Dynamic Planar Graph Isomorphism Is in DynFO

Samir Datta
Chennai Mathematical Institute & UMI ReLaX, Chennai, India

Asif Khan
Chennai Mathematical Institute, India

Felix Tschirbs
Ruhr University Bochum, Germany

Nils Vortmeier
Ruhr University Bochum, Germany

Thomas Zeume
Ruhr University Bochum, Germany

Abstract

Consider two planar graphs which are subject to edge insertions and deletions. We show that whether the two graphs are isomorphic can be maintained with first-order logic formulas and auxiliary data of polynomial size. This places the dynamic planar graph isomorphism problem into the dynamic descriptive complexity class DynFO. As a consequence, there is a dynamic constant-time parallel algorithm with polynomial-size auxiliary data which maintains whether two dynamic planar graphs are isomorphic.

2012 ACM Subject Classification Theory of computation → Complexity theory and logic

Keywords and phrases Dynamic complexity theory, parallel computation, dynamic algorithms

Digital Object Identifier 10.4230/LIPIcs.LICS.2026.36

Related Version *Full Version*: `http://arxiv.org/abs/2604.22365`

Funding Samir Datta and Asif Khan are partially funded by a grant from Infosys foundation. Felix Tschirbs, Nils Vortmeier and Thomas Zeume are supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation), grant 532727578.

1 Introduction

The *graph isomorphism problem* asks whether two given graphs are isomorphic, i.e., whether there is a bijection between their vertices that preserves their edges. It is one of the few algorithmic problems known to be in NP, but not known to be either in P or NP-complete. In fact, it is only known that the problem can be solved in quasipolynomial time [1] and that it is at least as hard as computing determinants [28] (i.e., it is not even known to be hard for P).

While it is open whether the graph isomorphism problem can be solved in polynomial time for general graphs, several restricted classes of graphs have been identified for which polynomial time or even logarithmic space suffices. For instance, early on it was shown that isomorphism of trees is complete for logarithmic space [22], which was generalized in subsequent work to graphs of bounded tree width [4, 16, 6, 14], planar graphs [10], and graphs of bounded genus [13]. In particular, the graph isomorphism problem on these graph classes can be solved efficiently in parallel. It can be solved by a CRCW PRAM with $n^{O(1)}$ processors in $O(\log n)$ time or equivalently by AC^1-circuits, i.e., circuits with $\neg$-gates and unbounded fan-in $\wedge$- and $\vee$-gates of polynomial size and logarithmic depth. From a logical perspective this means that the graph isomorphism problem for these restricted graph classes can be expressed by first-order formulas of quantifier-depth $\mathcal{O}(\log n)$ [2].

In this work we study the isomorphism problem for graphs that change dynamically over time, i.e., graphs where edges may be inserted or deleted. Our guiding question is whether less than $\mathcal{O}(\log n)$ parallel time suffices to solve the graph isomorphism problem for dynamic graphs for restricted graph classes. Almost three decades ago, Etessami made a first step into this direction by proving that the dynamic graph isomorphism problem for trees can be maintained in constant parallel time by AC^0-circuits with auxiliary data of polynomial size [15]. Later Mehta showed that also 3-connected planar isomorphism can be maintained in constant parallel time, assuming the graph stays 3-connected at all times [23].

In this work, we generalize both results and establish that the graph isomorphism problem for planar graphs can be maintained in constant parallel time.

▶ **Theorem** (Main theorem). *The dynamic graph isomorphism problem for planar graphs can be maintained by* AC^0*-circuits with auxiliary data of polynomial size.*

Similar to Etessami's and Mehta's results, the changing graphs are promised to remain planar and one bit of the auxiliary data indicates whether the graphs are isomorphic after each change.

For proving the result, like Etessami and Mehta, we work in the dynamic descriptive complexity framework introduced by Patnaik and Immerman [24, 25]. In this framework, the input graphs as well as the auxiliary data are represented by relational structures. After each change, all auxiliary relations are updated by first-order formulas with the changed edge and a tuple of nodes as parameters. The tuple of nodes will be included into an updated auxiliary relation if the formula evaluates to true in the relational auxiliary structure before the change. The problems maintainable in this way by first-order formulas constitute the dynamic descriptive complexity class DynFO and coincide with the problems maintainable by AC^0-circuits with auxiliary data of polynomial size (due to the correspondence of AC^0 and first-order logic, see [2]). In their proofs, Etessami and Mehta[1] established that dynamic tree isomorphism and dynamic 3-connected planar isomorphism are in DynFO.

▶ **Theorem** (Main theorem, rephrased). *The dynamic graph isomorphism problem for planar graphs is in* DynFO*.*

This result continues the exploration of the power of first-order logic as an update mechanism for dynamic problems, which intensified with the positive resolution of a conjecture by Patnaik and Immerman that directed graph reachability is in DynFO [9]. This and further results of the last decade – including, for instance, that MSO-definable algorithmic problems on bounded treewidth are in DynFO [11] and that testing planarity and maintaining plane embeddings are in DynFO [7] – are proof that DynFO or dynamic constant-parallel time, respectively, is surprisingly powerful. However, while DynFO contains NL-complete problems like directed graph reachability and artificial P-complete problems [25], there are also simple queries like "Is the number of nodes that have an edge to a red node odd?", which is definable in first-order logic extended by parity quantifiers, which are not known to be in DynFO [30].

Outline. We recall basic notions for graphs and dynamic complexity theory in Section 2. After outlining high-level proof ideas in Section 3 and discussing the structural impact of edge changes on planar graphs in Section 4, we provide a more detailed proof sketch in Section 5. More proof details are in the full version.

[1] Mehta showed membership in the class DynFO^+ that allows for polynomial-time initialization.

2 Preliminaries and notations

We recall elementary notions from graph theory based on [10].

Basic graph notions. In this work, all graphs $G = (V, E)$, where V is the set of vertices (also called nodes) and E is the set of edges, are simple and undirected. For a vertex $v \in V$, the neighbourhood $N(v) = \{u \mid (v, u) \in E\}$ contains all vertices adjacent to v. The graph induced by a subset $X \subseteq V(G)$ is denoted by $G[X]$ and $G - X$ denotes the graph $G[V(G) \setminus X]$. Two graphs $G = (V, E)$ and $G^* = (V^*, E^*)$ are *isomorphic*, denoted by $G \simeq G^*$, if there is an isomorphism from G to G^*, i.e., a bijection $\pi : V \to V^*$ such that $(u, v) \in E$ if and only if $(\pi(u), \pi(v)) \in E^*$ for all $u, v \in V$.

A graph is *planar* if it can be drawn on the plane such that edges do not cross except at their endpoints. Connected regions in a drawing of a planar graph are called *faces* [12].

A *(undirected) tree* is a graph in which there is exactly one path between any pair of nodes; a forest is a disjoint union of trees. Given a tree T, we denote by $\mathrm{ST}(x, r)$ the subtree rooted at r, in the tree T rooted at x. A *(rooted) context* $X = (V, E, r, h)$ is a tree (V, E) with root $r \in V$ and one distinguished leaf $h \in V$, called the *hole*. Two contexts $X = (V, E, r, h)$, $X^* = (V^*, E^*, r^*, h^*)$ are isomorphic if there is a root- and hole-preserving isomorphism between them, i.e., an isomorphism that maps r to r^* and h to h^*. For a forest F and nodes x, r, h occurring in this order on some path, the context $\mathrm{X}(x, r, h)$ is defined as the context we obtain by taking $\mathrm{ST}(x, r)$, removing all children of h, and taking r as root and h as hole.

Connectivity and decomposition trees. A graph is *connected* if there is a path between any two vertices u and v. A subset $S \subseteq V$ with $|S| = k$ is a *k-separating* set if $G - S$ is not connected. Such a set S is a *separating vertex* or *cut vertex* if $k = 1$ and a *separating pair* if $k = 2$. A graph G is k-connected if it has no $(k-1)$-separating set. If a graph is 2-connected, we also call it *biconnected*.

A biconnected subgraph $C = G[U]$ is a *biconnected component* of G if U is an inclusion-maximal set of nodes such that $G[U]$ is biconnected. The decomposition of a connected graph into its biconnected components can be given as a tree:

▶ **Definition 1** (Biconnected component tree). *The* biconnected component tree *bi-tree*(G) *of a connected graph G has a node for each biconnected component and each cut vertex of G. There is an edge in bi-tree*(G) *between the node for a biconnected component C and the node for a cut vertex u if u occurs in C.*

We further aim to divide biconnected components (that are not just single edges). A separating pair $\{u, v\}$ of a biconnected graph G is a *3-connected separating pair* if there are three vertex-disjoint paths between u and v in G. Let $U \subseteq V$ be an inclusion-maximal set such that $G[U]$ is not separated by any 3-connected separating pair, that is, $G[U \setminus \{u, v\}]$ is connected for any 3-connected separating pair $\{u, v\}$. The *triconnected component* associated with U is obtained from $G[U]$ by adding an edge (u, v) for each 3-connected separating pair $\{u, v\}$ of G with $u, v \in U$, if it is not already present. Such added edges are called *virtual edges*. A triconnected component is either 3-connected or a (chordless) cycle and we call every graph *triconnected* if it is either 3-connected or a cycle [20, p. 137].

▶ **Definition 2** (Triconnected component tree). *The* triconnected component tree *tri-tree*(G) *of a biconnected graph G has a node for each triconnected component and for each 3-connected separating pair of G. There is an edge between the node for a triconnected component C and the node for a 3-connected separating pair $\{u, v\}$, if u and v both occur in C.*

Both the biconnected component tree of a connected graph and the triconnected component tree of a biconnected graph are indeed trees and unique (up to isomorphism). Triconnected component trees are also known under the name SPQR trees [3, 17].

If X is a context of a biconnected or triconnected component tree T for a graph G, we denote by $\text{graph}(X)$ the subgraph of G induced by the set of vertices represented in X, that is, that occur as a cut vertex or in a biconnected component respectively in a separating pair or a triconnected component represented in X.

We also use the notion of *coherent paths* defined in [7].

▶ **Definition 3** (Coherent paths). *Let C_1, C_2 be two triconnected components of a triconnected component tree tri-tree(G) and let a_1, a_2 be vertices of C_1, C_2, respectively. The path between C_1, C_2 in tri-tree(G) is an* (a_1, a_2)-coherent path *if $G + \{(a_1, a_2)\}$ is planar.*

We also denote the (a_1, a_2)-coherent path from C_1 to C_2 by $\rho(C_1, C_2, (a_1, a_2))$. Often, we do not mention the vertices a_1, a_2 explicitly and say that the path between C_1, C_2 in tri-tree(G) is a *coherent path* if there are a_1, a_2 such that the path is (a_1, a_2)-coherent.

Note that the path between triconnected components C_1, C_2 may be non-coherent: for example, if C_1 and C_2 need to be embedded into different faces of another triconnected component, any edge insertion between C_1 and C_2 destroys planarity.

Dynamic complexity. The goal of dynamic (descriptive) complexity theory is to study the complexity of maintaining the answer to a query where the input structure is subject to changes [25]. In this paper, the input structure is a planar graph G and the change operations are insertions INS_e and deletions DEL_e of edges e such that G remains planar.

A *dynamic program* Π stores the input structure as well as a set $\mathcal{A}$ of auxiliary relations over some (relational) schema σ_{aux} and over the same domain as the input structure. For every auxiliary relation symbol $R \in \sigma_{\text{aux}}$ and change operations $\delta \in \{\text{INS}, \text{DEL}\}$, the program Π has an *update formula* $\varphi^R_\delta(\bar{x}; e)$, which can access input and auxiliary relations, as well as the changed edge e. After a change δ_e, an auxiliary relation $R^{\mathcal{A}}$ is defined by $\{\bar{x} \mid (\bar{x}, e) \models \varphi^R_\delta\}$.

A dynamic program Π *maintains* a dynamic query Q, if after applying a sequence α of changes to an initial input structure $\mathcal{I}_0$ with empty relations and applying the corresponding update formulas to $(\mathcal{I}_0, \mathcal{A}_0)$, where $\mathcal{A}_0$ is an initial auxiliary structure, a dedicated auxiliary relation always equals the result Q on the input structure.

The class DynFO contains all dynamic queries that are maintained by a dynamic program with $\mathsf{FO}(\leq, +, \times)$ formulas[2] as update formulas and first-order definable initialization of $\mathcal{A}_0$.

3 Proof overview

The inspiration for our dynamic program for planar isomorphism comes from algorithms that test planar isomorphism in linear time [19] and logarithmic space [10]. These algorithms decompose given graphs G and G^* into their biconnected and triconnected component trees. They then proceed by identifying pairs of isomorphic triconnected components; isomorphic biconnected components via their triconnected component trees; and finally isomorphic connected components via their biconnected component trees.

