no two P nodes) are adjacent. Every edge in \( E(G) \) appears in the skeleton of exactly one node. The tree \( T_G \) has \( O(|E(G)|) \) nodes and it can be computed in \( O(|E(G)|) \) time [9].

It is also known that \( T_G \) represents all combinatorial embeddings of \( G \) in \( \mathbb{R}^2 \) in the following manner [9]. Choosing a root node for \( T_G \) and an embedding of its skeleton. Then successively replace each virtual edge \( uv \) by the skeleton of the corresponding node \( \mu \) minus the virtual edge \( uv \) in skeleton(\( \mu \)). In each step of the recursion, if \( \mu \) is of type R, skeleton(\( \mu \)) can be flipped (reflected) around \( u \) and \( v \), and if \( \mu \) is of type P, the parallel edges between \( u \) and \( v \) can be permuted arbitrarily.

**Combinatorial representation of weak embeddings.** Given an embedding \( \psi_\varphi : G \rightarrow \mathcal{H} \), where \( \varphi \) is in normal form, we define a combinatorial representation \( \pi_\varphi \) as the set of total orders of edges in \( R_{uv} \), for all pipes \( uv \in E(H) \). Specifically, for every pipe \( uv \in E(H) \), fix an orientation of the boundary of \( R_{uv} \) (e.g., the one used in the construction of the strip system \( \mathcal{H} \)). Record the order in which the pipe edges in the embedding \( \psi_\varphi \) intersect a fixed side of \( \partial R_{uv} \) (say, the side \( \partial R_{uv} \cap \partial D_u \)) when we traverse it in the given orientation. Let \( \Pi(\varphi) \) be the set of combinatorial representations \( \pi_\varphi \) of all embeddings \( \psi_\varphi : G \rightarrow \mathcal{H} \). We say that two components \( C_1 \) and \( C_2 \) of \( G_u \) cross if and only if their terminals interleave in the cyclic order around \( \partial D_u \) (i.e., there exists no cut in the cyclic order in which all terminals of \( C_1 \) appear before all terminals of \( C_2 \)). If \( \pi_\varphi \in \Pi(\varphi) \), then \( \pi_\varphi \) induces no two components to cross.

**Lemma 2.** Given a set of total orders \( \pi_\varphi \), we can reconstruct an embedding \( \psi_\varphi : G \rightarrow \mathcal{H} \) if \( \pi_\varphi \in \Pi(\varphi) \), or detect that \( \pi_\varphi \notin \Pi(\varphi) \) in \( O(m) \) time.

**Proof.** For each \( uv \in E(H) \), draw \( \varphi^{-1}[uv] \) parallel Jordan arcs in \( R_{uv} \) connecting \( \partial D_u \) and \( \partial D_v \). For each \( u \in V(H) \), \( \pi_\varphi \) defines a ccw cyclic order of terminals around \( \partial D_u \). Compute \( \overline{C} \) for each component \( C \) and its SPQR trees. Use the SPQR trees to determine if there exists an embedding of the corresponding component \( C \) in which the terminals appear in the same cyclic order as the
one defined by $\pi$. If not, report that $\pi \not\in \Pi(\varphi)$. If yes, embed $C$ inside $D_u$ given the position of the already embedded terminals on $\partial D_u$. This decomposes $D_u$ into different faces. If a component connects terminals in different faces, then two components in $G_u$ cross. If no two components cross, we can incrementally embed the components $C$, as there is always a face of $D_u$ that contains all terminals of $C$.

**Simplified form.** Given an instance $\varphi : G \rightarrow H$ in normal form, we simplify a graph $G$ by removing parts of the graph $G_u$, for all $u \in V(H)$, that are locally “irrelevant” for the embedding, such as 0-, 1-, and 2-connected components that are not adjacent to edges in any pipe incident to $u$. Formally, for each component $C$ of $G_u$, we call a split component defined by a 2-cut $\{p, q\}$ of $C$ irrelevant if it contains no pipe-vertices. An instance is in simplified form if it is in normal form and every $u \in V(H)$ satisfies properties (P3)–(P4) below.

(P3) For every component $C$ of $G_u$, $\overline{C}$ is biconnected and every 2-cut of $\overline{C}$ contains at least one pipe-vertex.

Assuming that a cluster $u$ satisfies (P3), we define $T_C$ as the SPQR tree of $\overline{C}$ where $C$ is a component of $G_u$. Given a node $\mu$ of $T_C$, let the core of $\mu$, denoted core($\mu$), be the subgraph obtained from skeleton($\mu$) by deleting all pipe-vertices. Property (P4) below will allow us to bound the number of vertices of $G_u$ in terms of its number of terminals (cf. Lemma 4).

(P4) For every component $C$ of $G_u$, and every R node $\mu$ of $T_C$, core($\mu$) is isomorphic to a wheel $W_k$, for some $k \geq 4$, whose external vertices have degree 4 in $G_u$.

We now describe subroutine simplify($u$) that, for a given instance $\varphi : G \rightarrow H$ in normal form and a cluster $u \in V(H)$, returns an instance $\varphi' : G' \rightarrow H$ such that $u$ satisfies (P1)–(P4). We break the subroutine into two steps.

**simplify($u$).** Input: an instance $\varphi : G \rightarrow H$ in normal form and a cluster $u \in V(H)$.

**For every component $C$ of $G_u$, do the following.**

(1) If $C$ is not planar, report that $\varphi$ is not a weak embedding and halt. If $\text{pipe-deg}(C) = 0$, then delete $C$. Else compute $\overline{C}$, and find the maximal biconnected component $\overline{C}$ of $\overline{C}$ that contains all pipe-vertices. The component $\overline{C}$ trivially exists if $\text{pipe-deg}(C) \in \{1, 2\}$, and if $\text{pipe-deg}(C) \geq 3$, it exists since $E_C$ forms a cycle containing all pipe-vertices. Modify $C$ by deleting all vertices of $\overline{C} \setminus \overline{C}$, and update $\overline{C}$ (by deleting the same vertices from $\overline{C}$, as well); refer to Fig. 3(b)–(c). Consequently, we may assume that $\overline{C}$ is biconnected and contains all pipe-vertices. Compute the SPQR tree $T_C$ for $\overline{C}$. Set a node $\mu_r$ in $T_C$ whose core contain a pipe-vertex as the root of $T_C$. Traverse $T_C$ using DFS. If a node $\mu$ is found such that skeleton($\mu$) contains no pipe-vertex, let $\{p, q\}$ be the 2-cut of $\overline{C}$ shared by skeleton($\mu$) and skeleton(parent($\mu$)). Replace all irrelevant split components defined by $\{p, q\}$ by a single edge $pq$ in $C$. If $p$ or $q$ now have degree 2, suppress $p$ or $q$, respectively. Update $\overline{C}$ accordingly, and update $T_C$ to reflect the changes in $\overline{C}$ by changing $pq$ from virtual to real and possibly suppressing $p$ and/or $q$ in skeleton(parent($\mu$)), which also deletes node $\mu$ and its descendants since their skeletons contain edges in the deleted irrelevant split components; refer to Fig. 3(c)–(d). Continue the DFS ignoring deleted nodes.

(2) While there is an R node $\mu$ in $T_C$, of a component $C$ in $G_u$, that does not satisfy (P4), do the following. Let $Y$ be the set of edges in skeleton($\mu$) adjacent to core($\mu$) (i.e., edges between a vertex...
Lemma 3. Given an instance \( \varphi : G \to H \) in normal form and a cluster \( u \in V(H) \), the instance \( \varphi' = \text{simplify}(u) \) and \( \varphi \) are equivalent and \( u \) satisfies \([P1],[P4]\) in \( \varphi' \). The operation runs in \( O(|E(G_u)|) \) time. By successively applying simplify to all clusters in \( V(H) \), we obtain an equivalent instance in simplified form in \( O(m) \) time.

Proof. First we prove that \( u \) satisfies \([P1],[P4]\) in \( \varphi' \). Since \( \varphi \) is in normal form, \( u \) satisfies \([P1],[P4]\) in \( \varphi \). By construction, \( u \) still satisfies \([P1],[P4]\) in \( \varphi' \). After step 1, \( \overline{C} \) is clearly biconnected and every node \( \mu \) of \( T_C \) contains a pipe-vertex in its skeleton. Step 2 does not change this property. This implies that core(\( \mu \)) contains only real edges for every node \( \mu \). Suppose for contradiction that there is a 2-cut \( \{p,q\} \) such that neither \( p \) nor \( q \) is a pipe-vertex. Then \( \{p,q\} \) must be in core(\( \mu \)) where \( \mu \) is a S node. Then one split component of \( \{p,q\} \) is a path of length two or more. But \( G_u' \) has no degree-2 vertex by \([P2]\), a contradiction. Hence, \( u \) satisfies \([P3]\) in \( \varphi' \). By definition, after step 2 \( u \) satisfies \([P4]\) in \( \varphi' \).

We now show that the operation takes \( O(|E(G_u)|) \) time. In step 1, planarity testing is done in linear time for each component \( C \) of \( G_u \). We obtain \( \overline{C} \) by a DFS. We compute \( T_C \) in \( O(|E(C)|) \) time \cite{9}. Replacing irrelevant split components by one edge can be done in \( O(|E(C)|) \) overall time. In step 2, we can obtain a list of R nodes in \( O(|E(C)|) \) time. The changes in step 2 are local, both in \( C \) and \( T_C \), and do not influence whether other R nodes satisfy \([P4]\). Step 2 takes \( O(|E(G_u)|) \) time overall by processing each R node sequentially. All the changes are local to \( u \) and, by successively applying simplify, we obtain a simplified form in \( O(|E(G)|) \) time.

