

Reachability in $K_{3,3}$ -free Graphs and K_5 -free Graphs is in Unambiguous Log-Space

Thomas Thierauf
Fak. Elektronik und Informatik
HTW Aalen
73430 Aalen, Germany

Fabian Wagner*
Inst. für Theoretische Informatik
Universität Ulm
89069 Ulm, Germany

{thomas.thierauf,fabian.wagner}@uni-ulm.de

April 3, 2009

Abstract

We show that the reachability problem for directed graphs that are either $K_{3,3}$ -free or K_5 -free is in unambiguous log-space, $\text{UL} \cap \text{coUL}$. This significantly extends the result of Bourke, Tewari, and Vinodchandran that the reachability problem for directed planar graphs is in $\text{UL} \cap \text{coUL}$.

Our algorithm decomposes the graphs into biconnected and triconnected components. This gives a tree structure on these components. The non-planar components are replaced by planar components that maintain the reachability properties. For K_5 -free graphs we also need a decomposition into fourconnected components. A careful analysis finally gives a polynomial size planar graph which can be computed in log-space.

We show the same upper bound for computing distances in $K_{3,3}$ -free and K_5 -free directed graphs and for computing longest paths in $K_{3,3}$ -free and K_5 -free directed acyclic graphs.

1 Introduction

For undirected graphs, the reachability problem is L-complete [Rei05]. For general graphs, reachability is NL-complete. Bourke, Tewari and Vinodchandran [BTV07] proved that reachability on planar graphs is in $\text{UL} \cap \text{coUL}$ and is hard for L. Jacoby and Tantau [JT07] showed for series-parallel graphs that reachability is complete for L. They also showed that the problem to compute distances between vertices or longest paths are complete for L. Thierauf and Wagner [TW08] proved that the distance problem for planar graphs is in $\text{UL} \cap \text{coUL}$. For general graphs and even undirected planar graphs, the longest path problem is complete for NP. It is NL-complete for directed acyclic graphs (DAG). Limaye, Mahajan and Nimbhorkar [LMN09] prove that longest paths in planar DAGs can be computed in $\text{UL} \cap \text{coUL}$.

We study reachability on extensions of planar graphs. Our main result is that reachability for directed $K_{3,3}$ -free graphs and directed K_5 -free graphs logspace-reduces to planar reachability. Thus, the current upper bound for planar reachability, $\text{UL} \cap \text{coUL}$, carries over to reachability for directed $K_{3,3}$ -free graphs and directed K_5 -free graphs.

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

In the case of a $K_{3,3}$ -free graph G , our technique is to decompose G into biconnected components. Then these biconnected components are decomposed further into planar components triconnected components and K_5 -components. We construct a tree where the nodes are associated with these components (cf. [Vaz89]), the *PlaK₅-component tree*.

For the reachability problem from node s to t in graph G we consider the simple path P in the PlaK₅-component tree from component nodes S to T , where s and t are contained, respectively. We split the graph into components along the separating pairs we have on path P . A path from s to t in G must contain vertices of all these separating pairs. Thus, we make the reachability test for all these components. The difficulty is to handle the non-planar components. The crucial step is to replace all the K_5 -components in the tree by planar components such that the reachability is not changed. Then we recombine the planar components into a planar graph H such that there is a path from s to t in G if and only if this holds too in H . The construction can be carried out in log-space.

There also exists a decomposition of K_5 -free graphs (cf. Khuller [Khu88]). This is obtained by decomposing the graph into triconnected components. Each triconnected component is either planar, the four-rung Mobius ladder, also called V_8 , or it is constructed by taking 3-clique sums of planar 4-connected components [Wag37]. We replace the V_8 -components by planar components such that the reachability is not changed. Then we recombine the planar components into a planar graph. A difficulty that arises here is that we cannot use the 3-clique sum to recombine the components, because this would result again in a nonplanar graph. Instead, we carefully add copies of the components that can be arranged in a planar way such that the reachability is not altered. All the steps can be accomplished in log-space.

It is easy to see that our transformations from $K_{3,3}$ -free or K_5 -free graphs to planar graphs maintain not just reachability, but also the distances of the vertices. Therefore it follows from our results that distances in $K_{3,3}$ -free or K_5 -free graphs can be computed $\text{UL} \cap \text{coUL}$.

The same is true with respect to longest paths when considering DAGs instead. This is easy to see in the case of $K_{3,3}$ -free DAGs and requires some extra arguments in the case of K_5 -free DAGs. Hence, longest paths in $K_{3,3}$ -free or K_5 -free DAGs can be computed $\text{UL} \cap \text{coUL}$.

2 Definitions and Notations

A graph $G = (V, E)$ contains a set of vertices V and edges E . For $U \subseteq V$ let $G - U$ be the *induced subgraph* of G on $V - U$.

An undirected graph G is *connected* if there exists a path between any two vertices. A vertex v is an *articulation point* if $G - \{v\}$ is not connected. The connected components of $G - \{v\}$ are called the *split components* of v . G is *biconnected* if it contains no articulation points.

Let G be a biconnected graph. A pair of vertices u, v is a *separating pair* if $G - \{u, v\}$ is not connected. The connected components of $G - \{u, v\}$ are called the *split components* of u, v . G is *3-connected* if it contains no separating pairs, i.e. there are three vertex-disjoint paths between any pair in 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.

Let G be a 3-connected graph. A triple of vertices u, v, w is a *separating triple* if $G - \{u, v, w\}$ is not connected. The connected components of $G - \{u, v, w\}$ are called the *split components* of u, v, w . G is *4-connected* if it contains no separating triples.

A *$K_{3,3}$ -free graph* is an undirected graph which does not contain a $K_{3,3}$ as a minor. A

K_5 -free graph is an undirected graph which does not contain a K_5 as a minor.

Tree decomposition of connected planar graphs. The decomposition of graphs into their biconnected components is used in many applications. For example, for computing the number of perfect matchings of $K_{3,3}$ -free [Vaz89] or for the canonization of planar graphs [KHC04, DLN⁺08a]. We use the decomposition in the context of reachability testing. The first step is to decompose the connected graph G into biconnected components.

The tree decomposition of a connected planar graph from Hopcroft and Tarjan [HT73] is defined as follows. A connected planar graph G can be decomposed into biconnected components. For this find the set of articulation points. Start with an arbitrary articulation point a of G and consider the split components in $G - \{a\}$. In all the split components, a copy of a is contained. Recursively decompose the split components this way. The resulting connected components contain no articulation points and are biconnected. We define nodes for these components and the articulation points and a new graph as follows. An *articulation point node* is connected to a *biconnected component node* if this articulation point is contained as vertex in the corresponding component. The resulting graph is a tree, the *biconnected component tree*.

Lemma 2.1 *The biconnected component tree of an undirected graph can be computed in log-space.*

Proof: We can find all the articulation points of G in log-space: a vertex a is an articulation point if we can find two further vertices u and v such that there is no path from u to v in $G - \{a\}$.

Similarly, we can compute all biconnected components of G : two vertices u and v are in the same biconnected component, if there is a path from u to v in $G - \{a\}$, for every articulation point $a \notin \{u, v\}$ of G .

The biconnected component tree has an arbitrary articulation point as root, say a_0 . To walk along the tree, we have to identify the *parent articulation point* and the *child articulation points* of a biconnected component with respect to the root a_0 . Let B be a biconnected component with articulation points a_1, \dots, a_k . Then a_i is the parent articulation point of B , if there is a path from a_i to a_0 in $G - a_j$, for all $j \neq i$. Hence, we can traverse the biconnected component tree in log-space. \square

Tree decomposition of biconnected planar graphs. For biconnected graphs, Hopcroft and Tarjan [HT73] introduced the decomposition into separating pairs and *triconnected components*. The latter are cycles, 3-bonds and 3-connected components. Separating pairs will be connected by a *virtual edge*. The 3-bonds come from separating pairs that are connected by an edge in the given graph. Cycles are a special case. Although cycles are not 3-connected, they are not further decomposed for technical reasons. Hopcroft and Tarjan [HT73] defined the *triconnected component tree* that has these components as nodes and showed that it can be computed in linear time. There is an edge between the separating pair node for (a, b) and a triconnected component node for G_0 , if (a, b) is contained in G_0 (connected by a virtual edge). The resulting graph on these nodes is a tree \mathcal{T} . We consider an arbitrary separating pair as the root node of \mathcal{T} . For an example see Figure 1. Datta et. al. [DLN⁺08a] showed that when the graph is in addition *planar*, then the triconnected component tree can be computed even in log-space.

