


One-Way Trail Orientations


Anders Aamand¹

University of Copenhagen, Copenhagen, Denmark
andersaamand@hotmail.com

 <https://orcid.org/0000-0002-0402-0514>


Niklas Hjuler²

University of Copenhagen, Copenhagen, Denmark
niklashjuler@gmail.com

 <https://orcid.org/0000-0002-0815-670X>


Jacob Holm³

University of Copenhagen, Copenhagen, Denmark
jaho@di.ku.dk

 <https://orcid.org/0000-0001-6997-9251>

Eva Rotenberg⁴

Technical University of Denmark, Lyngby, Denmark
erot@dtu.dk

 <https://orcid.org/0000-0001-5853-7909>

Abstract

Given a graph, does there exist an orientation of the edges such that the resulting directed graph is strongly connected? Robbins' theorem [Robbins, Am. Math. Monthly, 1939] asserts that such an orientation exists if and only if the graph is 2-edge connected. A natural extension of this problem is the following: Suppose that the edges of the graph are partitioned into trails. Can the trails be oriented consistently such that the resulting directed graph is strongly connected?

We show that 2-edge connectivity is again a sufficient condition and we provide a linear time algorithm for finding such an orientation.

The generalised Robbins' theorem [Boesch, Am. Math. Monthly, 1980] for mixed multigraphs asserts that the undirected edges of a mixed multigraph can be oriented to make the resulting directed graph strongly connected exactly when the mixed graph is strongly connected and the underlying graph is bridgeless.

We consider the natural extension where the undirected edges of a mixed multigraph are partitioned into trails. It turns out that in this case the condition of the generalised Robbins' Theorem is not sufficient. However, we show that as long as each cut either contains at least 2 undirected edges or directed edges in both directions, there exists an orientation of the trails such that the resulting directed graph is strongly connected. Moreover, if the condition is satisfied, we may start by orienting an arbitrary trail in an arbitrary direction. Using this result one obtains a very simple polynomial time algorithm for finding a strong trail orientation if it exists, both in the undirected and the mixed setting.

¹ Research supported by Thorup's Advanced Grant DFF-0602-02499B from the Danish Council for Independent Research and by his Investigator Grant 16582, Basic Algorithms Research Copenhagen (BARC), from the VILLUM Foundation.

² This work is supported by the Innovation Fund Denmark through the DABAI project.

³ Research supported by Thorup's Advanced Grant DFF-0602-02499B from the Danish Council for Independent Research and by his Investigator Grant 16582, Basic Algorithms Research Copenhagen (BARC), from the VILLUM Foundation.

⁴ This research was conducted during the fourth author's time as a PhD student at University of Copenhagen.



© Anders Aamand, Niklas Hjuler, Jacob Holm, and Eva Rotenberg;
licensed under Creative Commons License CC-BY

45th International Colloquium on Automata, Languages, and Programming (ICALP 2018).
Editors: Ioannis Chatzigiannakis, Christos Kaklamani, Dániel Marx, and Donald Sannella;
Article No. 6; pp. 6:1–6:13



Leibniz International Proceedings in Informatics
LIPIC Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



2012 ACM Subject Classification Mathematics of computing → Graph algorithms, Mathematics of computing → Paths and connectivity problems

Keywords and phrases Graph algorithms, Robbins’ theorem, Graph orientation

Digital Object Identifier 10.4230/LIPIcs.ICALP.2018.6

1 Introduction and motivation

Suppose that the mayor of a small town decides to make all the streets one-way such that it is possible to get from any place to any other place without violating the orientations of the streets⁵. If all the streets are initially two-way then Robbins’ theorem [10] asserts that this can be done exactly when the corresponding graph is 2-edge connected. If, on the other hand some of the streets were already one-way in the beginning then the generalised Robbins’ theorem by Boesch [1] states that it can be done exactly when the corresponding “mixed” graph is strongly connected and the underlying graph is bridgeless.

However, the proofs of both of these results assume that every street of the city corresponds to exactly one edge in the graph. This assumption hardly holds in any city in the world and therefore a more natural assumption is that every street corresponds to a trail (informally, a potentially self-crossing path) in the graph and that the edges of each trail must be oriented consistently⁶.

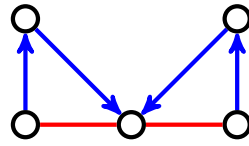
In this paper we consider such graphs having their edges partitioned into trails. We prove that the trails can be oriented to make the resulting directed graph strongly connected exactly if the initial graph is 2-edge connected (note that this is precisely the condition of Robbins’ theorem).

Not only do we show that the strong trail orientation problem in undirected 2-edge connected graphs always has a solution, we also provide a linear time algorithm for finding such an orientation. In doing so, we use an interesting combination of techniques that allow us to reduce to a graph with a number of 3-edge connected components that is linear in the number of edges. Using that the average size of these components is constant and that we can piece together solutions for the individual components we obtain an efficient algorithm.

