View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Dagstuhl Research Online Publication Server

Graph Isomorphism for $K_{3,3}$-free and K_5-free graphs is in Log-space

Samir Datta[1], **Prajakta Nimbhorkar**[2], **Thomas Thierauf**[3], **Fabian Wagner**[4]*

[1]Chennai Mathematical Institute
sdatta@cmi.ac.in

[2]The Institute of Mathematical Sciences
prajakta@imsc.res.in

[3]Fak. Elektronik und Informatik, HTW Aalen

[4]Institut für Theoretische Informatik,
Universität Ulm, 89073 Ulm
{thomas.thierauf,fabian.wagner}@uni-ulm.de

ABSTRACT. Graph isomorphism is an important and widely studied computational problem with a yet unsettled complexity. However, the exact complexity is known for isomorphism of various classes of graphs. Recently, [8] proved that planar isomorphism is complete for log-space. We extend this result further to the classes of graphs which exclude $K_{3,3}$ or K_5 as a minor, and give a log-space algorithm.

Our algorithm decomposes $K_{3,3}$ minor-free graphs into biconnected and those further into triconnected components, which are known to be either planar or K_5 components [20]. This gives a triconnected component tree similar to that for planar graphs. An extension of the log-space algorithm of [8] can then be used to decide the isomorphism problem.

For K_5 minor-free graphs, we consider 3-connected components. These are either planar or isomorphic to the four-rung mobius ladder on 8 vertices or, with a further decomposition, one obtains planar 4-connected components [9]. We give an algorithm to get a unique decomposition of K_5 minor-free graphs into bi-, tri- and 4-connected components, and construct trees, accordingly. Since the algorithm of [8] does not deal with four-connected component trees, it needs to be modified in a quite non-trivial way.

1 Introduction

The graph isomorphism problem GI consists of deciding whether there is a bijection between the vertices of two graphs, preserving the adjacencies among vertices. It is an important problem with a yet unknown complexity. The problem is clearly in NP and is also in SPP [2]. It is unlikely to be NP-hard [5, 16], because otherwise the polynomial time hierarchy collapses to the second level. As far as lower bounds are concerned, GI is hard for DET [18], the class of problems NC^1-reducible to the determinant [6].

*Supported by DFG grants TO 200/2-2.

© S. Datta, P. Nimbhorkar, T. Thierauf, F. Wagner; licensed under Creative Commons License-NC-ND.
Foundations of Software Technology and Theoretical Computer Science (Kanpur) 2009.
Editors: Ravi Kannan and K. Narayan Kumar; pp 145–156
Leibniz International Proceedings in Informatics (LIPIcs), Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany.
Digital Object Identifier: 10.4230/LIPIcs.FSTTCS.2009.2314

While this enormous gap has motivated a study of isomorphism in *general* graphs, it has also induced research in isomorphism restricted to special cases of graphs where this gap can be reduced. Tournaments are an example of directed graphs where the DET lower bound is preserved [21], while there is a quasi-polynomial time upper bound [4]. The complexity of isomorphism is settled for trees [11, 12], partial 2-trees [1], and for planar graphs [8]. We extend the result of [8] to isomorphism of $K_{3,3}$ and K_5 minor-free graphs. The previously known upper bound for these graph classes is P due to [14]. Both of these graph classes include planar graphs, and hence are considerably larger than the class of planar graphs.

We consider undirected graphs without parallel edges and loops, also known as *simple* graphs. For directed graphs or graphs with loops and parallel edges, there are log-space many-one reductions to simple undirected graphs (cf. [10]). Our log-space algorithm relies on the following properties of $K_{3,3}$ and K_5 minor-free graphs:

- The 3-connected components of $K_{3,3}$ minor-free graphs are either planar graphs or complete graphs on 5 vertices i.e. K_5's [20].
- The 3-connected components of K_5 minor-free graphs are either planar or V_8's (where V_8 is a four-rung mobius ladder on 8 vertices) or the following holds. The 4-connected components of the remaining non-planar 3-connected components are planar [9].

There is a related result [17] where reachability in $K_{3,3}$ and K_5 minor-free graphs are reduced to reachability in planar graphs under log-space many-one reductions. The basic idea is that the non-planar components are transformed into new planar components. This technique preserves the reachability properties but not the isomorphism. We give a log-space algorithm to get these decompositions in a *canonical* way, and construct the biconnected and triconnected component trees for $K_{3,3}$ minor-free graphs. Then we extend the log-space algorithm of [8] for isomorphism testing and canonization of two such graphs. The isomorphism of K_5 minor-free graphs is more complex, as in addition it has bi-, tri- *and* four-connected component trees. This needs considerable modifications and new ideas.

The rest of the paper is organized as follows: Section 2 gives the necessary definitions and background. Section 3 gives the decomposition of $K_{3,3}$ and K_5 minor-free graphs and proves the uniqueness of such decompositions. In Section 4 we give a log-space algorithm for isomorphism and canonization of $K_{3,3}$ and K_5 minor-free graphs. We omit some proofs due to space constraints and refer to full version of the paper for those proofs.

