Temporal Graph Reconfiguration for Always-Connected Graphs

Paul Sievers
Hasso Plattner Institute, University of Potsdam, Germany

George Skretas
Hasso Plattner Institute, University of Potsdam, Germany

Georg Tennigkeit
Hasso Plattner Institute, University of Potsdam, Germany

Abstract

Network redesign problems ask to modify the edges of a given graph to satisfy some properties. In temporal graphs, where edges are only active at certain times, we are sometimes only allowed to modify when the edges are going to be active. In practice, we might not even be able to perform all of the necessary modifications at once; changes must be applied step-by-step while the network is still in operation, meaning that the network must continue to satisfy some properties. To initiate a study in this area, we introduce the temporal graph reconfiguration problem.

As a starting point, we consider the Layered Connectivity Reconfiguration problem in which every snapshot of the temporal graph must remain connected throughout the reconfiguration. We provide insights into how bridges can be reconfigured into non-bridges based on their reachability partitions, which lets us identify any edge as either changeable or unchangeable. From this we construct a polynomial-time algorithm that gives a valid reconfiguration sequence of length at most $2M^2$ (where M is the number of temporal edges), or determines that reconfiguration is not possible. We also show that minimizing the length of the reconfiguration sequence is NP-hard via a reduction from vertex cover.

2012 ACM Subject Classification Replace ccsdesc macro with valid one

Keywords and phrases temporal graphs, reconfiguration, layered networks, network redesign

Funding *Georg Tennigkeit*: HPI Research School on Foundations of AI (FAI)

1 Introduction

The network redesign problem is an umbrella term for one of the fundamental problems of network science. An informal description of this problem is the following: Given a graph G, change the edges of the graph such that the new graph G' satisfies some properties. Several variants of the problem have been studied in the temporal graphs literature [7, 9, 6, 14]. A temporal graph is a static graph where edges are equipped with labels. These labels indicate the time at which the edge is active.

The general framework of network redesign problems in temporal graphs is that we are given an input temporal graph $\mathcal{G}$, and we want to find the minimum number of temporal edge changes that turn it into a temporal graph $\mathcal{G}'$ that satisfies a given property. This problem formulation assumes that in the application domain, we can change the connections of the network all together to instantly arrive at the network $\mathcal{G}'$. However with many applications, such as transportation networks or computer networks, the connections of the network can only be changed gradually in small batches, or maybe even one by one. Therefore, we would want to find an order of changing these connections such that the network remains operable after each change. Motivated by this, we introduce the temporal graph reconfiguration problem. In this problem, we are given a starting temporal graph $\mathcal{G}_1$ and a target temporal graph $\mathcal{G}_2$ with the same vertex set V, along with a property P. We are tasked with finding a sequence of changes to the edges of the graph such that $\mathcal{G}_1$ transforms into $\mathcal{G}_2$ while the graph maintains the property P after every single change. This problem formulation is very similar to reconfiguration problems often considered in distributed problems in dynamic graphs [10, 15], as well as reconfiguration problems in static graphs [4, 3]. Similarities and differences are discussed in the related work section.

1.1 Our Contribution

To begin our study of temporal graph reconfiguration problems, we will focus on a simple variant where we can only change the label of one edge during each reconfiguration step. The property we want to maintain after every label change is that every snapshot of the temporal graph remains connected.

To formalize this problem, we define a *temporal graph* with lifetime T as $\mathcal{G} = (V, \mathcal{E})$ where V is the set of vertices and $\mathcal{E} \subseteq \binom{V}{2} \times [T]$ is the set of temporal edges. We will refer to a temporal edge simply as an edge when it is clear from context. A temporal graph can be viewed as a collection of *snapshot*s, with the t-th snapshot $\mathcal{G}(t) = (V, \{(e, t) \in \mathcal{E}\})$, for a $t \in [T]$, representing the static graph containing only edges active at time t. We call a temporal graph $\mathcal{G}$ *always-connected* if every snapshot of $\mathcal{G}$ is connected.

With this we define the Layered Connectivity Reconfiguration problem: Given two always-connected temporal graphs $\mathcal{G}_1$ and $\mathcal{G}_2$, determine if it is possible to transform $\mathcal{G}_1$ into $\mathcal{G}_2$ by changing the time a single edge is active in each step, such that every intermediate temporal graph is always-connected. As an example, Figure 1 shows such a transformation between two temporal graphs.