Finally, we will consider the generalised Robbins’ theorem in this new setting by allowing some edges to be oriented initially and supposing that the remaining edges are partitioned into trails. We will show that if each cut (V_1, V_2) in the graph has either at least 2 undirected edges going between V_1 and V_2 or at least 1 directed edge in each direction then it is possible to orient the trails making the resulting graph strongly connected. In fact, we show that if this condition is satisfied we may start by orienting an *arbitrary* trail in an *arbitrary* direction. Although this condition is not necessary it does give a simple algorithm for finding a trail orientation if it exists. Indeed, initially the graph may contain undirected edges that are *forced* in one direction by some cut. For finding a trail orientation if it exists we can thus orient forced trails in the forced direction. If there are no forced trails we orient any trail arbitrarily.

⁵ The motivation for doing so is that the streets of the town are very narrow and thus it is a great hassle when two cars unexpectedly meet.

⁶ This version of the problem was given to us through personal communication with Professor Robert E. Tarjan.



■ **Figure 1** The graph is strongly connected and the underlying graph is 2-edge connected, but irrespective of the choice of orientation of the red trail, the graph will no longer be strongly connected.

Note that in the mixed setting the feasibility depends on the trail decomposition which is not the case for the other results. That the condition from the generalised Robbins' theorem is not sufficient can be seen from Figure 1.

Earlier methods

Several methods have already been applied for solving orientation problems in graphs where the goal is to make the resulting graph strongly connected.

One approach used by Robbins [10] is to use that a 2-edge connected graph has an *ear-decomposition*. An ear decomposition of a graph is a partition of the set of edges into a cycle C and paths P_1, \dots, P_t such that P_i has its two endpoints but none of its internal vertices on $C \cup \left(\bigcup_{j=1}^{i-1} P_j\right)$. Given the existence of an ear decomposition of a 2-edge connected graph it is easy to prove Robbins' theorem. Indeed, any choice of consistent orientations of the paths and the cycle gives a strongly connected graph.

A second approach introduced by Tarjan [4] gives another simple proof of Robbins' theorem. One can create a DFS tree in the graph rooted at a vertex v and orient all edges in the DFS tree away from v . The remaining edges are all back edges (see [4]) and are oriented towards v . It is easily verified that this gives a strong orientation if the graph is 2-edge connected. A similar approach was used by Chung et al. [2] in the context of the generalized Robbins' theorem for mixed multigraphs.

The above methods not only prove Robbins' theorem, they also provide linear time algorithms for finding strong orientations of undirected or mixed multigraphs.

However, none of the above methods have proven fruitful in our case. In case of the ear decomposition we would need one that is somehow compatible with the partitioning into trails, and this seems hard to guarantee. Similar problems appear when trying a DFS-approach. Neither does the proof by Boesch [1] of Robbins' theorem for mixed multigraphs generalise to prove our result. Most importantly, the corresponding theorem would no longer be true for trail orientations as is shown by the example in Figure 1.

Since the classical linear time algorithms rely on ear-decompositions and DFS searches, and since these approaches do not immediately work for trail partitions, our linear time algorithm will be a completely new approach to solving orientation problems.

Structure of the paper

The structure of this paper is as follows. In section 3 we prove our generalisation of Robbins' theorem for undirected graphs partitioned into trails. In section 4 we study the case of mixed graphs. Finally in section 5 we provide our linear time algorithm for trail orientation in an undirected graph.

2 Preliminaries

Let us briefly review the concepts from graph theory that we will need.

A graph having some subset of its edges oriented is said to be a *mixed* graph. We will write $\{u, v\}$ for an undirected edge between u and v and (u, v) for an edge directed from u to v .

A *walk* in a graph is an alternating sequence of vertices and edges $v_0, e_1, v_1, e_2, \dots, v_k$, such that for $1 \leq i \leq k$ the edge e_i has v_{i-1} and v_i as its two endpoints. In a directed or mixed graph we further require that either e_i be undirected or directed from v_{i-1} to v_i .

A *trail* is a walk without repeated edges. A *path* is a trail without repeated vertices (except possibly $v_0 = v_k$). Finally, a *cycle* is a path for which $v_0 = v_k$.

Next, a mixed multigraph $G = (V, E)$ is called *strongly connected* if for each pair of vertices $u, v \in V$ there exists a walk from u to v . In case the graph is undirected this is equivalent to saying that it consists of exactly one connected component. If $A \subseteq V$ we will say that A is *strongly connected in G* if for each pair of vertices $u, v \in A$ there is a walk in G from u to v .

A *cut* or *edge-cut* (V_1, V_2) in a graph is a partition of its vertices into two non-empty subsets V_1, V_2 . We recall the definition of k -edge connectivity. A graph $G = (V, E)$ is said to be *k -edge connected* if and only if $G' = (V, E - X)$ is connected for all $X \subseteq E$ where $|X| < k$. A trivially equivalent condition is that each cut (V_1, V_2) in the graph has at least k edges going between V_1 and V_2 .

Finally, if $G = (V, E)$ is a mixed multigraph and $A \subseteq V$ we define G/A to be the graph obtained by contracting A to a single vertex (maintaining duplicate edges and self-loops) and $G[A]$ to be the subgraph of G induced by A . The following simple observation will be used repeatedly in this paper.