2 Definitions and Notations

For $U \subseteq V$ let $G \setminus U$ be the *induced subgraph* of G on $V \setminus U$. Let $S \subseteq V$ with $|S| = k$. S is a *k-separating set* if $G \setminus S$ is not connected. The vertices of a k-separating set are called *articulation point (or cut vertex)* for $k = 1$, *separating pair* for $k = 2$, and *separating triple* for $k = 3$. G is *k-connected* if it contains no $(k-1)$-separating set, i.e. there are k vertex-disjoint paths between any pair in G. A 1-connected graph is simply called *connected* and a 2-connected graph *biconnected*. The connected components of $G \setminus S$ are called the *split components of S*.

DEFINITION 1. The biconnected component tree. *We define nodes for the biconnected components and articulation points. There is an edge between a* biconnected component node *and an* articulation point node *if the articulation point is contained in the corresponding component. The resulting graph is a tree, the* biconnected component tree $\mathcal{T}^{\mathrm{B}}(G)$.

A graph is *triconnected* if it is either 3-connected, a cycle or a 3-bond. A k-bond is a pair of vertices connected by k edges. A separating pair $\{a, b\}$ is called 3-*connected* if there are three vertex-disjoint paths between a and b. In the rest of the paper a separating pair is always considered to be a 3-connected separating pair.

DEFINITION 2. The triconnected component tree. *Define nodes for the triconnected components and (3-connected) separating pairs for a biconnected graph G. There is an edge between a* triconnected component node *and a* separating pair node *if the separating pair is contained in the corresponding component. In a triconnected component, the vertices of a separating pair are connected by a* virtual edge. *If a separating pair is connected in the original graph G then there is a node for a 3-bond connected to the separating pair node. The resulting graph is a tree, the* triconnected component tree $\mathcal{T}^{\mathrm{T}}(G)$.

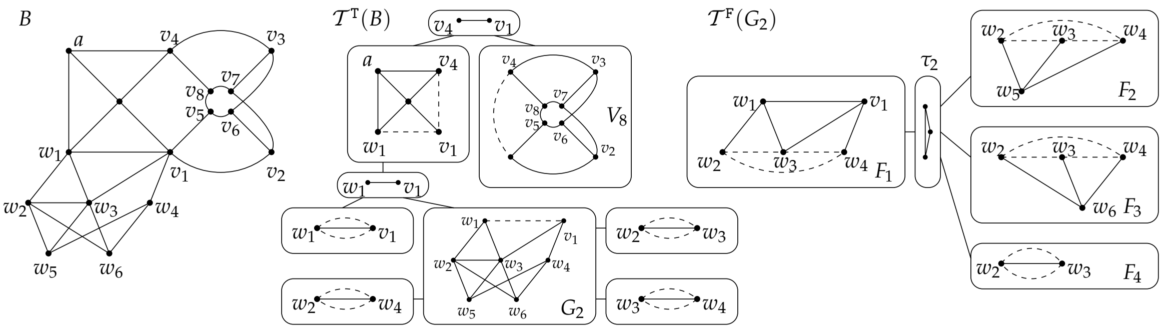


Figure 1: Decomposition of biconnected component B into triconnected components, and G_2 further into four-connected components. Virtual edges are indicated by dashed lines. There is a 3-bond connected to τ_2 because $\{w_2, w_3\}$ is an edge in G_2.

For a component tree T, the *size of an individual component node* C of T is the number of nodes in C. The vertices of the separating sets are counted in in every component where they occur. The *size of the tree* T, denoted by $|T|$, is the sum of the sizes of its component nodes. The size of T is at least as large as the number of vertices in $\mathsf{graph}(T)$, the graph corresponding to the component tree T. Let T_C be T when rooted at node C. A child of C is called a *large child* if $|T_C| > |T|/2$. $\#C$ denotes the number of children of C.

A graph H is a minor of a graph G if and only if H can be obtained from G by a finite sequence of edge-removal and edge-contraction operations. A *$K_{3,3}$-free graph* (*K_5-free graph*) is an undirected graph which does not contain a $K_{3,3}$ (or K_5) as a minor.

For two isomorphic graphs we write $G \cong H$. A *canon* for G is a sorted list of edges with renamed vertices $f(G)$, such that for all graphs G, H we have $G \cong H \Leftrightarrow f(G) = f(H)$. We also use *canon* with respect to some fixed starting edge. A *code* of G is the lexicographically sorted list of edges when given an arbitrary labeling of vertices.

By L we denote the languages computable by a log-space bounded Turing machine.

3 Decomposition into triconnected components

3.1 Decomposition of $K_{3,3}$-free graphs

We consider the decomposition of biconnected $K_{3,3}$-free graphs into triconnected components. The decomposition is unique [19] and has the following form.

