Reconfiguration of Graph Minors

Benjamin Moore
Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Canada
brmoore@uwaterloo.ca

Naomi Nishimura
David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, Canada
nishi@uwaterloo.ca

Vijay Subramanya
David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, Canada
v7subram@uwaterloo.ca

Abstract

Under the reconfiguration framework, we consider the various ways that a target graph H is a *minor* of a host graph G, where a subgraph of G can be transformed into H by means of *edge contraction* (replacement of both endpoints of an edge by a new vertex adjacent to any vertex adjacent to either endpoint). Equivalently, an H*-model* of G is a labeling of the vertices of G with the vertices of H, where the contraction of all edges between identically-labeled vertices results in a graph containing representations of all edges in H.

We explore the properties of G and H that result in a connected *reconfiguration graph*, in which nodes represent H-models and two nodes are adjacent if their corresponding H-models differ by the label of a single vertex of G. Various operations on G or H are shown to preserve connectivity. In addition, we demonstrate properties of graphs G that result in connectivity for the target graphs K_2, K_3, and K_4, including a full characterization of graphs G that result in connectivity for K_2-models, as well as the relationship between connectivity of G and other H-models.

2012 ACM Subject Classification Mathematics of computing → Graph theory

Keywords and phrases reconfiguration, graph minors, graph algorithms

Digital Object Identifier 10.4230/LIPIcs.MFCS.2018.75

Related Version A full version of the paper is available at `https://arxiv.org/pdf/1804.09240.pdf`.

Funding Supported by the Natural Sciences and Engineering Research Council of Canada

Acknowledgements We wish to thank Tesshu Hanaka, Venkatesh Raman, Krishna Vaidyanathan, and Marcin Wrochna for helpful discussions.

1 Introduction

Graph minors have been studied extensively as a means for categorizing graphs and exploiting their properties. A graph H is a *minor* of a graph G if H can be formed from a subgraph of G by a series of edge contractions, where the *contraction* of the edge uv results in the replacement of both u and v by a new vertex w that is adjacent to any vertex that was adjacent to u or v (or both). Much of the research in the area has focused on classes of graphs that are closed under the taking of minors, and on exploiting properties of graphs known not to contain certain graphs as minors. For example, it is known that for every minor closed class, that class is characterized by a finite set of forbidden minors [12]. Additionally, it has been shown that for any fixed graph H, every H-minor-free graph of treewidth w has an $\Omega(w) \times \Omega(w)$ grid as a minor [2].

In our work, we instead focus on the solution space of H-models of a graph G using the reconfiguration framework [6, 11, 14], where an H-model is a mapping that labels the vertices of G with the vertices of H. A *reconfiguration graph* for an instance of a problem consists of a node for each possible feasible solution and an edge between any two nodes representing solutions that are adjacent. The definition of adjacency may be presented as a *reconfiguration step* used to transform a solution into a neighbouring solution. Structural properties of the reconfiguration graph, including its diameter and whether it is connected, are of interest both in their own right and as keys to solving algorithmic problems, such as determining whether there is a path (or *reconfiguration sequence*) in the graph between two given vertices and, if so, finding the shortest such path.

The reconfiguration framework has been used to explore the relationships among solutions to instances of many graph problems, such as Independent Set, Shortest Path, and k-Colouring. Practical uses of such work include determining whether and how it is possible to make incremental changes to form one solution from another, ensuring that at each step along the way, the intermediate configuration is also a solution. The study of minor reconfiguration is a natural extension of work that determines conditions under which reconfiguration graphs are guaranteed to be connected, as well as recent work on mappings between graphs, in which each node in the reconfiguration graph represents a subgraph in a specified class [5, 10]. In addition, our results widen the scope of work in which configurations are represented as labelings, which previously had been limited to problems entailing moving labels from a source to a target configuration using the minimum number of exchanges of labels on adjacent vertices (detailed in a survey of reconfiguration [11]), and the use of labeled edges in the reconfiguring of triangulations [8].

In this paper, we consider how the connectivity of the reconfiguration graph depends on the choices of the host G and target H. We consider an instance of Minor Reconfiguration to consist of a *host graph* G and *target graph* H such that H is a minor of G. Each node in the reconfiguration graph consists of a labeling of the vertices of G with the vertices of H (or, more simply, integers in $\{1, \ldots, |V(H)|\}$) such that the contraction of each edge with identically-labeled endpoints results in a graph that, upon deletion of zero or more edges, yields H. We consider two H-models to be adjacent if they differ by a single label.

We begin by establishing properties of k-connected graphs and minors in Section 2, based on which we form a toolkit of techniques used in reconfiguration (Section 3). We consider various properties of G and H that determine whether or not the reconfiguration graph is connected. For a target graph H, we define $\text{host}(H)$ to be the set of host graphs G such that the reconfiguration graph for G and H is connected (Definition 8). We then focus on characterizing $\text{host}(K_2)$ (Section 4), $\text{host}(K_3)$ (Section 5), and $\text{host}(K_4)$ (Section 6). Finally, in Section 7, we summarize the results and present directions for future work.

2 Preliminaries

We define key terms used in the description of graphs; for common terms not defined in this paper, the reader is referred to a resource on graph theory [3]. We will frequently focus on subsets of the vertices; for a subgraph $V \subseteq V(G)$, the *induced subgraph* $G[V]$ is the subgraph with vertex set V and edge set $\{uv \in V(G) \mid u, v \in V\}$. As shorthand, for G a graph and S a set of vertices, we use $G \setminus S$ to denote $G[V(G) \setminus S]$. In order to avoid confusion with the vertices in graphs G and H, we refer to the *nodes* of a reconfiguration graph.

2.1 Properties of k-connected graphs

We focus on various ways of connecting vertices in the graph. A *cut set* S of G is a set of vertices such that $G \setminus S$ consists of at least two components; the member of a cut set of size one is also called a *cut vertex*. A *bridge* is an edge whose deletion disconnects the graph. A graph is k-connected if there is no cut set of size k. Equivalently, in a k-connected graph there exist k vertex-disjoint paths between any pair of vertices in the graph. At times we will focus on how highly connected a specific vertex might be. A *universal vertex* is adjacent to all other vertices in the graph. In a *complete graph* on j vertices, denoted K_j, all vertices are universal vertices.

To characterize the behaviour of various host and target graphs, we make use of characterizations of graphs in terms of a base graph class and a series of operations. *Adding an edge* consists of adding an edge between two vertices in $V(G)$. To *split* a vertex v is to first delete v from G, and then add two vertices v_1 and v_2 to G such that $v_1v_2 \in E(G)$, each neighbour of v in G is a neighbour of exactly one of v_1 or v_2, and $\deg(v_i) \geq 3$ for $i \in \{1, 2\}$.