As the name suggests, this specific problem is not inherently temporal in that the temporal order of edges is not relevant but it serves as a good starting point for investigating temporal graph reconfiguration problems. However, this problem can also be applied to scenarios where the edges of a graph are partitioned into different layers and each layer must be connected. As a token example, consider a pipe network transporting various liquids across long distances. Different liquids naturally should not be mixed, so each pipe transports only one type of liquid. This partitions the pipes into layers, with one layer for each type of liquid.

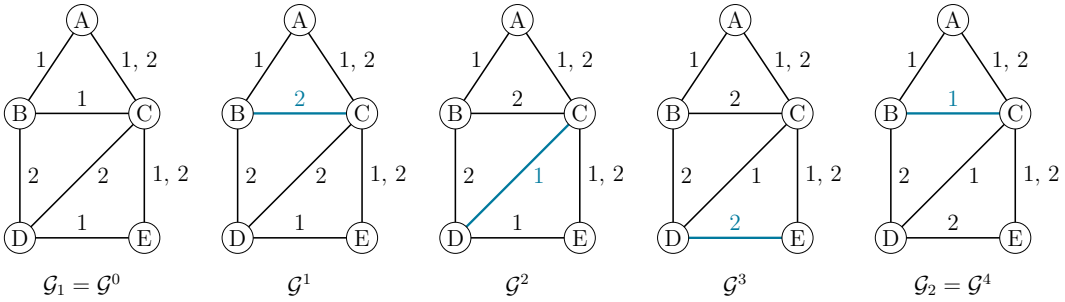

Figure 1 Example of a valid reconfiguration sequence $(\mathcal{G}^0, \cdots, \mathcal{G}^4)$ between $\mathcal{G}_1$ and $\mathcal{G}_2$. Time labels are shown on each edge, and edges marked in blue differ from the previous graph in the sequence.

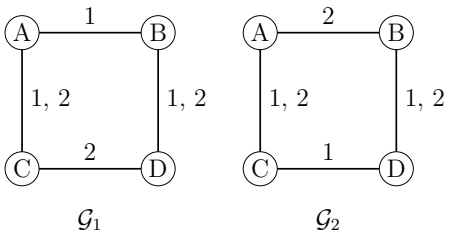

Figure 2 Example with no valid reconfiguration sequence between $\mathcal{G}_1$ and $\mathcal{G}_2$, as changing any label in $\mathcal{G}_1$ disconnects a snapshot.

We can change the type of liquid transported by a pipe by closing and draining the pipe, adjusting the connections at the endpoints, and reopening the pipe. During this step, we must ensure that each layer is still connected.

The first natural question to ask, given such a problem instance, is whether reconfiguration is even possible. See Figure 2 for an example where reconfiguration is not possible. Towards answering this question, we first define reachability partitions of bridge edges (Section 3). Bridges are significant because changing their label disconnects their snapshot, however, their reachability partitions provide some crucial insights for how a bridge edge can be turned into a non-bridge edge by reconfiguring other edges.

We use this to construct an algorithm that can decide the problem in polynomial-time (Section 4). First, we use dynamic programming to determine for any other edge if (and how) it can be turned into a non-bridge. If we identify an edge that is unchangeable no matter what reconfiguration happens before, and it has different labels in $\mathcal{G}_1$ and $\mathcal{G}_2$, reconfiguration is not possible. Otherwise, we repeatedly identify an edge that differs in $\mathcal{G}_1$ and $\mathcal{G}_2$ and apply the steps given by the DP in order to change the label of the differing edge. The trick here is to apply changes to $\mathcal{G}_1$ and $\mathcal{G}_2$ simultaneously so that we never increase the number of differences between them. This gives a reconfiguration sequence from both $\mathcal{G}_1$ and $\mathcal{G}_2$ to the same canonical labeling; Because every reconfiguration step is reversible, reversing the sequence from $\mathcal{G}_2$ to this canonical labeling yields a reconfiguration from $\mathcal{G}_1$ to $\mathcal{G}_2$.