LEMMA 3. [3] *Each triconnected component of a $K_{3,3}$-free graph is either planar or exactly the graph K_5.*

We state a more general result below, which is used in our decomposition:

LEMMA 4. *In a simple undirected biconnected graph G, the removal of 3-connected separating pairs gives a unique decomposition, irrespective of the order in which they are removed. This decomposition can be computed in log-space.*

Miller and Ramachandran [13] showed that the triconnected component tree of a $K_{3,3}$-free graph can be computed in NC^2. Thierauf and Wagner [17] describe a construction that works in log-space. This now follows from Lemma 4:

COROLLARY 5. *For a biconnected $K_{3,3}$-free graph, the triconnected planar components and K_5 components can be computed in log-space.*

3.2 Decomposition of K_5-free graphs

We decompose the given K_5-free graph into 3-connected and 4-connected components. It follows from a theorem of Wagner [22] that besides planar components we obtain the following non-planar components that way:

- the four-rung Mobius ladder (also called V_8), a 3-connected graph on 8 vertices, which is non-planar because it contains a $K_{3,3}$.
- The remaining 3-connected non-planar components are further decomposed into 4-connected components which are all planar.

Khuller [9] described a decomposition of K_5-free graphs with a clique-sum operation. If two graphs G_1 and G_2 each contain cliques of equal size, the *clique-sum* of G_1 and G_2 is a graph G formed from their disjoint union by identifying pairs of vertices in these two cliques to form a single shared clique, and then possibly deleting some of the clique edges. A *k-clique-sum* is a clique-sum in which both cliques have at most k vertices.

If G can be constructed by repeatedly taking k-clique-sums starting from graphs isomorphic to members of some graph class $\mathcal{G}$, then we say $G \in \langle \mathcal{G} \rangle_k$. The class of K_5-free graphs can be decomposed as follows.

THEOREM 6. [22] *Let $\mathcal{C}$ be the class of all planar graphs together with the four-rung Mobius ladder V_8. Then $\langle \mathcal{C} \rangle_3$ is the class of all graphs with no K_5-minor.*

Theorem 6 and the following observations lead to Corollary 7:

- If we build the 3-clique-sum of two planar graphs, then the three vertices of the joint clique are a separating triple in the resulting graph. Hence, the 4-connected components of a graph which is built as the 3-clique-sum of planar graphs must all be planar.

- The V_8 is non-planar and 3-connected and cannot be part of a 3-clique sum, because it does not contain a triangle as subgraph.

COROLLARY 7.(cf. [9]) *A non-planar 3-connected component of a K_5-free undirected graph is either the V_8 or its 4-connected components are all planar.*

Similar to the decomposition algorithm of Vazirani [20], we decompose the K_5-free graph into triconnected components. That is, we first decompose it into biconnected components and then the biconnected components further into triconnected components.

Unique decomposition of 3-connected K_5-free graphs. Let $G \neq V_8$ be a non-planar 3-connected component of a K_5-free graph, which needs to be decomposed into 4-connected components. The decomposition by [17] is not unique up to isomorphism. Therefore we describe a different way of decomposition. The main idea is to decompose G at only those separating triples which cause the non-planarity.

DEFINITION 8. *Let G be a 3-connected component of a graph G^* and let $\tau \subseteq V(G)$ be a separating triple. Then τ is called* 3-divisive *if in $G^* \setminus \tau$ the component G is split into at least three connected components.*

Intuitively, to see that a 3-divisive separating triple τ causes always non-planarity, collapse the split components of τ to single vertices and a $K_{3,3}$ is obtained. If G is not the $K_{3,3}$ then we can split G at one 3-divisive separating triple and all the other 3-divisive separating triples remain. We prove now that the $K_{3,3}$ is the only special case where this is different.

DEFINITION 9. *Let G be an undirected K_5-free 3-connected graph. Two 3-divisive separating triples $\tau \neq \tau'$ are* conflicting *if τ is no 3-divisive separating triple in a 3-connected component of $(G \setminus \tau') \cup \tau$.*

In general there are no conflicting 3-divisive separating triples except for the $K_{3,3}$. This is important to obtain a decomposition for G which is unique up to isomorphism.

LEMMA 10. *Let G be an undirected and 3-connected graph. There is a conflicting pair of 3-divisive separating triples in G if and only if G is the $K_{3,3}$.*

The four-connected component tree. If we fix one 3-divisive separating triple as root then we get a unique decomposition for G up to isomorphism, also if G is the $K_{3,3}$. We decompose the given graph G at 3-divisive separating triples and obtain *four-connected components*. Two vertices u, v belong to a *four-connected component* if for all 3-divisive separating triples τ the following is true: At least one of u, v belongs to τ *or* there is a path from u to v in $G \setminus \tau$. Note, a four-connected component is planar and 3-connected.

We define a graph with nodes for the four-connected components and 3-divisive separating triples. A *four-connected component node* is connected to a *3-divisive separating triple node* τ if the vertices of τ are also contained in the corresponding four-connected component. The resulting graph is a tree, the *four-connected component tree* $\mathcal{T}^{\mathrm{F}}(G)$. This unique decomposition can be computed in log-space, because every computation step can be queried to the reachability problem in undirected graphs which is in log-space [15].