► **Observation 2.1.** *If $G = (V, E)$ is k -edge connected and $A \subseteq V$ then G/A is k -edge connected. Also, if G is a strongly connected mixed multigraph then G/A is too.*

3 Robbins Theorem Revisited

We are now ready to state and prove our generalisation of Robbins' theorem.

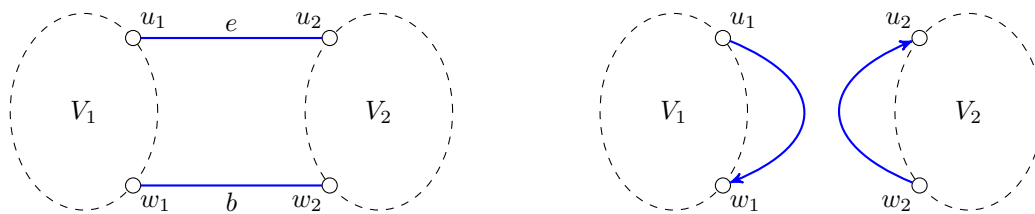
► **Theorem 3.1.** *Let $G = (V, E)$ be an undirected multigraph with E partitioned into trails. An orientation of each trail such that the resulting directed graph is strongly connected exists if and only if G is 2-edge connected.*

We note that this theorem could also be proven using a general result from Király and Szigeti [6] which relies on theorems by Nash-Williams [9]. However, our proof is significantly simpler (in fact we believe that restating the theorem by Király and Szigeti and explaining the reduction would be more cumbersome) and more suitable for constructing algorithms.

Proof. If G is not 2-edge connected, such an orientation obviously doesn't exist, so we only need to prove the converse. Suppose therefore that G is 2-edge connected.

Our proof is by induction on the number of edges in G . If there are no edges, the graph consists of a single vertex, and the statement is obviously true. Assume now the statement holds for all graphs with strictly fewer edges than G . Pick an arbitrary edge e that sits at the end of its corresponding trail.

If $G - e$ is 2-edge connected, then by the induction hypothesis there is a strong orientation of $G - e$ that respects the trails of G . Such an orientation clearly extends to the required orientation of G .



■ **Figure 2** A 2-edge cut and the two graphs $G_1 = G[V_1] \cup \{\{u_1, w_1\}\}$ and $G_2 = G[V_2] \cup \{\{u_2, w_2\}\}$. The orientations of the two new edges are obtained from the strong trail orientations of G_1 and G_2 .

If $G - e$ is not 2-edge connected, there exists a bridge b in $G - e$ (see Figure 2). Let V_1, V_2 be the two connected components of $G - \{e, b\}$, and let $e = \{u_1, u_2\}$ and $b = \{w_1, w_2\}$ such that for $i \in \{1, 2\}$, $u_i, w_i \in V_i$ (note that we don't necessarily have that u_i and w_i are distinct for $i \in \{1, 2\}$).

Now for $i \in \{1, 2\}$ construct the graph $G_i = G[V_i] \cup \{\{u_i, w_i\}\}$ (note that $\{u_i, w_i\}$ might be a self-loop but this causes no problems for the argument), and define the trails in G_i to be the trails of G that are completely contained in G_i , together with a single trail combined from the (possibly empty) partial trail of e contained in G_i and ending at u_i , followed by the edge $\{u_i, w_i\}$, followed by the (possibly empty) partial trail of b contained in G_i starting at w_i . Both G_1 and G_2 are 2-edge connected since they can each be obtained as contractions of G with some self-loops deleted. Furthermore, they each have strictly fewer edges than G , so inductively each has a strong orientation that respects the given trails. Further, we can assume that the orientations are such that the new edges are oriented (u_1, w_1) and (w_2, u_2) by flipping the orientation of all edges in either graph if necessary. We claim that this orientation, together with e oriented as (u_1, u_2) and b oriented as (w_2, w_1) , is the required orientation of G . To see this first note that (by our choice of flips) this orientation respects the trails. Secondly, suppose $v_1 \in V_1$ and $v_2 \in V_2$ are arbitrary. Since G_1 is strongly connected $G[V_1]$ contains a directed path from v_1 to u_1 . Similarly, $G[V_2]$ contains a directed path from u_2 to v_2 . Thus G contains a directed path from v_1 to v_2 . A similar argument gives a directed path from v_2 to v_1 and since v_1 and v_2 were arbitrary this proves that G is strongly connected and our induction is complete. ◀

The construction in the proof can be interpreted as a naive algorithm for finding the required orientation when it exists.

► **Corollary 3.2.** *The one-way trail orientation problem on a graph with n vertices and m edges can be solved in $\mathcal{O}(n + m \cdot f(m, n))$ time, where $f(m, n)$ is the time per operation for fully dynamic bridge finding (a.k.a. 2-edge connectivity).*

At the time of this writing [3], this is $\mathcal{O}(n + m(\log n \log \log n)^2)$. In Section 5 we will provide a less naive algorithm which runs in linear time.

4 Extension to Mixed graphs

Now we will extend our result to the case of mixed graphs. We are going to prove the following.