We make use of Tutte's characterization of 3-connected graphs and Ding and Qin's characterization of a subset of the 4-connected graphs, given below. The base case for Tutte's characterization is a wheel, defined as follows.

▶ **Definition 1.** A *k-wheel W_k* is a graph on $k+1$ vertices, the *rim vertices* $r_1, \ldots, r_k$ and the *hub vertex* h, where there is a cycle induced on the rim vertices and an edge between h and each rim vertex.

▶ **Theorem 2.** *[13] A graph is 3-connected if and only if it is obtained from a wheel by repeatedly adding edges and splitting vertices.*

To state Ding and Qin's result, we need a few additional definitions. The *line graph* $L(G)$ of a graph G has a vertex corresponding to each edge of G and two vertices are adjacent if their corresponding edges share an endpoint in G. A graph is *cubic* if each vertex has degree three. Furthermore, a cubic graph with at least six vertices is *internally 4-connected* if its line graph is 4-connected. One of the base classes for their characterization is a square of a cycle, as defined below.

▶ **Definition 3.** The *square of a cycle* C_k^2 is formed from the cycle C_k by adding an edge between any pair of vertices joined by a path of length two.

Finally, we say a sequence of 4-connected graphs $G_1, \ldots, G_n$ form a *(G_1, G_n)-chain* if for all $i \in \{1, \ldots, n-1\}$, there exists an edge e such that G_{i+1} is formed from G_i by removal of the edge e. Theorem 4 is a generalization of a well-known theorem of Martinov [9].

▶ **Theorem 4.** *[4] Let $\mathcal{C} = \{C_k^2 : k \geq 5\}$ and $\mathcal{L} = \{H : H$ is the line graph of an internally 4-connected cubic graph$\}$. Let G be a 4-connected graph not in $\mathcal{C} \cup \mathcal{L}$. Then if G is planar, there is a (G, C_6^2)-chain. Otherwise, there is a (G, K_5)-chain.*

2.2 Branch sets, H-models, host(H), and block trees

For the purposes of reconfiguration, we make use of an equivalent definition of a minor as a mapping of each vertex of host graph G to a vertex of target graph H. For convenience, we sometimes represent the vertices of H as integer labels.

▶ **Definition 5.** For graphs G and H and mapping $f : V(G) \to V(H)$, we refer to $f(v)$ as the *label* of v and define the *branch set* $G(f, i)$ to be the subgraph of G induced on the set of vertices with label i.

For ease, we will make use of $|G(f,i)|$ to denote $|V(G(f,i))|$. Given a mapping between $V(G)$ and $V(H)$, an edge of G is a *connecting edge* if its endpoints are members of two different branch sets, and we say that it *connects* those two branch sets. A mapping is equivalent to a minor when two additional properties hold, as indicated in Definition 6.

▶ **Definition 6.** For graphs G and H and mapping $f : V(G) \to V(H)$, we say that H is a *minor of* G and that f is an H*-model of* G if the following conditions hold:
1. for each $i \in V(H)$, each branch set $G(f,i)$ is nonempty and connected, and
2. for each edge $ij \in E(H)$, there exists an edge in $E(G)$ connecting $G(f,i)$ and $G(f,j)$.

▶ **Definition 7.** For any graphs G and H such that H is a minor of G, the *branch set reconfiguration graph* of G and H consists of a node for each H-model of G and, for each pair of H-models f and g, an edge between the nodes corresponding to f and g if and only if there exists a $v \in V(G)$ such that $f(v) \neq g(v)$ and for all $u \neq v$, $f(u) = g(u)$.

▶ **Definition 8.** For any graphs G and H such that H is a minor of G, $G \in \text{host}(H)$ if and only if the branch set reconfiguration graph of G and H is connected.

We will often find it convenient to view each branch set in terms of the tree structure of its 2-connected components. A *block* of a connected graph is either a maximal 2-connected subgraph or one of the endpoints of a bridge. The *block tree* of a connected graph consists of a node for each block B; there is an edge between the nodes corresponding to blocks B and B' if there exists a cut vertex v of G such that $V(B) \cap V(B') = \{v\}$. Given a graph G, an H-model f, and a label a, we use $T(G,f,a)$ to denote the block tree for $G(f,a)$. In addition, for a subgraph A of G, we use $T(G,f,a,A)$ to denote the subtree of $T(G,f,a)$ induced by the blocks containing vertices in $V(G(f,a)) \cap V(A)$. For convenience, we sometimes use "block of $G(f,a)$" to refer to a block of $T(G,f,a)$.

To make use of the tree structure, our algorithms typically process a block tree starting with blocks that are leaves of their block trees, or *leaf blocks*; a branch set that is 2-connected can be viewed as having a block tree consisting of a single leaf block. For ease of description, we will refer to the cut vertices of G that appear in multiple blocks as *joining vertices* and all other vertices as *interior vertices*.

2.3 Essential edges, crucial vertices, weak connections, and lynchpins

When considering how labelings can be reconfigured, we need to ensure that we retain the connecting edges as required in Definition 6. In doing so, we need to pay particular attention to vertices and edges whose relabeling will cause problems.

When there exists only a single edge that connects a pair of branch sets with labels a and b, $ab \in E(H)$, we call such an edge an *essential edge*, and denote it as $\text{ess}(a,b)$. If all the edges between branch sets with labels a and b have the same endpoint in a, we call that vertex an *essential vertex for* b; clearly every endpoint of an essential edge is an essential vertex, but not every essential vertex is the endpoint of an essential edge.

The presence of essential vertices will be important in determining when it is easy to relabel vertices. For any two labels, if the branch set with label a contains an essential vertex for b or if the branch set with label b contains an essential vertex for a, we will say that the branch sets with labels a and b are *weakly connected*, or form a *weak connection*.

Our results rely on the interplay between the presence of weak connections and the connectivity of a graph. For each weak connection, we identify a vertex as the *lynchpin* for the connection. When the branch sets with labels a and b are weakly connected by an essential edge, then either of the endpoints of the essential edge can be designated as the lynchpin. Otherwise, the (single) essential vertex giving rise to the weak connection is the lynchpin for that connection. We will use lynchpins to form cut sets between non-lynchpins and other branch sets.