THEOREM 11. *A unique decomposition of a* 3*-connected non-planar* K_5*-free graph (not the* V_8*) into four-connected components can be computed in log-space.*

The triconnected component tree of K_5-free graphs. From Lemma 4, it follows that the triconnected component tree of a K_5-free graph is log-space computable (also see [8, 17]). For technical reasons we make some changes to this tree structure. Let B be a biconnected K_5-free graph with G_0 a triconnected non-planar component node in $\mathcal{T}^{\mathrm{T}}(B)$. In $\mathcal{T}^{\mathrm{T}}(B)$ there is a separating pair node s for each edge which is part of a 3-divisive separating triple in G. In $\mathcal{T}^{\mathrm{T}}(B)$ the node s is connected to the node G. We call s a *leaf separating pair* of $\mathcal{T}^{\mathrm{T}}(B)$ if it is connected to only one component node. It can be seen that the set of leaf separating pairs can be computed in log-space.

4 Canonization of $K_{3,3}$-free and K_5-free graphs

4.1 Isomorphism ordering and canonization of $K_{3,3}$-free graphs

We decompose $K_{3,3}$-free graphs as in Section 3.1 and extend [8] for K_5-components.

Isomorphism ordering for K_5-components. For a K_5 component we have a node in the triconnected component tree. There are 5! ways of labeling the vertices of a K_5. The first two vertices are from the parent separating pair. There remain $2 \cdot 3! = 12$ ways of labelling the vertices, e.g. $a = 1, b = 2, c = 3, d = 4, e = 5$ is one possibility to label the vertices a, b, c, d, e. The canonical description of the graph is then $(1,2)(1,3),(1,4),(1,5),(2,1),(2,3),\ldots,(5,4)$. The canonical descriptions of all these labelings are candidates for the canon of the K_5. To keep notation short, we say *code* instead of *candidate for a canon*.

For each code, the isomorphism ordering algorithm compares two codes edge-by-edge, thereby going into recursion at child separating pairs and comparing their subtrees. If the subtrees are not isomorphic, the larger code is eliminated. The comparison and the elimination of codes is done similarly as for the planar triconnected components in Datta et.al. [8]. The comparison takes $O(1)$ space on the work-tape to keep counters for the not eliminated codes. The orientation of a K_5-component given to its parent depends on the direction of $\{a, b\}$ in the codes.

CLAIM 12. *Let* G_0 *be a* K_5*-node in a triconnected component tree and let* $V(G_0) = \{a, b, c, d, e\}$ *and* (a, b) *be the parent separating pair of* G_0*. Either all minimum remaining codes start with* (a, b) *(or reverse,* (b, a)*) or there is an equal number of codes starting with* (a, b) *and* (b, a)*.*

Once we can canonize K_5-components, we can use the algorithm of [8] to check the isomorphism ordering of triconnected and biconnected component trees.

THEOREM 13. *A* $K_{3,3}$*-free graph can be canonized in log-space.*

4.2 Isomorphism order of K_5-free graphs

Isomorphism order of K_5-free 3-connected graphs

The isomorphism order of two triconnected component trees S and T rooted at separating pairs $s = \{a, b\}$ and $t = \{a', b'\}$ is defined the same way as for planar graphs in [8] with one difference. When comparing nodes of the tree we first distinguish between the new types of nodes. We define planar triconnected components $<_{\mathrm{T}}$ V_8-components $<_{\mathrm{T}}$ non-planar 3-connected components. The isomorphism order for planar components is as in [8] and refine it now for the new types of non-planar components.

Isomorphism order of subtrees rooted at V_8-components: Consider S_{G_i} and T_{H_j} rooted at V_8-component nodes G_i and H_j. We construct the codes of G_i and H_j and compare them bit-by-bit. To canonize the V_8-components, we traverse it starting from the parent separating pairs $\{a, b\}$ and $\{a', b'\}$ and then traversing the components as follows. We define the codes with help of Hamiltonian cycles of the V_8-component. We define E' to be the set of edges which are contained in four Hamiltonian cycles.

LEMMA 14. *Each directed edge of a V_8 appears in two or four Hamiltonian cycles.*

Basically, there are two possible traversals of each Hamiltonian cycle starting from $\{a, b\}$, one in each direction. We define the code for G_i and the starting edge (a, b) in this direction as follows. We distinguish the situation whether $\{a, b\} \in E'$ or not. We will fix one Hamiltonian cycle starting with (a, b). We rename the vertices in that order of their first occurrence in the fixed Hamiltonian cycle exactly in that order. The code is then the list of edges in lexicographical order with the new labels.

Isomorphism order of the 3-connected non-planar components

Let S_{G_i} and T_{H_j} be trees rooted at 3-connected non-planar component nodes G_i and H_j which are different to the V_8. Let $s = \{a, b\}$ and $t = \{a', b'\}$ be the parent separating pairs of G_i and H_j, respectively. We are interested in the orientation given to s and t. After this, we discuss the comparison algorithm of G_i with H_j. We further partition G_i and H_j into their four-connected components and consider their trees $\mathcal{T}^{\mathrm{F}}(G_i)$ and $\mathcal{T}^{\mathrm{F}}(H_j)$.

Overview of the steps in the isomorphism order.