► **Theorem 4.1.** *Let $G = (V, E)$ be a strongly connected mixed multigraph. Then $G - e$ is strongly connected for all undirected $e \in E$ if and only if for any partition \mathcal{P} of the undirected edges of G into trails, and any $T \in \mathcal{P}$, any orientation of T can be extended to a strong trail orientation of (G, \mathcal{P}) .*

Algorithm 1: Algorithm for mixed graphs.

Input: A mixed multigraph G and a partition \mathcal{P} of the undirected edges of G into trails.

Output: True if (G, \mathcal{P}) has a strong trail orientation, otherwise false. If G has a bridge or is not strongly connected, G is left unmodified. Otherwise G is modified, either to have such a strong trail orientation, or to a forced graph that is not strongly connected.

```

1 if  $G$  has a bridge or is not strongly connected then
2   | return false
3 end
4 while  $|\mathcal{P}| > 0$  do
5   | if for some undirected edge  $e$ ,  $G - e$  is not strongly connected then
6     |   Let  $T \in \mathcal{P}$  be the trail containing  $e$ .
7     |   if some orientation of  $T$  leaves  $G$  strongly connected then
8       |     Apply such an orientation of  $T$  to  $G$ 
9     |   else
10    |     return false
11    |   end
12  | else
13    |   Let  $T \in \mathcal{P}$  be arbitrary.
14    |   Update  $G$  by orienting  $T$  in an arbitrary direction.
15  | end
16  | Remove  $T$  from  $\mathcal{P}$ .
17 end
18 return true

```

Suppose $G = (V, E)$ is as in the theorem. We will say that $e \in E$ is *forced* if it is undirected and satisfies that $G - e$ is not strongly connected. This terminology is natural as it is equivalent to saying that there exists a cut (V_1, V_2) in G such that e is the only undirected edge in this cut and such that all the directed edges go from V_1 to V_2 . If we want an orientation of the trails making the graph strongly connected we are clearly forced to orient e from V_2 to V_1 .

Theorem 4.1 is a proper extension of Theorem 3.1 since if G is undirected and 2-edge connected then no $e \in E$ is forced. Furthermore, the theorem suggests a very simple polynomial time algorithm (see Algorithm 1) for finding a strong orientation of the trails if it exists. Indeed, if the mixed graph contains forced edges we direct the corresponding trails in the forced direction. If there are no forced edges then either the graph is no longer strongly connected in which case we know that a strong trail orientation doesn't exist. Otherwise, we may by Theorem 4.1 orient any trail in an arbitrary direction.

For proving Theorem 4.1 we will need the following lemma.

► **Lemma 4.2.** *Let G be a directed graph, and let (A, B) be a cut with exactly one edge crossing from A to B . Then G is strongly connected if and only if G/A and G/B are.*

Proof. Strong connectivity is preserved by contractions, so if G is strongly connected then G/A and G/B both are. For the other direction, let (a_1, b_1) be the edge going from A to B . As G/A is strongly connected and (a_1, b_1) is the only edge going from A to B we can for any edge (b_2, a_2) going from B to A find a path from b_1 to b_2 that stays in B . Since G/B is



■ **Figure 3** A cut with two undirected edges and all directed edges going from A to B followed by a contraction of B .

strongly connected, it follows that A is strongly connected in G . By a symmetric argument, B is also strongly connected in G and since the cut has edges in both directions (as e.g. G/A is strongly connected), G must be strongly connected. ◀

Now we provide the proof of Theorem 4.1.

Proof of Theorem 4.1. If there exists an undirected edge e such that $G - e$ is not strongly connected, then the trail T containing e can at most be directed one way since e is forced, so there is an orientation of T that does not extend to a strong trail orientation of (G, \mathcal{P}) . To prove the converse suppose $G - e$ is strongly connected for all undirected $e \in E$.

The proof is by induction on $|\mathcal{P}|$. If $|\mathcal{P}| = 0$ the result is trivial. So suppose $|\mathcal{P}| \geq 1$ and that the theorem holds for all (G', \mathcal{P}') with $|\mathcal{P}'| < |\mathcal{P}|$.

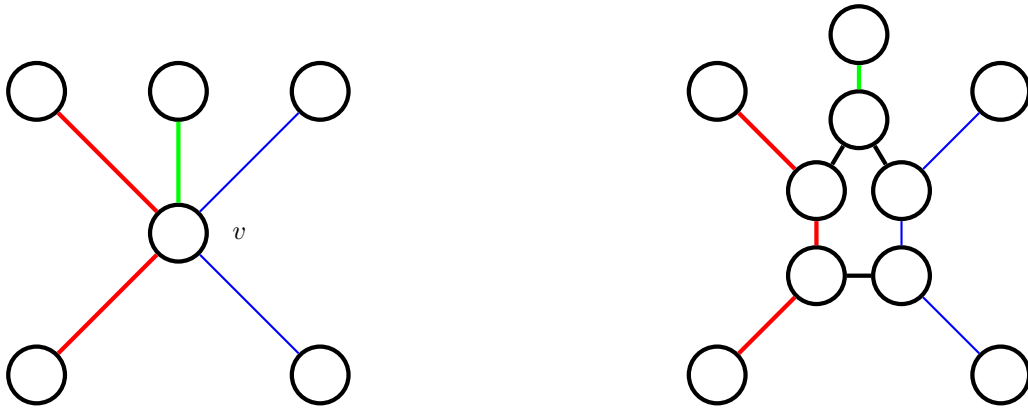
Consider the chosen trail $T \in \mathcal{P}$. If both orientations of T leave a graph where the condition in the theorem is still satisfied we are home by induction. Otherwise, there must exist a cut (A, B) of the following form: (1) T crosses the cut exactly once, (2) exactly one undirected edge from a different undirected trail $T' \in \mathcal{P}$ crosses the cut and (3) every directed edge crossing the cut goes from A to B .