Finally, we show that \( \varphi \) and \( \varphi' \) are equivalent. Notice that there is a bijection between the terminals of \( \varphi \) and \( \varphi' \). We show that \( \Pi(\varphi) = \Pi(\varphi') \), i.e., given \( \pi_\varphi \in \Pi(\varphi) \), then \( \pi_\varphi \in \Pi(\varphi') \) and vice versa. Notice that, given an order \( \pi_\varphi \), for each \( u \in V(H) \), \( \pi_\varphi \) induces two components \( C_1 \) and \( C_2 \) of \( G_u \) to cross if and only if the corresponding components \( C'_1 \) and \( C'_2 \) of \( G'_u \) also cross. Then, it suffices to show that the SPQR trees of \( \overline{C} \) and \( \overline{C}' \) for corresponding components \( C \) of \( G_u \) and \( C' \) of \( G'_u \) represent the same constraints in the cyclic order of terminals around \( D_u \). Step 1 deletes components of pipe-degree 0, which do not pose any restriction on the cyclic order of terminals. The subgraphs represented by irrelevant subtrees in the SPRQ tree of \( \overline{C} \) can be flipped independently and, since they do not contain pipe-vertices, their embedding does not interfere with the order of edges adjacent to pipe-vertices. Hence, step 1 does not alter any constraint on the cyclic order of terminals. By construction, step 2 does not change the circular order of edges in core(\( \mu \)). Replacing core(\( \mu \)) by a wheel \( W_{|Y|+1} \) does not change any of the constraints on the cyclic order of terminals.
Lemma 4. After simplify$(u)$, every component $C$ of $G_u$ contains $O(t_C)$ edges, where $t_C$ is the number of terminals in $C$.

Proof. Let us contract every wheel that is a maximal biconnected components in $C'$ by $[P4]$ into a single vertex and remove any loops created by the contraction. Let $\hat{C}$ denote the resulting component. We have $|E(C)| \leq 5|E(\hat{C})|$, since the number of contracted edges is at most $4|E(\hat{C})|$. Indeed, a wheel $W_{k+1}$ has $2k$ edges, which are contracted, and its $k$ external vertices are incident to $k$ edges that are not contracted by $[P4]$. We charge each of these $k$ edges in $E(\hat{C})$ for two edges of $W_k$. Then every edge in $E(\hat{C})$ receives at most 2 units of charge from each of its endpoints, hence at most 4 units of charge overall. The component $\hat{C}$ is a tree without degree-2 vertices whose leaves are precisely the terminals of $C$ by $[P1],[P3]$. Since the number of edges in a tree is at most twice the number its leaves, by the above inequality we have $|E(C)| \leq 5|E(\hat{C})| \leq 5 \cdot 2 \cdot t_C$, as claimed. \hfill\Box

3 Operations

In this section, we present our two main operations, clusterExpansion and pipeExpansion, that we use successively in our recognition algorithm. Given an instance $\varphi$ and a cluster $u$ in simplified form, operation clusterExpansion$(u)$ either finds a configuration that cannot be embedded locally in the neighborhood of $u$ and reports that $\varphi$ is not a weak embedding, or replaces cluster $u$ with a group of clusters and pipes (in most cases reducing the number of edges in pipes). It first modifies the embedded graph $H$, and then handles each component of $G_u$ independently.

Operation clusterExpansion$(uv)$ first merges two adjacent clusters, $u$ and $v$ (and the pipe $uv$) into a single cluster $(uv)$ and invokes clusterExpansion$((uv))$. We continue with the specifics.

Cluster expansion. Input: an instance $\varphi : G \rightarrow H$ in simplified form and a cluster $u \in V(H)$. We either report that $\varphi$ is not a weak embedding or return an instance $\varphi' : G' \rightarrow H'$. The instance $\varphi'$ is computed incrementally: initially $\varphi'$ is a copy of $\varphi$. Let the expansion disk $\Delta_u$ be a topological closed disk containing a single cluster $u \in V(H)$ in its interior and intersecting only pipes incident to $u$. Steps 0–3 will insert new clusters and pipes into $H'$ that are within $\Delta_u$, step 4 will determine the rotation system for the new clusters and check whether the rotation system induces any crossing between new pipes within $\Delta_u$, and step 5 brings $\varphi'$ to its simplified form.

Step 0. For each pipe $uv \in E(H)$ incident to $u$, subdivide $uv$ by inserting a cluster $u_v$ in $H'$ at the intersection of $uv$ and $\partial \Delta_u$. If $\deg(u) \geq 3$, then add a cycle $C_u$ of pipes through all clusters in $\partial \Delta_u$ (hence the clusters $u_v$ appear along $C_u$ in the order given by the rotation of $u$); and if $\deg(u) = 2$, then add a pipe between the two clusters in $\partial \Delta_u$. Delete $u$ (and all incident pipes). As a result, the interior of $\Delta_u$ contains no clusters.

Step 1: Components of pipe-degree 1. For each component $C$ of $G_u$ such that pipe-deg$(C) = 1$, let $uv$ be the pipe to which the pipe-edges incident to $C$ are mapped to. Move $C$ to the new cluster $u_v$, i.e., set $\varphi'(C) = u_v$. (For example, see the component in $u_x$ in Fig. 4.)

Step 2: Components of pipe-degree 2. For each pair of clusters $\{v, w\}$ adjacent to $u$, denote by $B_{vw}$ the set of components of $G_u$ of degree 2 adjacent to pipe-edges in $\varphi^{-1}[uv]$ and $\varphi^{-1}[uw]$. For all nonempty sets $B_{vw}$, do the following.

(a) Insert the pipe $u_vu_w$ into $H'$ if it is not already present.
(b) For every component $C \in B_{vw}$, do the following:
(b1) Compute \( \overline{C} \) (by \([P3]\) \( \overline{C} \) is biconnected). Compute the SPQR tree \( T_C \) of \( \overline{C} \). Set a node \( \mu_r \) as the root of \( T_C \) so that \( \text{skeleton}(\mu_r) \) contains both pipe-vertices, which we denoted by \( v' \) and \( w' \) (i.e., consistently with Section 2). Note that \( \mu_r \) cannot be of type P as otherwise \( C \) would not be connected.

(b2) If \( \mu_r \) is of type S, then \( E(\text{skeleton}(\mu_r)) \setminus \overline{E}_C \) forms a path between the pipe vertices \( v' \) and \( w' \), that we denote by \( P \), where the first and last edges may be virtual. Notice that path \( P \) contains at most 3 edges otherwise \([P2]\) or \([P3]\) would not be satisfied. If \( P = (v', w') \) has length 1, then subdivide \( P \) into 3 edges \( P = (v', p_1, p_2, w') \). If \( P = (v', p, w') \) has length 2, then \( \{v', p\} \) is a 2-cut in \( C \) that defines two split components, \( C_v \) and \( C_w \), containing \( v' \) and \( w' \), respectively. Split \( p \) into two vertices \( p_1 \) and \( p_2 \) connected by an edge so that \( p_1 \) (resp., \( p_2 \)) is adjacent to every vertex in \( C_v \) (resp., \( C_w \)) that was adjacent to \( p \). This also changes \( \text{skeleton}(\mu_r) \) so that the length of \( P \) increases to 3. Finally, assume that \( P = (v', p_1, p_2, w') \) has length 3. Then the edge \( p_1p_2 \) defines an edge cut in \( C \) that splits \( C \) into two components each incident to a single pipe, one to \( uw \) and the other to \( uw' \). We define \( \varphi' \) so that it maps each of the two components into \( u_v \) or \( u_w \) accordingly. (See the components incident to pipe \( u_v, u_w \) in Fig. 4.)

(b3) If \( \mu_r \) is of type R, by \([P4]\) core(\( \mu_r \)) is a wheel subgraph \( W_k \). Let \( k_v \) and \( k_w \), where \( k_v + k_w = k - 1 \), be the number of edges between \( W_k \) and \( v' \), and between \( W_k \) and \( w' \), respectively. Replace \( W_k \) by two wheel graphs \( W_{k_v+4} \) and \( W_{k_w+4} \) connected by three edges so that the circular order of the edges around \( v' \) and \( w' \) is maintained (recall that an R node has a unique embedding). The triple of edges between \( W_{k_v+4} \) and \( W_{k_w+4} \) is called a thick edge. The thick edge defines a 3-edge-cut that splits \( C \) into two components, each with a wheel graph. We define \( \varphi' \) so that each of the two components is mapped to its respective vertex \( u_v \) or \( u_w \). (See the components incident to pipe \( u_v, u_w \) in Fig. 4.)

Step 3: Components of pipe-degree 3 or higher. For all the remaining components \( C \) (pipe-deg(\( C \)) \( \geq 3 \)) of \( G_u \), do the following. Assume \( C \) is incident to pipe-edges mapped to the pipes \( uw_1, uw_2, \ldots, uw_d \).

(a) Compute \( \overline{C} \) and its SPQR tree \( T_C \) and let \( v_i' \) be the pipe-vertex corresponding to terminals adjacent to edges in \( uw_i \). Set the node \( \mu_r \) as the root of \( T_C \) such that \( \text{skeleton}(\mu_r) \) contains the cycle \( \overline{E}_C \). The type of \( \mu_r \) is R, otherwise \( C \) would be disconnected.