DEFINITION 15. *For a four-connected component tree T, the* size *of an individual component node C of T is the number n_C of vertices in C. The separating triple nodes are counted in every component where they occur. The* size *of the tree T, denoted by $|T|$, is the sum of the sizes of its component nodes.*

The isomorphism order of two four-connected component trees S and T rooted at 3-divisive separating triples τ and τ' where given an order $\mathsf{order}(\tau)$ and $\mathsf{order}(\tau')$ is defined $S_\tau \leq_{\mathrm{F}} T_{\tau'}$ if:

1. $|S_\tau| < |T_{\tau'}|$ or
2. $|S_\tau| = |T_{\tau'}|$ but $\#\tau < \#\tau'$ or
3. $|S_\tau| = |T_{\tau'}|$, $\#\tau = \#\tau' = k$, but $(S_{F_1}, \ldots, S_{F_k}) <_{\mathrm{F}} (T_{F_1'}, \ldots, T_{F_k'})$ lexicographically, where we assume that $S_{F_1} \leq_{\mathrm{F}} \ldots \leq_{\mathrm{T}} S_{F_k}$ and $T_{F_1'} \leq_{\mathrm{T}} \ldots \leq_{\mathrm{T}} T_{F_k'}$ are the ordered subtrees of S_τ and $T_{\tau'}$, respectively. For the isomorphism order between the subtrees S_{F_i} and $T_{F_i'}$ we compare lexicographically the codes of F_i and F_i' and *recursively* the subtrees rooted at the children of F_i and F_i'. Note, that these children are again separating triple nodes.
4. $|S_\tau| = |T_{\tau'}|$, $\#\tau = \#\tau' = k$, $(S_{F_1} \leq_{\mathrm{F}} \ldots \leq_{\mathrm{F}} S_{F_k}) =_{\mathrm{F}} (T_{F_1'} \leq_{\mathrm{F}} \ldots \leq_{\mathrm{F}} T_{F_k'})$, but the following holds. For all i, the return value from the recursion of S_{F_i} with $T_{F_i'}$ is an *orientation*

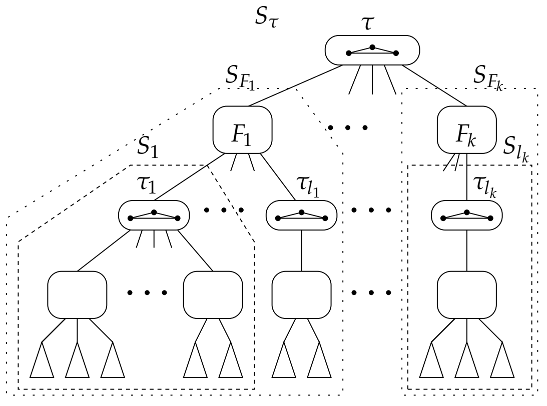
graph X_i and X'_i with $V(X_i) = \tau$ and $V(X'_i) = \tau'$ and colored edges, respectively. We compute a *reference orientation graph* X and X' from all the X_i and X'_i. We compare lexicographically whether X with $\mathsf{order}(\tau) < X'$ with $\mathsf{order}(\tau')$. We describe the notion of (reference) orientation graph and $\mathsf{order}(\tau)$ below in more detail.

We say that two four-connected component trees S_τ and $T_{\tau'}$ are *equal according to the isomorphism order*, denoted by $S_\tau =_{\mathtt{F}} T_{\tau'}$, if neither $S_\tau <_{\mathtt{F}} T_{\tau'}$ nor $T_{\tau'} <_{\mathtt{F}} S_\tau$ holds.

Orientation given to the parent separating pair by a non-planar component, not the V_8. Given two non-planar 3-connected components G_i and H_j and their trees $\mathcal{T}^{\mathtt{F}}(G_i)$ and $\mathcal{T}^{\mathtt{F}}(H_j)$. There is a set of candidates for root separating triples such that we obtain the minimum codes when the trees are rooted at them. For the canonization algorithm, the isomorphism ordering algorithm is used as a sub-routine. For the isomorphism ordering procedure, we give distinct colors to the parent separating pair and the parent articulation point in the trees. We also have colors for the child separating pairs and child articulation points, according to their isomorphism order. We recompute these colors by interrupting the current isomorphism ordering procedure and going into recursion at the corresponding subtrees.

Finally, we just consider the first occurrence of the parent separating pair in all the minimum codes. If the first occurrence is (a, b) in this direction in all the codes, then G_i gives this orientation to the parent. Similarly for (b, a). If both (a, b) and (b, a) occur first in different minimum codes, then there is no orientation given to the parent.

Isomorphism order of four-connected component trees. We describe what is different between isomorphism ordering for four-connected and triconnected component trees in Section 4.2 (also see [8]). Instead of separating pairs we have 3-divisive separating triples. In the isomorphism order algorithm for two triconnected component trees there was one task, where the orientations of the separating pairs were compared. An orientation of a pair $\{a, b\}$ in a triconnected component tree $T_{\{a,b\}}$ is the set of permutations which partially map $T_{\{a,b\}}$ to its canon. This is a subgroup of the symmetric group $Sym(\{a, b\})$. For a four-connected component tree S_τ, we consider the set of permutations of the triple $\tau = \{a, b, c\}$. This set contains all partial automorphisms which partially map S_τ to its canon. This is a subgroup of the symmetric group $Sym(\{a, b, c\})$. Instead of 3-connected planar components we have four-connected planar components in S_τ and $T_{\tau'}$.