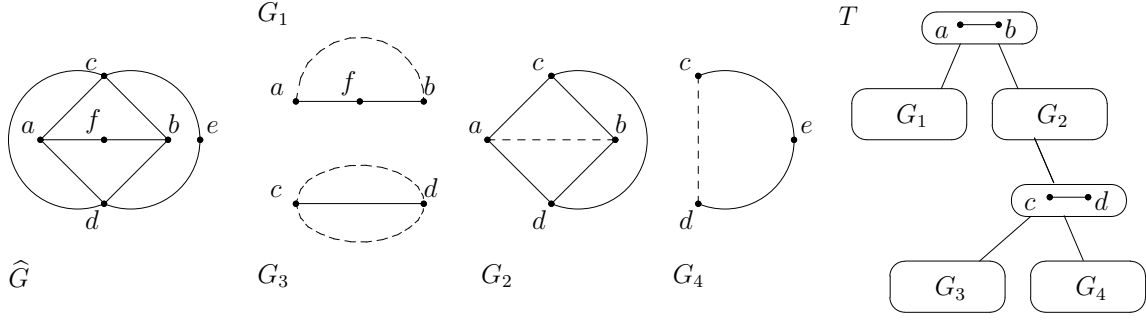


Figure 1: ([DLN⁺08a]) The decomposition of a biconnected planar graph G . Its triconnected components are G_1, \dots, G_4 and the corresponding decomposition into the tree T of triconnected components. The split pairs are (a, b) and (c, d) . Since the 3-connected separating pair (c, d) is connected by an edge in G , we also get $\{c, d\}$ as triple-bond G_3 . The virtual edges corresponding to the 3-connected separating pairs are drawn with dashed lines.

The size of a component tree. Let T be a biconnected or a triconnected component tree. We define the size of such a tree. The *size of an individual component node* of T is the number of nodes in the component. The *size of the tree T* , denoted by $|T|$, is the sum of the sizes of its component nodes.

Let \mathcal{T}_C be a component tree rooted at some component node C and let $\mathcal{T}_{C'}$ be a subtree of \mathcal{T} rooted at a child C' of C . We call C' a *large child* of C , if $|\mathcal{T}_{C'}| > |\mathcal{T}_C|/2$.

Reachability. Let \mathcal{G} be a class of directed graphs. We consider the following problems restricted to \mathcal{G} .

$$\begin{aligned} \mathcal{G}\text{-Reachability} &= \{ (G, s, t) \mid G \in \mathcal{G} \text{ contains a path from } s \text{ to } t \} \\ \mathcal{G}\text{-Distance} &= \{ (G, s, t, k) \mid G \in \mathcal{G} \text{ contains a path from } s \text{ to } t \text{ of length } \leq k \} \\ \mathcal{G}\text{-Long-Path} &= \{ (G, s, t, k) \mid G \in \mathcal{G} \text{ contains a simple path from } s \text{ to } t \text{ of length } \geq k \} \end{aligned}$$

\mathbb{L} is the class of languages accepted by deterministic log-space Turing-machines and \mathbb{NL} by nondeterministic log-space Turing-machines. \mathbb{UL} is the class of languages accepted by unambiguous nondeterministic log-space machines, i.e. there exists at most one accepting computation path. coUL is the class of complements of languages in \mathbb{UL} . We also use the fact that $\mathbb{L}^{\mathbb{UL} \cap \text{coUL}} = \mathbb{UL} \cap \text{coUL}$ (c.f. Thierauf and Wagner [TW08]).

3 Reachability in $K_{3,3}$ -free Graphs

We give a logspace reduction from the reachability problem for directed $K_{3,3}$ -free graphs to the reachability problem for directed planar graphs. The latter problem is known to be in $\mathbb{UL} \cap \text{coUL}$ [BTV07].

For the reduction, we decompose the given graph G into triconnected components. For the decomposition, we consider G as *undirected*. That is, each directed edge of G is considered as an undirected edge. After the decomposition we consider the components again as directed graphs.

We will see that every non-planar component is precisely the K_5 . We define a tree based on planar and K_5 -components. There are *component nodes* for the planar components and the K_5 -components which are connected via nodes for separating pairs of the components. We call this tree the *Pla K_5 -component tree*.

The key step in the reduction is to replace the K_5 -components by planar components while maintaining the reachability properties. Then we recombine the components to a directed planar graph H . The construction is such that node t is reachable from node s in G if and only if this holds in H , too.

3.1 Tree decomposition of biconnected graphs

The decomposition of graphs into their biconnected components is used in many applications. For example, for computing the number of perfect matchings of $K_{3,3}$ -free [Vaz89] or for the canonization of planar graphs [KHC04, DLN⁺08a]. We use the decomposition in the context of reachability testing. The first step is to decompose the connected graph G into biconnected components.

Lemma 3.1 *The biconnected component tree of an undirected graph can be computed in log-space.*

Proof: We can find all the articulation points of G in log-space: a vertex a is an articulation point if we can find two further vertices u and v such that there is no path from u to v in $G - \{a\}$.

Similarly, we can compute all biconnected components of G : two vertices u and v are in the same biconnected component, if there is a path from u to v in $G - \{a\}$, for every articulation point $a \notin \{u, v\}$ of G .

The biconnected component tree has an arbitrary articulation point as root, say a_0 . To walk along the tree, we have to identify the *parent articulation point* and the *child articulation points* of a biconnected component with respect to the root a_0 . Let B be a biconnected component with articulation points a_1, \dots, a_k . Then a_i is the parent articulation point of B , if there is a path from a_i to a_0 in $G - a_j$, for all $j \neq i$. Hence, we can traverse the biconnected component tree in log-space. \square

For biconnected graphs, Hopcroft and Tarjan [HT73] introduced the decomposition into triconnected components, namely cycles, bonds and 3-connected components. They defined the triconnected component tree that has these components as nodes, and showed that it can be computed in linear time. Datta et. al. [DLN⁺08a] showed that when the graph is in addition *planar*, then the triconnected component tree can be computed even in log-space.

We consider $K_{3,3}$ -free graphs. Hence, there can be K_5 -minors. Tutte [Tut66] proved that the decomposition of a $K_{3,3}$ -free graph into triconnected components is unique. Moreover, Asano [Asa85] proved that it has the following form.

Lemma 3.2 [Asa85] *Each triconnected component of a $K_{3,3}$ -free graph is either planar or exactly the graph K_5 .*

The triconnected components are the nodes of the triconnected component tree. Two nodes are connected by an edge, if they share a separating pair. For our purpose it suffices to distinguish between planar and non-planar components. Vazirani [Vaz89] recombines the *planar* triconnected components that are neighbors in the tree into one planar component.

This defines a new tree with alternating planar and K_5 -component nodes which we call the *Pla K_5 -component tree*. Vazirani [Vaz89] showed that the Pla K_5 -component tree is unique and can be computed in NC_2 . Here we give a simpler and more direct construction that works in log-space. We use Lemma 3.2 and the following lemmas which are implicitly in [Asa85].

Lemma 3.3 *Let $K = \{v_1, \dots, v_5\}$ be a K_5 -minor in graph G . Let $p_{i,j}$ be a simple path from v_i to v_j , for all $i \neq j$, such that these paths are pairwise vertex disjoint (except for their endpoints), and let $P = \bigcup_{v_i, v_j \in K} p_{i,j}$.*

If there are two vertices a, b and a path $p_{a,b}$ from a to b that is vertex disjoint from all paths in P (except for their endpoints) such that

(i) *a is an intermediate point on some path in P , w.l.o.g. say $p_{1,2}$ and*

(ii) *either $b \in \{v_3, v_4, v_5\}$ or b is an intermediate point on some path $p_{i,j} \neq p_{1,2}$,*

then G has a $K_{3,3}$ -minor.

Proof: We consider the two cases for node b separately. First, let $b \in \{v_3, v_4, v_5\}$, w.l.o.g. say $b = v_3$. Then we have a $K_{3,3}$ -minor in G with $\{v_1, v_2, v_3\}$ on the left side and $\{v_4, v_5, a\}$ on the right side. See Figure 2 (a).

For the second case, let b be an intermediate point on some path $p_{i,j} \neq p_{1,2}$. At least one of v_i and v_j is different from v_1 and v_2 , say v_i . We distinguish two cases. The cases are shown in Figure 2 (b) and (c).

- $v_j \notin \{v_1, v_2\}$. Then there is a $K_{3,3}$ -minor in G with $\{v_1, v_2, b\}$ on the left side and $\{v_i, v_j, a\}$ on the right side
- $v_j \in \{v_1, v_2\}$. Then there is a $K_{3,3}$ -minor in G that consists of the nodes $\{a, v_1, \dots, v_5\}$ with $\{v_1, v_2, v_i\}$ on the left side and the remaining nodes on the right side. Note that we have a path from a to v_i by following path $p_{a,b}$ to b and then going along path $p_{i,j}$ to v_i . By construction this path is vertex disjoint from all paths in $P - \{p_{1,2}, p_{i,j}\}$ (except for their endpoints). Note also that we do not need path $p_{i,j}$ otherwise, because v_i and v_j are on the same side of the $K_{3,3}$ -minor.

□

The next lemma is similar to Lemma 3.3 but takes a slightly different point of view.