(b) Changes in \( H' \). By \([P4]\) we have that core(\( \mu_r \)) is a wheel graph \( W_k \), \( k - 1 \geq d \). Let \( p_j \) be the \( j \)-th external vertex of \( W_k \) and \( p_C \) be its central vertex. Create a copy of \( W_k \) using clusters and pipes: Create a cluster \( u_{p_j} \) that represents each vertex \( p_j \), a cluster \( u_{p_C} \) that represents vertex \( p_C \), see Fig. 4 (middle). Insert the copy of \( W_k \) in \( H' \). Insert a pipe \( u_{p_j}u_{v_i} \) if \( p_jv_i \) is in \( E(\text{skeleton}(\mu_C)) \).

(c) Changes in \( G' \). Delete all edges and the central vertex of \( W_k \) from \( G' \), which splits \( C \) into \( k - 1 \) components. Set \( \varphi'(p_j) = u_{p_j} \) for \( j \in \{1, \ldots, k - 1\} \). By \([P4]\) every \( u_{p_j} \) is adjacent to a single edge \( u_{p_j}u_{v_i} \). We define \( \varphi' \) so that it maps the vertices of the component of \( C \) containing \( p_j \) to \( u_{v_i} \) except for \( p_j \). (Note that the cluster \( u_{p_C} \) and all incident pipes are empty.)

Step 4: Local Planarity Test. Let \( H_u \) be the subgraph induced by the newly created clusters and pipes. Let \( \overline{H_u} \) denote the graph obtained as the union of \( H_u \) and a star whose center is a new vertex (not in \( V(H_u) \)), and whose leaves are the clusters in \( \partial \Delta_u \). Use a planarity testing algorithm to test whether \( \overline{H_u} \) is planar. If \( \overline{H_u} \) is not planar, report that \( \varphi \) is not a weak embedding and halt. Otherwise, find an embedding of \( \overline{H_u} \) in which the center of the star is in the outer face. This defines a rotation system for \( H_u \). The rotation system of \( H' \) outside of \( \Delta_u \) is inherited from \( H \).

Step 5: Normalize. Finally, apply normalize to each new cluster in \( H' \). (This step subdivides edges so that \([P1]\) is satisfied as shown in Fig. 4 (right).)
Lemma 5. Given an instance \( \varphi : G \to H \) in simplified form containing a cluster \( u \), \text{clusterExpansion}(u) either reports that \( \varphi \) is not a weak embedding or produces an instance \( \varphi' : G' \to H' \) in simplified form that is equivalent to \( \varphi \) in \( O(|E(G_u)| + \deg(u)) \) time.

Proof. Step 5 of the operation receives as input an instance \( \varphi^* : G^* \to H^* \) and return an equivalent instance \( \varphi' \) by Lemma 1. By construction, the new clusters satisfy [P1]–[P4] in \( \varphi' \). It remains to show that \( \varphi^* \) and \( \varphi \) are equivalent, and to analyze the running time.

First assume that \( \varphi \) is a weak embedding, and so there is an embedding \( \psi_\varphi : G \to H \). We need to show that there exists an embedding \( \psi_{\varphi^*} : G^* \to H^* \), and hence \( \varphi^* \) is a weak embedding. This can be done by performing steps 0–3 on the graphs \( G \) and \( H \), and the embedding \( \psi_\varphi \), which will produce \( G^* \) and \( H^* \), and an embedding \( \psi_{\varphi^*} : G^* \to H^* \).

Next, assume that \( \varphi^* \) is a weak embedding. Given an embedding \( \psi_{\varphi^*} : G^* \to H^* \), we construct an embedding \( \psi_\varphi : G \to H \) as follows. Let \( H^*_u \) be the subgraph of \( H^* \) induced by the clusters created by \text{clusterExpansion}(u). Note that \( H^* \) is a connected plane graph: the clusters created in step 0 are connected by a path (if \( \deg(u) \leq 2 \)) or a cycle (if \( \deg(u) \geq 3 \)); and any clusters created in step 3 (when \( \deg(u) \geq 3 \)) are attached to this cycle. Since \( H^*_u \) is a connected plane graph, we may assume that there is a topological disk containing only the pipes and clusters of \( H^*_u \); let \( D_u \) denote such a topological disk.

Let \( G^*_u \) be the subgraph of \( G^* \) mapped to \( H^*_u \). We show that steps 0–3 of the operation can be reversed without introducing crossings. For each component \( C \) of \( G_u \) with pipe-deg\((C) \geq 3 \), embed a wheel \( W_k \) in the disk \( D_{uC} \) around the cluster \( uC \), and connect its external vertices to the vertices \( p_i, i = 1, \ldots, k - 1 \). Since the pipes incident to \( uC \) are empty, and each \( p_i \) is a unique vertex its cluster, this can be done without crossings. Now, every component \( C \) of \( G_u \) corresponds to a component \( C^* \) of \( G^*_u \), and by [P1] every terminal vertex in \( C \) corresponds to terminal in \( C^* \).

If we delete a component \( C^* \) from \( G^* \), there will be a face \( F \) of \( D_u \) (a component of \( D_u \setminus \psi_{\varphi^*}(G^*) \)) that contains all terminals of \( C \) on its boundary. Denote by \( \pi_C \) the ccw cyclic order in which these terminals appear in the facial walk of \( F \). If \( C \) admits an embedding in which the terminals appear in the outer face in the same order as \( \pi_C \), we can embed \( C \) in \( F \) on the given terminals. We show that \( C \) admits such an embedding by proving that the SPQR trees of \( C \) and \( C^* \) impose the same constraints on the cyclic order of terminals. Subdividing edges do not change these constraints. Step 1 does not change \( C \). If step 2(b2) adds an edge in the skeleton of an S node, the possible...
combinatorial embeddings remain the same. Step 2(b3) maintains the rotation of $v'$ and $w'$. If pipe-deg$(C) \geq 3$, $C$ and $C^*$ are identical apart from subdivided edges. Because all steps maintain the same constraints on the rotation system of the terminals, we can construct $\psi : G \rightarrow H$ by incrementally replacing the embedding of $C^*$ by an embedding of $C$ in $D_u$ for every component $C$ of $G_u$. By Lemma 2 this is possible without introducing crossings since no two components of $G_u$ cross and every component is planar since $\varphi$ is in simplified form.

Finally, we show that $\text{clusterExpansion}(u)$ runs in $O(|E(G_u)| + \deg(u))$ time. Step 0 takes $O(\deg(u))$ time. Steps 1–3 are local operations that take $O(|E(C)|)$ time for each component $C$ of $G_u$. Step 4 takes $O(|E(G_u)| + \deg(u))$ since each component $C$ inserts at most $O(|E(C)|)$ pipes in $H'$; the graph $H_u$ has $\deg(u)$ more edges than $H_u$, and planarity testing takes linear time in the number of edges $[16]$. Step 5 takes $O(|E(G_u)|)$ time by Lemma 1 and (P1).

Pipe Expansion. A cluster $u \in V(H)$ is called a base of an incident edge $uv \in E(H)$ if every component of $G_u$ is incident to: (i) at least one pipe edge in $uv$; and (ii) at most one pipe-edge or exactly three pipe edges that form a thick edge in any other pipe $uw$, $w \neq v$. We call a pipe $uv$ safe if both of its endpoints are bases of $uv$; otherwise it is unsafe. Given an instance $\varphi : G \rightarrow H$ and a safe pipe $uv \in E(H)$, operation $\text{pipeExpansion}(uv)$ consists of the following steps: First, produce an instance $\varphi^* : G \rightarrow H^*$ by contracting the pipe $uv$ into the new cluster $\langle uv \rangle \in V(H^*)$ while mapping the vertices of $G_u$ and $G_v$ to $\langle uv \rangle$, see Fig. 5(top); and then apply simplify and $\text{clusterExpansion}$ to $\langle uv \rangle$. The operation either reports that $\varphi$ is not a weak embedding or returns an instance $\varphi' : G' \rightarrow H'$.

![Pipe Expansion](image)

Figure 5: Pipe Expansion. A safe pipe $uv$ (top left). The cluster $\langle uv \rangle$ obtained after contraction of $uv$ (top right). The result of contracting the components in $G_u$ and $G_v$ (bottom left). The subsequent contraction of all components incident to $\partial_u D_{(uv)}$ and $\partial_v D_{(uv)}$, resp., and a Jordan curve that cross every edge of the resulting bipartite plane multigraph (bottom right).

We use the following folklore result in the proof of correctness of operation $\text{pipeExpansion}(uv)$. This result is obtained by an Euler tour algorithm on the dual graph of a plane bipartite multigraph.

**Theorem 2 (Belyi [4]).** For every embedded connected bipartite multigraph $G^*$, there exists a Jordan curve that crosses every edge of $G^*$ precisely once. Such a curve can be computed in $O(|E(G^*)|)$ time.
Lemma 6. Given an instance $\varphi : G \to H$ and a safe pipe $uv \in E(H)$, pipeExpansion$(uv)$ either reports that $\varphi$ is not a weak embedding or produces an equivalent instance $\varphi' : G' \to H'$.

Proof. Let $\varphi : G \to H$ be an instance in simple form, and let $uv$ be a safe pipe. Recall that pipeExpansion$(uv)$ starts by producing an instance $\varphi^* : G \to H^*$ by contracting the pipe $uv$ into the new cluster $\langle uv \rangle \in V(H^*)$ while mapping the vertices of $G_u$ and $G_v$ to $\langle uv \rangle$. It is enough to prove that $\varphi$ and $\varphi^*$ are equivalent, the rest of the proof follows from Lemmas 1 and 5.

One direction of the equivalence proof is trivial: Given an embedding $\psi : G \to \mathcal{H}$, we can obtain an embedding $\psi_{\varphi^*} : G \to H^*$ by defining $D_{\langle uv \rangle}$ as a topological disk containing only $D_u$, $D_v$, and $R_{uv}$.