Now suppose there is such a cut (A, B) (see Figure 3). Consider the graph G/B and let b be the node corresponding to B in G/B . Let \mathcal{P}_A consist of all trails in \mathcal{P} that are completely contained in A , together with a single trail T_A combined from the (possibly empty) fragments of T and T' , joined at b . Since any cut in G/B corresponds to a cut in G , G/B is strongly connected and remains so after deletion of any single undirected edge. By construction $|\mathcal{P}_A| \leq |\mathcal{P}| - 1$, so by induction any orientation of T_A in G/B extends to a strong orientation of $(G/B, \mathcal{P}_A)$. Let $G/A, a, \mathcal{P}_B$ and T_B be defined symmetrically, then by the same argument any orientation of T_B in G/A extends to a strong orientation of $(G/A, \mathcal{P}_B)$. Now for any orientation of T , we can choose orientations of T_A and T_B that are compatible. The result then follows by Lemma 4.2. ◀

Theorem 4.1 gives a sufficient condition for the existence of a strong orientation and we deal with the other cases by first orienting all forced edges. However, the generalised Robbins' theorem provides a simple equivalent condition, which we lack. Finding such an equivalent condition in our setting is an essential open problem for strong trail orientations. As seen by the example of Figure 1 such a condition will necessarily have to depend on the structure of the trail partition.

5 Linear time algorithm

In this section we provide our linear time algorithm for solving the trail orientation problem in undirected graphs. For this, we make two crucial observations. First, we show that there is an easy linear time reduction from general graphs or multigraphs to cubic multigraphs.



■ **Figure 4** A node of degree 5 turns into a cycle of length 5.

Second, we show that in a cubic multigraph with n vertices, we can in linear time find and delete a set of edges that are at the end of their trails, such that the resulting graph has $\Omega(n)$ 3-edge connected components. We further show that we can compute the required orientation recursively from an orientation of each 3-edge connected component together with the cactus graph of 3-edge connected components. Since the average size of these components is constant, we can compute the orientations of most of them in constant time individually and thus in linear time taken together. The rest contains at most a constant fraction of the vertices, and so a simple geometric sum argument tells us that the total time is also linear.

We start out by making the following reduction.

► **Lemma 5.1.** *The one-way trail problem on a 2-edge connected graph or multigraph with n vertices and m edges, reduces in $\mathcal{O}(m + n)$ time to the same problem on a 2-edge connected cubic multigraph with $2m$ vertices and $3m$ edges.*

Proof. Cyclically order the edges adjacent to each vertex such that two edges that are adjacent on the same trail are consecutive in the order. Replace each single vertex v with a cycle of length $\deg(v)$, with each vertex of the new cycle inheriting a corresponding neighbour of v such that the order of the vertices on the cycle corresponds to the cyclic ordering (see Figure 4). Note that for a vertex of degree 2, this creates a pair of parallel edges, so the result may be a multigraph. By the choice of cyclic ordering, we can make the cycle-edge between the two vertices on the same trail belong to that trail. The rest of the cycle edges form new length 1 trails. Clearly the new graph is also 2-edge connected so by Theorem 3.1 it has a strong trail orientation, and any strong trail orientation on this graph translates to a strong trail orientation of the original graph. The new graph has exactly $2m$ vertices and $3m$ edges, and is constructed in $\mathcal{O}(m + n)$ time. ◀

Recall now that a multigraph C is called a *cactus* if it is connected and each edge is contained in at most one cycle. If G is any connected graph we let C_1, \dots, C_k be its 3-edge connected components. It is well known that if we contract each of these we obtain a cactus graph. For a proof of this result see section 2.3.5 of [8]. As the cuts in a contracted graph are also cuts in the original graph we have that if G is 2-edge connected then the cactus graph is 2-edge connected. The edges of the cactus are exactly the edges of G which are part of a 2-edge cut. We will call these edges *2-edge critical*.

It is easy to check that if a cactus has m edges and n vertices then $m \leq 2(n - 1)$. We will be using this result in the proof of the following lemma.

► **Lemma 5.2.** *Let $G = (V, E)$ be a cubic 2-edge connected multigraph, let $X \subseteq E$, and let $F \subseteq E$ with $F \supseteq E \setminus X$ minimal (w.r.t inclusion) such that $H = (V, F)$ is 2-edge connected. Then H has at least $\frac{2}{5}|X|$ distinct 3-edge connected components.*

Proof. Let $X_{\text{del}} = X \setminus F$ be the set of edges deleted from G to obtain H , and let $X_{\text{keep}} = X \setminus X_{\text{del}}$ be the remaining edges in X .