Lemma 3.4 *Let $G = (V, E)$ be a biconnected undirected graph. Let $K \subseteq V$ be a K_5 -minor in G . If there is a node $u \notin K$ such that there are pairwise vertex disjoint paths from u to three nodes in K then G has a $K_{3,3}$ -minor.*

Proof: Let $K = \{v_1, \dots, v_5\}$. Because K is a K_5 -minor, there is a simple path $p_{i,j}$ between v_i and v_j for all $i \neq j$, such that these paths are vertex disjoint (except for their endpoints). Let $P = \bigcup_{v_i, v_j \in K} p_{i,j}$. Let furthermore $p_{u,1}, p_{u,2}, p_{u,3}$ be pairwise vertex disjoint paths from u to three vertices in K , say w.l.o.g. to v_1, v_2, v_3 . We assume that these paths do not pass through any of the other two vertices of K , i.e. v_4 or v_5 . Otherwise shorten such a path and rename the vertices. We distinguish three cases.

Case 1: $p_{u,1}, p_{u,2}, p_{u,3}$ reach w.l.o.g. v_1, v_2, v_3 respectively, without intersecting any of the paths $p_{i,j}$ in beforehand. Then we have a $K_{3,3}$ -minor with $\{v_1, v_2, v_3\}$ on the left side and $\{v_4, v_5, u\}$ on the right side, see Figure 3 (a).

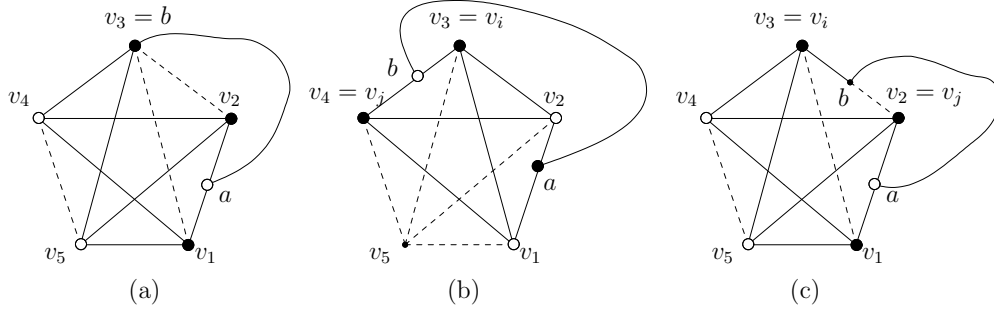


Figure 2: $K = \{v_1, \dots, v_5\}$ is a K_5 -minor. (a) There is a path from a to $b = v_3$ that is disjoint from all paths in P . Then the black and white nodes are the two sides of a $K_{3,3}$ -minor. The paths that are drawn dashed do not contribute to the $K_{3,3}$ -minor. (b) There is a path from a to b that is disjoint from all paths in P such that b is on some path $p_{i,j}$ where $v_i, v_j \notin \{v_1, v_2\}$. Then there is a $K_{3,3}$ -minor as indicated. (c) The same situation as in (b), but with $v_j \in \{v_1, v_2\}$.

Case 2: One of the paths $p_{u,1}, p_{u,2}, p_{u,3}$ intersects ≥ 2 paths in P before reaching the end vertex. Let a and b be two such consecutive intersection points on two different paths of P . Then we have a $K_{3,3}$ -minor by Lemma 3.3.

Case 3: In the remaining case, all paths $p_{u,1}, p_{u,2}, p_{u,3}$ intersect at most one path in P and at least one of the paths, say $p_{u,1}$, intersects some path of P , say path $p_{i,j}$ at an intermediate point. Let b be this point of intersection.

At least one of the paths $p_{u,1}, p_{u,2}, p_{u,3}$ does not have v_i or v_j as an endpoint. Let $p_{u,k}$ be such a path, for some $k \in \{1, 2, 3\}$, and let a be the first intersection of $p_{u,k}$ with some path in P . Note, that a might be in K . We distinguish the following subcases.

- $a \in p_{i,j}$. In this case we have a path from a to v_k that is vertex disjoint from the paths in P , see Figure 3 (b). Then there is a $K_{3,3}$ -minor by Lemma 3.3.
- $a \notin p_{i,j}$. This also includes the case that $a \in K$. Then we have a path from a to b via u that is vertex disjoint from the paths in P , see Figure 3 (c). Again there is a $K_{3,3}$ -minor by Lemma 3.3.

□

There are some easy consequences of Lemma 3.4. We consider how many nodes the components of a $\text{Pla}K_5$ -component tree can have in common.

Corollary 3.5 [Asa85] *A nonplanar 3-connected graph G with ≥ 6 vertices has a $K_{3,3}$ -minor.*

Proof: Assume that G is nonplanar because it contains a K_5 -minor. Let $K = \{v_1, \dots, v_5\}$ be a K_5 -minor in G . Let w be a vertex in G that is not in K . There must be paths p_1, p_2, p_3 from w to v_1, v_2, v_3 in C , respectively, such that path p_1 goes from w to v_1 without passing through v_2 or v_3 , and similarly from w to p_2 and p_3 . Note that if all paths from w to, say v_3 would pass through v_1 or v_2 , then v_1, v_2 would be a separating pair for G and G would not be 3-connected.

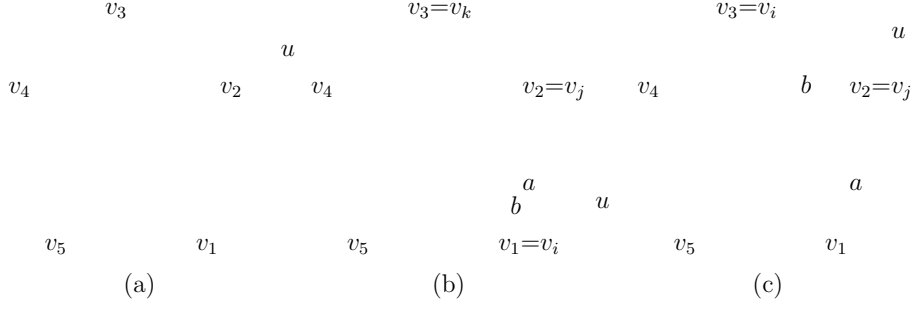


Figure 3: $K = \{v_1, \dots, v_5\}$ is a K_5 -minor. (a) There are paths from u to v_1, v_2, v_3 that are disjoint from all paths in P . Then the black and white nodes are the two sides of a $K_{3,3}$ -minor. The paths that are drawn dashed do not contribute to the $K_{3,3}$ -minor. (b) There is a path from $a \in p_{1,2}$ to $v_k \notin \{v_1, v_2\}$ (here $v_k = v_3$) which is disjoint from all paths in P . Then there is a $K_{3,3}$ -minor as indicated. (c) The paths from u intersect two different paths of the K_5 -minor at a and b , respectively. Then there is again a $K_{3,3}$ -minor as indicated, where the path from a to $v_k = v_3$ passes through u and b .

Next, go along p_i , starting from v_i for all i . Let u_1 be the first vertex on p_1 that intersects p_2 or p_3 . Let u_2 be the first vertex on p_2 that intersects p_1 or p_3 . There are two cases.

- $u_1 = u_2$. In this case we define $u = u_1$.
- $u_1 \neq u_2$. Then both vertices also belong to p_3 . Let u be the one which comes first on p_3 starting from v_3 .

Then we can use p_1, p_2, p_3 to construct three vertex disjoint paths from u to v_1, v_2, v_3 , respectively. By Lemma 3.4, there is a $K_{3,3}$ -minor in G . □

Corollary 3.6 *Two (different) K_5 -minors of a $K_{3,3}$ -free graph G have ≤ 2 vertices in common.*

Proof: Let K_1, K_2 be two K_5 -minors that have ≥ 3 vertices in common. Then G has a 3-connected component that contains K_1 and K_2 . This component has ≥ 6 vertices and therefore contains a $K_{3,3}$ by Corollary 3.5. □

Corollary 3.7 *Let $G = (V, E)$ be a $K_{3,3}$ -free biconnected undirected graph. Let K be a K_5 -minor and B be a biconnected planar component in G . Then K and B have ≤ 2 vertices in common.*

Proof: A biconnected planar component B consists of triconnected components that are connected by separating pairs. Note that by Corollary 3.5, each 3-connected planar component has ≤ 2 vertices in common with K . Hence, if B contains three or more vertices of K , then these vertices must be distributed in more than one 3-connected component of B . By a similar argument as in Corollary 3.5 we can find a node u such that there are vertex disjoint

paths in B to three nodes of K . The point is that at least one of the paths must leave a 3-connected component via a separating pair and can therefore be made disjoint from the other two paths. \square

The next lemma is the crucial lemma for identifying the K_5 -minors in a $K_{3,3}$ -free biconnected graph.