[2] Using a linear order $\leq$, we can identify the vertex set V with an initial segment of the natural numbers. The relations $+$ and $\times$ provide addition and multiplication on these numbers. Assuming the existence of these relations does not increase the expressive power of DynFO, see [9, Proposition 7]. We use that $\mathsf{FO}(\leq, +, \times)$ corresponds to uniform AC^0 [2].

Our dynamic program follows this decomposition and maintains

(1) isomorphisms between triconnected components,

(2) which biconnected components are isomorphic by maintaining isomorphism information for their triconnected component trees, and

(3) which connected components are isomorphic by maintaining isomorphism information for their biconnected component trees.

Our main theorem follows: the input to our dynamic program is the disjoint union of G and G^*; it holds $G \simeq G^*$ if these connected components of the input graph are isomorphic.

(1) Isomorphisms between triconnected components. Crucial for maintaining isomorphisms between triconnected components is that 3-connected planar graphs have a unique embedding (up to reflection and choice of outer face) in the plane [32]. This fact has been used for deciding isomorphism between two 3-connected planar graphs in various settings, see [31, 27]. Under the promise that graphs remain 3-connected, Mehta used their unique embeddings to maintain whether two 3-connected planar graphs are isomorphic by maintaining canonical breadth first search trees [23].

Without the promise that graphs remain 3-connected, there are changes that (i) do not change the triconnected component tree and only modify existing 3-connected components, and changes that (ii) change the triconnected component tree, e.g., by merging triconnected components along a path of the tree into a new 3-connected component. Mehta's approach handles (i), but not (ii).

We use, instead, Tutte's canonical spring embedding [29] which is obtained by fixing three arbitrary vertices on a common face of the graph, putting identical springs between adjacent pairs of vertices, and computing an equilibrium. Tutte showed that this equilibrium yields a canonical plane embedding of 3-connected planar graphs, called the *Tutte embedding*. To check whether two 3-connected planar graphs are isomorphic, their Tutte embeddings can be compared for all choices of three vertices (cf. [21]).

Our dynamic algorithm maintains representations of Tutte embeddings for all 3-connected components as well as for all such components that arise from edge insertions that merge multiple triconnected components along a path of the triconnected component tree into a new 3-connected component. The latter helps to deal with changes of type (ii). For maintaining embeddings, we use that coordinates of graph vertices in the plane – $\mathbf{x} \in \mathbb{Q}^n$ and $\mathbf{y} \in \mathbb{Q}^n$ for the x and y-coordinates of the n vertices – can be computed by solving a system of linear equations of the form $\mathbf{T}\,\mathbf{x} = \mathbf{b}$ and $\mathbf{T}\,\mathbf{y} = \mathbf{b}$. The *Tutte matrix* $\mathbf{T}$ is almost the Laplacian of the graph, except that the three rows corresponding to the fixed vertices have 1 at diagonal entries and 0 elsewhere. For solving these systems, it is sufficient to maintain the inverse of $\mathbf{T}$ and then compute $\mathbf{T}^{-1}\mathbf{b}$ (exploiting that $\mathbf{b}$ only has a constant number of non-zero entries).

We use that edge insertions and deletions in the graph can be translated to the following operations on Tutte matrices:

(a) Modifying constantly many entries of a Tutte matrix $\mathbf{T}$, or

(b) Combining two Tutte matrices $\mathbf{T}_1$ and $\mathbf{T}_2$ into a new Tutte matrix $\mathbf{T}$ in an overlapped block diagonal fashion with small overlap.

We show that the inverses of Tutte matrices can be maintained under both kind of changes using the Sherman-Morrison-Woodbury identity (see, e.g., [18]).

A technical issue in this approach is that entries of $\mathbf{T}^{-1}$ may be large, and hence the arithmetic operations required may not be carried out in $\mathsf{FO}(\leq, +, \times)$. Therefore we maintain inverses modulo polynomially many small ($\mathcal{O}(\log n)$ bits) primes and compute the Chinese

remainder representation of the positions of the vertices in the Tutte embedding. During updates, primes may become invalid because $\mathbf{T}^{-1}$ does not exist for them. For this reason we refresh available primes regularly using a "muddling technique" from [11].

From this information we can infer whether any two 3-connected components C and C^* of the graph are isomorphic, as well as all possible isomorphisms between them if they are.

(2) Isomorphic biconnected components. For maintaining isomorphic biconnected components, we exploit that two biconnected components are isomorphic if and only if there is an isomorphism between their triconnected component trees that maps triconnected components to triconnected components and separating pairs to separating pairs in such a way that

- triconnected components C are mapped to isomorphic triconnected components C^*;
- separating pairs $\{u_1, v_1\}, \ldots, \{u_k, v_k\}$ of C are mapped to separating pairs $\{u_1^*, v_1^*\} \stackrel{\text{def}}{=} \pi(\{u_1, v_1\}), \ldots, \{u_k^*, v_k^*\} \stackrel{\text{def}}{=} \pi(\{u_k, v_k\})$ of C^* such that the subtree rooted at $\{u_i, v_i\}$ in the triconnected component tree is isomorphic to the subtree rooted at $\{u_i^*, v_i^*\}$, for all $i \in \{1, \ldots, k\}$; and
- the isomorphism from C to C^* is compatible with the mapping π of the separating pairs.

Thus, we have to maintain whether (a) triconnected component trees are isomorphic, and (b) their isomorphisms can be "extended" to isomorphisms of the underlying graphs. We extend the dynamic algorithm for tree isomorphism due to Datta et al. [8] – a variant of Etessami's algorithm [15] – for (a), but exploit our knowledge of isomorphisms between triconnected components for (b).

The dynamic algorithm for tree isomorphism by Datta et al. maintains auxiliary relations

- X-ISO that stores tuples $(X, X^*) \stackrel{\text{def}}{=} (x, r, h, x^*, r^*, h^*)$ such that the contexts $X \stackrel{\text{def}}{=} \mathrm{X}(x, r, h)$ and $X^* \stackrel{\text{def}}{=} \mathrm{X}(x^*, r^*, h^*)$ are isomorphic,
- #ISO-SIBLINGS that stores tuples (x, r, y, m) such that y has m isomorphic siblings within the subtree $\mathrm{ST}(x, r)$, and
- DIST that stores tuples (x, y, d) such that the distance between nodes x and y is d.

For maintaining isomorphic biconnected components, we maintain similar relations for the triconnected component trees but ensure that both contexts within those trees as well as their induced subgraphs within the biconnected components are isomorphic.

The main challenge is that the triconnected component tree can change significantly. For instance, a triconnected component node can unfurl into a path of triconnected component nodes and separating pair nodes, when an edge is deleted.[3]

For updating whether contexts X and X^* are isomorphic where one of the contexts, say X, contains parts of such an unfurled path, we follow Etessami's approach and match subtrees of nodes along this path with their counterparts in X^* using the distance information (see below). Isomorphism of these unchanged subtrees can be verified using the stored auxiliary data. A difference to tree isomorphism is that we also need to map the separating pair nodes, for which, a priori, there are two possible ways for each pair $\{u, v\}$ and $\{u^*, v^*\}$ which might lead to exponentially many possible mappings if the path contains many such pairs. Fortunately, the unfurled path is a coherent path which implies that if $\{u_1, v_1\}$ and $\{u_2, v_2\}$ are adjacent separating pairs on the path, then the mapping of $\{u_1, v_1\}$ to $\{u_1^*, v_1^*\}$ determines the mapping of $\{u_2, v_2\}$ to $\{u_2^*, v_2^*\}$. Thus, the mapping of the first separating pair on the unfurled path determines the mappings for all subsequent pairs. To exploit this, we use auxiliary information from [7] for determining this mapping (see Section 5.2).

[3] Similarly, such a path of biconnected component nodes and separating pair nodes can merge into a single triconnected component node when an edge is inserted.

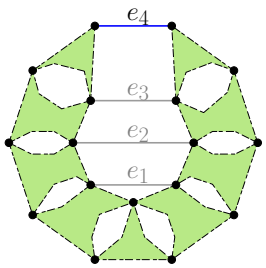


Figure 1 A 3-connected component that unfurls into a path of 3-connected components after deleting e_4.

For updating distance information when unfurling a path, it is necessary to maintain additional information for triconnected components. To see this, consider the 3-connected component in Figure 1, which remains 3-connected when deleting the edges e_1, e_2, and e_3; and unfurls into a path only when also deleting e_4. When the path unfurls, distance information for this path needs to be available immediately. A careful examination of the separating pair vertices of the unfurled path just before the deletion of an edge e reveals that they lie on a disk formed by the faces adjacent to e (see Figure 4 and Section 5.2).

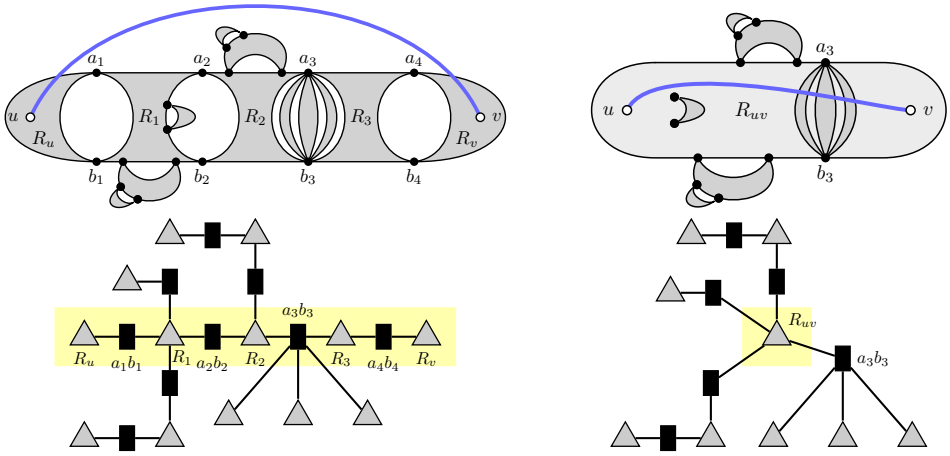
Technically, we show that isomorphism can be maintained even for coloured biconnected components, where colours need to be observed as well. Considering coloured biconnected components is helpful for maintaining isomorphic connected components. Intuitively, two nodes of a biconnected component will have the same colour, if they are cut vertices in the biconnected component tree of a connected component that have isomorphic subtrees.

(3) Isomorphic connected components. For maintaining isomorphic connected components, similarly to biconnected components, we exploit that two connected components are isomorphic if and only if there is an isomorphism between their biconnected component trees that maps biconnected components to biconnected components and cut vertices to cut vertices such that

- biconnected components C are mapped to isomorphic biconnected components C^*;
- cut vertices $v_1, \dots, v_k$ of C are mapped to cut vertices $v_1^* \stackrel{\text{def}}{=} \pi(v_1), \dots, v_k^* \stackrel{\text{def}}{=} \pi(v_k)$ of C^* such that the subtree rooted at v_i in the biconnected component tree is isomorphic to the subtree rooted at v_i^*, for all $i \in \{1, \dots, k\}$; and
- some isomorphism from C to C^* is compatible with the mapping π of the cut vertices.

Thus, we have to maintain whether (a) the biconnected component trees are isomorphic, and (b) the isomorphism can be "extended" to an isomorphism of the underlying graphs. We again build on the dynamic algorithm for tree isomorphism and maintain auxiliary information for isomorphic contexts, distances, and number of isomorphic siblings in the biconnected component tree. Again, a challenge is to handle when an edge deletion unfurls a biconnected component into a path of biconnected components and cut vertices. However, updating isomorphic contexts is easier to handle as there is only one mapping from a cut vertex v to its corresponding cut vertex v^* (as opposed to potentially two such mappings from separating pairs $\{u, v\}$ to $\{u^*, v^*\}$ in the case of the triconnected component trees). Also, maintaining distances is easier in this case.

One complication arises from the fact that there may be many ways to isomorphically map a biconnected component C to a biconnected component C^*. For triconnected components (see (2)), there is a unique mapping (up to reflection and choice of outer face) which can be inferred from the Tutte information. For biconnected components, we cannot deduce such



(a) Before the insertion. **(b)** After the insertion.

Figure 2 Illustration of the effect of inserting an edge (u, v) of type $[2 \xrightarrow{+} 3]$ on a graph and its triconnected component tree. In the triconnected component tree, the path from R_u to R_v is merged into a new triconnected component node R_{uv}.

a mapping from the isomorphism information maintained for biconnected components. To ensure that a biconnected component C can be mapped to a biconnected component C^* such that cut vertices v are mapped to cut vertices v^* with isomorphic subtrees, we colour cut vertices of C and C^* with their subtree isomorphism types. If there is an isomorphism between such coloured biconnected components then this isomorphism maps cut vertices v to cut vertices v^* with isomorphic subtrees.