The algorithm finds a reconfiguration sequence of length at most $2M^2$ where M is the number of temporal edges. However, this sequence may not be the shortest possible sequence. We ask then whether one can also find a *shortest* sequence in polynomial-time. We answer this negatively even for graphs that have lifetime $T = 2$ by giving a reduction from Vertex Cover. (Section 5). The reduction represents vertices as cycles on a snapshot; choosing a vertex corresponds to breaking its cycle in order to close a different (more useful) cycle. In

order to find a shortest reconfiguration sequence, one has to find a minimal number of cycles to break and later reconnect.

1.2 Related Work

To the best of our knowledge, the Layered Connectivity Reconfiguration problem is the first graph reconfiguration problem to be considered within the context of temporal graphs. The research most closely related to ours investigates the reconfiguration of arborescences consisting of non-decreasing edge-labels in temporal graphs [13, 8]. The key distinction between [13, 8] and our work is that our reconfiguration changes the temporal graph itself whereas in [13, 8], the authors modify a solution set on the graph.

In the static graph literature, numerous papers consider a solution set on a graph G and the goal is to reconfigure it into a different solution set in the same graph G. Bousquet et al. [4] examined spanning tree reconfiguration with different constraints on degree and diameter, while Bousquet et al. [3] considered constraints on the number of leaves. The complexity of different types of induced tree reconfiguration problems was analyzed by Wasa, Yamanaka and Arimura [21]. Hanaka et al. [11] described several reconfiguration problems in graphs under different graph structure properties, including paths, cycles, trees, and cliques, and with different operations. A polynomial-time algorithm for the matching reconfiguration problem was given by Ito et al. [12]. Here, the task is to transform two matchings M and M' into each other by adding or deleting an edge in each step, such that there is no intermediate matching of size smaller than $\min(|M|, |M'|) - 1$. A generalization of this problem, allowing for more changes at each step, was analyzed by Solomon and Solomon[19]. Bonamy et al.[2] studied the perfect matching reconfiguration problem with a focus on its complexity in different graph classes and showed it to be PSPACE-complete.

Moving on to static graph reconfiguration problems, motivated by detecting and redistricting gerrymandering, Akitaya et al.[1] studied the graph reconfiguration of connected graph partitions through node swapping between partitions. Deciding if there exists a reconfiguration sequence when swapping a single node at a time was shown to be PSPACE-complete. A plethora of papers also studies graph reconfiguration problems in distributed systems, particularly in the context of programmable matter. The amoebot model considers embedded modules on a triangular grid that form a shape [5]. The main problem investigated within the amoebot model is to develop algorithms that can transform a given starting shape A into a given starting shape B while maintaining connectivity [15]. Other programmable matter models have also studied this question [18, 16]. Finally, graph reconfiguration problems have also been studied in the overlay networks literature [10, 17].

2 Preliminaries

For $n \in \mathbb{N}$ we denote $[n] = \{1, \cdots, n\}$. For a set X, we write $\binom{X}{2}$ for the set of unordered pairs of elements in X.

A temporal graph with lifetime T is defined as $\mathcal{G} = (V, \mathcal{E})$ where V is the set of vertices and $\mathcal{E} \subseteq \binom{V}{2} \times [T]$ is the set of temporal edges. The t-th snapshot of $\mathcal{G}$ is defined as $\mathcal{G}(t) = (V, \{(e, t) \in \mathcal{E}\})$, for a $t \in [T]$, representing the static graph containing only edges active at time t. We call a temporal graph $\mathcal{G}$ always-connected if every snapshot of $\mathcal{G}$ is connected. For a temporal graph $\mathcal{G} = (V, \mathcal{E})$ and an edge $(e, t) \in \mathcal{E}$ we denote by $\mathcal{G} - (e, t)$ the temporal graph $\mathcal{G}' = (V, \mathcal{E} \setminus \{(e, t)\})$. We write $n = |V|$ for the number of vertices and $M = |\mathcal{E}|$ for the number of temporal edges.

We say a snapshot $\mathcal{G}(t)$ is *connected* if there exists a path in $\mathcal{G}(t)$ between every pair of nodes in V. A *connected component* of $\mathcal{G}(t)$ is a subset of vertices $C \subseteq V$ such that for all pairs of nodes $a, b \in C$ there exists a path between a and b in $\mathcal{G}(t)$ and for all nodes $a \in C$ and $b \notin C$ there does not exist a path between a and b in $\mathcal{G}(t)$. A temporal edge $(e,t) \in \mathcal{E}$ is called a *bridge* if its removal increases the number of connected components of $\mathcal{G}(t)$ by one. In static graphs (and thus also in snapshots of a temporal graph), all bridges can be computed in $\mathcal{O}(|V| + |E|)$ using the algorithm described in [20].