Lemma 3.8 *Let $G = (V, E)$ be a $K_{3,3}$ -free biconnected undirected graph. A set $K \subseteq V$ of 5 vertices is a K_5 -minor in G , and hence a K_5 -component in the $\text{Pla}K_5$ -component tree, if and only if for every pair $u, v \in K$ either $(u, v) \in E$ or $\{u, v\}$ is a separating pair in G such that the three remaining vertices of K are all in one split component of $G - \{u, v\}$.*

Proof: For the direction from right to left let $u, v \in K$ be a separating pair. By definition a separating pair has at least two split components. Then there is a path from u to v in G that passes purely through a split component of $G - \{u, v\}$ that does *not* contain the other three vertices of K (except for the endpoints). We have analogous paths for all separating pairs with both vertices in K . By construction, these paths are pairwise vertex disjoint (except for the endpoints). Hence K is a K_5 -minor in G .

For the direction from left to right let $K = \{v_1, \dots, v_5\}$ be a K_5 -minor in G and let $(v_1, v_2) \notin E$. Assume, that $\{v_1, v_2\}$ is not a separating pair in G . We show that in this case G has a $K_{3,3}$ -minor which contradicts the assumption that G is $K_{3,3}$ -free.

Because K is a K_5 -minor, there is a simple path $p_{i,j}$ between v_i and v_j for all $i \neq j$ such that these paths are vertex disjoint (except for their endpoints). Let $a \neq v_1, v_2$ be a node on path $p_{1,2}$. Since $\{v_1, v_2\}$ is not a separating pair, there is a simple path $p_{a,3}$ from a to v_3 which does not pass through v_1 or v_2 . By Lemma 3.4, there is a $K_{3,3}$ -minor in G . \square

As a consequence, we can compute all the K_5 -components of a biconnected undirected graph in log-space: cycle through all the $\binom{n}{5}$ sets of 5 nodes and check the condition of Lemma 3.8 for each set. With the K_5 -components in hand, we show that we can compute the $\text{Pla}K_5$ -component tree in log-space. We start with a technical lemma.

Lemma 3.9 *Let $G = (V, E)$ be a $K_{3,3}$ -free biconnected undirected graph. The $\text{Pla}K_5$ -component tree of G can be computed in log-space.*

Proof: We have already seen how to compute the K_5 components. We show how to compute the planar components. Two vertices u and v are in the same planar component, if there is a path p from u to v in G such that p contains ≤ 2 vertices from any K_5 -component. Note that by Corollary 3.7, path p can go through at most two vertices of a K_5 -component. Intuitively, this allows p to touch a K_5 -component, but not to go through the other components connected to the K_5 -component.

The $\text{Pla}K_5$ -component tree of G consists of

- K_5 -component nodes,
- planar component nodes, and
- K_5 -separating pair nodes. These are pairs of vertices that are separating pairs and belong to a K_5 .

The tree has alternating K_5 -component nodes and planar component nodes and a K_5 -separating pair node between these two. That is, there is an edge between a K_5 - or planar component node and a K_5 -separating pair node if the corresponding component contains both vertices of the separating pair. Figure 4 shows an example.

Let the tree be rooted at node N and let N' be a component node in the tree. The parent node of N' is the K_5 -separating pair u, v incident to N' in the tree such that there is no path from vertices in N' to vertices to the root N in $G - \{u, v\}$. All the other K_5 -separating pairs are children of N' . Hence, we can navigate in log-space through the tree. \square

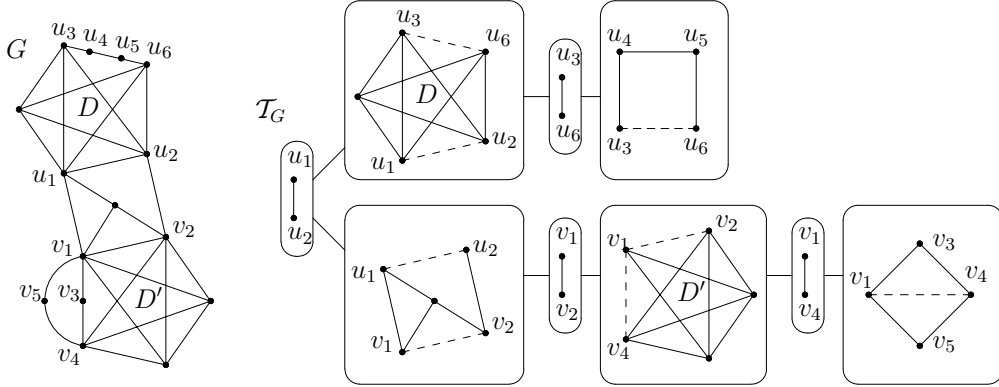


Figure 4: Two K_5 -component nodes D and D' with planar components.

We summarize.

Theorem 3.10 [DLN⁺08b]¹ *The decomposition of a $K_{3,3}$ -free biconnected graph into a planar component tree can be computed in log-space.*

3.2 Reduction to the planar case

In this section, we construct a reduction from the reachability problem for directed $K_{3,3}$ -free graphs to the reachability problem for directed planar graphs. We prove the following theorem.

Theorem 3.11 *$K_{3,3}$ -free Reachability \leq_T^L Planar Reachability.*

We start by showing that it suffices to consider *biconnected* directed $K_{3,3}$ -free graphs.

Lemma 3.12 *Reachability \leq_T^L biconnected Reachability.*

Proof: Let G be a graph and s and t be two given nodes in G . Compute the biconnected components of G in log-space by Lemma 3.1. Let S be the biconnected component that contains s , and T the one that contains t . Let $S = V_1, \dots, V_k = T$ with k odd be a simple path from S to T of nodes of the biconnected component tree of G . Let a_2, a_4, \dots, a_{k-1} be the articulation points shared by these components, i.e. a_i is shared by V_{i-1} and V_{i+1} . Recall,

¹The result came up in discussion with Samir Datta, Nutan Limaye, and Prajakta Nimbhorkar. The proof presented here was developed in this paper.

this path is an alternating path on articulation point nodes and separating pair nodes. Then a path from s to t has to go through all these articulation points. Hence, there is a path from s to t in G if and only if there is a path from a_{i-1} to a_{i+1} in the biconnected component of V_i , for all odd i . \square

Note that the reduction in the proof considers subgraphs of G . Hence the reduction maintains the property that G is $K_{3,3}$ -free or K_5 -free, respectively.

Let $G = (V, E)$ be a biconnected directed $K_{3,3}$ -free graph and s, t be two vertices in G . The problem is to find a path from s to t in G . Let \mathcal{T}_G be the PlaK_5 -component tree of G . Let S be the biconnected component that contains s and T the one that contains t .

We partition the tree into subtrees and consider the reachability problem for these subtrees. Then we replace *non-planar* components of the PlaK_5 -component tree \mathcal{T}_G by *planar* components such that the reachability condition remains unchanged.

Partitioning of G into subgraphs. Consider the simple path from S to T in \mathcal{T}_G , say $S = C_1, C_2, \dots, C_l = T$. A path from s to t always contains vertices of separating pairs, which are shared by the component nodes C_{i-1} and C_{i+1} . Let (u_i, v_i) be the separating pair which separates C_{i-1} from C_{i+1} . For an example see Figure 5.

Observe, that a path p from s to t must visit at least one vertex of each of these separating pairs. Once we have reached C_i , then p will not go back to C_{i-1} , because otherwise p would not be simple. Note, p goes through the siblings of C_{i-1} before it goes to the parent C_{i+1} .

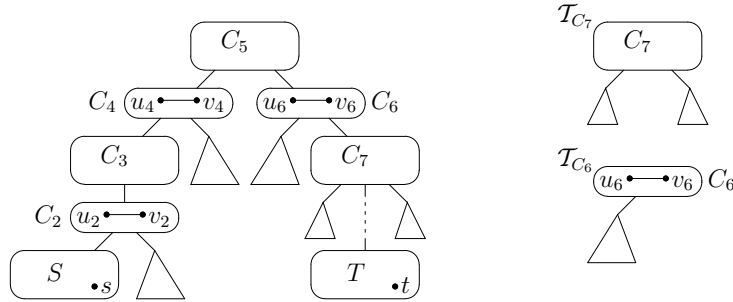


Figure 5: Partitioning of the PlaK_5 -component tree \mathcal{T}_G into pieces \mathcal{T}_{C_i} . The boxes indicate component nodes and the triangles indicate subtrees.

We partition the reachability problem into subproblems. For a component node C_i define the tree \mathcal{T}_{C_i} as the subtree of \mathcal{T}_G rooted at C_i , where the branches to C_{i-1} and C_{i+1} are cut off. Let G_i be the graph corresponding to \mathcal{T}_{C_i} . If C_i is a component node then $\{u_{i-1}, v_{i-1}\}$ and $\{u_{i+1}, v_{i+1}\}$ are separating pairs of C_i . In the case, when C_i is a separating pair node then we define $u_{i-1} = u_{i+1}$ and $v_{i-1} = v_{i+1}$ such that we have only one separating pair in C_i .

The following lemma states that the reachability problem in G can be partitioned into reachability problems in G_i .

Lemma 3.13 Any simple path p from s to t in G can be written as a concatenation of paths, $p = p_1, \dots, p_l$, such that

- path p_1 goes from s to u_2 or v_2 in G_1 ,

- path p_i is a path from u_{i-1} or v_{i-1} to u_{i+1} or v_{i+1} in G_i , for all $2 \leq i \leq l-1$,
- path p_l is a path from u_{l-1} or v_{l-1} to t in G_l .