For the other direction, assume that we are given an embedding $\psi_{\varphi^*} : G \to H^*$. We need to show that there exists an embedding $\psi : G \to \mathcal{H}$. We shall apply Theorem 2 after contracting certain subgraphs of $G_u$ and $G_v$ (as described below). If a component contains cycles, then the contraction of its embedding creates a bouquet of loops. We study the cycles induced by $G_u$ and $G_v$ and by thick edges to ensure that no other component is embedded in the interior of such cycles.

Components of $G_{\langle uv \rangle}$ of pipe-degree 0. Note that the terminals corresponding to the pipes incident to $u$ and $v$ lie in two disjoint arcs of $\partial D_{\langle uv \rangle}$, which we denote by $\partial_u D_{\langle uv \rangle}$ and $\partial_v D_{\langle uv \rangle}$, respectively. The components of graph $G_{\langle uv \rangle}$ with positive pipe-degree are incident to terminals in $\partial_u D_{\langle uv \rangle}$ or $\partial_v D_{\langle uv \rangle}$ (possibly both). The components of pipe-degree 0 can be relocated to any face of the embedding of all other components. Without loss of generality, we may assume that all components of pipe-degree 0 lie in a common face incident to both $G_u$ and $G_v$ in $\psi_{\varphi^*}$.

Cycles induced by $G_u$ and $G_v$, and by thick edges. Notice that [P3] and [P4] imply that every maximal biconnected component in $G_u$ and $G_v$ is a wheel. Since each wheel is 3-connected, the circular order of their external vertices is determined by the embedding $\psi_{\varphi^*}$. We may assume that no cycle through the external vertices of a wheel subgraph encloses any vertex except for the center of the wheel. Indeed, suppose a 3-cycle $(p_1, p_2, p_c)$ of a wheel encloses some vertex, where $p_c$ is the center of the wheel, and $p_1$ and $p_2$ are two consecutive external vertices. We can modify the embedding of the edge $p_1p_2$ in $\psi_{\varphi^*}$ so that it follows closely the path $(p_1, p_c, p_2)$ so that the 3-cycle does not contain any vertex, see Fig. 6.

Consider a thick edge $\theta$ in $G$ between a wheel in $G_u$ (or $G_v$) and a wheel in $G_w$ for some adjacent cluster $w \notin \{u, v\}$. Recall that a thick edge consists of three paths, say $P_1 = (p_1, t_1, t_2, p_2)$, $P_2 = (p_3, t_3, t_4, p_4)$, and $P_3 = (p_5, t_5, t_6, p_6)$, where $(p_1, p_3, p_5)$ and $(p_2, p_4, p_6)$ are consecutive external vertices of the two wheels, resp., and $t_i$ is the unique vertex in a cluster adjacent to terminal $t_i$ for $i \in \{1, \ldots, 6\}$; cf. [P1]. If we suppress the terminals, then the two wheels incident to the thick edge would be in the same maximal 3-connected component of $G$ and, therefore, their relative embedding is fixed. We can assume that no vertex is enclosed by the cycles induced by the vertices of the thick edge (i.e., by any pairs of paths from $P_1$, $P_2$, and $P_3$). Indeed, we can modify the embedding of the path $P_1$ and $P_3$ in $\psi_{\varphi^*}$ so that they closely follow the path $p_1p_3 \cup P_2 \cup p_4p_2$ and $p_5p_3 \cup P_2 \cup p_6p_6$, respectively. By [P4], such modification of the embedding is always possible without introducing crossings. We conclude that a cycle induced by the thick edge $\theta$ does not enclose any vertices of $G$. A similar argument can be used to show that we can also assume no cycle induced by a thick edge in $uv$ encloses a vertex in $\psi_{\varphi^*}$.

Separating $G_u$ and $G_v$. We next show that there exits a closed Jordan curve that separates $G_u$ and $G_v$, and crosses every edge between $G_u$ and $G_v$ precisely once. We contract subgraphs of $G_{\langle uv \rangle}$ in two steps.
(1) Contract each component \( C \) of \( G_u \) (resp., \( G_v \)) to a single vertex \( w_C \). This results in a bouquet of loops at \( w_C \). As argued above, a cycle through the external vertices of a wheel in \( G_u \) or \( G_v \) encloses only its center. Hence, none of the loops at \( w_C \) encloses any other vertex of \( G \), and they can be discarded.

(2) As \( uv \) is safe, every component \( C \) of \( G_u \) and \( G_v \) is incident to either at most one terminal in \( \partial D_{(uv)} \) or precisely three terminals corresponding to a thick edge. Contract the arc \( \partial_u D_{(uv)} \) (resp., \( \partial_v D_{(uv)} \)) and for every component \( C \) of \( G_u \) (resp., \( G_v \)) all edges between \( w_C \) and terminals into a new vertex \( u_\infty \in \partial D_{(uv)} \) (resp., \( v_\infty \in \partial D_{(uv)} \)), see Fig. 5(bottom-left); and insert an edge \( u_\infty v_\infty \) in the exterior of \( D_{(uv)} \). This results in a bouquet of loops at \( u_\infty \) (resp., \( v_\infty \)), corresponding to thick edges. As argued above, these loops do not enclose any vertices, and can be eliminated. The subgraph of all components of pipe-degree 1 or higher has been transformed into a connected bipartite plane multigraph, and each component of pipe-degree 0 is also transformed into a connected bipartite plane multigraph.

We apply Theorem 2 for each of these plane multigraphs independently. By our assumption, the components \( C \), pipe-deg\((C) = 0 \), lie in a common face. Consequently, we can combine their Jordan curves with the Jordan curve of all remaining components into a single Jordan curve that crosses every edge between \( G_u \) and \( G_v \) precisely once, see Fig. 5(bottom-right). After reversing the contractions and loop deletions described above, we obtain a Jordan curve that separates \( G_u \) and \( G_v \), and crosses every edge between \( G_u \) and \( G_v \) precisely once.

4 Algorithm and Runtime Analysis

In this section, we present our algorithm for recognizing weak embeddings. We describe the algorithm in Section 4.1 and prove that it recognizes weak embeddings in Section 4.1. The naive implementation would take \( O(m^2) \) time as explained below; we describe how to implement it in \( O(m \log m) \) time using additional data structures in Section 4.2.

4.1 Main Algorithm

We are given a piecewise linear simplicial map \( \varphi : G \rightarrow H \), where \( G \) is a graph and \( H \) is an embedded graph in an orientable surface. The graph \( G \) is stored using adjacency lists for each vertex \( p \in V(G) \). We store the combinatorial embedding of \( H \) using the rotation system of \( H \) (ccw order of pipes incident to each \( u \in V(H) \)). The mapping \( \varphi \) is encoded by its restriction to \( V(G) \): that is, by the images \( \varphi(p) \) for all \( p \in V(G) \). For each cluster \( u \in V(H) \), we store the set of components of \( G_u \), each stored as the ID of a representative edge and the pipe-degree of the component. Every pipe \( uv \in E(H) \) has two Boolean variables to indicate whether the respective endpoints are its bases.
We introduce some terminology for an instance $\varphi : H \to G$.

- A component $C$ of $G_u$, in a cluster $u \in V(H)$, is **simple** if $\text{pipe-deg}(C) = 2$ and in each of the two pipes, $C$ is incident to exactly one edge or exactly one thick edge. Otherwise $C$ is **complex**.
- Recall that a pipe $uv \in E(H)$ is **safe** if both $u$ and $v$ are bases of $uv$; otherwise it is **unsafe**.
- A cluster $u \in V(H)$ (resp., a pipe $uv \in E(H)$) is **empty** if $\varphi^{-1}[u] = \emptyset$ (resp., $\varphi^{-1}[uv] = \emptyset$). Otherwise $u$ is **nonempty**.
- A safe pipe $uv \in E(H)$ is **useless** if $uv$ is empty or if $u$ and $v$ are each incident to two nonempty pipes and every component of $G_u$ and $G_v$ is simple. Otherwise, the safe pipe $uv$ is **useful**.

Notice that the pipe expansion of a useless pipe may not change the instance combinatorially. As we show below (Lemma 10), our algorithm reduces $G$ to a collection of **thick cycles**, defined as a cycle in which each node is a single vertex or a wheel, and each edge is either a single edge or a thick edge. (If we contract every wheel to a single vertex we obtain a cycle, possibly with multiple edges coming from thick edges.)

**Algorithm($\varphi$).** Input: an instance $\varphi : G \to H$ in simplified form.

**Phase 1.** Apply $\text{clusterExpansion}$ to each $u \in V(H)$. Denote the resulting instance by $\varphi' : G' \to H'$. Build the data structures described above for $\varphi' : G' \to H'$.

**Phase 2.** While there is a useful pipe in $H'$, let $uv \in E(H')$ be an arbitrary useful pipe and apply $\text{pipeExpansion}(uv)$.

**Phase 3.** If any component of $G'$ that contains a wheel is nonplanar, then report that $\varphi$ is not a weak embedding and halt. Otherwise, in each cluster contract every wheel component to a single vertex, and turn every thick edge into a single edge by removing multiple edges. Denote the resulting instance by $\varphi'' : G'' \to H''$. If any component $C$ of $G''$ is a cycle with $k$ vertices but $\varphi''(C)$ is not a cycle with $k$ clusters in $H''$, then report that $\varphi$ is not a weak embedding, else report that $\varphi$ is a weak embedding. This completes the algorithm.