An *edge relabeling* operation on a temporal graph takes an edge $(e,t) \in \mathcal{E}$ and changes the time it is active, replacing it with (e,t') for some $t' \in [T] \setminus \{t\}$. We refer to *relabeling* an edge $(e,t) \to (e,t')$ as a shorthand for this operation. A *reconfiguration sequence* between temporal graphs $\mathcal{G}_1 = (V, \mathcal{E}_1)$ and $\mathcal{G}_2 = (V, \mathcal{E}_2)$ is a sequence of temporal graphs $(\mathcal{G}_1 = \mathcal{G}^0, \mathcal{G}^1, \cdots, \mathcal{G}^k = \mathcal{G}_2)$ with $\mathcal{G}^j = (V, \mathcal{E}^j)$ for $0 \leq j \leq k$, such that for $0 \leq i < k$ the graph $\mathcal{G}^{i+1}$ can be obtained from $\mathcal{G}^i$ using a single edge relabeling operation. More formally, for all $0 \leq i < k$, there exists $t, t' \in [T]$ such that for all $\tilde{t} \in [T] \setminus \{t, t'\} : \mathcal{E}^i(\tilde{t}) = \mathcal{E}^{i+1}(\tilde{t})$ and it holds that $|\mathcal{E}^i(t) \setminus \mathcal{E}^{i+1}(t)| = |\mathcal{E}^{i+1}(t') \setminus \mathcal{E}^i(t')| = 1$. An *intermediate graph* in such a reconfiguration sequence is any temporal graph G^i with $0 \leq i \leq k$. The length of such a reconfiguration sequence is k. We call a reconfiguration sequence valid if all intermediate graphs are always-connected. Similarly, an edge relabeling operation on an always-connected temporal graph $\mathcal{G}$ is valid if the resulting graph is always-connected.

As the edge relabeling operation does not change the number of temporal edges between two vertices, we assume that for all pairs $e \in \binom{V}{2}$ it holds that

$$|\{(e,t) \in \mathcal{E}_1 \mid t \in [T]\}| = |\{(e,t) \in \mathcal{E}_2 \mid t \in [T]\}|$$

as otherwise reconfiguration between $\mathcal{G}_1$ and $\mathcal{G}_2$ would not be possible. We also assume that any given temporal graph is always-connected.

3 Reachability Partitions

In this section, we introduce important definitions and properties of a temporal graph in relation to bridges. These will serve as building blocks for our proofs in the later sections.

Since removing a bridge disconnects a snapshot, a valid edge relabeling operation can only be performed on non-bridges in order to preserve the always-connected property. To find a reconfiguration sequence between two temporal graphs $\mathcal{G}_1$ and $\mathcal{G}_2$, we need to change the edges only present in $\mathcal{G}_1$ to the ones in $\mathcal{G}_2$. Once all these different edges have been relabeled, the graphs will be equal. If one of the different edges is a non-bridge, it can be changed directly, which gives some progress in reconfiguration. However, as illustrated in Figure 3, all different edges may initially be bridges. Thus we need a systematic way to first turn these bridges into non-bridges.

To address this, we define the *reachability partition* of a bridge $(\{u,v\},t)$. For a temporal graph $\mathcal{G}$, it is the partition of nodes into the two connected components of $\mathcal{G}(t) - (\{u,v\},t)$, i.e. the nodes reachable from u and v in $\mathcal{G}(t) - (\{u,v\},t)$. See Figure 4.