Note, if s and t are in the same component C_1 then $S = C_1 = T$ and $\mathcal{T}_G = \mathcal{T}_{C_1}$.

In the reachability problems for G_i , we search for a path from u_{i-1} (or v_{i-1}) to u_{i+1} (or v_{i+1}). Each separating pair is connected by a virtual edge. If we have the virtual edge $\{a, b\}$ in C_i on our path, then we have to check whether there is a path from a to b in a child of C_i in \mathcal{T}_{C_i} . Note, in the child component the same situation can occur, recursively. If e.g. (a, b) is also a directed edge in G then there es a child component, a leaf node in the tree which corresponds to a 3-bond. This node indicates this directed edge (a, b) .

Lemma 3.14 *There is a path from u_{i-1} or v_{i-1} to u_{i+1} or v_{i+1} in G_i if and only if there exists a path in C_i such that for virtual edges $\{a, b\}$ on this path there exists a path from a to b in the child component of C_i , recursively.*

Because we have K_5 -component nodes, it is not clear yet, how we can test reachability in $\text{UL} \cap \text{coUL}$. We transform the $K_{3,3}$ -free graph into a planar graph. We also need the following definition.

Definition 3.15 *Let \mathcal{T} be a PlaK_5 -component tree. The size of a separating pair node is defined as 2. The size of a K_5 -component node is defined as 5. The size of a planar component node is defined as the number of vertices in the component. The size of \mathcal{T} rooted at C is the sum of the sizes of its nodes.*

Transforming a K_5 -component into a planar component. Let \mathcal{T}_{C_i} be a PlaK_5 -component tree rooted at C_i as described above. We start with the root C_i and traverse the tree in depth first manner. When we reach a K_5 -component node D , then we replace it by a planar component D' as described next such that the reachability problem does not change. This results in a new PlaK_5 -component tree of a planar graph G' .

Lemma 3.16 *There is a log-space algorithm that transforms G into a planar graph G' such that there is a path from s to t in G if and only if there is such a path in G' .*

Let D be a K_5 -component node with vertices v_1, \dots, v_5 . Let \mathcal{T}_{C_i} be the subtree that contains D . Let N be the size of the subtree rooted at D in \mathcal{T}_{C_i} . Since our algorithm works recursively and in order to have a log-space bound, we would like to have recursive calls only on subtrees of small size, i.e. a fraction of N . Recall, there can be at most one large child of node D in \mathcal{T}_{C_i} . In the following, we consider the situation that we search a path from v_1 to v_2 in D and we have a large child at v_3, v_4 . The same construction works if there is no large child, and it can be easily adapted to the case that the large child is at another pair (e.g. v_2, v_4). The graph D' is defined as shown in Figure 6.

The new component D' has the following properties:

1. D' is planar.
2. Every path from v_1 to v_2 in D exists as well in D' , possibly going through one of the copies v_5' or v_5'' instead of v_5 .
3. D' contains the edge $\{v_3, v_4\}$ which corresponds to a large child only once.

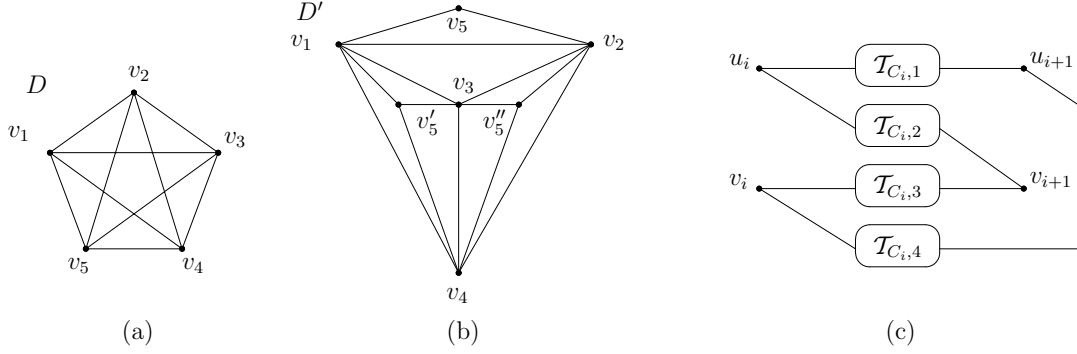


Figure 6: (a) A K_5 -component node D and (b) the planar component node D' . The two nodes v'_5 and v''_5 are copies of v_5 . For example, an edge (v_1, v_5) in D occurs twice in D' , as (v_1, v_5) and (v_1, v'_5) . The edges of D and D' are drawn undirected to not overload the picture. But note that only the virtual edges are undirected. The edges that come from graph G have the same direction as in G . (c) The construction if D is the root of \mathcal{T}_{C_i} . Because there are four reachability problems, we have four versions of D , say D_1, \dots, D_4 that replace the root in \mathcal{T}_{C_i} . This leads to four trees $\mathcal{T}_{C_i,1}, \dots, \mathcal{T}_{C_i,4}$. As we can see, the resulting graph is planar.

4. Vertices v_1 and v_2 are on the outer face of D' .

The last property is important for the special case when D is the root of the subtree, i.e. $D = C_i$. Then we have two vertices u_i, v_i and ask whether we can reach two other vertices u_{i+1} and v_{i+1} and all these vertices belong to D . For a planar arrangement, we make the construction as shown in Figure 6 (c). For example, D_1 is a copy of D where v_1 is identified with u_i and v_2 with u_{i+1} . In total, this gives four combinations of node-to-node reachability questions. Hence, we make four copies of the whole planar graph corresponding to \mathcal{T}_{C_i} , one for each path from a vertex of the incoming separating pair to a vertex of the outgoing separating pair. Note, that this case can occur only at the root, and not in the recursion in the tree \mathcal{T}_{C_i} . Therefore, we can afford to make the four copies.

The replacement of the K_5 -components is done recursively in depth-first manner with all the copies of children (i.e. the subtrees rooted at separating pairs) which we have in the new components D' . Consequently, we give new names to vertices in the copies of the subtrees.

We do this for all edges on all paths in D' . The order of the edges is given by the order they appear on the input tape. We can always recompute the new planar component D' , because we can recompute the sizes of the subtrees of D . We can always refer to which copy of a separating pair we went into recursion by storing $O(1)$ bits on the work-tape when we go into recursion. Hence, whenever we have to change vertex names, we can recompute the new vertex names of the copy of a separating pair. We need such bits at each level of recursion. Since the sizes of the copied subtrees are at most $1/2$ the size of the tree, there are at most $O(\log n)$ levels of recursion. Hence, the algorithm runs in log-space.

Lemma 3.17 *The resulting graph G' after the transformation of K_5 components has the following properties for all i .*

- G'_i is a planar graph.

- There are simple paths from u_i or v_i to u_{i+1} or v_{i+1} in G_i if and only if there are such simple paths in G' .
- The size of the resulting graph G'_i remains polynomial.

Proof: The planarity and the reachability test on G'_i can be proven by induction on the number of K_5 -components, when replaced one by one. The same way as in a K_5 -component node D , G' contains the same copies of vertices we have in D' and for each copied edge in D' we have a copy of the according split component in G' .

The resulting graph is of polynomial size $\mathcal{S}(N)$, because we recursively copy subgraphs of size smaller than $N/2$. The recursion equation is the following for some constant k :

$$\mathcal{S}(N) = k\mathcal{S}(N/2) + O(N)$$

□

This finishes the proof of Theorem 3.11 and we get the following corollary.

Corollary 3.18 $K_{3,3}$ -free graph Reachability is in $\text{UL} \cap \text{coUL}$.

Distance and longest paths in $K_{3,3}$ -free graphs. For the distance problem and the longest path problem it suffices again to consider biconnected graphs, because we can pass only once through every articulation point on a simple path from s to t . Hence we can consider longest paths or distances in the biconnected components, and then sum up these lengths appropriately.

For biconnected graphs we use the transformation from Lemma 3.16. It suffices to observe that simple paths in a K_5 -component D of graph G have the same length as the corresponding paths in the planar component D' in graph G' . Hence, the following lemma holds.

Lemma 3.19 1. $K_{3,3}$ -free Distance \leq_m^L planar Distance.

2. $K_{3,3}$ -free Long-Path \leq_m^L planar Long-Path.

Thierauf and Wagner [TW08] proved that computing the distance in planar directed graphs is in $\text{UL} \cap \text{coUL}$. Limaye, Mahajan and Nimbhorkar [LMN09] proved that computing a longest path in planar DAGs is in $\text{UL} \cap \text{coUL}$. As a consequence we get the following corollary.

Corollary 3.20 1. $K_{3,3}$ -free Distance $\in \text{UL} \cap \text{coUL}$.

2. $K_{3,3}$ -free Long-Path $\in \text{UL} \cap \text{coUL}$.