A vertex v with label a is a *crucial vertex* if it is an essential vertex for b and an essential vertex for c, for $b \neq c$, and a *non-crucial vertex* otherwise. If for some distinct labels a, b, and c, a vertex $v \in G(f,a)$ is essential for c and also has at least one neighbour in $G(f,b)$, then v is a *b-crucial vertex*. Clearly, a vertex in $G(f,a)$ that is essential for b and c is crucial, b-crucial, and c-crucial.

2.4 Properties of H-models of k-connected graphs

When G is k-connected, we are able to establish properties of connecting edges of branch sets, as shown in Lemma 9 and Lemma 10, as well as the structure of weak edges (Lemma 11). The results make use of the fact that in a k-connected graph there cannot be a cut set of size less than k separating any two vertices; cut sets are typically formed from the joining vertices of leaf blocks and lynchpins, and vertices separated by cut sets are typically non-lynchpins. Proofs of results marked with (*) have been omitted due to space limitations.

▶ **Lemma 9 (*).** *Given a k-connected graph G and an H-model f of G for some graph H, for any branch set $G(f,a)$, each leaf block has $k-1$ interior vertices that are endpoints of connecting edges.*

▶ **Lemma 10.** *Given a k-connected graph G and an H-model f of G, where $|V(H)| = k$, suppose there exist branch sets $G(f,\ell)$ and $G(f,m)$ such that $\ell m \in E(H)$ and there are weak connections between $G(f,\ell)$ and each branch set other than $G(f,m)$ (where a weak connection between $G(f,\ell)$ and $G(f,m)$ is possible but not required). Then, the following hold:*

1. *Each leaf block in $G(f,\ell)$ must contain an interior vertex that is the endpoint of a connecting edge to $G(f,m)$.*
2. *If it is possible to designate lynchpins of the weak connections such that $G(f,\ell)$ contains a non-lynchpin, then each leaf block in $G(f,m)$ must contain an interior vertex that is the endpoint of a connecting edge to $G(f,\ell)$.*

Proof. To see why the first point holds, suppose instead that no such interior vertex existed in a leaf block of $G(f,\ell)$. Then, each path between an interior vertex u in the leaf block in $G(f,\ell)$ and any vertex v in $G(f,m)$ must pass through either the joining vertex of the leaf block or one of the lynchpins for the weak connections. However, u and v are thus separated by a cut set of size at most $k-1$, contradicting the k-connectivity of G.

The argument for the second point is similar; we can show that the joining vertex of the leaf block in $G(f,m)$ and the lynchpins of the weak connections form a cut set of size at most $k-1$ separating any interior vertex in the leaf block and the non-lynchpin in $G(f,\ell)$. ◀

▶ **Lemma 11 (*).** *Given a k-connected graph G and a K_k-model f of G such that there is a branch set B with weak connections to all other branch sets, it is not possible to designate lynchpins such that B contains at least one vertex x that is not a lynchpin for any of the weak connections, and at least one other branch set contains a vertex y that is not a lynchpin for any of the weak connections.*

3 Toolkit for reconfiguration of minors

In this section, we introduce techniques and properties that are exploited in the results found in the rest of the paper. In particular, we focus on the types of steps used in reconfiguration and the properties that need to be satisfied for each type of transformation. In Lemmas 12 and 13 we determine conditions under which a vertex can be relabeled in a single step. In the remainder of the section, we present results that can be used to handle more complex situations in which one or more of the conditions do not hold.

Lemma 12 delineates the conditions necessary for a vertex to be able to be relabeled from a to b in a single reconfiguration step: it cannot be the only vertex with label a, it cannot be a cut vertex in its branch set, it must be connected to a vertex with label b, and it is not incident with every edge between the branch sets for labels a and c, where $c \neq b$.

▶ **Lemma 12 (*).** *Given a graph G and an H-model f of G, a vertex v can be relabeled from a to b in a single reconfiguration step if and only if the following conditions hold:*
1. $|G(f, a)| > 1$*;*
2. *v is not a cut vertex in $G(f, a)$;*
3. *v has at least one neighbour in $G(f, b)$; and*
4. *v is not a b-crucial vertex.*

Because several of the conditions hold automatically for a universal vertex, the following lemma lists a smaller number of conditions:

▶ **Lemma 13 (*).** *Given a graph G and an H-model f of G, a vertex v can be relabeled from a to b in a single reconfiguration step if the following conditions hold:*
1. *$G(f, a)$ contains a universal vertex u such that $u \neq v$; and*
2. *v has at least one neighbour in $G(f, b)$.*

When neither Lemma 12 nor Lemma 13 applies, the relabeling of a vertex requires a series of reconfiguration steps. When the vertex to be relabeled is the only member of its branch set or a crucial vertex, we first need to fill its branch set with new vertices that can provide the necessary connecting edges to other branch sets. When the vertex to be relabeled is a cut vertex, we will need to siphon away vertices from its branch set so that it is no longer a cut vertex among the remaining vertices with its label.

When a branch set is a block tree, both filling and siphoning entail the relabeling of vertices in a branch set block by block, starting at the leaf blocks. If we are able to relabel all the interior vertices of a leaf block, we can simplify the block tree by removing the leaf block. We will show in Lemma 20 that such relabeling is possible as long as we can avoid certain bad situations involving *leaf-crucial models* and *leaf-ℓ-crucial models*, as outlined in Definitions 14, 15, and 16.

▶ **Definition 14.** Given a graph G and an H-model f of G, we say that a vertex v is a *leaf-crucial vertex* if v is a crucial vertex that is an interior vertex in a leaf block in its branch set. An H-model that contains a leaf-crucial vertex is a *leaf-crucial model.*

▶ **Definition 15.** Given a graph G and an H-model f of G, we say that a vertex v is a *leaf-ℓ-crucial vertex* if v is an ℓ-crucial vertex that is an interior vertex in a leaf block in its branch set. An H-model that contains a leaf-ℓ-crucial vertex is a *leaf-ℓ-crucial model.*

▶ **Definition 16.** Given a graph G, an H-model f of G, a label a, and a subgraph A of $G(f, a)$, we say that *f hits a leaf-crucial model on relabeling A* if any relabeling of f that changes only the vertices of A can be extended by relabeling only the vertices of A to reach a leaf-crucial model, and that *f hits a leaf-ℓ-crucial model on relabeling A* if any relabeling of f that changes only the vertices in A can be extended by relabeling only the vertices of A to reach a leaf-ℓ-crucial model.