▶ **Definition 1** (Reachability Partition). *Given a temporal graph $\mathcal{G} = (V, \mathcal{E})$ and a bridge $(e,t) = (\{u,v\},t) \in \mathcal{E}$, we define the components of the reachability partition as* $\mathrm{Comp}(u,e,t) := \{x \in V \mid \exists$*path between u and x in $\mathcal{G}(t) - (e,t)\}$ and analogously* $\mathrm{Comp}(v,e,t) := \{x \in V \mid \exists$*path between v and x in $\mathcal{G}(t) - (e,t)\}$.*

We refer to the partition of V into $\mathrm{Comp}(u,e,t)$ *and* $\mathrm{Comp}(v,e,t)$ *as the reachability partition of (e,t).*

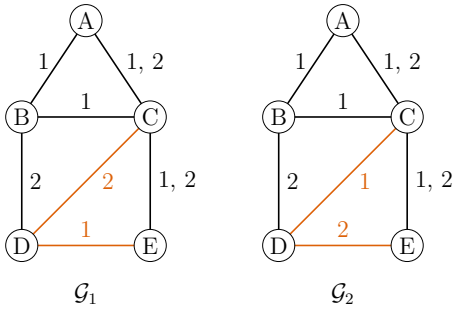

Figure 3 Example of temporal graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ in which all different edges in $\mathcal{G}_1$ (marked in orange) are bridges.

An edge $(\{u', v'\}, t')$ is a *crossing edge* in the reachability partition of a bridge $(\{u, v\}, t) = (e, t)$ if $u' \in \text{Comp}(u, e, t)$ and $v' \in \text{Comp}(v, e, t)$.

For a bridge $(\{u, v\}, t) = (e, t)$, note that in an always-connected graph, $\mathcal{G}(t)$ has exactly one connected component. By definition of a bridge, $\mathcal{G}(t) - (e, t)$ has exactly two connected components. Since every node belongs to a connected component, these sets form a partition of V and correspond to our reachability partition, as u and v are in different connected components.

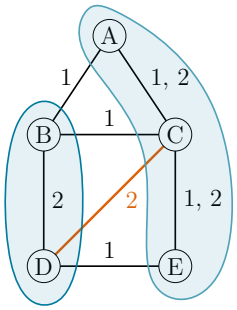

Figure 4 Example of the reachability partition of the edge $(e, 2) = (\{C, D\}, 2)$.

Using the definition of the reachability partition, we can now characterize when a bridge becomes a non-bridge after an edge relabeling operation.

▶ **Lemma 2** (Bridges and Reachability Partition). *Given a temporal graph $\mathcal{G} = (V, \mathcal{E})$ and a bridge (e, t), a valid edge relabeling operation $(e', t') \to (e', t)$ on $\mathcal{G}$ turns (e, t) into a non-bridge if and only if (e', t) is a crossing edge in the reachability partition of (e, t).*

Proof. Let $(e, t) = (\{u, v\}, t)$ and $(e', t') = (\{x, y\}, t')$ be edges in $\mathcal{G}$. Let $\mathcal{G}'$ be the graph obtained by applying the edge relabeling operation $(e', t') \to (e', t)$ on $\mathcal{G}$.

Suppose (e', t) is a crossing edge in the reachability partition of (e, t). W.l.o.g., assume $x \in \text{Comp}(u, e, t)$ and $y \in \text{Comp}(v, e, t)$. We show that (e, t) is not a bridge in $\mathcal{G}'$ by proving that $\mathcal{G}'(t) - (e, t)$ is connected.

Since $\text{Comp}(u, e, t)$ and $\text{Comp}(v, e, t)$ are connected components of $\mathcal{G}(t) - (e, t)$, any two nodes in the same connected component remain connected in $\mathcal{G}'(t) - (e, t)$, because $\mathcal{G}'(t) - (e, t)$ contains all edges of $\mathcal{G}(t) - (e, t)$ and the new edge $\{e', t'\}$.

For nodes $a \in \text{Comp}(u, e, t)$ and $b \in \text{Comp}(v, e, t)$, there exist paths a to x within $\text{Comp}(u, e, t)$ and y to b within $\text{Comp}(v, e, t)$ in $\mathcal{G}(t) - (e, t)$. In $\mathcal{G}'(t) - (e, t)$ the added edge

$(\{x, y\}, t)$ connects these paths, giving a path from a to b. Therefore $\mathcal{G}'(t) - (e, t)$ is connected and (e, t) is no longer a bridge after the edge relabeling operation.

Suppose (e', t) is not a crossing edge in the reachability partition of (e, t). W.l.o.g., assume $x, y \in \text{Comp}(u, e, t)$. Since $\text{Comp}(u, e, t)$ is already connected, there exists a path x to y in $\mathcal{G} - (e, t)$. Thus after adding $(\{x, y\}, t)$, the connected components do not change and $\mathcal{G}' - (e, t)$ is still not connected. Therefore (e, t) remains a bridge after the relabeling. ◀

Since a bridge may require multiple edge relabeling operations before it becomes a non-bridge, we need to analyze how its reachability partition changes after edge relabeling operations which leave it a bridge. This scenario is formalized in the next lemma.