4 Reachability in K_5 -free graphs

We give a logspace reduction from the reachability problem for directed K_5 -free graphs to the reachability problem for directed planar graphs.

For the reduction, we decompose the given graph G into 3-connected and 4-connected components. For the decomposition we consider G as undirected. It follows from a theorem of Wagner [Wag37] that besides planar components we obtain the following non-planar components that way:

- the four-rung Mobius ladder, also called V_8 (see Figure 7), 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.

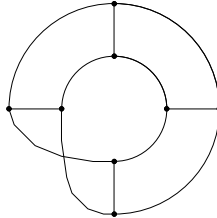


Figure 7: The four-rung Mobius ladder, also called V_8 .

We define trees based on these components. There are nodes for the tri-connected components and the V_8 -components which are connected via separating pair nodes. This is the triconnected component tree of a K_5 -free graph. The non-planar 3-connected components are further decomposed into 4-connected components for which we define a 4-connected component tree. We will show, that this can be done in logspace.

The key step in the reduction is to replace the V_8 -components by planar components such that the reachability properties are not altered.

4.1 The Tree Decomposition

Khuller [Khu88] 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 4.1 [Wag37] *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.*

We make two easy observations with respect to the above clique-sum operation.

- If we build the 3-clique-sum of two planar graphs, then the three nodes of the joint clique are a separating triple in the resulting graph. Hence the 4-connected components of a graph which is build as the 3-clique-sum of planar graphs must all be planar.
- The V_8 is non-planar and 3-connected, but not 4-connected. Furthermore, the V_8 cannot be part of a 3-clique sum, because it does not contain a triangle as subgraph.

By Theorem 4.1 and the two observations we have the following situation.

Corollary 4.2 (cf. [Khu88]) *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.*

In the following we argue that the 3-connected and 4-connected components can be computed in log-space. Similar to the decomposition algorithm of Vazirani [Vaz89], 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. By Lemma 3.1, the biconnected component tree of an undirected K_5 -free graph can be computed in log-space. The same holds for the triconnected component tree of a K_5 -free biconnected graph.

The triconnected component tree for biconnected K_5 -free graphs.

Datta et.al. [DLN⁺08a] show how to construct the triconnected component tree for a planar biconnected graph in log-space. We give a different construction which is suitable for K_5 -free biconnected graphs. The difference is, that the 3-connected components must not be planar.

We define a graph whose nodes are associated to the triconnected components and separating pairs of G . A *separating pair node* is connected to a *triconnected component node* if the separating pair is contained in the corresponding component. The resulting graph is a tree, the *triconnected component tree* \mathcal{T} for a K_5 -free biconnected graph G where edges are considered as undirected. We describe how to compute these nodes.

Let $\{a_0, b_0\}$ be an arbitrary separating pair in G . Let S be a split component of $G - \{a_0, b_0\}$. We define a triconnected component C with respect to S which has the following properties. For C there is a node in \mathcal{T} . The node for C is incident to the separating pair node for $\{a_0, b_0\}$. Intuitively, C is obtained from S by collapsing split components at separating pairs in S to virtual edges. More precisely, we distinguish whether C is a cycle or a 3-connected component as follows (cf. Battista and Tamassia [BT89], [BT96]).

- If $S - \{a_0, b_0\}$ contains articulation points then C is a *cycle*. Let c_1, \dots, c_l be this set of articulation points. Then, $a_0, c_1, c_2, \dots, c_l, b_0$ is the cycle C . Two consecutive vertices c_i, c_{i+1} are connected by an edge as in C or form a separating pair. In the latter case we connect the consecutive separating pair in C by a virtual edge which replaces the original edge, in case there is one in S .
- If $S - \{a_0, b_0\}$ does not contain articulation points, then C is a *3-connected component* which is defined as follows. Two vertices u and v of S are in C if for all separating pairs $\{a, b\}$ different from $\{a_0, b_0\}$ in G there are simple paths from u and v to a_0 (or b_0) and from u to v in $S - \{a, b\}$. Note that $\{a_0, b_0\}$ also belongs to C . Connect each separating pair of S in C by a virtual edge which replaces the original edge if it is present in S . Then C is a 3-connected component, because it does not contain separating pairs by construction.
- If there is an edge between a_0 and b_0 in G then this edge is maintained in an extra component called a *3-bond*, i.e. the directed edge and two virtual edges. This also is defined to be a triconnected component.

We further argue that this decomposition can be computed in log-space.

Lemma 4.3 *The triconnected component tree for a K_5 -free biconnected graph can be computed in log-space.*

Proof: All the tasks, in particular the detection of separating pairs and articulation points or finding paths in undirected graphs, where some vertices are deleted can be done in log-space via queries to reachability testing [Rei05]. The construction of the triconnected component tree can be done in log-space, since we operate on a tree structure.

Note that the navigation on the triconnected component tree is in logspace because there are logspace computable functions to compute the parent, first child and next sibling for each node in the tree. To see this, it suffices to locally store information of the active separating pair a_0, b_0 the active child (i.e. one vertex of a split component) and for finding the unique parent we store the root node of the tree if it is a separating pair and if the root is a triconnected component node then we store a vertex of the corresponding component. \square

Decomposition into 4-connected components. It remains to further decompose the 3-connected components which are non-planar and not the V_8 . To define the decomposition, we need the notion of maximum separating triples.

Definition 4.4 *Let τ be a separating triple and G a split component of τ . A separating triple $\tau_{\max} \neq \tau$ is a candidate separating triple in G with respect to τ if for any separating triple $\tau' \notin \{\tau, \tau_{\max}\}$ there is a path from a vertex of τ_{\max} to a vertex of τ in $G - \tau'$.*

Two candidate separating triples τ_1, τ_2 are crossing if $\tau_1 \cap \tau_2 \neq \emptyset$ and τ_1 and τ_2 have a split component in common (i.e. if there is a vertex v in $G - (\tau \cup \tau_1 \cup \tau_2)$ which belongs to a split component G_1 of τ_1 and a split component G_2 of τ_2). Note, if τ_1, τ_2 are crossing then there is no τ_3 which is also crossing τ_1 or τ_2 . See Figure 8 for an example.

A maximum separating triple with respect to τ is either a candidate separating triple which is not crossing with any other candidate separating triple or the lexicographical smaller one of two crossing candidate separating triples.

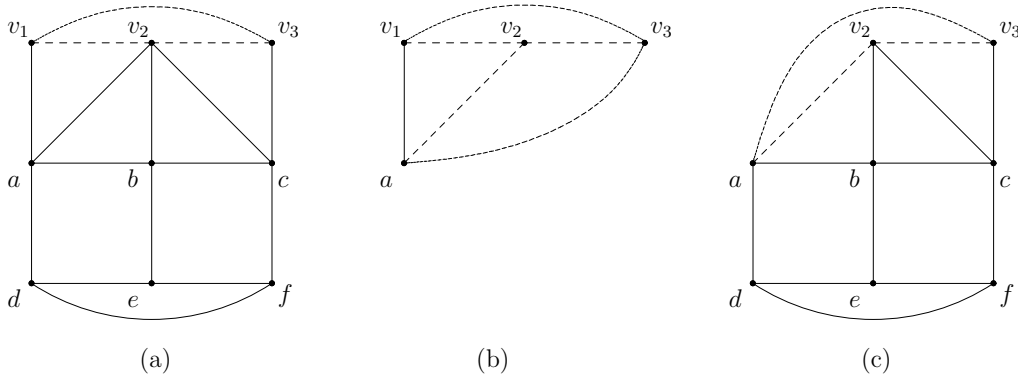


Figure 8: (a) The parent separating triple $\tau = \{v_1, v_2, v_3\}$ and two pairwise crossing candidate separating triples $\{a, v_2, v_3\}$ and $\{v_1, v_2, c\}$.
(b) The 4-connected component split by τ and $\{a, v_2, v_3\}$.
(c) The split component split by $\{a, v_2, v_3\}$ which has to be further decomposed recursively.

Let C be a 3-connected component in a K_5 -free graph and let τ be a separating triple in C . We define a 4-connected component D with respect to τ as follows. Two nodes u and v are in D if there are simple paths from u and v to some node of τ and from u to v in $C - \tau'$, for all maximum separating triples τ' with respect to τ .

Connect each pair of vertices of τ or of a maximum separating triple in D by a virtual edge. If for such a pair there is an edge in G then this is not contained in D and we define a 3-bond instead.

The resulting component D is 4-connected, because it does not contain separating triples by construction.

We define a tree T as follows. There is a node for each separating triple and each 4-connected component. A *separating triple node* is connected to a *4-connected component node* if the separating triple is contained in the 4-connected component. Choose one separating triple τ_{root} in C as the root node of T . The resulting graph is a tree, the *4-connected component tree of C* . This tree can be computed in log-space, since the tasks are similar to those of the Plak_5 -component tree in the previous section.

Lemma 4.5 *The 4-connected component tree of a 3-connected K_5 -free graph can be computed in log-space.*

4.2 Reachability on K_5 -free Graphs

In this section, we prove the following theorem.