**Analysis of Algorithm.** We show that the Algorithm recognizes whether the input $\varphi : G \to H$ is a weak embedding. The running time analysis follows in Section 4.2. We start by showing that the algorithm terminates.

**Lemma 7.** The algorithm terminates and Phase 2 is executed $O(m)$ times.

**Proof.** It is enough to show that Phase 2 terminates as both Phase 1 and Phase 3 consist of for-loops only. We show that Phase 2 is executed $O(m)$ times using the potential function

$$\Phi_1(\varphi) = \sum_{uv \in E(H)} \min\{\sigma(uv) - 1, 0\}$$

for $\varphi : G \to H$, where $\sigma(uv)$ is defined as the number of pipe-edges in $\varphi^{-1}[uv] \setminus 3$ minus three times the number of thick edges (so that each thick edge is counted as one). By definition, $0 \leq \Phi_1(\varphi) = O(m)$. It suffices to show that for every safe and useful pipe $uv$, $\text{pipeExpansion}(uv)$ decreases the potential by at least one.
Let $C$ be a component of $G_{uv}$ and let $\sigma_C(uv)$ be the number of pipe-edges of $C$ minus three times the number of thick edges of $C$ in $\varphi^{-1}[uv]$ before $\text{pipeExpansion}(uv)$. Since $uv$ is safe, every component of $G_u$ and $G_v$ is incident to a pipe-edge in $uv$, consequently

$$\sigma(uv) = \sum_C \sigma_C(uv),$$

where the summation is over all components $C$ of $G_{uv}$. By the definition of safe pipes, we have $\sigma_C(uv) \geq 1$, and $C$ contains at least one component from each of $G_u$ and $G_v$ as subgraphs. Since $uv$ is useful, at least one component $C$ of $G_{uv}$ contains a complex component of $G_u$ or $G_v$, and consequently $\sigma_C(uv) > 1$.

If a component $C$ of $G_{uv}$ consists of two simple components, one from $G_u$ and $G_v$ each, then $\sigma_C(uv) = 1$, and $\text{pipeExpansion}(uv)$ transforms $C$ into two simple components in two adjacent new clusters along the disk $\Delta_{uv}$, connected by a single edge or a thick edge. In this case, $\text{pipeExpansion}(uv)$ does not change the contribution of $C$ to the potential.

If $C$ contains a complex component of $G_u$ or $G_v$, we show that $\text{pipeExpansion}(uv)$ decreases $\Phi_1$ due to the edges in $C$ by at least $\sigma_C(uv) - 1$. Distinguish three cases based on the pipe-degree of $C$. If pipe-deg($C$) $\in \{0, 1\}$, no edge of the component will be a pipe edge after $\text{pipeExpansion}(uv)$. If pipe-deg($C$) $= 2$, $C$ will produce a single (thick) edge mapped to a pipe inside $\Delta_{uv}$ and therefore will contribute at most one unit to the potential. If pipe-deg($C$) $\geq 3$, the pipe edges produced by $\text{pipeExpansion}(uv)$ due to $C$ are the only edges mapped to the corresponding pipes in $H'$, which do not contribute to potential $\Phi_1$.

Overall, the contribution of a component $C$ of $G_{uv}$ to $\Phi_1$ never increases, but it strictly decreases for at least one of components. It follows that each operation $\text{pipeExpansion}(uv)$ decreases $\Phi_1$ by at least one, and so the Phase 2 is executed $O(m)$ times, as claimed.

We show next that Phase 2 reduces $G'$ to a collection of cycles and thick cycles (see Lemma 10 below). First, we need a few observations.

**Lemma 8.** Every cluster $u'$ produced by an operation $\text{clusterExpansion}(u)$ satisfies the following:

(B1) $u'$ is either empty or the base for some nonempty pipe.

(B2) If $u'$ lies in the interior of the disk $\Delta_u$, then $u'$ is either empty or the base of a unique nonempty pipe $u'v'$, where $v'$ in on the boundary of $\Delta_u$.

(B3) If $u'$ lies on the boundary of the disk $\Delta_u$, then $u'$ is either empty, or the base of at least one and at most two pipes, exactly one of which is outside of $\Delta_u$.

(B4) If $u'$ is nonempty, then $u'$ is incident to at most three empty pipes.

**Proof.** Operation $\text{clusterExpansion}(u)$ creates a cluster $u_v$ on the boundary of the disk $\Delta_u$ for every pipe $uv$ incident to $u$. By construction, $u_v$ is either empty (if the pipe $uv$ was empty), or a base for the pipe $vu_v$, which lies outside of $\Delta_u$. When $u_v$ is nonempty, the only incident pipes that could be empty are created in Step 0 (either two pipes of the cycle $C_u$ or a single pipe if deg($u$) $= 2$).

For each component $C$ of $G_u$ with pipe-deg($C$) $\geq 3$, $\text{clusterExpansion}(u)$ creates a wheel where the center cluster is empty and every external cluster is incident to exactly one nonempty pipe (to a cluster on the boundary of $\Delta_u$), consequently, it is a base for that pipe. The external clusters are incident to exactly three empty pipes by **(P4).**
The following hold for the instance

**Lemma 10.**

Let

**Proof.** Phase 1 of the algorithm performs \( \text{clusterExpansion}(u) \) for every cluster of the input independently. At the end of Phase 1 (i.e., beginning of Phase 2), each cluster in \( H' \) has been created by a \( \text{clusterExpansion} \) operation, and \( (B1) \) follows from Lemma 8.

Subsequent steps of Phase 2 successively apply \( \text{pipeExpansion} \) operations thereby creating new clusters. By Lemma 8, property \( (B1) \) is established for all new clusters, and it continues to hold for existing clusters.

**Lemma 9.** In every step of Phase 2, if \( H' \) contains a nonempty unsafe pipe, then it also contains a nonempty useful pipe.

**Proof.** Let \( u_0u_1 \) be an nonempty unsafe pipe in \( H' \). Without loss of generality, assume that \( u_1 \) is not a base for \( u_0u_1 \). We iteratively define a path starting with \( u_0u_1 \) as follows. For \( i \geq 1 \), assume that \( u_{i-1}u_i \) is a nonempty unsafe pipe in \( H' \) for which \( u_1 \) is not a base. By \( (B2) \), \( u_i \) is a base for some nonempty pipe \( u_iu_{i+1} \), where \( u_{i+1} \neq u_i \). This iterative process either finds a safe pipe or a cycle.

**Case 1:** The iterative process finds a safe pipe \( u_iu_{i+1} \). We claim that \( u_iu_{i+1} \) is useful. Suppose, to the contrary, that \( u_iu_{i+1} \) is useless. Then \( u_i \) is incident to two nonempty pipes (which are necessarily \( u_{i-1}u_i \) and \( u_iu_{i+1} \)), and every component in \( G_{u_i} \) has pipe-degree 2. Consequently, \( u_i \) is a base for both \( u_{i-1}u_i \) and \( u_iu_{i+1} \). This contradicts our assumption that \( u_i \) is not a base for \( u_{i-1}u_i \); and proves the claim.

**Case 2:** The iterative process finds a cycle \( U = (u_k, u_{k+1}, \ldots, u_\ell) \) where each \( u_i, k \leq i \leq \ell \), is a base for \( u_{i-1}u_i \) but not a base for \( u_iu_{i+1} \). We show that this case does not occur. All clusters in the cycle have been created by \( \text{clusterExpansion} \) operation. Not all clusters in \( U \) are created by the same \( \text{clusterExpansion} \) operation. Indeed, if \( u_i \) and \( u_{i+1} \), for some \( i \), are on the boundary of an expansion disk \( \Delta_u \) then \( u_{i+1} \) is not the base of \( u_iu_{i+1} \) by the construction of \( U \). Furthermore, \( u_iu_{i+1} \) is nonempty by the definition of a base. By \( (B3) \), \( u_{i+1} \) is the base of a single pipe, which is outside of \( \Delta_u \), as the other base would have to be \( u_iu_{i+1} \). Due to the previous claim and since \( U \) is a cycle, we can find \( u_i \) and \( u_{i+1} \), created by two different applications of \( \text{clusterExpansion} \), such that \( u_i \) was created before \( u_{i+1} \). By \( (B3) \), \( u_{i+1} \) is a base for \( u_iu_{i+1} \). This contradicts our assumption, and proves that Case 2 does not occur.

**Lemma 10.** The following hold for the instance \( \varphi' : G' \to H' \) at the end of Phase 2:

1. every pipe in \( E(H') \) is empty or useless,
2. every component of \( G' \) is either a cycle or a thick cycle, and
3. any two components of \( G' \) are mapped to the same or two vertex-disjoint cycles in \( H' \).

**Proof.** 1. When the while loop of Phase 2 terminates, there are no useful pipes in \( H' \). Consequently, every safe pipe is useless. By Lemma 9, every unsafe pipe is empty at that time. Overall, every pipe is empty or useless, as claimed.

2. Since every pipe \( uv \in E(H') \) is empty or useless, every component in every cluster is simple. It follows that every component of \( G' \) must be a cycle or a thick cycle.

3. Since every pipe \( uv \in E(H') \) is empty or useless, every cluster is incident to at most two nonempty pipes. Consequently, any two (thick) cycles of \( G' \) are mapped to either the same cycle or two disjoint cycles in \( H' \).
Lemma 11. The algorithm reports whether $\varphi$ is a weak embedding.