4 Effects of Changes on Planar Graph Decompositions

When an edge of a planar graph G is modified, the decomposition of G into its biconnected and triconnected component trees can change significantly. In this section, building on [7, Definition 8], we systematically discuss the structural impact of edge changes.

The effect of modifying an edge (u, v) depends on whether (i) the edge (u, v) is inserted or deleted, (ii) the vertices u and v are in the same k-connected component before the change, for $k \leq 3$, and whether (iii) u and v are in the same k-connected component after the change, for $k \leq 3$. We denote the *type* of an edge insertion (respectively edge deletion) with $[k \xrightarrow{+} k']$ (respectively $[k \xrightarrow{-} k']$), where $k, k' \in \{0, \ldots, 3\}$ are the maximal numbers such that the two affected vertices are in the same k-connected component before and in the same k'-connected component after the change.

Depending on the type of a change, the change may affect only triconnected component trees, only biconnected component trees, or both.

Changes only relevant for the triconnected component tree. The following types of changes only affect the triconnected components, any biconnected component trees remain unchanged.

$[3 \xrightarrow{+} 3]$. The vertices u and v belong to a common 3-connected component before and after the edge change. In this case, the triconnected component tree remains unchanged.

[2 $\xrightarrow{+}$ 3]. The nodes u and v lie in distinct triconnected components R_u and R_v in the same biconnected component of G; in the resulting graph G' they are in a common 3-connected component R_{uv}. See Figure 2 for an illustration.

Let the R_u to R_v path[4] in tri-tree(G) be $R_uP_1R_1P_2\cdots P_kR_kP_{k+1}R_v$, where the R_i are triconnected component nodes and the P_i are 3-connected separating pair nodes. We first assume that all nodes R_i represent 3-connected components (and not cycle components). After the insertion, all 3-connected components on the path merge into a 3-connected component R_{uv}. The new triconnected component tree tri-tree(G') is obtained from tri-tree(G) by replacing the R_u to R_v path by a new node for R_{uv}. To this end, the separating pair nodes of degree 2 and all triconnected component nodes are replaced by R_{uv}; all remaining separating pair nodes of degree more than 2 are connected by an edge to the new node R_{uv}.

If any R_i on the R_u to R_v path in tri-tree(G) is a cycle component, let $P_i = \{a_1, b_1\}$ and $P_{i+1} = \{a_2, b_2\}$ be the two separating pairs adjacent to R_i on the path and let $(a_1, \ldots, a_2, b_2, \ldots, b_1)$ be the cyclic order of the vertices in the cycle. In G', the vertices a_1, a_2, b_1, b_2 get absorbed into the merged 3-connected component R_{uv}; the cycle R_i is split into two cycles $C_u = (a_1, \ldots, a_2)$ and $C_v = (b_2, \ldots, b_1)$ with corresponding triconnected component nodes in tri-tree(G'). New separating pair nodes for $\{a_1, a_2\}$ and $\{b_1, b_2\}$ are introduced, which connect R_{uv} to C_u and to C_v, respectively. Any separating pair node that is attached to R_i in tri-tree(G) but does not lie on the path from R_u to R_v gets attached to either C_u or C_v in tri-tree(G'), depending on whether the separating pair nodes lie in C_u or C_v. In case R_u (resp. R_v) itself is a cycle component, the cycle R_u (resp. R_v) is to be split into possibly two cycles, analogous to a cycle component that lies internally on the R_u to R_v path, by insertion of a chord from u (resp. v) to each vertex of P_1 (resp. P_{k+1}).

[2 $\xrightarrow{+}$ 2]. If u and v belong to the same 2-connected component but not to the same 3-connected component before and after the insertion of the edge (u, v), they belong to a common cycle component C in G (otherwise this would be a change of type [2 $\xrightarrow{+}$ 3]). The triconnected component tree tri-tree(G') is obtained from tri-tree(G) by replacing the triconnected component node of C with a path $C_1P_{uv}C_2$, where P_{uv} corresponds to the new separating pair $\{u, v\}$ and C_1, C_2 correspond to the two cycles obtained by splitting C along the chord (u, v). Each neighbour P of C in tri-tree(G) is made a neighbour of either C_1 or C_2 in tri-tree(G'), depending on the location of the vertices of the separating pair P.

Edge changes of types [3 $\xrightarrow{-}$ 3] and [3 $\xrightarrow{-}$ 2], [2 $\xrightarrow{-}$ 2] reverse edge changes of type [3 $\xrightarrow{+}$ 3], [2 $\xrightarrow{+}$ 3], and [2 $\xrightarrow{+}$ 2]. Effects on the decompositions are also reversed.

Changes affecting both component trees. Changes that create or destroy biconnected components affect both the triconnected and biconnected component trees.

[1 $\xrightarrow{+}$ 2]. Before the insertion of (u, v), the vertices u and v belong to distinct biconnected components B_u and B_v; after the insertion, u and v end up in a common biconnected component B_{uv}.

Let $(B_u = B_0)c_1B_1c_2\cdots c_{k-1}B_{k-1}c_k(B_k = B_v)$ be the B_u to B_v path in bi-tree(G), where the B_i are biconnected component nodes and the c_i are cut vertex nodes. After the insertion, all the biconnected components on this path merge into one biconnected component B_{uv}. The new biconnected component tree bi-tree(G') is obtained from bi-tree(G) by replacing

[4] If u or v are in multiple triconnected components, R_u and R_v are defined such that they are the only triconnected components on the path that contain u and v, respectively. This choice is unique.

the B_u to B_v path by a new node for B_{uv}. To this end, the cut vertex nodes of degree 2 and all biconnected component nodes are replaced by B_{uv}; all remaining cut vertex nodes of degree more than 2 are connected by an edge to the new node B_{uv}.

The triconnected component tree for the newly formed biconnected component B_{uv} is obtained from the triconnected component trees of the old biconnected components that merge into B_{uv} as follows. Let $c_0 = u, c_1, \ldots, c_k, c_{k+1} = v$ be the list of cut vertex nodes between B_u and B_v in the order they appear, preceded by u and followed by v. Every pair $\{c_i, c_{i+1}\}$ of consecutive vertices in the list forms a 3-connected separating pair in B_{uv}. In every component B_i the edge (c_i, c_{i+1}) is inserted as virtual edge if it is not present. A new separating pair node for $\{c_i, c_{i+1}\}$ is added to the resulting triconnected component tree for the component, if it not already exists (and if B_i not only consists of the single edge between these nodes), and it becomes a neighbour of the unique triconnected component node whose component contains c_i and c_{i+1}.

The tree tri-tree(B_{uv}) contains all resulting triconnected component trees and an additional new triconnected component node for the cycle that consists of the (potentially virtual) edges (c_i, c_{i+1}) and the inserted edge (u, v). The latter node becomes a neighbour of the separating pair nodes $\{c_i, c_{i+1}\}$.

Edge changes of type $[2 \xrightarrow{-} 1]$ reverse edge changes of type $[1 \xrightarrow{+} 2]$. Effects on the decompositions are also reversed.

Changes only relevant for the biconnected component tree. The following types of changes only affect bi-tree(G), the triconnected component tree tri-tree(G) is unchanged.

$[0 \xrightarrow{+} 2]$. Suppose u and v belong to distinct connected components C_u and C_v before the edge insertion. After the insertion, C_u and C_v merge into one connected component C_{uv}. The biconnected component tree bi-tree(C_{uv}) of C_{uv} is obtained from bi-tree(C_u) and bi-tree(C_v) as follows. It contains a node B_{uv} for the trivial biconnected component that only consists of the edge (u, v). If u is already a cut vertex node in bi-tree(C_u), this cut vertex node is connected to B_{uv}. Otherwise, first a new cut vertex node for u is introduced to bi-tree(C_u) as a neighbour to the unique biconnected component node whose component includes u. Analogously, the cut vertex node for v from bi-tree(C_v) is connected to B_{uv}, potentially after introducing that cut vertex node.

Edge changes of type $[2 \xrightarrow{-} 0]$ reverse edge changes of type $[0 \xrightarrow{+} 2]$. Effects on the decompositions are also reversed.

5 Maintaining Planar Graph Isomorphism

In this section we show our main result.

▶ **Theorem** (Main theorem, rephrased)**.** *The dynamic graph isomorphism problem for planar graphs is in* DynFO*.*

Our proof follows the proof overview provided in Section 3. Our dynamic program represents connected components, biconnected components, and triconnected components by tuples of vertices of the underlying graph. Biconnected and triconnected component trees are then maintained as auxiliary relations.

▶ **Lemma 4.** *Biconnected (resp. triconnected) component trees for connected (resp. biconnected) components of arbitrary graphs can be maintained in* DynFO*.*

Proof sketch. Reachability and connectivity in undirected graphs can be maintained in DynFO [25]. By maintaining connectivity information when ignoring a constant number k of vertices and their edges, for all choices of k vertices, also k-connectivity can be maintained. From this information we can identify k-connected components, cut vertices and 3-connected separating pairs in FO. ◀

We show that isomorphisms between triconnected components can be maintained in Section 5.1, and that isomorphic biconnected and connected components can be maintained in Sections 5.2 and 5.3.

5.1 Maintaining Isomorphisms between Triconnected Components

We show how to maintain isomorphisms between triconnected components of planar graphs.

▶ **Proposition 5.** *Let G be a planar graph. The relation $\textsc{iso}_3$ that contains all tuples $(a, b, c, d, a^*, b^*, c^*, d^*)$ such that*

- *a, b, c, d and a^*, b^*, c^*, d^* are vertices of triconnected components C and C^* of G, respectively,*
- *C and C^* are isomorphic via an isomorphism π_{abc} fixed by $\pi_{abc}(a) = a^*, \pi_{abc}(b) = b^*$, $\pi_{abc}(c) = c^*$, and*
- *$\pi_{abc}(d) = d^*$ (i.e., π_{abc} can also be accessed via $\textsc{iso}_3$).*

can be maintained in DynFO *under insertions and deletions of edges, provided that G stays planar.*

For triconnected components C, C^* that are cycles, this follows as distances are maintained for them (see Lemma 9).

For the proof for 3-connected components, we maintain Tutte embeddings for all tuples (a, b, c) and (a^*, b^*, c^*). Finding isomorphic components then boils down to checking that two such embeddings are equal and the induced isomorphism maps a vertex to the vertex of the other component with the same position in the embedding.

We formalize Tutte embeddings and the auxiliary data required to maintain them in Section 5.1.1. In Section 5.1.2 we introduce linear algebra tools for maintaining the data under several operations and use them to prove Proposition 5 in Section 5.1.3.

5.1.1 The Tutte embedding and maintained auxiliary data

We make the definition of the Tutte embedding more formal. Assuming three vertices $v_{i_1}, v_{i_2}, v_{i_3}$ of the input graph G are pinned and the edges are replaced by springs, we consider the forces that act on the non-pinned vertices through the springs. At equilibrium, every vertex is stationary and hence the forces acting on it must all sum to zero. We obtain the following system of equations for the first coordinate x_i of each vertex v_i of G (and a second, analogous system for the second component, which we omit in the following), where the first coordinates of the pinned vertices $v_{i_1}, v_{i_2}, v_{i_3}$ are $p_{i_1}, p_{i_2}, p_{i_3}$, respectively:

$$\deg(v_i)x_i = \sum_{(v_i,v_j)\in E} x_j \qquad \forall v_i \notin \{v_{i_1}, v_{i_2}, v_{i_3}\}$$

$$x_i = p_i \qquad \forall v_i \in \{v_{i_1}, v_{i_2}, v_{i_3}\}$$

Writing the equations in linear algebraic terms, we have $\mathbf{T}\,\mathbf{x} = \mathbf{b}$ where $\mathbf{T} = \mathbf{T}(G, v_{i_1}, v_{i_2}, v_{i_3})$ is a square matrix with entries

$$T_{ij} = \begin{cases} -1 & \text{if } i \neq j, (v_i, v_j) \in E \text{ and } v_i \notin \{v_{i_1}, v_{i_2}, v_{i_3}\} \\ \deg(v_i) & \text{if } i = j \text{ and } v_i \notin \{v_{i_1}, v_{i_2}, v_{i_3}\} \\ 1 & \text{if } i = j \text{ and } v_i \in \{v_{i_1}, v_{i_2}, v_{i_3}\} \\ 0 & \text{otherwise} \end{cases}$$

and $\mathbf{b}$ is such that

$$b_i = \begin{cases} 0 & \text{if } v_i \notin \{v_{i_1}, v_{i_2}, v_{i_3}\} \\ p_i & \text{if } v_i \in \{v_{i_1}, v_{i_2}, v_{i_3}\}. \end{cases}$$

The matrix $\mathbf{T}$ is very similar to the *Laplacian* $\mathbf{L}$ of the graph G, which has the entries $L_{ii} = \deg(v_i)$ for all i and $L_{ij} = -1$ if $(v_i, v_j) \in E$ and 0 otherwise for all $i \neq j$. So, $\mathbf{T}$ is obtained from $\mathbf{L}$ by replacing the rows for the pinned vertices $v_{i_1}, v_{i_2}, v_{i_3}$ with the row that has 1 at the diagonal entry and 0 otherwise. Note that $\mathbf{L}$ is symmetric, but $\mathbf{T}$ is not.