Theorem 4.6 *K_5 -free Reachability \leq_m^L planar Reachability.*

Let G be a connected graph and s and t be two vertices in G . By Lemma 3.12 and Lemma 3.14, we can partition the reachability problem for G into reachability problems on the triconnected components of G . If a triconnected component is planar, then we are done with it. If it is non-planar, then we distinguish the two cases whether the triconnected component is the V_8 or not.

In a triconnected component tree, a triconnected component has an incoming separating pair $\{u_i, v_i\}$ and an outgoing separating pair $\{u_{i+1}, v_{i+1}\}$. We consider four reachability tests, from u_i to u_{i+1} , from u_i to v_{i+1} , from v_i to u_{i+1} and from v_i to v_{i+1} . For each of these reachability tests, we construct a planar copy of the triconnected non-planar component and connect them as shown in Figure 6 (c) on page 13.

Transforming a V_8 -component into a planar component. Let \mathcal{T}_C be a triconnected component tree rooted at some node C . Let D be a V_8 -component node in \mathcal{T}_C and v_1, \dots, v_8 the vertices in D . We transform D into a planar component D' such that the reachability question remains unchanged. The transformation is shown in Figure 9. For this, let v_1, \dots, v_4 be four vertices in D such that v_1, v_2 are pairwise different. Assume, we search for a path from v_1 to v_2 in D and that (v_3, v_4) is an edge in D and corresponds to a large child of C .

By construction, D' has the following properties.

- For each path from v_1 to v_2 that does *not* contain (v_3, v_4) , D' contains a copy of this path, i.e. a copy of all vertices and edges on this path.
- For all the paths from v_1 to v_2 which contain (v_3, v_4) , D' contains a copy of the sub-path from v_1 to v_3 and v_4 to v_2 or vice versa from v_1 to v_4 and v_3 to v_2 .

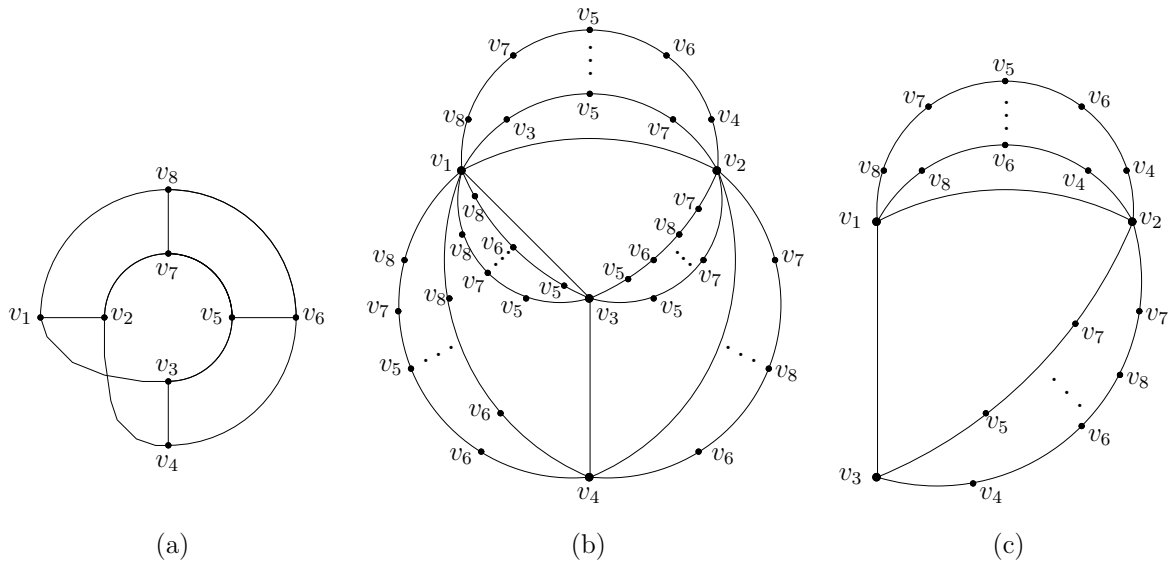


Figure 9: To simplify matters, all the copies of a vertex have the same label in the picture. We do not identify copies of vertex names.

(a) The V_8 component D with edges drawn undirected.

(b) The planar component D' is shown schematically. v_1 and v_2 can be any two vertices in D and v_3, v_4 correspond to a large child of D such that $\{v_1, v_2\} \cap \{v_3, v_4\} = \emptyset$. For every path from v_1 to v_2 that does not contain (v_3, v_4) , D' contains a copy of the path, i.e. a copy of all vertices and edges on this path. This is indicated above the the vertices v_1, v_2 .

The remaining paths go along (v_3, v_4) or (v_4, v_3) in D . This edge should occur only once in D' . Therefore these paths in D are subdivided into paths from v_1 to v_3 or to v_4 , and from these vertices to v_2 . This is indicated in the part below v_1, v_2 .

(c) $\{v_1, v_2\}$ and $\{v_3, v_4\}$ must not be disjoint. The construction is essentially the same as in (b). Here, we see D' in the case that (v_1, v_3) is the large child.

- D' contains the edge (v_3, v_4) exactly once.
- v_1 and v_2 are both on the outer-face of D' . This property is important in the case, that D is the root of \mathcal{T}_C (i.e. $D = C$).
- D' with all the copies of paths is planar and contains $O(1)$ copies of each edge.

The replacement of the V_8 -components is done recursively in depth-first manner with all the copies of children (i.e. the subtrees rooted at separating pairs) which we have in the new components D' . Consequently, we give new names to vertices in the copies of the subtrees.

We do this for all edges on all paths in D' . The order of the edges is given by the order in which they appear on the input tape. We can always recompute the new planar component D' , because we can recompute the sizes of subtrees of D . We can always recompute from which copy of a separating pair we went into recursion by storing $O(1)$ bits on the work-tape when we go into recursion. So, when we have to change vertex names, then we can recompute the new vertex names of the copied separating pair. We need such bits at each level of

recursion. Since the sizes of the copied subtrees are at most $1/2$ the size of the tree, there are at most $O(\log n)$ levels of recursion. Hence, the algorithm runs in log-space.

Lemma 4.7 *Let G_i be a K_5 -free biconnected graph with incoming separating pair u_i, v_i and outgoing separating pair u_{i+1}, v_{i+1} . Let G'_i be the resulting graph when all the V_8 -components D in G_i are replaced by the new gadgets D' . G'_i has the following properties.*

- G'_i is a planar biconnected graph.
- There are simple paths from u_i or v_i to u_{i+1} or v_{i+1} in G_i if and only if there are such simple paths in G'_i .
- The size of G'_i is polynomial in the size of G_i .

Planar arrangement of split components in a 4-connected component tree. After the replacement of V_8 -components by planar components, we have to consider the other non-planar 3-connected components. We have decomposed them into planar 4-connected components. We have to recombine all the components into one planar graph. However, we cannot simply reverse the decomposition process because the 3-clique sum of the 4-connected (planar) components could result in a non-planar 3-connected component.

To get around this problem we make copies of some of the components and arrange the copies in a planar way. This has to be done carefully such that the size of the graph constructed that way stays polynomial in the size of the input graph.

Consider a 4-connected component tree \mathcal{T} . Let S and T be the component nodes in \mathcal{T} where vertex s and t are contained in, respectively. Consider S as the root of \mathcal{T} i.e., let $\mathcal{T}_S = \mathcal{T}$ and let P be a simple path from S to T in \mathcal{T}_S . We describe how to find a planar arrangement of the components of \mathcal{T}_S .

We start by putting the component S in the new planar arrangement. Inductively assume that we have put some component C and let τ be some child separating triple node of C in \mathcal{T}_S . Let furthermore the children of τ be the 4-connected component nodes C_1, \dots, C_k . Precisely one of the children is put once in the planar arrangement, the other children are put three times, i.e. there are two additional copies which are connected to copies of the vertices of τ . Figure 10 shows the construction. By G' we denote the resulting planar arranged graph we obtain for G .

Note, the construction does not change the reachability properties. For example, if there is a path from v_1 to v_2 in G which goes through the component C_2 and also passes node v_3 , then there will be a path from v_1 to v_2 in G' which goes through the copy C_2'' of C_2 and passes the copy v_3'' instead of v_3 . If there is no path from v_1 to v_2 in G then there will be no path from v_1 to v_2 in the constructed planar graph as well. The children which is put only once is selected as follows:

1. If a child C_i of τ is a node on path P , then we select C_i .
2. If no child of τ is a node on path P but there is a large child C_j , then we select C_j .
3. If none of the first two cases occurs then we select an arbitrary component, say C_1 .