▶ **Observation 17.** *Given a graph G, an H-model f of G, a label a, and a subgraph A of $G(f,a)$, if f does not hit a leaf-ℓ-crucial model on relabeling A, then for each model g reachable from f by relabeling only the vertices of A, no interior vertex in a leaf block of $T(G,g,a)$ is ℓ-crucial.*

In Lemma 18, we show that each time we relabel an interior vertex in a leaf block, if the leaf block still has interior vertices, one will be a neighbour of the relabeled vertex. We use the result in Lemma 19 to show that if we can avoid leaf-crucial models, then it is possible to relabel all the interior vertices in a leaf block (and hence remove it from the branch set). By repeatedly relabeling leaf blocks, an entire connected component can be siphoned away, as shown in Lemma 20.

▶ **Lemma 18.** *Given a graph G and an H-model f of G, suppose there exist labels a and b and a vertex v such that $|G(f,a)| \geq 2$, v is in a leaf block L of $T(G,f,a)$, and relabeling v to b (and no other vertices) results in another H-model g. Then v has a neighbour in $G(g,a)$, and if $|V(L) \setminus \{v\}| \geq 2$, then v has a neighbour u in $G(g,a)$ such that $u \in V(L)$ and u is an interior vertex in a leaf block of $T(G,g,a)$.*

Proof. We observe that since Lemma 12 holds for the relabeling of v from a to b, $|G(f,a)| \geq 2$, and since each branch set is connected, v must then have a neighbour in $G(g,a)$.

We now suppose that $|V(L) \setminus \{v\}| \geq 2$. At least one neighbour $u \in V(L)$ of v is in a leaf block L_g of $T(G,g,a)$, as v is not a cut vertex of $G(f,a)$ (by Lemma 12, condition 2) and L is a 2-connected leaf block of $T(G,f,a)$. If u is an interior vertex in $G(g,a)$, we are done. If instead u is a joining vertex and there exists no interior vertex in L_g that is a neighbour of v, then for each interior vertex $w \in V(L_g)$, u lies on every path from v to w in L, which contradicts the fact that L is 2-connected. Hence, v is adjacent to an interior vertex in $G(g,a)$. ◀

▶ **Lemma 19 (*).** *Given a graph G, an H-model f of G, a label a, and a leaf block L of $T(G,f,a)$, suppose that $|V(G(f,a)) \setminus V(L)| \geq 1$, L contains at least one interior vertex that is the endpoint of a connecting edge, and f does not hit a leaf-crucial model on relabeling L. Then we can reconfigure f to a model g such that $g(v) \neq f(v)$ for each $v \in V(L)$ that is an interior vertex of $G(f,a)$ and $g(u) = f(u)$ for all other vertices u.*

▶ **Lemma 20.** *Given a 2-connected graph G and an H-model f of G, suppose there exist $ab \in E(H)$, a cut vertex x of $G(f,a)$, and a connected component C of $G(f,a) \setminus \{x\}$ that contains at least one vertex with a neighbour in $G(f,b)$ such that f does not hit a leaf-crucial model or a leaf-b-crucial model on relabeling C. Then we can reconfigure f to a model g such that $g(v) \neq f(v)$ for each $v \in V(C)$, $g(u) = f(u)$ for all $u \notin V(C)$, and x has a neighbour in $G(g,b)$.*

Proof. We use B to denote the block of $T(G,f,a,C)$ containing x, and view $T(G,f,a,C)$ as rooted at B. We observe that any leaf block of $T(G,f,a,C)$ is also a leaf block of $T(G,f,a)$.

To reconfigure f to g, we work up the tree $T(G,f,a,C)$ from leaf blocks up to B, at each step relabeling all the vertices in the current block with labels different from a. Specifically, if a leaf block does not have an interior vertex with a neighbour labeled b, then in Case 1 below, we can relabel the vertices in the block; such a relabeling removes the block from the branch set for label a. If instead a leaf block does have an interior vertex with a neighbour labeled b, then in Case 2 below, we can relabel the block with b. Such a relabeling not only removes the block from the branch set for label a, but also ensures that the joining vertex of the block has a neighbour with label b. Repeated applications of the two cases suffice to ensure that we eventually reach a point in the process at which B is a leaf block and contains an interior vertex with a neighbour labeled b; using Case 2, we can then satisfy the statement of the lemma by ensuring that every vertex in B except x receives label b.

Case 1: A leaf block L of $T(G, f, a, C)$ does not contain an interior vertex with a neighbour in $G(f, b)$.

Since G is 2-connected, by Lemma 9, L has at least one interior vertex that is an endpoint of a connecting edge. Because f does not hit a leaf-crucial model and $|V(G(f, a)) \setminus V(L)| \geq 1$, by Lemma 19, we can relabel the interior vertices of L.

Case 2: A leaf block L of $T(G, f, a, C)$ contains an interior vertex v with a neighbour in $G(f, b)$.

We first observe that we can relabel v to b: since f does not hit a leaf-b-crucial model on relabeling C, v is not b-crucial (Observation 17), and hence all the conditions of Lemma 12 hold. We can then repeat the same argument on the resulting model h, as follows. For L_h the leaf block of $T(G, h, a, C)$ such that $V(L_h) = V(L) \setminus \{v\}$, if one exists, by Lemma 18, L_h contains an interior vertex u that is a neighbour of v. We can then use the fact that f does not hit a leaf-b-crucial model on relabeling C to again apply Observation 17 to conclude that u is not b-crucial and that Lemma 12 is satisfied for the labeling of u to b. Further repetitions of the argument result in the relabeling of all interior vertices of L to b. ◀

4 Characterizing host(K_2)

Theorem 21 fully characterizes host(K_2); as a consequence, we can use membership in host(K_2) as an alternate definition of 2-connectivity. The reconfiguration of a 2-connected graph G is achieved by defining a canonical model (one in which one vertex has one label and all other vertices have the other label) and then showing it is possible both to reconfigure any K_2-model to a canonical model and to reconfigure between canonical models. In contrast, when G is not 2-connected, the presence of a cut vertex prevents reconfiguration, as no ordering of relabeling steps can prevent a branch set from being disconnected.

▶ **Theorem 21 (*).** *$G \in host(K_2)$ if and only if G is 2-connected.*

5 Characterizing host(K_3)

To show that every 3-connected graph is in host(K_3), we make use of Tutte's characterization in Theorem 2. In order to prove Theorem 22, it suffices to show that wheels are in host(K_3) (Corollary 24) and that connectivity is preserved under the splitting of vertices (Lemma 27) and adding of edges (Lemma 30).