Our goal is to maintain the inverse $\mathbf{T}^{-1}$ of $\mathbf{T}$. From this, we obtain $\mathbf{x}$ as $\mathbf{x} = \mathbf{T}^{-1}\,\mathbf{b}$. As $\mathbf{b}$ has only three non-zero values, this computation can be performed in $\mathsf{FO}(\leq, +, \times)$, assuming that $\mathbf{T}^{-1}$ is available in a suitable form and all values are polynomially bounded.

Note that $\mathbf{T}^{-1}$ is a matrix over $\mathbb{Q}$. It follows that $\mathbf{x}$ is also over the rationals if $p_{i_1}, p_{i_2}, p_{i_3}$ are rational numbers. As $\mathbf{T}^{-1}$ may have entries $\frac{a}{b}$ that involve exponentially large numbers, we maintain $\mathbf{T}^{-1}$ modulo many polynomially large primes.

We now present the auxiliary information that is maintained. Let G be a graph of size n (or a subgraph of a graph of size n) and let $v_{i_1}, v_{i_2}, v_{i_3}$ be three of its vertices. Let p be a prime that is polynomially bounded in n. Let $\mathbf{L}$ be the Laplacian of G and let $\mathbf{T} = \mathbf{T}(G, v_{i_1}, v_{i_2}, v_{i_3})$ be the matrix as explained above with respect to G and $v_{i_1}, v_{i_2}, v_{i_3}$. We call $(G, v_{i_1}, v_{i_2}, v_{i_3})$ *embedding-inducing* if $\mathbf{T}^{-1}$ exists and $(G, v_{i_1}, v_{i_2}, v_{i_3}, p)$ *modulo embedding-inducing* if $\mathbf{T} \bmod p = \mathbf{T}(G, v_{i_1}, v_{i_2}, v_{i_3}, p)$ has an inverse over $\mathbb{Z}_p$.

We often use the following fact.

▶ **Lemma 6.** *If G is a 3-connected planar graph and $v_{i_1}, v_{i_2}, v_{i_3}$ are on the same face then $(G, v_{i_1}, v_{i_2}, v_{i_3})$ is embedding-inducing.*

Proof. A matrix $\mathbf{A}$ is *weakly chained diagonally dominant* if it is weakly diagonally dominant, has strictly diagonally dominant rows and for every non-strictly diagonally dominant row i there is a strictly diagonally dominant row j and a sequence $A_{ii_1}, A_{i_1i_2}, \ldots A_{i_rj}$ of non-zero entries. Such a matrix is non-singular [26].

For any connected graph G with any vertices $v_{i_1}, v_{i_2}, v_{i_3}$, the matrix $\mathbf{T} = \mathbf{T}(G, v_{i_1}, v_{i_2}, v_{i_3})$ is weakly chained diagonally dominant: the rows of $v_{i_1}, v_{i_2}, v_{i_3}$ are strictly diagonally dominant and from every vertex of G, at least one of the nodes $v_{i_1}, v_{i_2}, v_{i_3}$ is reachable without using an outgoing edge of v_{i_1}, v_{i_2}, or v_{i_3}. ◀

Note that it can be maintained in DynFO whether three vertices of a 3-connected planar graph lie on a common face [7].

For a modulo embedding-inducing $(G, v_{i_1}, v_{i_2}, v_{i_3}, p)$, where G contains n vertices, let $\mathcal{M}(G, v_{i_1}, v_{i_2}, v_{i_3}, p)$ contain the following information:

1. the inverse $\mathbf{T}^{-1} \in \mathbb{Z}_p^{n \times n}$,
2. the products $\mathbf{T}^{-1}\,\mathbf{c} \in \mathbb{Z}_p^n$ and $\mathbf{c}^\mathsf{T}\,\mathbf{T}^{-1} \in \mathbb{Z}_p^{1 \times n}$ for each column $\mathbf{c}$ of the Laplacian $\mathbf{L}$,
3. the products $\mathbf{c}_1^\mathsf{T}\,\mathbf{T}^{-1}\,\mathbf{c}_2 \in \mathbb{Z}_p$ for each pair $\mathbf{c}_1, \mathbf{c}_2$ of columns of $\mathbf{L}$.

In the remainder of this section, we will show how to maintain $\mathcal{M}(G, v_{i_1}, v_{i_2}, v_{i_3}, p)$ under various operations and that this suffices to maintain isomorphism of 3-connected components.

5.1.2 Tools for Maintaining Tutte auxiliary information

We discuss now how $\mathcal{M}(G, v_{i_1}, v_{i_2}, v_{i_3}, p)$ can be maintained using $\mathsf{FO}(\leq, +, \times)$ formulas under operations like edge changes, exchanging of pinned vertices and merging of graphs (to be defined later), provided the inverses continue to exist. Subsection 5.1.3 uses these insights for maintaining isomorphisms for 3-connected components in DynFO.

Large parts of our reasoning rely on the Sherman-Morrison-Woodbury (SMW) formula, see [18], which explains how to obtain the inverse of a matrix that is the result of changing an $n \times n$ matrix $\mathbf{A}$ by adding the product $\mathbf{U}\mathbf{V}^\mathsf{T}$ of two arbitrary $n \times k$ matrices $\mathbf{U}$ and $\mathbf{V}$, as long as the original and the new matrix are invertible:

$$(\mathbf{A} + \mathbf{U}\mathbf{V}^\mathsf{T})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}(\mathbf{I} + \mathbf{V}^\mathsf{T}\mathbf{A}^{-1}\mathbf{U})^{-1}\mathbf{V}^\mathsf{T}\mathbf{A}^{-1} \tag{1}$$

In this formula, $\mathbf{I}$ denotes the $k \times k$ identity matrix.

Given the matrix $\mathbf{T}$ for an embedding-inducing $(G, v_{i_1}, v_{i_2}, v_{i_3})$, inserting an edge (v_i, v_j) leads to the change of four entries of $\mathbf{T}$, assuming that v_i, v_j are neither of $v_{i_1}, v_{i_2}, v_{i_3}$: the diagonal entries T_{ii} and T_{jj} are incremented, the entries T_{ij} and T_{ji} are decremented. The new matrix $\mathbf{T}'$ can be expressed as $\mathbf{T} + \mathbf{a}\mathbf{a}^\mathsf{T}$, where $\mathbf{a}$ is the (column) vector with entries $a_i = 1$, $a_j = -1$ and 0 everywhere else.

▶ **Lemma 7.** *Fix a domain of size n. Let G be a graph with at most n vertices, $v_{i_1}, v_{i_2}, v_{i_3}$ three of its vertices, and p a prime of magnitude $\mathcal{O}(n^c)$, for some constant c, such that $(G, v_{i_1}, v_{i_2}, v_{i_3}, p)$ is modulo embedding-inducing. If G' results from G by inserting or deleting some edge, then given $\mathcal{M}(G, v_{i_1}, v_{i_2}, v_{i_3}, p)$ one can determine whether $(G', v_{i_1}, v_{i_2}, v_{i_3}, p)$ is also modulo embedding-inducing in $\mathsf{FO}(\leq, +, \times)$ and if so, $\mathcal{M}(G', v_{i_1}, v_{i_2}, v_{i_3}, p)$ can be defined.*

Proof. Suppose that $(G, v_{i_1}, v_{i_2}, v_{i_3}, p)$ is modulo embedding-inducing and that G' results from G by inserting the edge (v_i, v_j). Let $\mathbf{T} \stackrel{\text{def}}{=} \mathbf{T}(G, v_{i_1}, v_{i_2}, v_{i_3}, p)$ and let $\mathbf{a}$ be the vector whose only non-zero entries are the entries $a_i = 1$ and $a_j = -1$. We assume that v_i, v_j are neither of $v_{i_1}, v_{i_2}, v_{i_3}$, as for these vertices the rows of $\mathbf{T}$ do not change. Then $\mathbf{T} + \mathbf{a}\mathbf{a}^\mathsf{T} = \mathbf{T}' = \mathbf{T}(G', v_{i_1}, v_{i_2}, v_{i_3}, p)$. According to the SMW formula it holds

$$(\mathbf{T} + \mathbf{a}\mathbf{a}^\mathsf{T})^{-1} = \mathbf{T}^{-1} - \mathbf{T}^{-1}\mathbf{a}(\mathbf{I} + \mathbf{a}^\mathsf{T}\mathbf{T}^{-1}\mathbf{a})^{-1}\mathbf{a}^\mathsf{T}\mathbf{T}^{-1}$$

and $(G', v_{i_1}, v_{i_2}, v_{i_3}, p)$ is modulo embedding-inducing, that is, $\mathbf{T}'$ is invertible over $\mathbb{Z}_p$, exactly if $\mathbf{I} + \mathbf{a}^\mathsf{T}\mathbf{T}^{-1}\mathbf{a}$ is invertible, that is, has a non-zero determinant. As $\mathbf{a}$ has only two non-zero entries, computing the scalar $\mathbf{a}^\mathsf{T}\mathbf{T}^{-1}\mathbf{a} \in \mathbb{Z}_p$ involves only a constant number of additions and multiplications of constantly many numbers, and therefore is possible in $\mathsf{FO}(\leq, +, \times)$. Then, it can be determined whether $1 + \mathbf{a}^\mathsf{T}\mathbf{T}^{-1}\mathbf{a} \equiv 0 \pmod{p}$ holds. Also, the column vector $\mathbf{T}^{-1}\mathbf{a}$ and the row vector $\mathbf{a}^\mathsf{T}\mathbf{T}^{-1}$ can be computed, and then the full right-hand side of the SMW formula can be evaluated.

It remains to explain how the products $\mathbf{T}'^{-1}\,\mathbf{c}, \mathbf{c}^\mathsf{T}\,\mathbf{T}'^{-1}$ and $\mathbf{c}_1^\mathsf{T}\,\mathbf{T}'^{-1}\,\mathbf{c}_2$ can be computed, for all columns $\mathbf{c}, \mathbf{c}_1, \mathbf{c}_2$ of the Laplacian $\mathbf{L}'$ of G'. For the products of the form $\mathbf{c}_1^\mathsf{T}\,\mathbf{T}'^{-1}\,\mathbf{c}_2$, we observe

$$\mathbf{c}_1^\mathsf{T}\,(\mathbf{T} + \mathbf{a}\mathbf{a}^\mathsf{T})^{-1}\,\mathbf{c}_2 = \mathbf{c}_1^\mathsf{T}\,\mathbf{T}^{-1}\,\mathbf{c}_2 - \mathbf{c}_1^\mathsf{T}\,\mathbf{T}^{-1}\mathbf{a}(\mathbf{I} + \mathbf{a}^\mathsf{T}\mathbf{T}^{-1}\mathbf{a})^{-1}\mathbf{a}^\mathsf{T}\mathbf{T}^{-1}\,\mathbf{c}_2$$