By minimality of H there are at least $|X_{\text{keep}}|$ 2-edge-critical edges in H i.e. edges of the corresponding cactus, and thus, if $|X_{\text{keep}}| \geq \frac{4}{5}|X|$, there are at least $\frac{1}{2}|X_{\text{keep}}| + 1 \geq \frac{2}{5}|X| + 1$ distinct 3-edge connected components.

If $|X_{\text{keep}}| \leq \frac{4}{5}|X|$ then $|X_{\text{del}}| \geq \frac{1}{5}|X|$, and since G is cubic and the removal of each edge creates two vertices of degree 2 we must have that H has at least $2|X_{\text{del}}| \geq \frac{2}{5}|X|$ distinct 3-edge connected components. ◀

► **Lemma 5.3.** *Let $G = (V, E)$ be a connected cubic multigraph with E partitioned into trails. Then G has a spanning tree that contains all edges that are not at the end of their trail.*

Proof. Let F be the set of edges that are not at the end of their trail. Since G is cubic, the graph (V, F) is a collection of vertex-disjoint paths, and in particular it is acyclic. Since G is connected, F can be extended to a spanning tree. ◀

Note that we can find this spanning tree in linear time e.g. by contracting all edges internal to a trail, finding a spanning tree of the resulting graph, and adding the internal trail edges to the edges of this spanning tree.

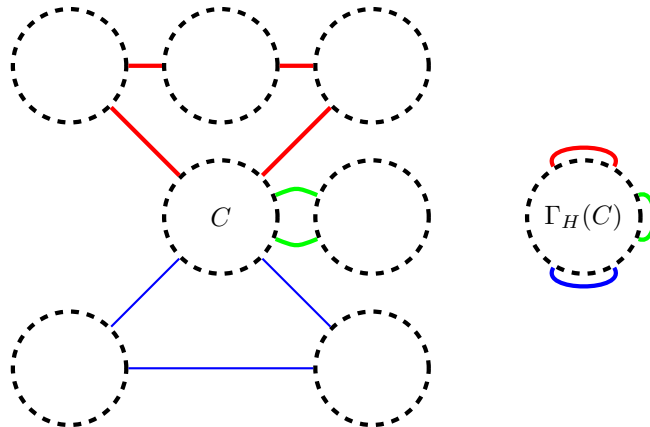
► **Lemma 5.4.** *Let $G = (V, E)$ be a cubic 2-edge connected multigraph with E partitioned into trails. Let T be a spanning tree of G containing all edges that are not at the end of their trail. Let H be a minimal subgraph of G (w.r.t inclusion) that contains T and is 2-edge connected. Then for any $k \geq 5$, less than $\frac{4}{5} \frac{k}{k-1} |V|$ of the vertices in H are in a 3-edge connected component with at least k vertices.*

Proof. Let X be the set of edges that are not in T . Since G is cubic, $|X| = \frac{1}{2}|V| + 1$. By Lemma 5.2 H has at least $\frac{2}{5}|X| > \frac{1}{5}|V|$ 3-edge connected components. Each such component contains at least one vertex, so the total number of vertices in components of size at least k is less than $\frac{k}{k-1} (|V| - \frac{1}{5}|V|) = \frac{4}{5} \frac{k}{k-1} |V|$. ◀

► **Definition 5.5.** Let C be a 3-edge connected component of some 2-edge connected graph H , whose edges are partitioned into trails. Define $\Gamma_H(C)$ to be the 3-edge connected graph obtained from C by inserting a new edge $\{e_1, f_1\}$ for each min-cut $\{e, f\}$ where $e = \{e_1, e_2\}$ and $f = \{f_1, f_2\}$ and $e_1, f_1 \in C$. Define the corresponding partition of the edges of $\Gamma_H(C)$ into trails by taking every trail that is completely contained in C , together with new trails combined from the fragments of the trails that were broken by the min-cuts together with the new edges that replaced them. See Figure 5.

At this point the idea of the algorithm can be explained. We remove as many of the edges that sit at the end of their trails as possible, while maintaining that the graph is 2-edge connected. Lemma 5.4 guarantees that we obtain a graph H with $\Omega(|V|)$ many 3-edge connected components of size $O(1)$. We solve the problem for each $\Gamma_H(C)$ for every 3-edge connected component. Finally, we combine the solutions for the different components like in the proof of Theorem 3.1.

► **Theorem 5.6.** *The one-way trail orientation problem can be solved in $\mathcal{O}(m + n)$ time on any 2-edge connected undirected graph or multigraph with n vertices and m edges.*



■ **Figure 5** The 3-edge connected components of a 2-edge connected graph. Notice that every edge leaving a 3-edge connected component C becomes part of a cycle if all 3-edge components are contracted. The right hand side shows $\Gamma_H(C)$ where C is the component in the middle.