▶ **Theorem 22 (*).** *Every 3-connected graph is in host(K_3).*

The result for wheels (Corollary 24) follows from a result on a generalization of wheels by allowing multiple hub vertices, each of which is a universal vertex, and replacing each rim vertex by a connected graph. To obtain a more general result, we use $W(G_1, G_2, \ldots, G_m, n, \ell, m)$ to denote a generalized wheel, for each G_i a connected graph on n vertices, $V(G_i) = \{v_{(i,1)} \ldots v_{(i,n)}\}$, and ℓ and m both positive integers. The graph $W(G_1, G_2, \ldots, G_m, n, \ell, m)$ consists of ℓ *hub vertices*, $V_H = \{h_1, \ldots, h_\ell\}$, and mn *subgraph vertices*, $V_S = \{s_{i,j} \mid 1 \leq i \leq m, 1 \leq j \leq n\}$, where $s_{i,j}$ corresponds to $v_{(i,j)}$. The edge set consists of the *hub edges*,

$E_H = \{h_i h_j \mid 1 \leq i \leq \ell, 1 \leq j \leq \ell, i \neq j\}$, the *subgraph edges*, $E_s = \{s_{i,j}s_{i,k} \mid v_{(i,j)}v_{(i,k)} \in E(G_i), 1 \leq i \leq m\}$, the *rim edges*, $E_R = \{s_{(i,j)}s_{(k,j)} \mid k \equiv i+1 \bmod m, 1 \leq j \leq n\}$, and the *connecting edges*, $E_C = \{h_k s_{(i,j)} \mid 1 \leq k \leq \ell, 1 \leq i \leq m, 1 \leq j \leq n\}$. Observe that $W(G_1, G_2, \ldots, G_m, n, l, m)$ has a K_{l+2}-minor when $m \geq 3$.

▶ **Lemma 23 (*).** *For any graphs G_i, $W(G_1, G_2, \ldots, G_m, n, \ell, m)$ is in host$(K_{\ell+2})$ for any $m \geq 3$.*

▶ **Corollary 24.** *W_n is in host(K_3).*

We consider the splitting of vertices in two steps. In Lemma 25, which applies more generally to K_k for any $k > 2$, we show that we can reconfigure between models in which the vertices resulting from the split have the same label. Then, in Lemma 27, which uses Lemma 26, we consider cases in which the vertices can have different labels.

▶ **Lemma 25.** *Let G be a 2-connected graph and G' be formed from G by splitting a vertex v into vertices x and y. For any $k > 2$, let f and g be K_k-models of G and f' and g' be K_k models of G' such that $f(v) = f'(x) = f'(y)$, $g(v) = g'(x) = g'(y)$ and for all $u \in V(G) \setminus \{v\}$, $f(u) = f'(u)$ and $g(u) = g'(u)$. If f and g are reconfigurable, then f' and g' are reconfigurable.*

Proof. Since f and g are reconfigurable, there is a reconfiguration sequence $\sigma = f = f_1, \ldots, f_\ell = g$ for some value of ℓ. Using σ, we wish to form a reconfiguration sequence σ' from f' to g' in the reconfiguration graph for G'. In forming the sequence, we observe that if there is a prefix τ of σ such that v is not relabeled in any of the steps, then we can form a prefix τ' of σ' by executing the same sequence of steps.

We now consider the first relabeling of v in σ, say from f_j to f_{j+1}; we wish to show that in σ', we can relabel both x and y in the same way. Without loss of generality, we assume that $f_j(v) = f'_j(x) = f'_j(y) = a$ and that $f_{j+1}(v) = b$.

We can use the fact that Lemma 12 holds for the labeling of v by b to establish useful properties of x and y. Because v is not in a branch set of size one (condition 1), x and y are not the only two vertices in $G'(f'_j, a)$. Since v is a not a cut vertex of its branch set (condition 2), x and y together cannot form a cut set. As v has a neighbour with label b (condition 3), either x or y (or both) must have a neighbour with label b under f'_j. Finally, because v is not a b-crucial vertex (condition 4), there must exist some vertex other than x or y in $G'(f'_j, a)$ that has a neighbour with label b.

Without loss of generality, we assume that x has a neighbour with label b, and consider two cases, depending on whether or not x is a joining vertex in the block tree of $G'(f'_j, a)$.

Case 1: x is a not a joining vertex of the block tree of $G'(f'_j, a)$

By our observations above, all conditions of Lemma 12 hold for the relabeling of x to b. As a consequence of the relabeling, y now has a neighbour with label b. Because v was not a cut vertex in $G(f_j, a)$, the removal of both x and y cannot disconnect $G'(f'_j, a)$, and hence condition 2 holds for y. As the remaining conditions of Lemma 12 were established in the argument above, we can now relabel y to b, as needed.

Case 2: x is a joining vertex in the block tree of $G'(f'_j, a)$

We first show that the component C of $G'(f'_j, a)$ containing y consists solely of the vertex y. If instead there existed another vertex in C, then the removal of both x and y would separate the vertex from the other components of $G'(f'_j, a) \setminus \{x\}$, and consequently v would be a cut vertex in $G(f_j, a)$, which contradicts Lemma 12 for the relabeling of v to b.

By the definition of the split operation, $\deg(y) \geq 3$; because x is y's only neighbour with label a, y must have at least one neighbour with label b or c, for some $c \notin \{a, b\}$. If y has a neighbour with label b, we can use Case 1 to complete the relabeling of y and then x. Otherwise, we will show that we can first relabel y to c, relabel x to b, and finally relabel y to b. In each case, we show that we can use Lemma 12.

To see that we can relabel y to c, it suffices to observe that y is not c-crucial, as x has a neighbour in b, and if y were essential for some $d \notin \{b, c\}$, then v would be essential for d in f_j, and hence b-crucial in f_j, which is a contradiction. Now, since x is now no longer a cut vertex, we can relabel x to b. Finally, since y now has a neighbour with label b (that is, x), we can relabel y with b, as needed. ◀

▶ **Lemma 26.** *Given a 3-connected graph G and a K_3-model f of G, suppose there exists a cut vertex x of $G(f, a)$ with a neighbour in $G(f, b)$ such that x is not essential for c. Then there exists a component D of $G(f, a) \setminus \{x\}$ such that we can reconfigure f to a model g in which $g(v) = f(v)$ for each $v \in V(D)$, $g(x) = b$, and $g(u) \neq f(u)$ for all other vertices of $G(f, a)$.*

Proof. We form model g by first siphoning all components of $G(f, a) \setminus \{x\}$ except D out of $G(f, a)$ and then by relabeling x to b.

Because x is not essential for c, there must exist at least one component of $G(f, a) \setminus \{x\}$ that contains a vertex with a neighbour in $G(f, c)$; we choose D to be one such component.