▶ **Lemma 3** (Reachability Partition invariant). *Given a temporal graph $\mathcal{G} = (V, \mathcal{E})$ and a bridge (e, t), the reachability partition of $(e, t) = (\{u, v\}, t)$ does not change after any valid edge relabeling operation $(e', t_1) \to (e', t_2)$ on $\mathcal{G}$ after which (e, t) is still a bridge.*

Proof. Let (e, t) be any bridge in $\mathcal{G}$ and (e', t_1) a non-bridge in $\mathcal{G}$. Let $\mathcal{G}'$ be the graph obtained by applying the valid edge relabeling operation $(e', t_1) \to (e', t_2)$ on $\mathcal{G}$ and suppose (e, t) is still a bridge in $\mathcal{G}'$.

In $\mathcal{G}$ let $a \in \text{Comp}(u, e, t)$. We show that $a \in \text{Comp}(u, e, t)$ still holds in $\mathcal{G}'$. The argument for $a \in \text{Comp}(v, e, t)$ is analogous.

Case 1: $t_1 \neq t$ and $t_2 \neq t$
In this case $\mathcal{G}(t) = \mathcal{G}'(t)$, so clearly $a \in \text{Comp}(u, e, t)$ in $\mathcal{G}'$ holds.

Case 2: $t_1 = t$
As the only change to $\mathcal{G}(t) - (e, t)$ is the removal of an edge, there is no new path in $\mathcal{G}'(t) - (e, t)$ between v and a and $a \notin \text{Comp}(v, e, t)$ still holds. Since the components form a partition of V, it follows that $a \in \text{Comp}(u, e, t)$ holds in $\mathcal{G}'$.

Case 3: $t_2 = t$
As the only change to $\mathcal{G}(t) - (e, t)$ is the addition of an edge, there is still a path in $\mathcal{G}'(t) - (e, t)$ between u and a. Therefore $a \in \text{Comp}(u, e, t)$ holds in $\mathcal{G}'$.

As the components of the reachability partition of (e, t) still form a partition of V in $\mathcal{G}'$, this shows that the reachability partition does not change after the edge relabeling operation. ◀

4 Decision Problem

In this section, we give a polynomial-time algorithm for deciding if there exists a reconfiguration sequence between two temporal graphs such that every intermediate graph is always-connected. If such a valid reconfiguration sequence exists, the algorithm finds one of length at most $2M^2$ where M is the number of temporal edges in the initial temporal graph.

The key idea is to classify edges as either *changeable* or *unchangeable*. A changeable edge is one that can eventually be changed, i.e. there exists a temporal graph, reachable through a valid reconfiguration sequence, in which it becomes a non-bridge. An unchangeable edge remains a bridge in all such reachable graphs.

With this classification we can state our main result, which characterizes exactly when a valid reconfiguration sequence exists. We prove this result in the remainder of this section.

▶ **Theorem 4** (Finding a reconfiguration sequence in polynomial-time). *For temporal graphs $\mathcal{G}_1 = (V, \mathcal{E}_1)$ and $\mathcal{G}_2 = (V, \mathcal{E}_2)$, there exists a valid reconfiguration sequence between $\mathcal{G}_1$ and $\mathcal{G}_2$ if and only if all edges in $\mathcal{E}_1 \setminus \mathcal{E}_2$ are changeable. If that is the case we can find a valid reconfiguration sequence of length at most $2M^2$ in $\mathcal{O}(M^3)$.*

To prove Theorem 4, we proceed in two steps. First, we establish Lemma 8, which shows that for any changeable edge, a shortest reconfiguration sequence to turn it into a non-bridge can be computed in polynomial-time. Since unchangeable edges do not have such a sequence, this lemma allows us to identify exactly which edges are changeable.

Secondly, we prove Lemma 9. For temporal graphs $\mathcal{G}_1$ and $\mathcal{G}_2$, the lemma establishes a way to reduce the number of different edges between them whenever there exists a changeable edge in $\mathcal{G}_1$ which is not in $\mathcal{G}_2$.