Proof. By Lemmas 5 and 6, every operation either reports that the instance is not a weak embedding and halts, or produces a instance equivalent to the input $\varphi$. Consequently, if any operation finds a negative instance, then $\varphi$ is not a weak embedding, and if the while loop in Phase 2 terminates, it yields an instance $\varphi' : G' \to H'$ equivalent to $\varphi$. By Lemma 10 every component of $G'$ is a cycle or a thick cycle, any two of which are mapped to the same or disjoint cycles in $H'$.

![Figure 7: A thick cycle that cannot be embedded in an annulus.](image)

Note that the strip system of the subgraph $\varphi'(C)$ of $H'$, where $C$ is a cycle or a thick cycle, is homeomorphic to the annulus, since $M$ is orientable. Hence, if $C$ is not planar, which can happen only when $C$ is thick and projective planar, then $\varphi'$ is clearly not a weak embedding (see Fig. 7). On the other hand, if $\varphi'(C)$ is planar the restriction of $\varphi'$ to $C$ is obviously embeddable in an annulus. Furthermore, a cycle or thick cycle $C$ that winds around $\varphi'(C)$ several times cannot be embeddable on an orientable surface (Fig. 1(d)), but one or more cycles that each wind around $\varphi'(C)$ once can be embedded in nested annuli in the strip system $H'$ (Fig. 1(c)). This completes the proof of correctness of Algorithm($\varphi$).

4.2 Efficient Implementation

Recall that $\text{pipeExpansion}(uv)$ first contracts the pipe $uv$ into the new cluster $\langle uv \rangle \in V(H^*)$. Each simple component $C$ of $G_{\langle uv \rangle}$ is composed of two simple components, in $G_u$ and $G_v$ respectively. Then $\text{clusterExpansion}$ performed at the end of $\text{pipeExpansion}(uv)$ splits $C$ into two simple components in two new clusters. That is, two adjacent simple components in $G_u$ and $G_v$ are replaced by two adjacent simple components in two new clusters. Consequently, we cannot afford to spend $O(|E_{\langle uv \rangle}|)$ time for $\text{pipeExpansion}(uv)$. We introduce auxiliary data structures to handle simple components efficiently: we use set operations to maintain a largest set of simple components in $O(1)$ time. A dynamic variant of the heavy path decomposition yields an $O(m \log m)$ bound on the total time spent on simple components in the main loop of the algorithm.

**Data structures for simple components.** For each pair $(u, uv)$ of a cluster $u \in V(H)$ and an incident pipe $uv \in E(H)$, we store a set $L(u, uv)$ of all simple components of $G_u$ adjacent to a (thick) edge in $\varphi^{-1}[uv]$. For every cluster $u \in V(H)$, let $w^*(u)$ be a neighbor of $u$ maximizing the size of $L(u, uv^*(u))$; we maintain a pointer from $u$ to a set $L(u, uv^*(u))$. The total number of sets $L(u, uv)$ and the sum of their sizes are $O(m)$. We can initialize them in $O(m)$ time. For each cluster $u \in V(H)$, we store the components in two sets: a set of simple and complex components, respectively, each stored as the ID of a representative edge of the component. Hence, for each pipe we can determine in $O(1)$ time whether $uv$ is useful or useless. Every simple component in $G_u$ has a pointer to the two sets $L(u, .)$ in which it appears.
In each iteration of Phase 2, we implement \texttt{pipeExpansion}(uv) for a useful pipe \( uv \) as follows. Compute the complex components of \( G'_w \) using DFS starting from each complex component of \( G'_u \) and \( G'_v \). If a simple component of \( G'_u \) and \( G'_v \) is absorbed by a new complex component of \( G'_w \), delete it from the set \( L(u,. \) or \( L(v,. \) in which it appears.

(a) We perform \texttt{clusterExpansion} only on complex components.
(b) We handle the simple components of \( G'_w \) as follows (refer to Fig. 8). Since \( uv \) is safe, every simple component of \( G'_u \) and \( G'_v \) appears in \( L(u,uv) \) and \( L(v,vu) \), respectively. For all \( w, w \notin \{v, w^*\} \), where \( w^* = w^*(u) \), move all components from \( L(u,uv) \) to the cluster \( \langle uv \rangle_w \). Similarly, for all \( x, x \notin \{u, x^*\} \), where \( x^* = w^*(v) \), move all components from \( L(v,vx) \) to the cluster \( \langle vx \rangle_x \).

(c) We add the new simple components (created by \texttt{clusterExpansion} from complex components) that fit the definition of \( L(\langle u,v \rangle_{w^*}, \langle u,v \rangle_{w^*}\langle u,v \rangle_{x^*}) \) to the current set \( L'(u,uv) \) to obtain \( L(\langle u,v \rangle_{w^*}, \langle u,v \rangle_{w^*}\langle u,v \rangle_{x^*}) \). Analogously, add new simple components to the current sets \( L'(v,vu) \), \( L(u,uv) \) where \( w \neq v \), and \( L(v,vx) \) where \( x \neq u \) to obtain the new sets \( L(\langle u,v \rangle_{x^*}, \langle u,v \rangle_{x^*}\langle u,v \rangle_{w^*}) \), \( L(\langle u,v \rangle_{w}, \langle u,v \rangle_{w^*}) \), and \( L(\langle u,v \rangle_{x}, \langle u,v \rangle_{x^*}) \), respectively. Compute any other sets of simple components from scratch.

We show that the running time of the algorithm improves to \( O(m \log m) \). In order to design a new potential function (\( \Phi_3(\varphi) \), defined below), we need an upper bound on the total number of simple components created in Phase 2.

**Lemma 12.** At most \( 12m \) simple components are created in Phase 2 of the algorithm.

**Proof.** We use a potential function to charge the creation of new simple components. Let

\[
\Phi_2(\varphi) = \sum_{uv \in E(H)} |\varphi^{-1}[uv]| - 2\ell(G),
\]

where \( \ell(G) \) denotes the number of leaves in \( G \). At the end of Phase 1, there are at most \( 3m \) pipe-edges: \texttt{clusterExpansion}(u) creates at most one new edge (or one thick edge) for each connected component of \( G_u \). The maximum number of leaves is always two times the number of pipe-edges.
edges. Hence, the term $-2\ell(G)$ in (1) is at least $-2 \cdot 2 \cdot 3m = -12m$. Consequently, we have $-10m = 2m - 12m \leq \Phi_2(\varphi') \leq 2m$ for the instance $\varphi'$ obtained at the end of Phase 1.

We show that Phase 2 can only decrease the value of $\Phi_2$. Additionally, we show that the creation of a new simple component can be charged to the decrease of $\Phi_2$, which will conclude the proof.

Let $C$ be a component of $G_{uv}$ in pipeExpansion($uv$), where $uv$ is useful (hence safe). If pipe-deg($C$) = 1, by the definition of safe pipes, the resulting instance will have at least one fewer pipe-edge inherited from $C$ than $\sigma_C(uv)$. If pipe-deg($C$) = 2, and $C$ is simple, $\Phi_2(\varphi)$ remains unchanged and no simple component is created. If pipe-deg($C$) = 2, and $C$ is complex, then at most one simple component is created, or else $C$ would be simple. As argued in the proof of termination (Lemma 7), the resulting instance will have at least one fewer pipe-edge inherited from $C$ than $\sigma_C(uv)$, and no leaf is created. If pipe-deg($C$) $\geq$ 3, let $W_k$, $k \geq 4$, be the wheel described in the definition of clusterExpansion. By the definition of safe pipes, $C$ contains at least $k - 2$ edges between $G_u$ and $G_v$ (which are pipe-edges before the contraction of $uv$). The operation creates $k - 1$ pipe-edges, all of which are leaves. Then the difference between the new and the old value of $\Phi_2$ is at most $[(k - 1) - 2(k - 1)] - [k - 2] = 3 - 2k$. For $k \geq 4$ we have $3 - 2k < -k$, and the operation creates at most $k$ simple components, this completes the proof. \hfill \Box

**Lemma 13.** Our implementation of the algorithm runs in $O(m \log m)$ time.

**Proof.** Phases 1 and 3 take $O(m)$ time by Lemmas 5, 10 and 12. The while loop in Phase 2 terminates after $O(m)$ iterations by Lemma 7. Using just Lemmas 3 and 5 each iteration of Phase 2 would take $O(m)$ time leading to an overall running time of $O(m^2)$. We define a new potential function for an instance $\varphi : G \to H$ to show that each iteration of Phase 2 takes $O(\log m)$ amortized time.

For every $u \in V(H)$, let $L(u)$ be the number of simple components in $G_u$. Let $s$ be the number of simple components created from the beginning of Phase 2 up to the current iteration. We define a new potential function as

$$\Phi_3(\varphi) = \Phi_1(\varphi) + (12m - s) \log(28m) + \sum_{u \in V(H)} L(u) \log L(u).$$

By Lemma 12, the second term is nonnegative. Note that $\Phi_3(\varphi) = O(m \log m)$ since $\Phi_1(\varphi) = O(m)$ and $\sum_{u \in V(H)} L(u) = O(m)$. We show that $\Phi_3$ monotonically decreases in Phase 2. As argued above (cf. Lemma 7), $\Phi_1$ monotonically decreases. The second term of $\Phi_3$ can only decrease since $s$ increments when simple clusters are created (but never decrements). The term $\sum_{u \in V(H)} L(u) \log L(u)$ increases when new simple components are created. However, this increase is offset by the decrease in the second term of $\Phi_3$. It suffices to consider the case that $L(u)$ increments from $k$ to $k + 1$. Then

$$k \log k + \log(28m) \geq \left( k \log(k + 1) - k \log \frac{k + 1}{k} \right) + (2 + \log(k + 1))$$

$$\geq (k + 1) \log(k + 1) + 2 - \log \left( 1 + \frac{1}{k} \right)^k$$

$$\geq (k + 1) \log(k + 1) + 2 - \log e$$

$$\geq (k + 1) \log(k + 1),$$

---

2 All logarithms are of base 2.
that is, the decrease of $\log(28m)$ in the second term offsets the increase of $(k+1) \log(k+1) - k \log k$ in the third term.