By Lemma 9, each other component C must have a neighbour in either $G(f, b)$ or $G(f, c)$. Since x has a neighbour in $G(f, b)$ and D has a neighbour in $G(f, c)$, f does not hit a leaf-crucial model, a leaf-b-crucial model, or a leaf-c-crucial model on relabeling C, and hence we can use Lemma 20 to relabel all vertices of C with either b or c.

After the vertices of every component except D have been relabeled, all the conditions for Lemma 12 now hold for the relabeling of x to b: the branch set for label a has at least one vertex other than x, x is no longer a cut vertex, x has a neighbour labeled b, and a vertex in D has a neighbour labeled c. ◀

▶ **Lemma 27.** *Suppose G is a 3-connected graph such that $G \in host(K_3)$ and G' is a graph formed from G by splitting a vertex v into vertices x and y. Then G' is in $host(K_3)$.*

Proof. We show that for any source and target K_3-models of G', we can find a reconfiguration sequence from the source to the target. We know from Lemma 25 that we can reconfigure between any two K_3-models in which x and y have the same labels. Here, we show that we can reconfigure any K_3-model to a K_3-model in which x and y have the same labels. This suffices to demonstrate the existence of a reconfiguration sequence between the source and target K_3-models, as we reconfigure from the source K_3-model to a K_3-model in which x and y have the same labels, then to another K_3-model in which x and y have the same labels, and finally to the target K_3-model.

Without loss of generality, we assume that the labels are a, b, and c, and that in the starting K_3-model, $f'(x) = a$ and $f'(y) = b$. If Lemma 12 holds for either relabeling x to b or relabeling y to a, then we can accomplish the reconfiguration in a single step. Similarly, the reconfiguration can be accomplished using Lemma 26 if either x or y is a cut vertex that is not essential for c.

Thus, it suffices to consider the cases in which either condition 1 or 4 of Lemma 12 must be violated for both relabeling x to b and relabeling y to a. In any case, both x and y will be essential for c. By Lemma 11, it is not possible for both x and y to be essential for c unless $|G'(f', a)| = |G'(f', b)| = 1$. Thus, it suffices to consider the case $|G'(f', a)| = |G'(f', b)| = 1$.

Due to the 3-connectivity of G, x and y will each have at least two neighbours in $G'(f', c)$. We will show that one of y's neighbours w can be relabeled b, after which y can be relabeled a.

If Lemma 12 does not apply for the relabeling of w to b, then because w is not b-crucial, the only possible condition of Lemma 12 that can be violated is condition 2. Suppose that every neighbour of y in $G'(f', c)$ is a cut vertex. Since by Lemma 9, each leaf block in $G'(f', c)$ has two interior vertices that are endpoints of connecting edges, each of these edges must connect to x. By 3-connectivity, there must be three vertex-disjoint paths to y from an interior vertex in a leaf block in $G(f', c)$. As only one can pass through x and only one can pass through the joining vertex of the leaf block, there can only be two paths at most, forming a contradiction. Because all conditions of Lemma 12 must hold, we can relabel w to b.

To see that we can now relabel y to a, we observe that since $G'(f', c)$ was connected, z has a neighbour with label c. This implies that y is not essential for c, as needed to satisfy all conditions of Lemma 12. ◀

Finally, in Lemma 30, we show that for G' the graph formed by adding an edge xy to a 3-connected graph $G \in \text{host}(K_3)$, G' is also in $\text{host}(K_3)$. We achieve the result by showing that we can handle situations in which xy plays a role not played by any other edge, either as an essential edge or as a bridge within a branch set. The proof relies on the following results:

▶ **Lemma 28 (*).** *Given a 2-connected graph G and a K_3-model f of G, any branch set containing at least two vertices has no crucial vertex.*

▶ **Lemma 29.** *Given a 3-connected graph G and a K_3-model f of G, suppose that $x \in G(f, a)$, $y \in G(f, b)$, and xy is an essential edge. Then we can reconfigure f to a K_3-model g such that $g(v) \neq f(v)$ for each $v \in G(f, c)$, $c \notin \{a, b\}$, $g(u) = f(u)$ for all other vertices u, and xy is not an essential edge in g.*

Proof. We consider two cases, depending on whether or not $G(f, c)$ is 2-connected.

Case 1: $G(f, c)$ is not 2-connected.

By Lemma 9, each leaf block L in $G(f, c)$ has at least two interior vertices that are endpoints of connecting edges. Due to 3-connectivity, we can further show that each leaf block in $G(f, c)$ has edges to both $G(f, a)$ and $G(f, b)$, as otherwise the joining vertex of the leaf block and either x or y would form a cut set of size two separating internal vertices of the leaf block at one of the branch sets.

The fact that all other leaf blocks connect to both $G(f, a)$ and $G(f, b)$ ensure that f does not hit a leaf-crucial model on relabeling L. We can then use Lemma 19 to relabel all interior vertices of L. Due to the connectivity of L and the fact that it contained neighbours in both $G(f, a)$ and $G(f, b)$, it follows that xy is not an essential edge in the resulting model g.

Case 2: $G(f, c)$ is 2-connected.

We first use 3-connectivity to show that $|G(f, c)| > 1$ and that $G(f, c)$ contains vertices u and v such that u has a neighbour in $G(f, a)$ and v has a neighbour in $G(f, b)$. If $|G(f, c)| = 1$, then the vertex in $G(f, c)$ and either x or y form a cut set of size two separating the branch sets $G(f, a)$ and $G(f, b)$. Similarly, if $G(f, c)$ contained only a single endpoint of a connecting edge, then the endpoint and either x or y would also form a cut set of size two.

We can choose u and v such that P is a (u, v)-path in $G(f, c)$ such that no vertex in P other than u or v has a neighbour in $G(f, a)$ or $G(f, b)$. We let $u = v_1, \ldots, v_t = v$ be the vertices of P with edges $v_i v_{i+1}$, $i \in \{1, \ldots, t\}$. Since G is 3-connected and xy is an essential edge, $G(f, c)$ must contain vertices $w \notin \{u, v\}$ and $z \notin \{u, v\}$ such that w has an edge to $G(f, a)$ and z has an edge to $G(f, b)$.