By repeatedly applying this lemma, we obtain a valid reconfiguration sequence that transforms $\mathcal{G}_1$ and $\mathcal{G}_2$ into the same representation $\mathcal{G}_c$. This gives us a valid reconfiguration sequence between $\mathcal{G}_1$ and $\mathcal{G}_2$, proving Theorem 4.

Finding shortest reconfiguration sequences for every edge

As it will prove useful for computing a reconfiguration sequence, we introduce a finer classification of changeability. We classify edges by the minimum number of edge relabeling operations needed to turn them into non-bridges.

▶ **Definition 5** (k-Changeable Edge). *Given a temporal graph $\mathcal{G} = (V, \mathcal{E})$, a temporal edge $(e, t) \in \mathcal{E}$ is* k-changeable *if there exists a valid reconfiguration sequence of length k transforming $\mathcal{G}$ into a temporal graph $\mathcal{G}'$ in which (e, t) is a non-bridge, and no such sequence of length $l < k$ exists. We call an edge* changeable *if it is k-changeable for some $k \in \mathbb{N}$ and* unchangeable *if it is not k-changeable for any $k \in \mathbb{N}$.*

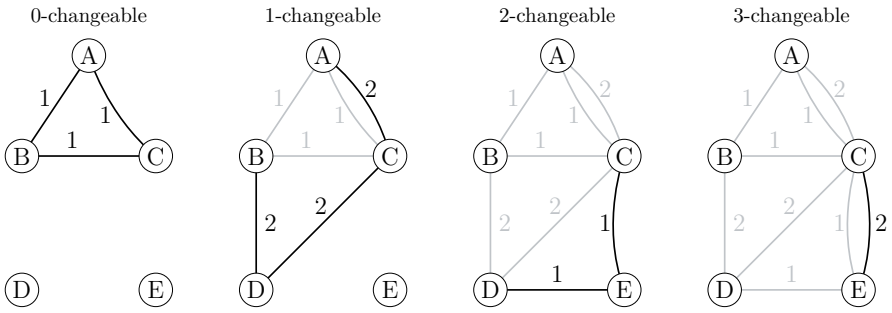

Figure 5 Illustration of k-changeable edges in a temporal graph. The 0-changeable edges are exactly the non-bridges. For $i \in \mathbb{N}$, the $(i+1)$-changeable edges are those bridges that become non-bridges by changing an i-changeable edge.

The following lemma uses the reachability partition to determine the number of operations needed for an edge to become changeable. It asserts that to identify all k-changeable edges, only $k-1$ changeable edges need to be considered, which already hints at the order of computation we will use later.

▶ **Lemma 6** (Reachability Partition and k-changeability). *Given a temporal graph $\mathcal{G} = (V, \mathcal{E})$ and a $k \in \mathbb{N}$, let (e, t) be an edge that is not l-changeable for any $l \leq k$. Then (e, t) is $k+1$-changeable if and only if there exists a k-changeable edge (e', t') that is a crossing edge in the reachability partition of (e, t).*

Proof. Let $k \in \mathbb{N}$ and let (e, t) be a bridge in $\mathcal{G}$ which is not l-changeable for any $l \leq k$.

Suppose there exists a k-changeable edge (e', t') which is a crossing edge in the reachability partition of (e, t). As (e', t') is k-changeable, there exists a valid reconfiguration sequence of length k from $\mathcal{G}$ to a temporal graph $\mathcal{G}'$ in which (e', t') is a non-bridge. Because (e, t)

is not l-changeable for any $l \leq k$, it remains a bridge in every intermediate graph of this sequence. By Lemma 3, the reachability partition of (e,t) remains unchanged throughout this sequence. Thus in $\mathcal{G}'$, (e',t') is a non-bridge and a crossing edge in the reachability partition of (e,t). By Lemma 2, the relabeling operation $(e',t') \to (e',t)$ turns (e,t) into a non-bridge. Therefore (e,t) is $k+1$-changeable.