Isomorphism order of two subtrees rooted at four-connected component nodes. We consider the isomorphism order of two subtrees S_{F_i} and $T_{F'_j}$ rooted at four-connected component nodes F_i and F'_j, respectively. We construct the codes of F_i and F'_j and compare them bit-by-bit. To canonize F_i, we use the log-space algorithm from [7]. Besides F_i, the algorithm gets as input a starting edge and a combinatorial embedding ρ of F_i. There are three choices of selecting a starting edge $\{a, b\}, \{b, c\}, \{a, c\}$ and two choices for the direction of each edge, e.g. for $\{a, b\}$ we have (a, b) and (b, a). Further, a 3-connected planar graph has two planar combinatorial embeddings [23]. There are 12 possible ways to canonize G_i.

We start the canonization of G_i and H_j in all the possible ways and compare these codes bit-by-bit. Let C and C' be two codes to be compared. The base case is that F_i and F'_j are leaf nodes and therefore contain no further virtual edges. In this case we use the lexicographic order between C and C'. If G_i and H_j contain a virtual edge, then this belongs to a child separating triple and is treated in a special way when comparing C and C':

First, if a virtual edge is traversed in the codes C or C' but not in the other, then we define the one without the virtual edge to be the *smaller code.*

Second, if C and C' encounter the virtual edges $\{u, v\}$ and $\{u', v'\}$ then we consider only the child separating triples which do not have virtual edges considered earlier in the codes C and C'. We order these triples according to the positions of all their virtual edges in the codes. We call this order the *position-order*. W.l.o.g. let τ_{i_0} (in C) and τ'_{j_0} (in C') be the separating triples which come first in this position-order. For τ_{i_0} and τ'_{j_0}, we will define below the *reference orientation graphs* X and X' with $V(X) = \tau_{i_0}$ and $V(X') = \tau'_{j_0}$, respectively. For all pairs in $\tau_{i_0} = \{u, v, w\}$ and $\tau'_{j_0} = \{u', v', w'\}$ we have virtual edges in C and C'. We compare X and X' with respect to the ordering of these virtual edges $\mathsf{order}(\tau_{i_0})$ and $\mathsf{order}(\tau'_{j_0})$ in the codes C and C', respectively. This is described below in more detail. If we find an inequality, say $X < X'$ then C is defined to be the *smaller code*. Proceed with the next separating triples in the position-order until we ran through all of them.

We eliminate the codes which were found to be the larger codes at least once. In the end, the codes that are not eliminated are the *minimum codes*. If we have minimum codes for both F_i and F'_j then we define $S_{F_i} =_{\mathrm{F}} T_{F'_j}$. The construction of the codes also defines an isomorphism between the subgraphs associated to S_{F_i} and $T_{F'_j}$, i.e. $\mathsf{graph}(S_{F_i}) \cong \mathsf{graph}(T_{F'_j})$. For a single four-connected component this follows from [7]. If the trees contain several components, then our definition of $S_{F_i} =_{\mathrm{F}} T_{F'_j}$ guarantees that we can combine the isomorphisms of the components to an isomorphism between $\mathsf{graph}(S_{F_i})$ and $\mathsf{graph}(T_{F'_j})$.

Finally, we define the *orientation given to the parent separating triple of* F_i and F'_j as follows.

- We compute an *orientation graph* X_i with $V(X_i) = \tau$.
- For each pair in τ when taken as starting edge for the canonization of S_{F_i} which leads to a minimum code (among all the codes for these edges) we have a directed edge in $E(X_i)$ with color (1).
- Also for the r-th minimum codes we have a directed edge in $E(X_i)$ with color (r), for all $1 \leq r \leq 6$. Here, 6 is the number of directed edges in τ.

We define a new graph X_i with $V(X_i) = \tau$ and X'_j with $V(X'_j) = \tau'$. For each of the remaining minimum codes we have a unique starting edge which is also contained as a directed edge in X_i or X'_j, respectively. Every subtree rooted at a four-connected component node gives an orientation graph to the parent separating triple. If the orientation is consistent, then we define $S_\tau =_{\mathrm{F}} T_{\tau'}$ and show that the corresponding graphs are isomorphic.