We next show that the time spent on each iteration of the while loop in Phase 2 is bounded from above by a constant times the decrease of the potential $\Phi_3$. This will complete the proof since the potential $\Phi_3$ is nonincreasing throughout the execution of the algorithm as we have just shown.

Let $\text{complex}(G_{uv})$ denote the number of edges of $G_{uv}$ in complex components, and $\Lambda$ be the collection of sets $L(u, uv)$ where $w \notin \{v, w^*(u)\}$ and $L(v, vx)$ where $x \notin \{u, w^*(v)\}$. Our implementation of $\text{pipeExpansion}(uv)$ spends $O(\text{complex}(G_{uv}) + \deg(\langle uv \rangle))$ time to process complex components, by Lemma 5, and $O(\sum_{L \in \Lambda} |L|)$ to process simple components. Hence, the decrease in $\Phi_3$ is nonincreasing throughout the execution of the algorithm as we have just shown.

First, let $C$ be a complex component in $G_{uv}$. We have seen (in the proof of Lemma 7) that $\text{pipeExpansion}(uv)$ decreases $\Phi_1$ by at least $\sigma_C(uv) - 1$ due to edges in $C$. By the definition of safe pipes, $C$ contains a nonempty set $A$ of components of $G_u$ and a nonempty set $B$ of components of $G_v$. By Lemma 6, each component $C' \in A \cup B$ contains $O(t_{C'})$ edges, where $t_{C'}$ is the number of terminals of $C'$. Notice that $\sigma_C(uv) \geq \sum_{C' \in A} (t_{C'} - 3)$, because at most three out of $t_{C'}$ terminals is not adjacent to a pipe-edge in $\varphi^{-1}(uv)$ by the definition of safe pipes (equality occurs when a thick edge is incident to $C'$ in a pipe other than $uv$). As such, for a complex component $C$ in $G_{uv}$, the decrease in $\Phi_1$ is $\Omega(|E(C)|)$. Therefore, the first term of (2) is charged to the decrease in $\Phi_1$. This takes care of steps (a) and (c) of the efficient implementation (Section 4.2). It remains to bound the time complexity of step (b), which deals exclusively with simple components.

Second, we show that the time that $\text{pipeExpansion}(uv)$ spends on simple components is absorbed by the decrease in the last term of $\Phi_3$. When we move the components of $L(u, uv)$, $w \notin \{v, w^*(v)\}$, to the cluster $\langle uv \rangle_w$, we spend linear time on all but a maximal set, which can be moved in $O(1)$ time using a set operation. In what follows we show that a constant times the corresponding decrease in the term $\sum_{u \in V(H)} L(u) \log L(u)$ subsumes this work.

We adapt the analysis from the classic heavy path decomposition. Suppose we partition a set of size $k = L(u) = L(v)$ into $\ell$ subsets of sizes $k_1 \geq \ldots \geq k_\ell$. Note that $k_j \leq k/2$ for $j \geq 2$. Then

$$k \log k = \sum_{i=1}^\ell k_i \log k \geq k_1 \log k_1 + \sum_{j=2}^\ell k_j \log(2k_j) = \sum_{i=1}^\ell k_i \log k_1 + \sum_{j=2}^\ell k_j.$$ 

Hence, the decrease in $\sum_{u \in V(H)} L(u) \log L(u)$, which is equal to $k \log k - \sum_{i=1}^\ell k_i \log k_i$, is bounded from below by $k - k_1$. Therefore if we spend $O(1)$ time on a maximal subset of size $k_1$, we can afford to spend linear time on all other subsets. Thus, the decrease in $\sum_{u \in V(H)} L(u) \log L(u)$ subsumes the actual work and this concludes the proof.

This completes the proof of Theorem 1(i). Part (ii) of Theorem 1 is shown in Section 5.

5 Constructing an embedding

Our recognition algorithm in Section 4 decides in $O(m \log m)$ time whether a given instance $\varphi$ is a weak embedding. However, if $\varphi$ turns out to be a weak embedding, it does not provide an
embedding $\psi_\varphi$, since at the end of the algorithm we have an equivalent “reduced” instance $\varphi'$ at hand. In this section, we show how to compute the combinatorial representation of an embedding $\psi_\varphi$ for the input $\varphi$.

Assume that $\varphi : G \to H$ is a weak embedding. By Lemmas 11 and 13 we can obtain a combinatorial representation of an embedding $\pi_{\varphi'} \in \Pi(\varphi')$ in $O(n \log n)$ time of the instance $\varphi' : G' \to H'$ produced by the algorithm at the end of Phase 2.

We sequentially reverse the steps of the algorithm, and maintain a combinatorial embeddings for all intermediate instances until we obtain a combinatorial representation $\pi_\varphi \in \Pi(\varphi)$. By Lemma 2 we can then obtain an embedding $\psi_\varphi : G \to \mathcal{H}$ in $O(m)$ time. Reversing a clusterExpansion($u$) operation is trivial: the total orders of pipes in $\Delta_u$ can be ignored and the total order for the pipes $uv$ are the same as orders for $u,v$. This can be done in $O(\deg(u))$ time.

Let $\varphi^{(1)} : G^{(1)} \to H^{(1)}$ be the input instance of pipeExpansion($uv$), $\varphi^{(2)} : G^{(1)} \to H^{(2)}$ be the instance obtained by contracting $uv$ and $\varphi^{(3)} : G^{(3)} \to H^{(3)}$ be the instance after clusterExpansion($\langle uv \rangle$). By the previous argument, the total orders of pipe-edges $\pi_{\varphi^{(2)}}(\langle uv \rangle w)$ of all pipes $\langle uv \rangle w$ can be obtained from a combinatorial representation $\pi_{\varphi^{(3)}}(\langle uv \rangle w)$ in $O(\deg(\langle uv \rangle w))$ time. These orders also correspond to $\pi_{\varphi^{(1)}}(uv)$ and $\pi_{\varphi^{(1)}}(vx)$ for pipes $uv$ and $vx$ in $\varphi^{(1)}$ where $v \neq u$ and $x \neq u$.

In order to obtain an order $\pi_{\varphi^{(1)}}(uv)$, we embed $G^{(1)}_{\langle uv \rangle}$ in $D_{\langle uv \rangle}$ using Lemma 2 in $O(|E(G^{(1)}_{\langle uv \rangle})|)$ time and find the Jordan curve defined in the proof of Lemma 6. The order $\pi_{\varphi^{(1)}}(uv)$ of the pipe-edges in $\varphi^{-1}[uv]$ is given by the order in which the Jordan curve intersects these edges. This takes $O(|E(G^{(1)}_{\langle uv \rangle})|)$ time by Theorem 2 though we first need to obtain an embedding of $G^{(1)}$, where triangles of wheels in $G^{(1)}_{\langle uv \rangle}$ and 4-cycles induced by thick edges in and incident to $G^{(1)}_{\langle uv \rangle}$ are empty. This can be done by changing the rotation at the vertices of the wheels and 4-cycles corresponding to thick edges one by one in $O(|E(G^{(1)}_{\langle uv \rangle})|)$ time, since $uv$ is safe in the resulting instance.

However, this would lead to a $O(m^2)$ worst case time complexity because of simple components. We show how to reduce the running time to $O(m \log m)$. Let us call a pipe-edge (thick edge) simple if it connects two simple components in two adjacent clusters. In each total order $\pi_{\varphi^{(2)}}(uv)$ for a pipe $uv$ in an instance $\varphi^*$, we arrange maximal blocks of consecutive simple edges into a bundle that takes a single position in the order, and store the order among the simple edges in the bundle in a separate linked list. We can substitute each bundle with one representative simple component. Then, using $\pi_{\varphi^{(3)}}(\langle uv \rangle w)$ for all pipes $\langle uv \rangle w$ incident to $\langle uv \rangle w$ we can obtain a list of at most $\deg(\langle uv \rangle w) + |C|$ representative simple components, where $C$ is the list of complex components in $G_{\langle uv \rangle}$. We can proceed by embedding all complex components in $C$ and the representatives of simple components in $D_{\langle uv \rangle}$. Obtaining a Jordan curve that encloses all vertices in $(\varphi^{(1)})^{-1}[u]$ now takes $O(\text{complex}(G_{\langle uv \rangle}) + \deg(\langle uv \rangle w))$ time. The order in which the Jordan curve crosses the edges in $G_{\langle uv \rangle}$ defines $\pi_{\varphi^{(1)}}(uv)$, where the size of $\pi_{\varphi^{(1)}}(uv)$ is $O(\text{complex}(G_{\langle uv \rangle}) + \deg(\langle uv \rangle w))$ and each pipe-edge obtained from a representative simple component represents a bundle of simple edges. We can merge consecutive bundles of simple edges in $\pi_{\varphi^{(1)}}(uv)$, as needed, in $O(\text{complex}(G_{\langle uv \rangle}) + \deg(\langle uv \rangle w))$ time. By [H4], this running time is bounded above by the running time of our implementation of pipeExpansion($uv$), as argued in the proof of Lemma 13. Therefore, we can reverse every operation and obtain an embedding $\psi_\varphi : G \to \mathcal{H}$ in $O(m \log m)$ time. This completes the proof of Theorem [Hii].
6 Algorithm for nonorientable surfaces

We show that our algorithm can be adapted to recognize weak embeddings $\varphi : G \to H$ when $H$ is embedded in a nonorientable manifold $M$. We discuss the adaptation to nonorientable surfaces in a separate section to reduce notational clutter in Sections 2–4.