If $\mathbf{c}_1$ and $\mathbf{c}_2$ are columns of $\mathbf{L}'$ that are equal in $\mathbf{L}$, the additionally required partial products are contained in $\mathcal{M}(G, v_{i_1}, v_{i_2}, v_{i_3}, p)$. If the columns are different from the respective columns $\mathbf{d}_1, \mathbf{d}_2$ of $\mathbf{L}$ then these columns differ in at most two entries, so $\mathbf{c}_k = \mathbf{d}_k + \mathbf{\Delta}_k$, where $\mathbf{\Delta}_k$ has at most two non-zero entries, for each $k = 1, 2$. So, we can compute the products as

$$\begin{aligned}\mathbf{c}_1^\mathsf{T}(\mathbf{T}+\mathbf{a}\mathbf{a}^\mathsf{T})^{-1}\mathbf{c}_2 &= (\mathbf{d}_1^\mathsf{T}+\boldsymbol{\Delta}_1^\mathsf{T})\,\mathbf{T}^{-1}\,(\mathbf{d}_2+\boldsymbol{\Delta}_2)\\ &\quad -(\mathbf{d}_1^\mathsf{T}+\boldsymbol{\Delta}_1^\mathsf{T})\,\mathbf{T}^{-1}\mathbf{a}(\mathbf{I}+\mathbf{a}^\mathsf{T}\mathbf{T}^{-1}\mathbf{a})^{-1}\mathbf{a}^\mathsf{T}\mathbf{T}^{-1}\,(\mathbf{d}_2+\boldsymbol{\Delta}_2)\\ &= \mathbf{d}_1^\mathsf{T}\mathbf{T}^{-1}\mathbf{d}_2+\boldsymbol{\Delta}_1^\mathsf{T}\mathbf{T}^{-1}\mathbf{d}_2+\mathbf{d}_1^\mathsf{T}\mathbf{T}^{-1}\boldsymbol{\Delta}_2+\boldsymbol{\Delta}_1^\mathsf{T}\mathbf{T}^{-1}\boldsymbol{\Delta}_2\\ &\quad -\mathbf{d}_1^\mathsf{T}\,\mathbf{T}^{-1}\mathbf{a}(\mathbf{I}+\mathbf{a}^\mathsf{T}\mathbf{T}^{-1}\mathbf{a})^{-1}\mathbf{a}^\mathsf{T}\mathbf{T}^{-1}\,\mathbf{d}_2\\ &\quad -\boldsymbol{\Delta}_1^\mathsf{T}\,\mathbf{T}^{-1}\mathbf{a}(\mathbf{I}+\mathbf{a}^\mathsf{T}\mathbf{T}^{-1}\mathbf{a})^{-1}\mathbf{a}^\mathsf{T}\mathbf{T}^{-1}\,\mathbf{d}_2\\ &\quad -\mathbf{d}_1^\mathsf{T}\,\mathbf{T}^{-1}\mathbf{a}(\mathbf{I}+\mathbf{a}^\mathsf{T}\mathbf{T}^{-1}\mathbf{a})^{-1}\mathbf{a}^\mathsf{T}\mathbf{T}^{-1}\,\boldsymbol{\Delta}_2\\ &\quad -\boldsymbol{\Delta}_1^\mathsf{T}\,\mathbf{T}^{-1}\mathbf{a}(\mathbf{I}+\mathbf{a}^\mathsf{T}\mathbf{T}^{-1}\mathbf{a})^{-1}\mathbf{a}^\mathsf{T}\mathbf{T}^{-1}\,\boldsymbol{\Delta}_2\end{aligned}$$

where the products containing $\mathbf{d}_k$ are contained in $\mathcal{M}(G, v_{i_1}, v_{i_2}, v_{i_3}, p)$ and the products involving $\boldsymbol{\Delta}_k$ can be computed in $\mathsf{FO}(\leq, +, \times)$, as these vectors have a constant number of non-zero entries and thus the matrix product requires a constant number of additions.

The remaining products can be computed analogously.

In case of an edge deletion, the new matrix $\mathbf{T}'$ can be expressed as $\mathbf{T} - \mathbf{a}\mathbf{a}^\mathsf{T}$ and the approach is analogous. ◀

Other operations on $\mathcal{M}(G, v_{i_1}, v_{i_2}, v_{i_3}, p)$ can be implemented similarly, see the full version for details, for instance,

- exchanging pinned vertices, i.e., given $\mathcal{M}(G, v_{i_1}, v_{i_2}, v_{i_3}, p)$ for some pinned vertices $v_{i_1}, v_{i_2}, v_{i_3}$, one can determine $\mathcal{M}(G, v'_{i_1}, v'_{i_2}, v'_{i_3}, p)$ (if it exists) for arbitrary other pinned vertices $v'_{i_1}, v'_{i_2}, v'_{i_3}$,
- merging of components, i.e., given $\mathcal{M}(G_1, v_{i_1}, v_{i_2}, v_{i_3}, p)$ and $\mathcal{M}(G_2, v_{i_1}, v_{i_2}, v_{i_4}, p)$ for graphs G_1 and G_2 sharing a separating pair $\{v_{i_1}, v_{i_2}\}$, for the graph G obtained from the union of G_1 and G_2 by adding the edge (v_{i_3}, v_{i_4}) one can determine $\mathcal{M}(G, v_{i_1}, v_{i_3}, v_{i_4}, p)$ (if it exists), and
- splitting of components, i.e., the reverse of the previous operation.

5.1.3 Maintaining Tutte information for coherent paths

We now show that the information $\mathcal{M}(H, v_{i_1}, v_{i_2}, v_{i_3}, p)$ can be maintained for 3-connected components H of planar graphs, as long as the involved inverses exist modulo the prime p. To do so, we also need this information for 3-connected components induced by coherent paths in triconnected component trees, since edge insertions may merge paths of such trees into one 3-connected component.

Let $\rho = \rho(C_1, C_2, (a_1, a_2))$ be a ((a_1, a_2)-) coherent path. By $G[\rho]$ we denote the 3-connected component that is created by the $[2 \xrightarrow{+} 3]$ insertion of the edge (a_1, a_2).

▶ **Lemma 8.** *Let G be a planar graph with n vertices and let p be a prime of magnitude $\mathcal{O}(n^c)$, for some constant c. Assume that $\mathcal{M}(H, v_{i_1}, v_{i_2}, v_{i_3}, p)$ is available for every $H, v_{i_1}, v_{i_2}, v_{i_3}$ where H is a*

- *3-connected components of G, or*
- *a graph $G[\rho]$ induced by a coherent path ρ in tri-tree(G)*

and $v_{i_1}, v_{i_2}, v_{i_3}$ are vertices on the same face of H.

Let G' be a planar graph that results from G by inserting or deleting some edge. Suppose H' is any 3-connected component of G' or equal to $G'[\rho']$ for any coherent path ρ' in tri-tree(G'), and $v'_{i_1}, v'_{i_2}, v'_{i_3}$ are any of its vertices that are on the same face.

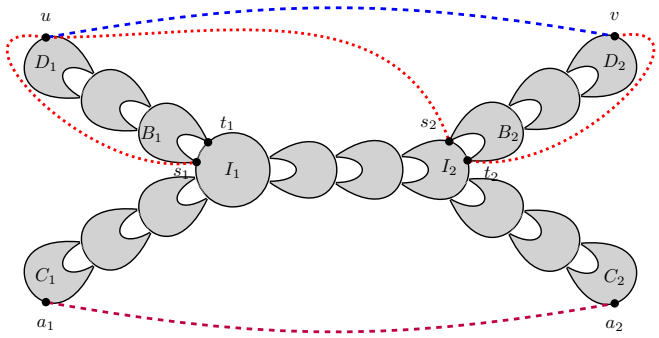


Figure 3 Illustration of Case $[2 \xrightarrow{+} 3]$ in Lemma 8's proof.

There is a constant number of embedding-inducing $(H_j, v_1^j, v_2^j, v_3^j)_{j \le d}$, where the H_j are 3-connected planar graphs and v_1^j, v_2^j, v_3^j are vertices that are on the same face of H_j, such that one can determine in $\mathsf{FO}(\le, +, \times)$ whether all of $(H', v'_{i_1}, v'_{i_2}, v'_{i_3}, p)$ and $(H_j, v_1^j, v_2^j, v_3^j, p)_{j \le d}$ are modulo embedding-inducing and if so, $\mathcal{M}(H', v'_{i_1}, v'_{i_2}, v'_{i_3}, p)$ can be defined (using $\mathcal{M}(H_j, v_1^j, v_2^j, v_3^j, p)$).

Proof. The proof distinguishes all edge type changes from Section 4 where the triconnected component tree is affected. We sketch Case $[2 \xrightarrow{+} 3]$ here, more details are in the full version.

Consider an edge change of type $[2 \xrightarrow{+} 3]$ and suppose G' results from G by insertion of the edge (u, v), where u and v are in different triconnected components D_1, D_2 on a coherent path $\rho(D_1, D_2, (u, v))$ in the triconnected component tree tri-tree(G) of G. Accordingly, the components of that path coalesce into a single 3-connected component in the triconnected component tree of G'.

We show how to maintain the auxiliary information for a coherent path $\rho(C_1, C_2, (a_1, a_2))$ in tri-tree(G') between some components C_1 and C_2 that include the vertices a_1, a_2, respectively. We assume that the coalesced component lies on $\rho(C_1, C_2, (a_1, a_2))$ in tri-tree(G'), as otherwise no update is necessary.

We only consider the general case that C_1, C_2 are not the newly coalesced component, otherwise the following explanations can easily be adapted. It follows that $\rho(C_1, C_2, (a_1, a_2))$ is also a coherent path in tri-tree(G). See Figure 3 for a sketch.

The coherent path $\rho(D_1, D_2, (u, v))$ in G can be divided into three subpaths: the intersection with the coherent path $\rho(C_1, C_2, (a_1, a_2))$ and the parts before and after the intersection. In Figure 3, these are (1) the path between the components D_1 and B_1, (2) the path between I_1 and I_2 and (3) the path between B_2 and D_2.

We assume here that the paths from D_1 to B_1 and from D_2 to B_2 are both not a single cycle component but either a single 3-connected component or a non-trivial path consisting of triconnected components. Let $\{s_1, t_1\}$ be the separating pair between the components B_1 and I_1 and let $\{s_2, t_2\}$ be the separating pair between the components B_2 and I_2. Let $G_1 = G[\rho(D_1, B_1, (u, s_1))]$ and $G_3 = G[\rho(D_2, B_2, (v, t_2))]$ be the graphs induced by the coherent paths before and after the intersection. Let $G_2 = G[\rho(C_1, C_2, (a_1, a_2))]$ be the 3-connected component associated with the coherent path under consideration in the unchanged graph. According to the assumption of the lemma, $\mathcal{M}(G_1, s_1, t_1, u, p)$, $\mathcal{M}(G_2, s_1, t_1, s_2, p)$ and $\mathcal{M}(G_3, s_2, t_2, v, p)$ are available.

We now explain how to compute $\mathcal{M}(H', v'_{i_1}, v'_{i_2}, v'_{i_3}, p)$, where $H' = G'[\rho(C_1, C_2, (a_1, a_2))]$ and $v'_{i_1}, v'_{i_2}, v'_{i_3}$ are three of its vertices that are on the same face. Notice that H' differs from the union of the three graphs G_1, G_2, G_3 only by a constant number of edges: Let H_1 be the graph that is obtained from the union of G_1 and G_2 by adding the edge (u, s_2) and let H_2 be the graph that is obtained from the union of H_1 and G_3 by adding the edge (u, v). Then H' is obtained from H_2 by removing the edges $(u, s_1), (u, s_2)$ and (v, t_2) (and possibly virtual edges (s_1, t_1) and (s_2, t_2)). According to Lemma 7 and the similar lemmas mentioned above, we can determine in $\mathsf{FO}(\leq, +, \times)$ for each graph of that sequence whether (together with suitably chosen vertices) it is modulo embedding-inducing and, if they all are, $\mathcal{M}(H', v'_{i_1}, v'_{i_2}, v'_{i_3}, p)$ can be obtained.

We give more details. Assuming that every intermediate structure is modulo embedding-inducing, from $\mathcal{M}(G_1, s_1, t_1, u, p)$ and $\mathcal{M}(G_2, s_1, t_1, s_2, p)$ one can obtain $\mathcal{M}(H_1, t_1, u, s_2, p)$ (merging of components) and $\mathcal{M}(H_1, s_2, t_2, u, p)$ (exchanging pinned vertices); then using $\mathcal{M}(G_3, s_2, t_2, v, p)$ one can obtain $\mathcal{M}(H_2, s_2, u, v, p)$ (again merging of components). Applying Lemma 7 three to five times, corresponding to the deletions of the edges $(u, s_1), (u, s_2)$ and (v, t_2) and possibly of virtual edges (s_1, t_1) or (s_2, t_2) if the corresponding vertex set is not a 3-connected separating pair any more in H', one can obtain $\mathcal{M}(H', s_2, u, v, p)$. Finally, one can obtain $\mathcal{M}(H', v'_{i_1}, v'_{i_2}, v'_{i_3}, p)$ (exchanging pinned vertices).

Note that all intermediate graphs are 3-connected planar graphs and their three distinguished vertices are on the same face, so all intermediate structures are embedding-inducing by Lemma 6.

It remains to explain the approach if at least one of the path from D_1 to B_1 or the path from D_2 to B_2 consists only of a single cycle component. We detail the case that the path from D_1 to B_1 is a single cycle component. Then only the vertices u, s_1, t_1 of this cycle are part of the coalesced 3-connected component in G'. In the explanation above we can therefore replace H_1 by the graph that results from G_2 by adding the vertex u and the three edges $(u, s_1), (u, t_1), (u, s_2)$. In the full version we show that we can obtain the required information $\mathcal{M}(H_1, t_1, u, s_2, p)$ also for this graph. ◀

5.1.4 Maintaining Isomorphisms between Triconnected Components

We have seen that information for the Tutte embedding can be maintained modulo primes, as long as the involved inverses exist modulo these primes. To prove Proposition 5, we show how to extract an isomorphism between isomorphic 3-connected components.

Proof of Proposition 5. If C, C^* are cycle components, then they are isomorphic if and only if their lengths are equal. There is an isomorphism that maps the given vertices as given in the proposition statement if the distances between the vertices are pairwise equal. This information is available according to the following lemma, which is proved in the full version of this paper.