Let N be the size of the triconnected component tree rooted at node C . We emphasize that in case 1, if a large child is copied three times because another child of τ is on path P

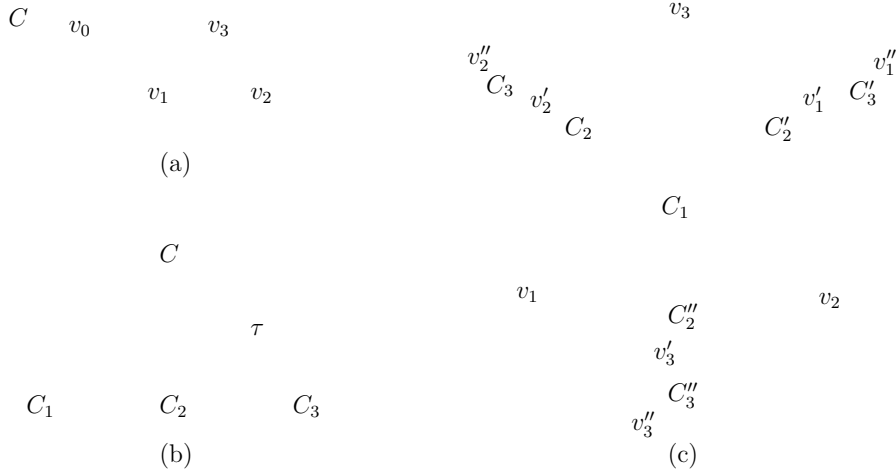


Figure 10: (a) A planar 4-connected component C with separating triple $\tau = \{v_1, v_2, v_3\}$. The pairwise edges among these vertices are virtual edges, indicated by dashed undirected lines. (b) The 4-connected component tree with C , separating triple τ and its children, the 4-connected component nodes C_1, C_2 and C_3 . (c) The planar arrangement of C_1, C_2 and C_3 at separating triple τ is obtained by making copies of C_2 and C_3 (i.e. C'_2, C''_2 and C'_3, C''_3) and vertices v_i (i.e. v'_i, v''_i) for $1 \leq i \leq 3$. The planar arrangement of C is obtained by connecting the vertices of τ to v_0 .

then this situation does not occur recursively. That is, because the ancestors of the copied large child do not belong to path P . Hence, the planar arranged graph G' is of polynomial size, because we just copy recursively subgraphs of size smaller than $N/2$ even if we consider the exception of case 1. The recursion equation for the size $\mathcal{S}(N)$ of G' with some constant k is

$$\mathcal{S}(N) = k\mathcal{S}(N/2) + O(N)$$

This finishes the proof of Theorem 4.6. Since reachability in planar DAGs is in $\text{UL} \cap \text{coUL}$ [BTV07], we get the following corollary.

Corollary 4.8 K_5 -free graph reachability is in $\text{UL} \cap \text{coUL}$.

Distance and longest paths in K_5 -free graphs. To compute distances, we search for shortest simple paths. Again it is easy to see that the graph transformations do not change the distances between the vertices. Also for the longest path problem we can guarantee the same length of a simple path in the resulting graph, but here we have to argue more carefully. Namely, we have to make sure that we do not get longer paths by the copies of the components introduced in the planar arrangement of the 4-connected components. We use the fact that subpaths of longest paths are again longest paths in a DAG.

Lemma 4.9 Let $p_{u,v}$ be a longest simple path from u to v in a DAG G . Let a, b be two vertices on path $p_{u,v}$ and let $p_{a,b}$ be the subpath of $p_{u,v}$ from a to b . Then $p_{a,b}$ is a longest simple path from a to b in G .

Since we make up to three copies of a component in our construction of a planar graph, we have to make sure that we do not get longer paths by using the copies. We show that it is not possible for a path to pass through a node and its copy.

Lemma 4.10 *Let v be node of DAG G and let v' be a copy of v in the planar graph G' constructed from G . Then there is no path from v to v' in G' .*

Proof: Let C be the 4-connected component that contains v and let $\{v_1, v_2, v_3\}$ be the separating triple that separates C from G . In G' , assume that there is a path p from v in C to v' in C' . The components C and C' are connected by one vertex of v_1, v_2, v_3 , say v_1 . Hence p can have the form $p = (v, u_1, \dots, u_l, v_1, u'_{l+1}, \dots, u'_k, v')$, where $u_1, \dots, u_l \in C$ and $u'_{l+1}, \dots, u'_k \in C'$. But then we have the cycle $(v, u_1, \dots, u_l, v_1, u_{l+1}, \dots, u_k, v)$ in C which is a contradiction.

Path p might also go through v_2 and v_3 . Then the argument is similar. \square

We argue that the length of longest paths are not changed by the transformations. Consider Figure 10. Let p be a longest path from v_1 to some node v in G . Assume that p goes through component C_2 and that $v \in C_1$. The interesting case is when p passes through v_2 and v_3 (in this order), because then we have two ways to go in the planar graph:

1. we can go through component C_2 and pass through v'_2 and then switch to C_1 via v_3 .
2. we can go to v_2 through component C''_2 and then to v_3 through component C'_2 .

The first case corresponds exactly to path p in G because after passing through v'_2 , the path cannot go through v_2 anymore by Lemma 4.10. In the second case the longest simple path from v_1 to v_2 in C''_2 has the same length as the first part of p from v_1 to v_2 in C_2 by Lemma 4.9. Also, the longest simple path from v_2 to v_3 in C'_2 has the same length as the second part of p from v_2 to v_3 in C_2 . Hence, the second possibility does not lead to longer paths. We conclude that the lengths of longest paths are not changed by the reduction.

Theorem 4.11 1. K_5 -free Distance \leq_m^L planar Distance.

2. K_5 -free Long-Path \leq_m^L planar Long-Path.

As a consequence, we have

Corollary 4.12 1. K_5 -free Distance $\in \text{UL} \cap \text{coUL}$.

2. K_5 -free Long-Path $\in \text{UL} \cap \text{coUL}$.

5 Conclusion

We showed a reduction from the reachability, the distance and the longest path problem on $K_{3,3}$ -free graphs and on K_5 -free graphs (DAGs) to the corresponding problem on planar graphs (DAGs), respectively. It would be interesting to extend the result to $K_{3,4}$ -free graphs or to K_6 -free graphs, for example or even to minor-closed graph classes.

6 Acknowledgement

We thank Samir Datta, Nutan Limaye, Prajakta Nimbhorkar, Mario Szegedy, and Jacobo Torán for helpful discussions.

References

- [Asa85] Tetsuo Asano. An approach to the subgraph homeomorphism problem. *Theoretical Computer Science*, 38, 1985.
- [BT89] Giuseppe Di Battista and Roberto Tamassia. Incremental planarity testing. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 436–441, 1989.
- [BT96] Giuseppe Di Battista and Roberto Tamassia. On-line maintenance of triconnected components with SPQR-trees. *Algorithmica*, 15, 1996.
- [BTV07] Chris Bourke, Raghunath Tewari, and N.V. Vinodchandran. Directed planar reachability is in unambiguous log-space. *Annual IEEE Conference on Computational Complexity (CCC)*, pages 217–221, 2007.
- [DLN⁺08a] Samir Datta, Nutan Limaye, Prajakta Nimbhorkar, Thomas Thierauf, and Fabian Wagner. A log-space algorithm for canonization of planar graphs. Technical report, arXiv:0809.2319, 2008.
- [DLN⁺08b] Samir Datta, Nutan Limaye, Prajakta Nimbhorkar, Thomas Thierauf, and Fabian Wagner. Personal communication, 2008.
- [HT73] John E. Hopcroft and Robert E. Tarjan. Dividing a graph into triconnected components. *SIAM Journal on Computing*, 2(3):135–158, 1973.
- [JT07] Andreas Jakobý and Till Tantau. Logspace algorithms for computing shortest and longest paths in series-parallel graphs. In *Proceedings of the 27th International Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 4855 of *Lecture Notes in Computer Science*, pages 216–227. Springer, 2007.
- [KHC04] Jacek P. Kukluk, Lawrence B. Holder, and Diane J. Cook. Algorithm and experiments in testing planar graphs for isomorphism. *Journal of Graph Algorithms and Applications*, 8(2):313–356, 2004.
- [Khu88] Samir Khuller. Parallel algorithms for K_5 -minor free graphs. Technical Report TR88-909, Cornell University, Computer Science Department, 1988.
- [LMN09] Nutan Limaye, Meena Mahajan, and Prajakta Nimbhorkar. Longest paths in planar dags in unambiguous logspace. In *Computing: The Australian Theory Symposium (CATS)*, volume 94, 2009.
- [Rei05] Omer Reingold. Undirected st-connectivity in log-space. In *Proceedings of the 37th annual ACM Symposium on Theory of Computing (STOC)*, pages 376–385, 2005.

- [Tut66] William T. Tutte. *Connectivity in graphs*. University of Toronto Press, 1966.
- [TW08] Thomas Thierauf and Fabian Wagner. The isomorphism problem for planar 3-connected graphs is in unambiguous logspace. In *Proceedings of the 25th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 633–644, 2008.
- [Vaz89] 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.
- [Wag37] Klaus Wagner. Über eine Eigenschaft der ebenen Komplexe. In *Mathematical Annalen*, volume 114, 1937.