First, we adapt the definition of the strips system. The embedding of a graph $H$ into a (orientable or nonorientable) surface $M$ is naturally inherited from the values of $\varphi$. For every edge $e = uv$, if $\lambda(e) = -1$, we identify $A_{u,v}$ with $\partial R_{uv}$ via an orientation reversing homeomorphism and $A_{v,u}$ with $\partial R_{uv}$ via an orientation preserving homeomorphism, or vice-versa. If we represent $M$ as a sphere with a finite number holes, that are turned into cross-caps, the signature of an edge is interpreted as the parity of the number of times an edge passes through a cross-cap.

Second, we adapt the operation of cluster expansion as follows. We put $\lambda(u,v) := \lambda(uv)$ for all neighbors $v$ of $u$, and $\lambda(e) := 1$ for all newly created edges $e$.

Third, we adapt the operation pipeExpansion$uv$ as follows. If $\lambda(uv) = -1$, before creating the cluster $(uv)$, we flip the value of $\lambda$ from $-1$ to $1$, and vice-versa, for every edge of $H$ adjacent to $u$. This corresponds to pushing the edge $uv$ off all the cross-caps that it passes through. Then the values of $\lambda$ on the edges incident to $(uv)$ in $H^*$ are naturally inherited from the values of $\lambda$ on the edges adjacent to $u$ and $v$ in $H$. The value of $\lambda$ for all other edges in $H^*$ remain the same as in $H$.

The first two phases of the algorithm remain the same except that they use the adapted operations of cluster and pipe expansion. Phase 3 is changed as follows. The changes affect only the (thick) cycles $C$ such that $\Pi_{e \in E(\varphi(C))} \lambda(e) = -1$, or in other words cycles $C$, for which the strip system of $\varphi(C)$ is homeomorphic to the Möbius band. For such thick cycles $C$, we report that the instance is negative if the underlying graph of $C$ is planar. If a cycle $C$ in $G''$ satisfies $\Pi_{e \in E(\varphi''(C))} \lambda(e) = 1$, we proceed as in the orientable case. Otherwise, we report that the instance is negative if $C$ winds more than two times around $\varphi''(C)$; or there exist two distinct cycles $C_1$ and $C_2$ in $G''$ winding once around $\varphi''(C)$ such that $\varphi''(C) = \varphi(C_1) = \varphi(C_2)$ in $H''$. Else we can report that $\varphi$ is a weak embedding at the end.

7 Conclusions

We have shown (Theorem 1) that it takes $O(m \log m)$ time to decide whether a piecewise linear simplicial map $\varphi : G \to H$ from an abstract graph $G$ with $m$ edges to an embedded graph $H$ is a weak embedding (i.e., whether, for every $\varepsilon > 0$, there exists an embedding $\psi : G \to \mathcal{H}$ of $G$ into a neighborhood $\mathcal{H}$ of $H$ such that $\|\varphi - \psi\| < \varepsilon$). The only previously known algorithm for this problem takes $O(n^{2\omega}) \leq O(n^{4.75})$ time, where $\omega$ is the matrix multiplication constant [14], and until recently no polynomial-time algorithm was available even in the special case that $H$ is embedded in the plane. Only the trivial lower bound of $\Omega(m)$ is known for the time complexity of recognizing weak embeddings in our setting. Closing the gap between $\Omega(m)$ and $O(m \log m)$ remains open.

If $\varphi : G \to H$ is a continuous map, but not necessarily simplicial (i.e., an image of an edge may pass through vertices of $H$), then the running time increases to $O(mn \log mn)$, where $n = |V(H)|$ (cf. Corollary 1). In the special case that $G$ is a cycle and $H$ is a planar straight-line graph, an $O(m \log m)$-time algorithm was recently given in [1]. It remains an open problem whether a similar improvement is possible for arbitrary $G$ and $H$. 

24
An interesting generalization of our problem is deciding whether a given 2-dimensional simplicial complex embeds into some 3-dimensional manifold, also known as the **thickenability of 2-dimensional simplicial complexes**. Indeed, [21, Lemma\(^3\)] implies that a polynomial-time algorithm for the thickenability of 2-dimensional simplicial complexes would directly translates into a polynomial-time algorithm for our problem. Studying this more general problem is one of the next natural steps in our investigation: Currently, we do not know whether this problem is tractable. Furthermore, due to the same argument a polynomial-time algorithm for deciding whether a 2-dimensional simplicial complex embeds in \(\mathbb{R}^3\) would already imply a polynomial-time algorithm for our problem if we restrict ourselves to orientable surfaces. However, this problem is NP-hard [8], so the existence of such an algorithm is highly unlikely.

**References**

[1] Hugo A. Akitaya, Greg Aloupis, Jeff Erickson, and Csaba D. Tóth. Recognizing weakly simple polygons. *Discrete Comput. Geom.*, to appear. Special issue dedicated to the best papers from the 32nd Symposium on Computational Geometry (SoCG 2016).

[2] Patrizio Angelini, Giordano Da Lozzo, Giuseppe Di Battista, and Fabrizio Frati. Strip planarity testing for embedded planar graphs. *Algorithmica*, 77(4):1022–1059, 2017.

[3] Patrizio Angelini and Giordano Da Lozzo. Clustered planarity with pipes. In Seok-Hee Hong, editor, *Proc. 27th Internat. Symp. on Algorithms and Computation (ISAAC)*, volume 64 of *LIPIcs*, pages 13:1–13:13. Schloss Dagstuhl, 2016.

[4] S. B. Belyi. Self-nonintersecting and non intersecting chains. *Mathematical notes of the Academy of Sciences of the USSR*, 34(4):802–804, 1983. Translated from Matematicheskie Zametki, Vol. 34, No. 4, pp. 625–628, 1983.

[5] Henning Bruhn and Reinhard Diestel. MacLane’s theorem for arbitrary surfaces. *J. Combin. Theory Ser. B*, 99:275–286, 2009.

[6] Hsien-Chih Chang, Jeff Erickson, and Chao Xu. Detecting weakly simple polygons. In *Proc. 26th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1655–1670, 2015.

[7] Pier Francesco Cortese, Giuseppe Di Battista, Maurizio Patrignani, and Maurizio Pizzonia. On embedding a cycle in a plane graph. *Discrete Math.*, 309(7):1856–1869, 2009.

[8] Arnaud de Mesmay, Yoav Rieck, Eric Sedgwick, and Martin Tancer. Embeddability in \(\mathbb{R}^3\) is NP-hard. Preprint, [arXiv:1708.07734](https://arxiv.org/abs/1708.07734), 2017.

[9] Giuseppe Di Battista and Roberto Tamassia. On-line planarity testing. *SIAM J. Comput.*, 25:956–997, 1996.

[10] Reinhard Diestel. *Graph theory*, volume 173 of *Graduate Texts in Mathematics*. Springer, Heidelberg, fifth edition, 2017.

---

\(^3\)The Lemma is stated only for connected graphs and only in the case when the target surface is a sphere. However, it is easy to see that an analogous statement holds for disconnected graphs and for orientable surfaces of arbitrary genus.
[11] Qing-Wen Feng, Robert F. Cohen, and Peter Eades. How to draw a planar clustered graph. In Proc. 1st Conference on Computing and combinatorics (COCOON), volume 959 of LNCS, pages 21–30. Springer, Berlin, 1995.

[12] Qing-Wen Feng, Robert F. Cohen, and Peter Eades. Planarity for clustered graphs. In Proc. 3rd European Symposium on Algorithms (ESA), volume 979 of LNCS, pages 213–226, Berlin, 1995. Springer.

[13] Radoslav Fulek. Embedding graphs into embedded graphs. Preprint, [arXiv:1608.02087], 2016.

[14] Radoslav Fulek and Jan Kynčl. Hanani–Tutte for approximating maps of graphs. Preprint, [arXiv:1705.05243], 2017.

[15] Haim Hanani. Über wesentlich un-plättbare Kurven im drei-dimensionalen Raume. Fundamenta Mathematicae, 23:135–142, 1934.

[16] John Hopcroft and Robert Tarjan. Efficient planarity testing. J. ACM, 21:549–568, 1974.

[17] Michael Jünger, Sebastian Leipert, and Petra Mutzel. Level planarity testing in linear time. In Proc. 6th Symposium on Graph Drawing (GD), volume 1547 of LNCS, pages 224–237. Springer, Berlin, 1998.

[18] Bojan Mohar. A linear time algorithm for embedding graphs in an arbitrary surface. SIAM J. Discrete Math., 12(1):6–26, 1999.

[19] Dušan Repovš and Arkadij B. Skopenkov. A deleted product criterion for approximability of maps by embeddings. Topology Appl., 87(1):1–19, 1998.

[20] Karol Sieklucki. Realization of mappings. Fund. Math., 65:325–343, 1969.

[21] Arkadiy B. Skopenkov. A generalization of Neuwirth’s theorem on thickening 2-dimensional polyhedra. Mathematical Notes, 58(5):1244–1247, 1995.

[22] Mikhail Skopenkov. On approximability by embeddings of cycles in the plane. Topology Appl., 134(1):1–22, 2003.

[23] Carsten Thomassen. The graph genus problem is NP-complete. J. Algorithms, 10(4):568–576, 1989.

[24] William T. Tutte. Toward a theory of crossing numbers. J. Combin. Theory, 8:45–53, 1970.