Isomorphism order of two subtrees rooted at separating triple nodes. We consider the subtrees $S_{F_1}, \ldots, S_{F_k}$ and $T_{F'_1}, \ldots, T_{F'_k}$. We order them $S_{F_1} \leq_{\mathrm{F}} \cdots \leq_{\mathrm{F}} S_{F_k}$ and $T_{F'_1} \leq_{\mathrm{F}} \cdots \leq_{\mathrm{F}} T_{F'_k}$, and verify that $S_{F_i} =_{\mathrm{F}} T_{F'_i}$ for all i. If we find an inequality then the one with the smallest index i defines the order between S_τ and $T_{\tau'}$. For all i, assume now that $S_{F_i} =_{\mathrm{F}}$

$T_{F'_i}$ and, inductively, the corresponding split components are isomorphic, i.e. $\mathsf{graph}(S_{F_i}) \cong \mathsf{graph}(T_{F'_i})$. The next comparison concerns the orientation of τ and τ'. We already explained above the orientation given by each of the S_{F_i}'s to τ. We define a *reference orientation* for the root nodes τ and τ' which is given by their children. This is done as follows. We partition $(S_{F_1}, \ldots, S_{F_k})$ into classes of isomorphic subtrees, say $I_1 <_{\mathtt{F}} \ldots <_{\mathtt{F}} I_p$ for some $p \leq k$, and similarly we partition $(T_{F'_1}, \ldots, T_{F'_k})$ into $I'_1 <_{\mathtt{F}} \ldots <_{\mathtt{F}} I'_p$. It follows that I_j and I'_j contain the same number of subtrees for every j.

Consider the orientation given to τ by an isomorphism class I_j: For each child F_i which belongs to I_j we compute an *orientation graph* X_i with vertices $V(X_i) = \tau$. The orientation graph is defined as above but with the following changes. Instead of colors $(1), \ldots, (6)$ we have the colors $(j, 1), \ldots, (j, 6)$ for the edges. The *reference orientation given to* τ is defined as follows. We define the orientation graph X with vertices $V(X) = \tau$ and edges $E(X) = \bigcup_{1 \leq i \leq k} E(X_i)$ the disjoint union of the edges of the orientation graphs from all children of τ. Thus, X has multiple edges. We call X the *reference orientation graph for* τ.

Comparison of two orientation graphs. For τ and τ', the isomorphism ordering algorithm compares X and X' for isomorphism. Assume now τ and τ' have isomorphic subtrees and the nodes F and F' as parents. In this situation we return from recursion with $=_{\mathtt{F}}$ and give the orientation graphs X and X' to the parent. We went into recursion because the virtual edges of τ and τ' appeared in the same positions in the codes of the parent. In these codes, we have a complete ordering on the vertices of τ and τ'. Let $V(X) = \{u, v, w\}$ and let $\mathsf{order}(\tau) = u < v < w$ be an ordering of τ. We compute a list of counters for $(X, \mathsf{order}(\tau))$:

- We order the edges of X according to the order of their vertices, lexicographically. That is, $(u, v) < (u, w) < (v, u) < (v, w) < (w, u) < (w, v)$.
- Among directed edges with the same ends, we order them according to their color. That is, an edge with color (i_1, i_2) comes before an edge with color (j_1, j_2) if $(i_1, i_2) < (j_1, j_2)$ lexicographically.
- We define a counter for the number of edges with the same ends and the same color. For the edge (u, v) we have the counters $c_{(u,v),1}, \ldots, c_{(u,v),6p}$. Note, we have at most $6p$ colors because there are 6 colors for edges from orientation graphs of one isomorphism class and there are p isomorphism classes.
- We order the counters according to the order of the edges. That is, we have a list of counters $L(X, \mathsf{order}(\tau)) = (c_{(u,v),1}, c_{(u,v),2}, \ldots, c_{(u,v),6p}, \ldots, c_{(w,v),1}, \ldots, c_{(w,v),6p})$.

Note, among isomorphic graphs, there must be edges having the same color up to a permutation of them. Counting the colored edges allows to combine the orientations of all isomorphic subtrees. Note, if an orientation graph X_i for τ has two equal colored edges then there is an automorphism that maps the one edge to the other same colored edge in τ. The permutation of one edge completely fixes the whole automorphism of τ. Hence, also when counting the edges from different orientation graphs X_1 and X_2, if w.l.o.g. there are the edges (u, v) with colors 1 and 2 then the mapping of (u, v) to other edges completely fixes the whole automorphism among τ and whether X_1 and X_2 are swapped. With an inductive argument, this can be generalized to the whole orientation graph X. Let X' be the corresponding reference orientation graph for τ'. We define the isomorphism order $(X, \mathsf{order}(\tau)) < (X', \mathsf{order}(\tau'))$ exactly when $L(X, \mathsf{order}(\tau)) < L(X', \mathsf{order}(\tau'))$ lexicographically. The preceding discussion leads to the following theorem.

THEOREM 16. *The 3-connected non-planar graphs G and H which contained 3-divisive separating triples are isomorphic if and only if there is a choice of separating triples τ, τ' in G and H such that $S_\tau =_{\mathrm{F}} T_{\tau'}$ when rooted at τ and τ', respectively.*

4.3 Complexity of the isomorphism order algorithm and canonization

For a log-space implementation, there are two tasks: We limit the number of choices for roots of triconnected and four-connected component trees, and ensure that nothing is stored on the work-tape while recursing on a large child. For the first task, we modify the algorithm of [8] to accommodate non-planar 3-connected components. The two root finding procedures are interdependent. We bound the number of roots for the four-connected component tree with respect to the number of child separating pairs and child articulation points of tri- and biconnected subtrees. The algorithm is based on an intricate case analysis which has to be extended to work with respect to the three tree structures.