▶ **Lemma 9.** *For any biconnected component B of a planar graph G and any cycle component C in the triconnected component tree of B, the distances between any two vertices of C can be maintained in* DynFO*, provided that G stays planar.*

Suppose C, C^* are 3-connected components. As explained in Section 5.1.1, for a 3-connected planar graph H, we can obtain its Tutte embedding with respect to three pinned vertices $v_{i_1}, v_{i_2}, v_{i_3}$ from the inverse of the matrix $\mathbf{T} = \mathbf{T}(G, v_{i_1}, v_{i_2}, v_{i_3})$. To determine whether two 3-connected planar graphs are isomorphic, given three vertices in one graph and the assumed isomorphic copies in the other graph, we compare the obtained embeddings to find matching pairs of vertices in the two graphs.

We face two problems: (1) we do not have $\mathbf{T}^{-1}$ available directly but only $\mathbf{T}^{-1} \bmod p$ for many primes p; and (2) we permanently "lose" the information $\mathbf{T}^{-1} \bmod p$ for a prime p if $(H, v_{i_1}, v_{i_2}, v_{i_3}, p)$ is not modulo embedding-inducing at some point, that is, if the encoded matrix inverses do not exist modulo the prime p.

To solve (1), observe that the entries of $\mathbf{T}$ are bounded by the number n of vertices. As the numbers in any entry $\frac{a}{b}$ in $\mathbf{T}^{-1}$ are bounded by the determinant of $\mathbf{T}$, the numbers in $\mathbf{T}^{-1}$ are at most $n!n^n$, which for large enough n is less than 2^{n^2}. This means that if $\mathbf{T}^{-1} \bmod p$ is available for n^2 many different primes p, we can determine for two embeddings whether they are equal, by checking whether they are equal modulo all n^2 different primes.

For (2), assume that the inverse $\mathbf{T}(H, v_{i_1}, v_{i_2}, v_{i_3})^{-1} \bmod p$ is available for all primes $p \in P$ for some set P, i.e., $\mathcal{M}(H, v_{i_1}, v_{i_2}, v_{i_3}, p)$ is stored as auxiliary information for each $p \in P$. According to Lemma 8, this information can be updated using first-order formulas, provided that (a) for the changed graph H' the structure $(H', v_{i_1}, v_{i_2}, v_{i_3}, p)$ is modulo embedding-inducing, that is, the inverse $\mathbf{T}(H', v_{i_1}, v_{i_2}, v_{i_3})^{-1} \bmod p$ exists, and (b) also a constant number of further matrix inverses exist modulo p. As the respective structures are embedding-inducing, that is, the inverses exist when evaluated over the rationals instead of modulo some prime, the only reason for non-existence of an inverse is that p divides the determinant of some of the involved matrices. As again the determinants are bounded by 2^{n^2}, less than n^2 many primes can divide each determinant, so only for $\mathcal{O}(n^2)$ many primes $p \in P$ the auxiliary information cannot be obtained for the new graph G'. In total we maintain the auxiliary information for $\mathcal{O}(n^6)$ many graphs H (one for each coherent path), in every step the auxiliary information might not be available any more for $\mathcal{O}(n^8)$ many primes.

To address that at some point we "run out" of primes, we use that whenever a graph problem can be maintained for $\log^i(n)$ many steps using first-order formulas, starting from auxiliary relations computed by AC^i circuits, then the problem is in DynFO [11, Theorem 4.2]. Here, AC^i circuits are uniform circuit families of polynomial size and depth $\log^i(n)$, and $i \in \mathbb{N}$ is arbitrary.

The 3-connected components can be identified in $\mathsf{LOGSPACE} \subseteq \mathsf{AC}^1$ [10, Lemma 4.3] and matrix inverses can be computed in $\mathsf{NC}^2 \subseteq \mathsf{AC}^2$ (cf. [5]). Therefore, one can compute in AC^2 the auxiliary information $\mathcal{M}(\cdot)$ for n^{11} many primes, which by the prime number theorem can be found among the first n^{12} many numbers.

According to Lemma 8, we can maintain this information for $\log^2(n)$ many steps, losing $\mathcal{O}(n^8)$ many primes in every step, but still having the information available for n^2 primes in the end.

As reasoned above, this is enough to express the relation $\textsc{iso}_3$ for the case that a, b, c and a^*, b^*, c^* are on the same face in their respective 3-connected components. Otherwise, one can existentially quantify two triples of vertices that are on the same face in their respective components such that an inferred isomorphism maps a to a^*, b to b^* and c to c^*, if such an isomorphism exists. ◀

5.2 Maintaining Isomorphism Information for Biconnected Components

In this section, we show how to maintain which biconnected components of a planar graph are isomorphic. Actually, slightly more generally, we show this result for *coloured graphs*. This will later be helpful for maintaining isomorphic connected components. Vertices in coloured graphs can be coloured with one of polynomially many colours. Formally, we use an additional relation F that for a vertex v may contain a tuple $(v, \bar{f})$, for some tuple $\bar{f}$ of vertices, with the meaning that v has colour $\bar{f}$.

▶ **Proposition 10.** *The relation* ISO_2 *that for a coloured planar graph* G *contains all tuples* (a, b, a^*, b^*) *such that:*

- a, b *and* a^*, b^* *are vertices of biconnected components* B *and* B^* *of* G*, respectively, and*
- B *and* B^* *are isomorphic via an isomorphism* π *with* $\pi(a) = a^*$ *and* $\pi(b) = b^*$

can be maintained in DynFO *under insertions and deletions of edges and changes of vertex colours, provided that* G *stays planar. Additionally, (i) for changes of type* $[1 \xrightarrow{+} 2]$ *and* $[2 \xrightarrow{-} 1]$*, all vertices of an emerging or vanishing cycle component may change their colour, and (ii) all vertices of a colour may be recoloured with an unused colour.*

As described in the proof overview in Section 3, we adapt the context-based dynamic tree isomorphism algorithm by Datta et al. [8]. In triconnected component trees, we specify contexts by tuples $\bar{t}, \bar{r}, \bar{h}$ of two or three graph vertices. Each tuple uniquely represents a triconnected component (if its three vertices are contained in the component) or a separating pair (if its two vertices are the vertices of the pair). For a context $\text{X}(\bar{t}, \bar{r}, \bar{h})$ and a tuple $\bar{f} = (\bar{f}_{\bar{r}}, \bar{f}_{\bar{h}})$ of colours, the *recoloured context* $\text{rcX}(\bar{t}, \bar{r}, \bar{h}, \bar{f})$ is obtained from the (coloured) graph of $\text{X}(\bar{t}, \bar{r}, \bar{h})$ by recolouring the vertices in $\bar{r}$ and $\bar{h}$ as given by $\bar{f}$. Two recoloured contexts $X = \text{rcX}(\bar{t}, \bar{r}, \bar{h}, \bar{f})$ and $X^* = \text{rcX}(\bar{t}^*, \bar{r}^*, \bar{h}^*, \bar{f}^*)$ of the triconnected component trees of G are *fully isomorphic* if the graphs $\text{graph}(X)$ and $\text{graph}(X^*)$ are isomorphic via an isomorphism π such that $\pi(\bar{r}) = \bar{r}^*$ and $\pi(\bar{h}) = \bar{h}^*$. Note that fully isomorphic (recoloured) contexts X and X^* are in particular isomorphic, so, there is a root and hole preserving isomorphism between the (recoloured) contexts. *Recoloured subtrees* $\text{rcST}(\bar{t}, \bar{r}, \bar{f})$ are defined similarly by recolouring the root $\bar{r}$ of a subtree. We often refer to recoloured contexts and subtrees simply as contexts and subtrees.

Our dynamic algorithm for ISO_2 maintains the relations

- X-ISO_2 that stores tuples $(X, X^*) \stackrel{\text{def}}{=} (\bar{x}, \bar{r}, \bar{f}, \bar{h}, \bar{x}^*, \bar{r}^*, \bar{h}^*, \bar{f}^*)$ such that the recoloured contexts $X \stackrel{\text{def}}{=} \text{X}(\bar{x}, \bar{r}, \bar{h}, \bar{f})$ and $X^* \stackrel{\text{def}}{=} \text{X}(\bar{x}^*, \bar{r}^*, \bar{h}^*, \bar{f}^*)$ are fully isomorphic,
- $\#\text{ISO-SIBLINGS}_2$ that stores tuples $(\bar{x}, \bar{r}, \bar{y}, \bar{f}, m)$ such that (i) the triconnected component node $\bar{y}$ is a child of the separating pair node $\bar{r}$, and (ii) the subtree rooted at $\bar{y}$ with nodes from $\bar{r}$ recoloured by $\bar{f}$ has exactly m fully isomorphic siblings,
- DIST_2 that stores tuples $(\bar{x}, \bar{y}, d)$ such that the distance between nodes $\bar{x}, \bar{y}$ of the triconnected component tree is d, and
- SAME-FACE that stores tuples $(\bar{c}_1, \bar{c}_2, a_1, a_2, s_1, s_2)$ such that $\rho(\bar{c}_1, \bar{c}_2, (a_1, a_2))$ is a coherent path and the vertices s_1 and s_2 that appear in two distinct separating pairs on this path are on the same face after inserting the edge (a_1, a_2). The maintenance of this relation is described in [7].

This implies Proposition 10, as ISO_2 is easily definable from X-ISO_2. We next show that X-ISO_2 and $\#\text{ISO-SIBLINGS}_2$ (Section 5.2.1) as well as DIST_2 (Section 5.2.2) can be maintained.

5.2.1 Maintaining Isomorphic Contexts and Sibling Counts

We discuss how X-ISO_2 and $\#\text{ISO-SIBLINGS}_2$ can be maintained.

▶ **Lemma 11.** *The relations* X-ISO_2 *and* $\#\text{ISO-SIBLINGS}_2$ *can be maintained in* DynFO *under insertions and deletions of edges and changes of vertex colours, provided that* G *stays planar. Additionally, (i) for changes of type* $[1 \xrightarrow{+} 2]$ *and* $[2 \xrightarrow{-} 1]$*, all vertices of an emerging or vanishing cycle component may change their colour, and (ii) all vertices of a colour may be recoloured with an unused colour.*

Without loss of generality, we assume that a relation ST-ISO$_2$ is available which contains all pairs $(\bar{t}, \bar{r}, \bar{f}, \bar{t}^*, \bar{r}^*, \bar{f}^*)$ representing fully isomorphic recoloured subtrees, as it can be easily defined from X-ISO$_2$.

An important building block for updating the relations X-ISO$_2$ and #ISO-SIBLINGS$_2$ is that first-order formulas can express whether two contexts rooted at $\bar{r}$ and $\bar{r}^*$ are fully isomorphic, given (i) isomorphisms between the triconnected components of $\bar{r}$ and $\bar{r}^*$ and (ii) the information whether the contexts rooted at the children of $\bar{r}$ and $\bar{r}^*$ are pairwise fully isomorphic. The first information is available thanks to Proposition 5; the latter information either because these contexts were not affected by the change and therefore the corresponding parts of X-ISO$_2$ are still valid, or because it was constructed by a previous application of these first-order formulas.

▷ Claim 12. Let $X = \mathrm{rcX}(\bar{t}, \bar{r}, \bar{h}, \bar{f})$ and $X^* = \mathrm{rcX}(\bar{t}^*, \bar{r}^*, \bar{h}^*, \bar{f}^*)$ be recoloured contexts rooted at the triconnected components R and R^* of $\bar{r}$ and $\bar{r}^*$, respectively. If

- the relation ISO$_3$ and
- the information for each child C of R and each child C^* of R^* whether the subcontext of X rooted at C is fully isomorphic to the subcontext of X^* rooted at C^*

are available, then some FO formula expresses whether X and X^* are fully isomorphic.

Proof. The first-order formula checks that R and R^* are isomorphic via an isomorphism π that maps $\bar{r}$ to $\bar{r}^*$ via ISO$_3$. In particular, it checks whether π respects the colouring of the vertices. Then, for every child separating pair $\{s_1, s_2\}$ of R it checks that $\{\pi(s_1), \pi(s_2)\}$ is a child separating pair of R^* and that the two subcontexts rooted at these separating pairs are fully isomorphic. Finally, it checks that R^* has no further child separating pairs. ◁

▷ Claim 13. Let $X = \mathrm{rcX}(\bar{t}, \bar{r}, \bar{h}, \bar{f})$ and $X^* = \mathrm{rcX}(\bar{t}^*, \bar{r}^*, \bar{h}^*, \bar{f}^*)$ be recoloured contexts that are rooted at separating pairs S and S^* described by $\bar{r}$ and $\bar{r}^*$, respectively. If

- the number of isomorphic siblings of each child of S and S^*, and
- the information whether the subcontext of X rooted at C is fully isomorphic to the subcontext of X^* rooted at C^* ,for each child C of S and each child C^* of S^*,

are available, then some FO formula expresses whether X and X^* are fully isomorphic.