Proof. By Lemma 5.1, we can assume the graph is cubic. For the algorithm we will use two subroutines. First of all, when we have found a minimum spanning tree T containing the edges that are not on the end of their trail we can use the algorithm of Kelsen et al. [5] to, in linear time, find a minimal (w.r.t. inclusion) subgraph H of G that contains T and is 2-edge connected. Secondly, we will use the algorithm by Mehlhorn et al. [7] to, in linear time, build the cactus graph of 3-edge connected components. The algorithm runs as follows:

1. Construct a spanning tree T of G that contains all edges that are not at the end of their trail.
2. Construct a minimal subgraph H of G that contains T and is 2-edge connected⁷.
3. Find the cactus of 3-edge connected components of⁸ H .
4. For each 3-edge connected component C_i , construct $\Gamma_H(C_i)$.
5. Recursively compute an orientation for each⁹ $\Gamma_H(C_i)$.
6. Combine the orientations from each component to a strong trail orientation of H . A such is also a strong trail orientation of G .

First we will show correctness and then we will determine the running time.

Recall that we can flip the orientation in each $\Gamma_H(C_i)$ and still obtain a strongly connected graph respecting the trails in $\Gamma_H(C_i)$. The way we construct the orientation of the edges of G is by flipping the orientation of each $\Gamma_H(C_i)$ in such a way that each cycle in the cactus graph becomes a directed cycle¹⁰. This can be done exactly because no edge of the cactus is contained in two cycles. By construction this orientation respects the trails so we need to argue that it gives a strongly connected graph.

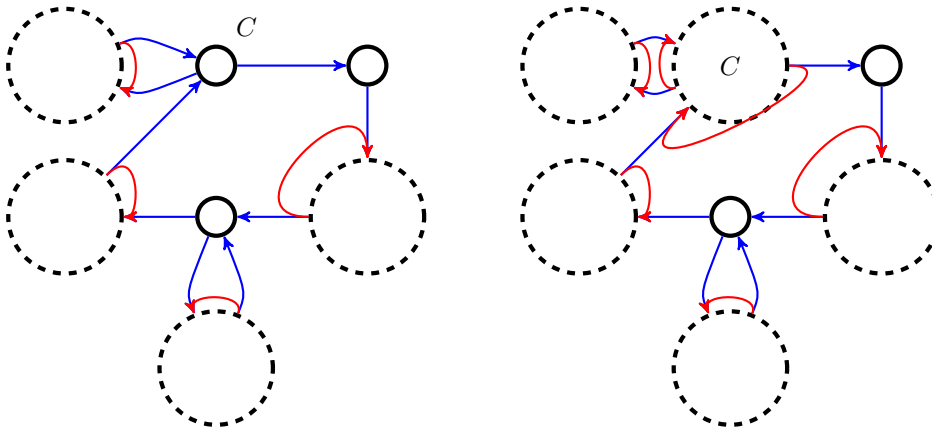
For showing that the resulting graph is strongly connected, consider the graph in which every 3-edge connected component is contracted to a single point. This is exactly the cactus of 3-edge connected component of G which is strongly connected as the cycles of the cactus

⁷ See Kelsen [5].

⁸ See Mehlhorn [7].

⁹ Note that $\Gamma_H(C_i)$ is cubic unless it consists of exactly one node. In this case however we don't need to do anything.

¹⁰ In practice this is done by making a DFS (or any other search tree one likes) of the cactus and repeatedly orienting each component in a way consistent with the previous ones.



■ **Figure 6** Before and after uncontracting component C . The blue edges are the edges of the cactus i.e. the 2-edge critical edges of H . The red edges are the ones obtained from the 2-edge cuts of H as described in the construction of the $\Gamma_H(C_i)$.

graph have become directed cycles. Now assume inductively that we have uncontracted some of the 3-edge connected components obtaining a graph G_1 which is strongly connected. We then uncontract another component C (see Figure 6) and obtain a new graph G_2 which we will show is strongly connected. If $u, v \in C$, then since $\Gamma_H(C)$ is strongly connected there is a path from u to v in $\Gamma_H(C)$. If this path only contains edges which are edges of C it will also exist in G_2 . If the path uses one of the added (now oriented) edges (e_1, f_1) , it is because there are edges (e_1, e_2) and (f_2, f_1) forming a cut and thus being part of a cycle in the cactus. In this case we use edge (e_1, e_2) to leave component C and then go from e_2 back to component C which is possible since G_1 was strongly connected. When we get back to the component C we must arrive at f_1 since otherwise there would be two cycles in the cactus containing the edge (e_1, e_2) . Hence we succeeded in disposing of the edge (e_1, f_1) with a directed path in G_2 . This argument can be used for any of the edges of $\Gamma_H(C)$ that are not in C and thus C is strongly connected in G_2 . Since G_1 was strongly connected this suffices to show that G_2 is strongly connected. By induction this implies that after uncontracting all components the resulting graph is strongly connected.