We will attempt to relabel all of P to label a. As $G(f, c)$ is 2-connected, we can relabel v_1 to a. If the resulting branch set is 2-connected, then we attempt relabel v_2,v_3, $\ldots$, v_t in that order until we relabel the entire path. If this succeeds, then we are done. Otherwise, at some step relabeling along the path we obtain a K_3 model g such that $G(g, c)$ is not 2-connected. Now we apply Case 1 to g to complete the claim. ◀

▶ **Lemma 30.** *Suppose G is a 3-connected graph such that $G \in host(K_3)$ and G' is formed from G adding an edge xy. Then $G' \in host(K_3)$.*

Proof. We first observe that any K_3-model of G is also K_3-model of G'. Consequently, to show that we can reconfigure between any K_3-models of G', it suffices to show that we can reconfigure between any K_3-model of G' and a K_3-model of G, as the fact that $G \in \mathrm{host}(K_3)$ ensures that we can reconfigure between any two K_3-models of G.

There are only two cases in which a K_3-model f of G' is not a K_3-model of G, namely cases in which the role xy plays in the K_3-model is not played by any other edge. In both cases we can assume that $G' \neq G$, and consequently that $|V(G')| > 3$.

Case 1: xy is the essential edge connecting $G'(f, f(x))$ and $G'(f, f(y))$

Without loss of generality, we assume $f(x) = a$ and $f(y) = b$. By Lemma 29, we can reconfigure f to a K_3-model f' by relabeling only vertices in $G(f, c)$ such that xy is not an essential edge in f', which means f' is also a K_3-model of G.

Case 2: $f(x) = f(y)$ and xy is a bridge in $G'(f, f(x))$

We show that we can reconfigure to a model in which x and y have different labels so that xy is a connecting edge. Depending on whether xy is then an essential edge, we have either completed the reconfiguration or we have reduced the situation to Case 1.

Without loss of generality, we assume that $f(x) = f(y) = a$, and observe that the removal of xy separates $G'(f, a)$ into two components C_1 (containing x) and C_2 (containing y), each of which contains at least one leaf block.

By Lemma 9, each of the leaf blocks has two vertices with neighbours in $G'(f, b)$ or $G'(f, c)$. We will show that we can reconfigure to a K_3-model in which either C_1 or C_2 has no vertex with label a, so that xy is no longer a bridge.

Case 2a: One component has edges to both $G'(f, b)$ and $G'(f, c)$.

Without loss of generality, let C_1 have connecting edges to both $G'(f, b)$ and $G'(f, c)$. Then f does not hit a leaf-b-crucial model or a leaf-c-crucial model on relabeling C_2 because there are necessary connecting edges from C_1. Also, f does not hit a leaf-crucial model on relabeling C_2 because there are at least two vertices labeled a in each model in the reconfiguration sequence, and so by Lemma 28, there never exists a crucial vertex labeled a. Now by Lemma 20, since x is a cut vertex of $G'(f, a)$, we can relabel all the vertices of C_2, which ensures xy is a connecting edge.

Case 2b: One component has two edges to $G'(f, b)$ and one component has two edges to $G'(f, c)$.

Without loss of generality, let C_1 have connecting edges to $G'(f, b)$ and C_2 have connecting edges to $G'(f, c)$. Then f does not hit a leaf-c-crucial model on relabeling C_2 because C_1 has edges to $G'(f, b)$. Also, it follows from Lemma 28 that f does not hit a leaf-crucial model on relabeling C_2 because there are at least two vertices labeled a in each model in the reconfiguration sequence. Hence, by Lemma 20, since x is a cut vertex of $G'(f, a)$, we can relabel all the vertices of C_2, which ensures xy is a connecting edge. ◀

6 Characterizing host(K_4)

In order to use Ding and Qin's characterization in Theorem 4, we show that $C_6^2 \in \text{host}(K_4)$ (Lemma 32) and $K_5 \in \text{host}(K_4)$ (a special case of Lemma 33) and present analogues of Lemmas 27 and 30 (Lemmas 37 and 38), showing that host(K_4) is closed under the splitting of vertices or adding of edges for 4-connected graphs. The four results are sufficient to prove the following theorem:

▶ **Theorem 31 (*).** *Every 4-connected graph is in host(K_4), provided it is not in $\mathcal{L}$, where $\mathcal{L} = \{H : H$ is the line graph of an internally 4-connected cubic graph$\}$.*

The results establishing the base cases of the characterization are relatively straightforward. Lemma 32 makes use of various properties of the structure of C_6^2, most notably the fact that for each vertex v, there is a unique vertex $s(v)$ such that there is no edge between v and $s(v)$. As an immediate consequence, any four vertices form a 4-cycle, which permits the use of Theorem 21 for reconfiguration of part of the graph. Lemma 33, a generalization of $K_5 \in \text{host}(K_4)$, follows easily from the high connectivity of cliques.

▶ **Lemma 32 (*).** *C_6^2 is in host(K_4).*

▶ **Lemma 33 (*).** *For any $m > \ell$, $K_m \in$ host(K_ℓ).*

As essential edges can result from either the splitting of vertices or the adding of edges, Lemma 36 plays a crucial role in the proofs of both Lemma 37 and Lemma 38. The proof of Lemma 36 makes extensive use of Lemmas 11, 34, and 35 in covering all possible cases of weak connections among branch sets, where branch sets of size one are handled separately.

▶ **Lemma 34 (*).** *Given a 4-connected graph G and a K_4-model f of G such that for labels a, b, c, d there exist weak connections between branch sets with labels a and b, b and c, c and d, and d and a, then it is not possible to designate lynchpins such that the branch set with label a contains at loeast one vertex x that is not a lynchpin and the branch set with label d contains at least one vertex y that is not a lynchpin.*

▶ **Lemma 35.** *Given a 3-connected graph G and an K_4-model f of G, suppose xy is an essential edge and there exists an essential edge e from $G(f, f(y))$ to $G(f, f(z))$, $z \neq x$, such that y is not the endpoint of e. Then it is possible to reconfigure f to a model g in which $g(x) = g(y) = f(x)$, and for all $a \neq f(y)$, for $v \in G(f, a)$, $g(v) = f(v)$.*

Proof. Without loss of generality, we let $f(x) = a$, $f(y) = b$, and $f(z) = c$, so that there is an essential edge ess(b, c). We consider two cases, depending on whether or not y is a cut vertex.

When y is not a cut vertex, we verify that all conditions of Lemma 12 hold for relabeling y to a: condition 1 follows as $G(f, b)$ contains at least y and an endpoint v of ess(b, c), condition 2 follows by assumption, condition 3 follows from the existence of xy, and condition 4 follows from the observation that if y is an a-crucial vertex, then it must be essential for d, which implies $\{y, z\}$ is a 2-cut in G, contradicting the fact that G is 3-connected.