Proof. The first-order formula checks for each child triconnected component of S whether there is a child triconnected component of S^* such that the respective subcontexts rooted at these components are fully isomorphic, that the number of isomorphic siblings in the respective triconnected component trees are the same, and that S^* has no child triconnected component whose subcontext is not fully isomorphic to the subcontext of a child of S. ◁

Proof of Lemma 11. We use the two facts above to express whether two contexts are fully isomorphic after a change. To this end, all types of changes have to be explored. Here, we only discuss changes of types $[3 \xrightarrow{+/-} 3]$ and $[3 \xrightarrow{-} 2]$ that affect only one of the contexts. The remaining cases are discussed in the full version.

Case $[3 \xrightarrow{+/-} 3]$. Let C be the changed 3-connected component in the context X. Let $\{c_1, c_2\}$ be the parent separating pair of C in X and let c_3 be another graph vertex in C. We distinguish two cases, depending on the position of C relative to $\bar{h}$.

First suppose that C is on the path from $\bar{h}$ to $\bar{r}$ in X. The information in #ISO-SIBLINGS$_2$ for $\{c_1, c_2\}$ can be updated by a first-order formula as follows. If C was isomorphic to any other child triconnected component of $\{c_1, c_2\}$ before the change, the counts for these children can be decremented by one. Because the isomorphism information for children of C is not affected by the change, according to Claim 12, a first-order formula can check whether

any other child of $\{c_1, c_2\}$ is isomorphic to C after the change, increment the counts for the corresponding children by one and copy the resulting count for C. Any further update of the relation #ISO-SIBLINGS$_2$ can be done along the same lines as for the children of $\{c_1, c_2\}$.

To update X-ISO$_2$ for X and X^*, a first-order formula can test whether X and X^* are fully isomorphic after the change by first existentially quantifying vertices $c_1^*, c_2^*, c_3^* = \bar{c}^*$, where $\{c_1^*, c_2^*\}$ is the parent separating pair of the component C^* of $\bar{c}^*$ that is the isomorphic copy of C, and verifying that the contexts $\mathrm{rcX}(\bar{t}, \bar{c}, \bar{h}, \bar{f}_1)$ and $\mathrm{rcX}(\bar{t}^*, \bar{c}^*, \bar{h}^*, \bar{f}_1^*)$ are fully isomorphic, which is again possible according to Claim 12. Also, the formula can check whether the contexts $\mathrm{rcX}(\bar{t}, (c_1, c_2), \bar{h}, \bar{f}_2)$ and $\mathrm{rcX}(\bar{t}^*, (c_1^*, c_2^*), \bar{h}^*, \bar{f}_2^*)$ are fully isomorphic, following Claim 13. It remains to check that also the contexts $\mathrm{rcX}(\bar{t}, \bar{r}, (c_1, c_2), \bar{f}_3)$ and $\mathrm{rcX}(\bar{t}, \bar{r}, (c_1^*, c_2^*), \bar{f}_3)$ with hole (c_1, c_2) and (c_1^*, c_2^*) respectively are fully isomorphic, which can be read from X-ISO$_2$, as these contexts are unaffected by the change. Here, $\bar{f}_i, \bar{f}_i^*$ encode the colours of the corresponding vertices in the recoloured graphs $\mathrm{graph}(X)$ and $\mathrm{graph}(X^*)$.

Now suppose that C is not on the path from $\bar{h}$ to $\bar{r}$ in X. Then let A be the least common ancestor of $\bar{h}$ and C in the context X, which can be identified using the distance information on the tree, and let $\bar{a}$ be a tuple of vertices representing it. Analogously to the explanation above for the contexts $\mathrm{rcX}(\bar{t}, \bar{r}, \bar{h}, \bar{f})$ and $\mathrm{rcX}(\bar{t}^*, \bar{r}^*, \bar{h}^*, \bar{f}^*)$, a first-order formula can determine whether the subtrees $\mathrm{rcST}(\bar{t}, \bar{b}, \bar{f}_4)$ and $\mathrm{rcST}(\bar{t}^*, \bar{b}^*, \bar{f}_4^*)$ are fully isomorphic, where $\bar{b}$ represents the child of A on the path to C and $\bar{b}^*$ is an existentially quantified tuple of vertices representing its isomorphic copy. The context rooted at the child $\bar{d}$ of A that includes the hole is not affected by the change, so using either Claim 12 or Claim 13, one can determine whether $\mathrm{rcX}(\bar{t}, \bar{a}, \bar{h}, \bar{f}_5)$ and $\mathrm{rcX}(\bar{t}^*, \bar{a}^*, \bar{h}^*, \bar{f}_5^*)$ are fully isomorphic, where $\bar{a}^*$ is the quantified tuple that represents the isomorphic copy of A.

If also the contexts $\mathrm{rcX}(\bar{t}, \bar{r}, \bar{a}, \bar{f}_6)$ and $X(\bar{t}^*, \bar{r}^*, \bar{a}^*, \bar{f}_6)$ are fully isomorphic, so are X and X^*. Again, $\bar{f}_i, \bar{f}_i^*$ encode the colours of the corresponding vertices in the respective recoloured graphs.

Case [3 $\overline{\longrightarrow}$ 2]. In the case that a 3-connected component unfurls into a path ρ of alternating separating pairs and triconnected components after an edge deletion, let C be the component of this path that is closest to the root in the context X. At most two children K_1 and K_2 of C are on the path ρ; for the other children, the old isomorphism information is still valid. Our goal is to compute the isomorphism information for K_1 and K_2, then the information for C can be computed according to Claim 12 or Claim 13, depending on the type of C. Isomorphisms of the full context X can then be computed as detailed in the previous cases.

We explain how the isomorphism information for K_1 is computed, the computation for K_2 is analogous. We assume that K_1 is a separating pair $\{a_1, b_1\}$; in case K_1 is a triconnected component, we apply the following to its child separating pair on the path ρ and apply Claim 12 to get the isomorphism information of K_1.

Let $(P_1 = K_1)D_1P_2 \cdots P_kD_k$ be the subpath of ρ that is below K_1, where the P_i are separating pair vertices and the D_i are triconnected component vertices. Note that the old isomorphism information for the subtrees below these vertices that do not intersect ρ are still valid. However, we cannot iteratively use Claim 12 and Claim 13 to lift this information to the subtree of K_1, as the length of the path is not bounded by a constant. Instead, a first-order formula will test in parallel for each ℓ from 2 to k whether the context with root $P_{\ell-1}$ and hole P_ℓ is fully isomorphic to its counterpart.

Putting this information together requires some care, as for each separating pair $P_\ell = \{a_\ell, b_\ell\}$ and its candidate isomorphic copy $P_\ell^* = \{a_\ell^*, b_\ell^*\}$ there is the possibility to map (a_ℓ, b_ℓ) to (a_ℓ^*, b_ℓ^*) or to (b_ℓ^*, a_ℓ^*). Note however that an ordering of P_1 and P_k as well as of the candidate isomorphic copies P_1^* and P_k^* fixes an ordering of the other separating pairs:

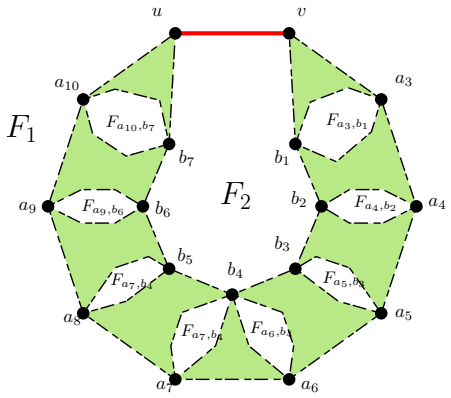


Figure 4 A 3-connected graph C that after deletion of edge (u, v) unfurls into a path of triconnected components with separating pairs $\{a_3, b_1\}, \{a_4, b_2\}, \{a_5, b_3\}, \{a_6, b_4\}, \{a_7, b_4\}, \{a_8, b_5\}, \{a_9, b_6\}, \{a_{10}, b_7\}$. These pairs are also the nodes of the disk graph $\mathsf{Disk}(C, F_1, F_2)$.

if the ordering of P_1 is (a_1, b_1) and the ordering of P_k is (a_k, b_k), after inserting an edge (a_1, a_k), every separating pair $P_2, \ldots, P_{k-1}$ has exactly one vertex that is on the same face as a_1 and a_k and this vertex can be identified using the auxiliary relation SAME-FACE. Any isomorphism has to respect this property, which allows a first-order formula to globally fix a mapping between the vertices of the separating pairs and then to check whether this mapping can be extended to an isomorphism of the whole context. ◀

5.2.2 Maintaining Distances

Distances in triconnected component trees are not affected by changes of type $[3 \xrightarrow{+/-} 3]$ and can be easily updated after changes of type $[2 \xrightarrow{+} 3]$.

Changes of type $[3 \xrightarrow{-} 2]$ are more difficult, as a node for a 3-connected component C can unfurl into a long path. We exploit the unique combinatorial embedding of C to determine the length of that path. A key observation is that the separating pairs $\{a_1, b_1\}, \ldots, \{a_k, b_k\}$ in the unfurled path appear in this order in the *disk* formed by the two faces F_1 and F_2 adjacent to the deleted edge in C (see Figure 4). Within C, for each such pair $\{a_i, b_i\}$, one of the vertices lies on F_1 and the other on F_2.

We next define such disks for arbitrary (not necessarily adjacent) faces F_1 and F_2, and show how to use them to maintain distances. A *combinatorial face* F of a 3-connected graph C is a face of C together with one of the two orders of its vertices as they appear around the face. We use that each combinatorial face already determines a unique (combinatorial) embedding $\mathsf{Emb}(C, F)$ of C with F as outer face [12].

▶ **Definition 14** (Disk graph). *Let C be a 3-connected graph, $\mathsf{Emb}(C, F_1)$ an embedding of C for a combinatorial face F_1, and F_2 another combinatorial face of $\mathsf{Emb}(C, F_1)$. The directed graph $\mathsf{Disk}(C, F_1, F_2)$ has nodes (a, b) where a, b are distinct vertices of G such that*

(a) *a appears on F_1 and b appears on F_2,*

(b) *there exists a face $F_{a,b}$ distinct from F_1 and F_2 such that both a and b appear on $F_{a,b}$,*

(c) *there does not exist another pair (a', b') with (i) $a \neq a'$ and $b \neq b'$, (ii) satisfying (a) and (b), and (iii) the vertices a, a', b, b' appear on another face of $\mathsf{Emb}(C, F_1)$ in that cyclic order.*

The graph $\mathsf{Disk}(C, F_1, F_2)$ has an edge from (a, b) to (a', b') if for any other node (a'', b'') of $\mathsf{Disk}(C, F_1, F_2)$ the vertices a, a', a'' and the vertices b, b', b'' appear in that (clockwise) cyclic order on the face F_1 and F_2 respectively.

We observe that $\mathsf{Disk}(C, F_1, F_2)$ is a directed cycle. Its distances correspond to distances in an unfurled path of the triconnected component tree, if an edge deletion merges the faces F_1 and F_2 and destroys a 3-connected component.

▶ **Lemma 15.** *Suppose C is a 3-connected component, e is an edge of C such that $C - \{e\}$ is not 3-connected, and F_1 and F_2 are the faces of C adjacent to e. Then the following holds:*

(i) *$\{a, b\}$ is a 3-connected separating pair of $C - \{e\}$ if and only if (a, b) or (b, a) is a node of $\mathsf{Disk}(C, F_1, F_2)$.*

(ii) *The triconnected component tree of $C - \{e\}$ is a path of length $2d$, where d is the number of nodes of $\mathsf{Disk}(C, F_1, F_2)$, in which the 3-connected separating pairs of $C - \{e\}$ appear in the same order as their corresponding nodes in $\mathsf{Disk}(C, F_1, F_2)$.*

Thus distances between separating pair nodes of the triconnected component tree tri-tree$(C - \{e\})$ can be inferred from the distances between the nodes of $\mathsf{Disk}(C, F_1, F_2)$.

Therefore, apart from maintaining distances on triconnected decomposition trees, we also maintain the distances on the cycles $\mathsf{Disk}(C, F_1, F_2)$, for every 3-connected component C and all combinatorial faces F_1, F_2. More formally, our dynamic algorithm additionally maintains the relation DiskDistance that contains tuples $(\bar{f}_1, \bar{f}_2, \bar{a}, \bar{b}, k)$ such that $\bar{f}_1, \bar{f}_2$ specify combinatorial faces F_1, F_2 of a 3-connected component C; $\bar{a}, \bar{b}$ specify nodes of $\mathsf{Disk}(C, F_1, F_2)$; and k is the distance of $\bar{a}$ and $\bar{b}$ in $\mathsf{Disk}(C, F_1, F_2)$. To this end, it also maintains combinatorial embeddings with respect to each combinatorial face using the dynamic algorithm from [7].