Now for the running time. By Lemma 5.4 each level of recursion reduces the number of vertices in “large” components by a constant fraction, for instance for $k = 10$ we reduce the number of vertices in components of size at least 10 by a factor of $\frac{8}{9}$. Let $f(n)$ be the worst case running time with n nodes for a cubic graph, and pick c large enough such that cn is larger than the time it takes to go through steps 1-4 and 6 as well as computing the orientations in the “small” components. This includes the linear time needed to construct the new set of trails (in 4), and the linear time to reassemble the directed trails (in 6). Let a_1, \dots, a_k be the number of vertices in the “large” 3-edge connected components. Then $\sum_i a_i \leq \frac{8n}{9}$ and

$$f(n) \leq cn + \sum_i f(a_i).$$

Inductively, we may assume that $f(a_i) \leq 9ca_i$ and thus obtain

$$f(n) \leq cn + \sum_i f(a_i) \leq cn + \sum_i 9ca_i = cn + 8cn = 9cn$$

proving that $f(n) \leq 9cn$ for all n . ◀

Algorithm 2: Linear time algorithm for cubic graphs.

Input: A 2-edge connected undirected cubic multigraph G and a partition \mathcal{P} of the edges of G into trails.

Output: G is modified to a strong trail orientation of (G, \mathcal{P}) .

- 1 Construct a spanning tree T of G that contains all edges that are not at the end of their trail.
- 2 Construct a minimal subgraph H of G that contains T and is 2-edge connected.
- 3 Find the cactus C of 3-edge connected components of H .
- 4 **for** each 3-edge connected component C_i in C in DFS preorder **do**
- 5 Construct $G_i = \Gamma_H(C_i)$.
- 6 Recursively compute an orientation for G_i .
- 7 **if** the orientation of G_i is not compatible with its DFS parent **then**
- 8 Flip orientation of G_i
- 9 **end**
- 10 **end**
- 11 **for** each edge e deleted from G to create H **do**
- 12 **if** no edge on the trail of e has been oriented yet **then**
- 13 Pick an arbitrary orientation for e .
- 14 **else**
- 15 Set the orientation of e to follow the trail.
- 16 **end**
- 17 **end**

6 Open problems

We here mention two problems concerning trail orientations which remain open.

First of all, our linear time algorithm for finding trail orientations only works for undirected graphs and it doesn't seem to generalise to the trail orientation problem for mixed graphs. It would be interesting to know whether there also exists a linear time algorithm working for mixed graphs. If so it would complete the picture of how fast an algorithm we can obtain for any variant of the trail orientation problem.

Secondly, our sufficient condition for when it is possible to solve the trail orientation problem for mixed multigraphs is clearly not necessary. It would be interesting to know whether there is a simple necessary and sufficient condition like there is in the undirected case. Since in the mixed case the answer to the problem actually depends on the given trail decomposition and not just on the structure of the mixed graph it is harder to provide such a condition. One can however give the following condition. It is possible to orient the trails making the resulting graph strongly connected if and only if when we repeatedly direct the forced trails end up with a graph satisfying our condition in Theorem 4.1. This condition is not simple and is not easy to check directly. Is there a more natural condition?

References

- 1 Frank Boesch and Ralph Tindell. Robbins's theorem for mixed multigraphs. *The American Mathematical Monthly*, 87(9):716–719, 1980. URL: <http://www.jstor.org/stable/2321858>.
- 2 Fan R. K. Chung, Michael R. Garey, and Robert E. Tarjan. Strongly connected orientations of mixed multigraphs. *Networks*, 15(4):477–484, 1985. doi:10.1002/net.3230150409.

- 3 Jacob Holm, Eva Rotenberg, and Mikkel Thorup. Dynamic bridge-finding in $\tilde{O}(\log^2 n)$ amortized time. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '18*, pages 35–52, 2018. URL: <http://dl.acm.org/citation.cfm?id=3174304.3175269>.
- 4 John Hopcroft and Robert Tarjan. Algorithm 447: Efficient algorithms for graph manipulation. *Commun. ACM*, 16(6):372–378, 1973. doi:10.1145/362248.362272.
- 5 Pierre Kelsen and Vijaya Ramachandran. On finding minimal 2-connected subgraphs. In *Proceedings of the Second Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms, 28-30 January 1991, San Francisco, California.*, pages 178–187, 1991. URL: <http://dl.acm.org/citation.cfm?id=127787.127824>.
- 6 Zoltán Király and Zoltán Szigeti. Simultaneous well-balanced orientations of graphs. *J. Comb. Theory, Ser. B*, 96(5):684–692, 2006.
- 7 Kurt Mehlhorn, Adrian Neumann, and Jens M. Schmidt. Certifying 3-edge-connectivity. *Algorithmica*, 77(2):309–335, 2017. doi:10.1007/s00453-015-0075-x.
- 8 Hiroshi Nagamochi and Toshihide Ibaraki. *Algorithmic Aspects of Graph Connectivity*. Cambridge University Press, New York, NY, USA, 1 edition, 2008.
- 9 J. A. Nash-Williams. On orientations, connectivity and odd-vertex-pairings in finite graphs. *Canad. J. Math.*, 12(5):555–567, 1960.
- 10 H. E. Robbins. A theorem on graphs, with an application to a problem of traffic control. *The American Mathematical Monthly*, 46(5):281–283, 1939. URL: <http://www.jstor.org/stable/2303897>.