For the second task, we extend the ideas from [8], because for the analysis of large children we consider the bi- tri- and four-connected component trees simultaneously. In the trees, the recursion goes from depth d to $d+2$ and large children are handled a priori at any level.For the space requirement of our algorithm we get:

$$\mathcal{S}(N) = \max_j \ \mathcal{S}\left(\frac{N}{k_j}\right) + O(\log k_j),$$

where $k_j \geq 2$ (for all j) is the number of bi-, tri- or four-connected subtrees having the same size. Hence, $\mathcal{S}(N) = O(\log N)$. It is helpful to imagine that we have three work-tapes, which are used when we go into recursion at articulation point nodes, separating pair nodes, and separating triple nodes respectively. We canonize K_5-free graphs exactly the same way as planar graphs. Thus we get

THEOREM 17. *The isomorphism order between two K_5-free graphs can be computed in log-space. The canonization of K_5-free graphs can be done in log-space.*

Acknowledgment We thank V. Arvind, Bireswar Das, Raghav Kulkarni, Nutan Limaye, Meena Mahajan and Jacobo Torán for helpful discussions.

References

[1] V. Arvind, Bireswar Das, and Johannes Köbler. A logspace algorithm for partial 2-tree canonization. In *Computer Science Symposium in Russia (CSR)*, 2008.

[2] V. Arvind and Piyush P. Kurur. Graph isomorphism is in SPP. *Information and Computation*, 204(5), 2006.

[3] Tetsuo Asano. An approach to the subgraph homeomorphism problem. *Theoretical Computer Science*, 38, 1985.

[4] László Babai and Eugene M. Luks. Canonical labeling of graphs. In *Proceedings of the 15th annual ACM symposium on Theory of computing (STOC)*, 1983.

[5] R. B. Boppana, J. Hastad, and S. Zachos. Does co-NP have short interactive proofs? *Information Processing Letters*, 25(2), 1987.

[6] Stephen A. Cook. A taxonomy of problems with fast parallel algorithms. *Information and Control*, 64(1-3), 1985.

[7] Samir Datta, Nutan Limaye, and Prajakta Nimbhorkar. 3-connected planar graph isomorphism is in log-space. In *Proceedings of the 28th annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, 2008.

[8] Samir Datta, Nutan Limaye, Prajakta Nimbhorkar, Thomas Thierauf, and Fabian Wagner. Planar graph isomorphism is in log-space. Technical Report TR09-052, Electronic Colloquium on Computational Complexity (ECCC), 2009.

[9] Samir Khuller. Parallel algorithms for K_5-minor free graphs. Technical Report TR88-909, Cornell University, Computer Science Department, 1988.

[10] Johannes Köbler, Uwe Schöning, and Jacobo Torán. *The Graph Isomorphism Problem*. Birkhäuser, 1993.

[11] Steven Lindell. A logspace algorithm for tree canonization (extended abstract). In *Proceedings of the 24th annual ACM Symposium on Theory of Computing (STOC)*, 1992.

[12] Pierre McKenzie, Birgit Jenner, and Jacobo Torán. A note on the hardness of tree isomorphism. In *Proceedings of the 13th Annual IEEE Conference on Computational Complexity (CCC)*. IEEE Computer Society, 1998.

[13] Gary L. Miller and V. Ramachandran. A new graph triconnectivity algorithm and its parallelization. In ACM, editor, *Proceedings of the nineteenth annual ACM Symposium on Theory of Computing*, pages 335–344, 1987.

[14] Ilia N. Ponomarenko. The isomorphism problem for classes of graphs closed under contraction. *Journal of Mathematical Sciences (JSM, formerly Journal of Soviet Mathematics)*, 55, 1991.

[15] Omer Reingold. Undirected st-connectivity in log-space. In *Proceedings of the 37th annual ACM Symposium on Theory of Computing (STOC)*, 2005.

[16] Uwe Schöning. Graph isomorphism is in the low hierarchy. *Journal on Computing and System Sciences*, 37(3), 1988.

[17] Thomas Thierauf and Fabian Wagner. Reachability in $K_{3,3}$-free graphs and K_5-free graphs is in unambiguous log-space. Technical Report TR09-029, Electronic Colloquium on Computational Complexity (ECCC), 2009.

[18] Jacobo Torán. On the hardness of graph isomorphism. *SIAM Journal on Computing*, 33(5), 2004.

[19] William T. Tutte. *Connectivity in graphs*. University of Toronto Press, 1966.

[20] Vijay V Vazirani. NC algorithms for computing the number of perfect matchings in $K_{3,3}$-free graphs and related problems. *Information and Computation*, 80, 1989.

[21] Fabian Wagner. Hardness results for tournament isomorphism and automorphism. In *32nd International Symposium on Mathematical Foundations of Computer Science (MFCS)*, 2007.

[22] Klaus Wagner. Über eine Eigenschaft der ebenen Komplexe. In *Mathematical Annalen*, volume 114, 1937.

[23] Hassler Whitney. A set of topological invariants for graphs. *American Journal of Mathematics*, 55, 1933.

This work is licensed under the Creative Commons Attribution-NonCommercial-No Derivative Works 3.0 License.