5.3 Maintaining Isomorphism Information for Connected Components

The techniques for maintaining isomorphic biconnected components can be lifted to connected components.

▶ **Proposition 16.** *The relation $\textsc{iso}_1$ that for a planar graph G contains all tuples (a, a^*) such that:*

- *a and a^* are vertices of connected components A and A^* of G, respectively, and*
- *A and A^* are isomorphic via an isomorphism π with $\pi(a) = a^*$*

can be maintained in DynFO *under insertions and deletions of edges, provided that G stays planar.*

For maintaining isomorphisms of triconnected component trees, we used the isomorphisms of triconnected components to identify isomorphic copies of separating pairs of those components and checked whether the isomorphism extended to the subtrees of the triconnected component tree below these separating pairs. For isomorphic biconnected components, however, we only know that an isomorphism exists, but cannot map cut vertices of these components to their isomorphic copies and then check that their subtrees in the biconnected component tree are isomorphic too.

We solve this problem by colouring cut vertices such that two cut vertices have the same colour if and only if their subtrees are isomorphic. If an isomorphism between biconnected components exists whose cut vertices are coloured this way (and by Proposition 10, isomorphism of coloured biconnected components can be maintained), we know that such an isomorphism extends to the subtree below these components in the biconnected component tree. As the isomorphism type of a cut vertex depends on the specific context, we maintain, for every possible context X of the biconnected component tree and recolouring $\bar{f}$ of its hole, a coloured copy col-graph$(X, \bar{f})$ of the graph that is described by that context.

Our dynamic algorithm maintains an auxiliary relation $\textsc{x-iso}_1$ that contains tuples $((X, \bar{f}), (X^*, \bar{f}^*)) \stackrel{\text{def}}{=} (\bar{x}, \bar{r}, \bar{h}, \bar{f}, \bar{x}^*, \bar{r}^*, \bar{h}^*, \bar{f}^*)$ such that the coloured graphs col-graph$(X, \bar{f})$ and col-graph$(X^*, \bar{f}^*)$ are isomorphic via an isomorphism that maps $\bar{r}$ to $\bar{r}^*$. For proving Proposition 16, we then show:

▶ **Lemma 17.** *The following can be maintained in* DynFO *under insertions and deletions of edges to a planar graph G, provided that G stays planar:*

(1) *a coloured graph col-graph$(X, \bar{f})$ for every context $X = (\bar{t}, \bar{r}, \bar{h})$ of bi-tree(G) and colouring $\bar{f}$ of the vertices in $\bar{h}$, such that*
- **(a)** *col-graph$(X, \bar{f})$ has the same vertices and edges as graph(X),*
- **(b)** *exactly the vertices of $\bar{h}$ and the cut vertices of col-graph$(X, \bar{f})$ are coloured,*
- **(c)** *the vertices of $\bar{h}$ that are not cut vertices in col-graph$(X, \bar{f})$ are coloured as given by $\bar{f}$, and*
- **(d)** *for two contexts $X_1 = X(\bar{t}_1, \bar{r}_1, \bar{h}_1)$ and $X_2 = X(\bar{t}_2, \bar{r}_2, \bar{h}_2)$ and colourings $\bar{f}_1, \bar{f}_2$, two cut vertices v_1 in col-graph$(X_1, \bar{f}_1)$ and v_2 in col-graph$(X_2, \bar{f}_2)$ have the same colour if and only if the subtrees $ST(\bar{r}_1, v_1)$ and $ST(\bar{r}_2, v_2)$ of the biconnected component trees of the respective coloured graphs are fully isomorphic.*

(2) *the relation* X-ISO$_1$ *with respect to these coloured graphs.*

6 Conclusion

We have shown that DynFO is powerful enough to maintain whether two planar graphs are isomorphic. For isomorphic 3-connected components of a planar graph, one can also maintain a witnessing isomorphism, which is an important ingredient of our proof. We believe that such a witness can also be maintained for arbitrary isomorphic planar graphs, but verifying this conjecture is future work. Whether one can maintain a canon of a planar graph, that is, a unique representative of the isomorphism type of the input graph, stays an open problem.

As for planar graphs, the isomorphism problem for bounded-treewidth graphs can be solved with logarithmic space [14]. At the same time, many properties of graphs with bounded treewidth can be maintained in DynFO [11]. It is interesting to study whether isomorphism of bounded-treewidth graphs can be maintained in DynFO as well. A major difficulty towards such a result is the problem of maintaining isomorphism-invariant tree decompositions of graphs: if two graphs with bounded treewidth are isomorphic, we would like to have tree decompositions of these graphs that are also isomorphic. The (approximations of) tree decompositions utilized in [11] do not have this property, as they depend on the order of edge changes to the input graph.

References

1 László Babai. Graph isomorphism in quasipolynomial time [extended abstract]. In *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '16, pages 684–697, New York, NY, USA, 2016. Association for Computing Machinery. doi:10.1145/2897518.2897542.

2 David A. Mix Barrington, Neil Immerman, and Howard Straubing. On uniformity within NC^1. *J. Comput. Syst. Sci.*, 41(3):274–306, 1990. doi:10.1016/0022-0000(90)90022-D.

3 G. Battista and R. Tamassia. On-line maintenance of triconnected components with spqr-trees. *Algorithmica*, 15(4):302–318, 1996. doi:10.1007/BF01961541.

4 Hans L Bodlaender. Polynomial algorithms for graph isomorphism and chromatic index on partial k-trees. *Journal of Algorithms*, 11(4):631–643, 1990. doi:10.1016/0196-6774(90)90013-5.

5 Stephen A. Cook. A taxonomy of problems with fast parallel algorithms. *Information and Control*, 64(1-3):2–21, 1985. doi:10.1016/S0019-9958(85)80041-3.

6 Bireswar Das, Jacobo Torán, and Fabian Wagner. Restricted space algorithms for isomorphism on bounded treewidth graphs. *Information and Computation*, 217:71–83, 2012. doi:10.1016/j.ic.2012.05.003.

7 Samir Datta, Asif Khan, and Anish Mukherjee. Dynamic planar embedding is in DynFO. In Jérôme Leroux, Sylvain Lombardy, and David Peleg, editors, *48th International Symposium on Mathematical Foundations of Computer Science (MFCS 2023)*, volume 272 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 39:1–39:15, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.MFCS.2023.39.

8 Samir Datta, Asif Khan, Anish Mukherjee, Felix Tschirbs, Nils Vortmeier, and Thomas Zeume. Query maintenance under batch changes with small-depth circuits. In Rastislav Královic and Antonín Kucera, editors, *49th International Symposium on Mathematical Foundations of Computer Science, MFCS 2024, August 26-30, 2024, Bratislava, Slovakia*, volume 306 of *LIPIcs*, pages 46:1–46:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.MFCS.2024.46.

9 Samir Datta, Raghav Kulkarni, Anish Mukherjee, Thomas Schwentick, and Thomas Zeume. Reachability is in DynFO. *J. ACM*, 65(5):33:1–33:24, 2018. doi:10.1145/3212685.

10 Samir Datta, Nutan Limaye, Prajakta Nimbhorkar, Thomas Thierauf, and Fabian Wagner. Planar graph isomorphism is in log-space. *ACM Trans. Comput. Theory*, 14(2), September 2022. doi:10.1145/3543686.

11 Samir Datta, Anish Mukherjee, Thomas Schwentick, Nils Vortmeier, and Thomas Zeume. A strategy for dynamic programs: Start over and muddle through. *Log. Methods Comput. Sci.*, 15(2), 2019. doi:10.23638/LMCS-15(2:12)2019.

12 Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

13 Michael Elberfeld and Ken-ichi Kawarabayashi. Embedding and canonizing graphs of bounded genus in logspace. In *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '14, pages 383–392, New York, NY, USA, 2014. Association for Computing Machinery. doi:10.1145/2591796.2591865.

14 Michael Elberfeld and Pascal Schweitzer. Canonizing graphs of bounded tree width in logspace. *ACM Trans. Comput. Theory*, 9(3):12:1–12:29, 2017. doi:10.1145/3132720.

15 Kousha Etessami. Dynamic tree isomorphism via first-order updates. In Alberto O. Mendelzon and Jan Paredaens, editors, *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 1-3, 1998, Seattle, Washington, USA*, pages 235–243. ACM Press, 1998. doi:10.1145/275487.275514.

16 Martin Grohe and Oleg Verbitsky. Testing graph isomorphism in parallel by playing a game. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part I*, volume 4051 of *Lecture Notes in Computer Science*, pages 3–14. Springer, 2006. doi:10.1007/11786986_2.

17 Carsten Gutwenger and Petra Mutzel. A linear time implementation of spqr-trees. In Joe Marks, editor, *Graph Drawing, 8th International Symposium, GD 2000, Colonial Williamsburg, VA, USA, September 20-23, 2000, Proceedings*, Lecture Notes in Computer Science, pages 77–90. Springer, 2000. doi:10.1007/3-540-44541-2_8.

18 H. V. Henderson and S. R. Searle. On deriving the inverse of a sum of matrices. *SIAM Review*, 23(1):53–60, 1981. doi:10.1137/1023004.

19 J. E. Hopcroft and J. K. Wong. Linear time algorithm for isomorphism of planar graphs (preliminary report). In *Proceedings of the Sixth Annual ACM Symposium on Theory of Computing*, STOC '74, pages 172–184, New York, NY, USA, 1974. Association for Computing Machinery. doi:10.1145/800119.803896.

20 John E. Hopcroft and Robert Endre Tarjan. Dividing a graph into triconnected components. *SIAM J. Comput.*, 2(3):135–158, 1973. doi:10.1137/0202012.

21 Sandra Kiefer, Ilia Ponomarenko, and Pascal Schweitzer. The Weisfeiler-Leman dimension of planar graphs is at most 3. *J. ACM*, 66(6):44:1–44:31, 2019. doi:10.1145/3333003.

22 Steven Lindell. A logspace algorithm for tree canonization (extended abstract). In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing, May 4-6, 1992, Victoria, British Columbia, Canada*, pages 400–404, 1992. doi:10.1145/129712.129750.

23 Jenish C. Mehta. Dynamic complexity of planar 3-connected graph isomorphism. In Edward A. Hirsch, Sergei O. Kuznetsov, Jean-Éric Pin, and Nikolay K. Vereshchagin, editors, *Computer Science - Theory and Applications - 9th International Computer Science Symposium in Russia, CSR 2014, Moscow, Russia, June 7-11, 2014. Proceedings*, Lecture Notes in Computer Science, pages 273–286. Springer, 2014. doi:10.1007/978-3-319-06686-8_21.

24 Sushant Patnaik and Neil Immerman. Dyn-FO: A parallel, dynamic complexity class. In Victor Vianu, editor, *Proceedings of the Thirteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 24-26, 1994, Minneapolis, Minnesota, USA*, pages 210–221. ACM Press, 1994. doi:10.1145/182591.182614.

25 Sushant Patnaik and Neil Immerman. Dyn-FO: A parallel, dynamic complexity class. *J. Comput. Syst. Sci.*, 55(2):199–209, 1997. doi:10.1006/JCSS.1997.1520.

26 P. N. Shivakumar and Kim Ho Chew. A sufficient condition for nonvanishing of determinants. *Proc. Amer. Math. Soc.*, 43:63–66, 1974. doi:10.2307/2039326.

27 Thomas Thierauf and Fabian Wagner. The isomorphism problem for planar 3-connected graphs is in unambiguous logspace. *Theory Comput. Syst.*, 47(3):655–673, 2010. doi:10.1007/S00224-009-9188-4.

28 Jacobo Torán. On the hardness of graph isomorphism. *SIAM Journal on Computing*, 33(5):1093–1108, 2004. doi:10.1137/S009753970241096X.

29 W. T. Tutte. How to draw a graph. *Proceedings of the London Mathematical Society*, s3-13(1):743–767, 1963. doi:10.1112/plms/s3-13.1.743.

30 Nils Vortmeier and Thomas Zeume. Dynamic complexity of parity exists queries. *Log. Methods Comput. Sci.*, 17(4), 2021. doi:10.46298/LMCS-17(4:9)2021.

31 L. Weinberg. A simple and efficient algorithm for determining isomorphism of planar triply connected graphs. *IEEE Transactions on Circuit Theory*, 13(2):142–148, 1966. doi:10.1109/TCT.1966.1082573.

32 Hassler Whitney. 2-isomorphic graphs. *American Journal of Mathematics*, 55(1):245–254, 1933. URL: http://www.jstor.org/stable/2371127.