If instead y is a cut vertex, by removing y we can break $T(G, f, b)$ into components such that one of the components C contains the endpoint of $\text{ess}(b, c)$. By Lemma 9, each leaf block of C must contain at least two vertices with neighbours in other branch sets, and hence C must contain an edge with a neighbour in $G(f, d)$. As C has all necessary connecting edges, we can apply Lemma 20 to siphon away the vertices in every other component $C' \neq C$. Consequently, y will no longer be a cut vertex, we can then relabel y to a, as needed. ◀

▶ **Lemma 36 (*).** *Suppose G is a 4-connected graph such that $G \in host(K_4)$, $|V(G)| \geq 5$, and xy is an essential edge under the K_4-model f. Then it is possible to reconfigure G to a K_4-model in which x and y have the same label.*

Lemma 37 follows the structure of the proof of Lemma 27, relying on Lemma 25 for the reconfiguring between K_4-models in which x and y have the same label and on Lemma 36 for the case in which xy is an essential edge. The two possible cases for Lemma 38 are xy being an essential edge (handled by Lemma 36) and xy being a bridge in a branch set.

▶ **Lemma 37 (*).** *Suppose G is a 4-connected graph such that $G \in host(K_4)$ and G' is a 4-connected graph formed from G by splitting a vertex v into vertices x and y. Then G' is in $host(K_4)$.*

▶ **Lemma 38 (*).** *Suppose G is a 4-connected graph such that $G \in host(K_4)$ and G' is formed from G adding an edge xy. Then $G' \in host(K_4)$.*

7 Conclusions and open questions

We have developed a toolkit for the reconfiguration of minors, and specific results for H-models of small cliques H. Our results imply an alternate definition of 2-connectivity, whereby a graph is 2-connected if and only if it is in $\text{host}(K_2)$. Furthermore, we have shown that every 3-connected graph is in $\text{host}(K_3)$ and that every 4-connected graph is in $\text{host}(K_4)$, provided that it is not in $\mathcal{L}$, where $\mathcal{L} = \{H : H \text{ is the line graph of an internally 4-connected cubic graph}\}$.

It remains to be shown whether similar results can be obtained for larger cliques, or for other graphs H. As our results rely on characterizations of k-connected graphs, further work is likely to depend on further progress on such results.

As there are alternate ways of defining adjacency relations, further work is needed to determine which definitions are equivalent and for those that are not, what results can be obtained. In our work, we can view each label as a token; based on this viewpoint, the adjacency relation we have considered can be viewed as *Token relabeling (TR)*, changing the label of one vertex in G. Two other possibilities worthy of consideration are *Token sliding (TS)*, swapping the labels of two adjacent vertices in G, and *Token jumping (TJ)*, swapping the labels of any two vertices in G. Both TS [6] and TJ [7] are well-studied for other types of reconfiguration problems, many of which have unlabeled or distinctly labeled tokens. The use of TS instead of TR is instrumental in handling degree-one vertices in G, which otherwise can rarely be relabeled.

Moreover, it is worth considering an alternate formulation in which solutions are considered to be adjacent if one can be formed from another by reassigning labels to vertices according to some permutation on the labels.

Future directions for research include considering other ways of assessing the reconfiguration graph, such as determining its diameter or, in cases in which the reconfiguration graph is connected, to form algorithms that determine whether there is a path between an input pair of solutions. It remains open how to characterize isolated vertices in the reconfiguration graph, known as *frozen configurations* [1].

Throughout the paper, we required every vertex of G to be a member of a branch set in an H-model. If instead we considered a subgraph of G, a solution might entail the labeling of a subset of the vertices of G. We observe that when the number of labels is equal to the number of vertices in H, the problem is reduced subgraph isomorphism [5]. Alternative mappings can be considered as well, such as topological embedding of one graph in another.

References

1 Richard C. Brewster, Jae-Baek Lee, Benjamin Moore, Jonathan A. Noel, and Mark Siggers. Graph homomorphism reconfiguration and frozen h-colourings. *CoRR*, arXiv:1712.00200, 2017.

2 Erik D. Demaine and MohammadTaghi Hajiaghayi. Graphs excluding a fixed minor have grids as large as treewidth, with combinatorial and algorithmic applications through bidimensionality. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 682–689. Society for Industrial and Applied Mathematics, 2005.

3 Reinhard Diestel. *Graph theory.* Springer-Verlag, Electronic Edition, 2005.

4 Guoli Ding and Chengfu Qin. Generating 4-connected graphs, 2015. URL: `https://www.math.lsu.edu/~ding/chain4.pdf`.

5 Tesshu Hanaka, Takehiro Ito, Haruka Mizuta, Benjamin Moore, Naomi Nishimura, Vijay Subramanya, Akira Suzuki, and Krishna Vaidyanathan. Reconfiguring spanning and induced subgraphs. In *Proceedings of the 24^{th} International Computing and Combinatorics Conference*, 2018.

6 Takehiro Ito, Erik D. Demaine, Nicholas J. A. Harvey, Christos H. Papadimitriou, Martha Sideri, Ryuhei Uehara, and Yushi Uno. On the complexity of reconfiguration problems. *Theoretical Computer Science*, 412(12-14):1054–1065, 2011.

7 Marcin Kamiński, Paul Medvedev, and Martin Milanič. Complexity of independent set reconfigurability problems. *Theoretical Computer Science*, 439:9–15, 2012.

8 Anna Lubiw, Zuzana Masárová, and Uli Wagner. Proof of the orbit conjecture for flipping edge-labelled triangulations. In *Proceedings of the 33^{rd} International Symposium on Computational Geometry*, 2017.

9 Nicola Martinov. Uncontractable 4-connected graphs. *Journal of Graph Theory*, 6(3):343–344, 1982.

10 Moritz Mühlenthaler. Degree-contrained subgraph reconfiguration is in P. In *40th International Symposium on Mathematical Foundations of Computer Science*, pages 505–516, 2015.

11 N. Nishimura. Introduction to reconfiguration. *Algorithms*, 11(4):52, 2018.

12 Neil Robertson and P.D. Seymour. Graph minors. XX. Wagner's conjecture. *Journal of Combinatorial Theory, Series B*, 92(2):325–357, 2004. Special Issue Dedicated to Professor W.T. Tutte.

13 W.T. Tutte. A theory of 3-connected graphs. *Indagationes Mathematicae (Proceedings)*, 64:441–455, 1961.

14 Jan van den Heuvel. The complexity of change. *Surveys in Combinatorics 2013*, 409:127–160, 2013.