Suppose (e,t) is $k+1$-changeable. Then there exists a valid reconfiguration sequence $(\mathcal{G} = \mathcal{G}^0, \mathcal{G}^1, \cdots, \mathcal{G}^{k+1})$ of length $k+1$ in which (e,t) becomes a non-bridge in $\mathcal{G}^{k+1}$. Since (e,t) is not l-changeable for any $l \leq k$, it remains a bridge in all intermediate graphs $\mathcal{G}^0 \cdots \mathcal{G}^k$. By Lemma 3, the reachability partition of (e,t) is the same in all these graphs. By Lemma 2, the last step in the sequence that makes (e,t) a non-bridge involves relabeling an edge $(e',t') \to (e',t)$ in $\mathcal{G}^k$, which is a crossing edge in the reachability partition of (e,t). Since (e,t) is not l-changeable for any $l \leq k$, (e',t') is not l'-changeable for any $l' < k$. As it becomes a non-bridge in $\mathcal{G}^k$ it is thus k-changeable. Therefore there exists a k-changeable edge in the reachability partition of (e,t). ◀

The following lemma provides a characterization of unchangeable edges. It shows that if there exists a $k \in \mathbb{N}$ with no k-changeable edges, there are no r-changeable edges for any $r > k$ as well. In that case, all edges which are not l-changeable for any $l < k$ are unchangeable.

▶ **Lemma 7** (Continuity of Changeable Edges). *Given a temporal graph $\mathcal{G} = (V,\mathcal{E})$ and a $k \in \mathbb{N}^+$, if $\mathcal{E}$ can be partitioned into sets U and C such that every edge in U is not l-changeable for any $l \leq k$, and every edge in C is l'-changeable for some $l' \leq k-1$, then all edges in U are unchangeable.*

Proof. Let $k \in \mathbb{N}^+$ such that $\mathcal{E}$ can be partitioned into sets U and C as in the statement of the lemma. We show by induction that all edges in U are not r-changeable for any $r \in \mathbb{N}$. The statement holds for all $r \leq k$.

Let $r \in \mathbb{N}$ with $r \geq k$ and suppose all edges in U are not l-changeable for any $l \leq r$. Let (e,t) be any edge in U. As $r \geq k$ no edge in C is r-changeable. Thus by Lemma 6, (e,t) is not $r+1$-changeable. This shows by induction that every edge in U is not r-changeable for any $r \in \mathbb{N}$ and therefore unchangeable. ◀

With the lemmas introduced previously, we can now develop an algorithm to compute, for any edge, the shortest reconfiguration sequence to turn it into a non-bridge. Before describing the algorithm, we introduce a convenient structure to find crossing edges in the reachability partitions of bridges.

For a temporal graph $\mathcal{G} = (V,\mathcal{E})$ and an edge $(e,t) \in \mathcal{E}$, we define $\mathrm{Cross}(e,t)$ as the set of bridges $(e',t') \in \mathcal{E}$, such that (e,t) is a crossing edge in the reachability partition of (e',t'). The following algorithm computes Cross.

For every bridge $(e',t') = (\{u',v'\},t) \in \mathcal{E}$, first do a DFS in $\mathcal{G}(t') - (e',t')$ starting from u' and from v' to determine $\mathrm{Comp}(u',e',t')$ and $\mathrm{Comp}(v',e',t')$. Then, for each edge $(e,t) \in \mathcal{E}$, add (e',t') to $\mathrm{Cross}(e,t)$ if (e,t) is a crossing edge in the reachability partition of (e',t').

Correctness follows directly from the definition of a crossing edge in the reachability partition. Each DFS identifies all nodes reachable from the endpoints of a bridge, giving its reachability partition. As the algorithm iterates over all edges for every bridge, Cross is computed correctly. Since there are at most M bridges, and for each bridge the DFS and the iteration over all edges both take linear time, the algorithm runs in $\mathcal{O}(M^2)$, and therefore in polynomial-time.

Now, to compute a shortest reconfiguration sequence to change any temporal edge, we define Change(k) for $0 \leq k \leq m$. The set Change(0) contains all non-bridges. For $0 < k \leq M$, the set Change(k) contains all k-changeable edges, each with an index pointing to the edge in Change($k-1$) that must be changed before it. The following algorithm computes Change (refer to Figure 5 for an example execution).

Using Tarjan's Algorithm to find all bridges, first compute Change(0). Then, starting with $k = 0$, iteratively increase k until Change(k) is empty. In each iteration, go through all edges (e, t) in Change(k), and for each, iterate through all edges (e', t') in Cross(e, t). If (e', t') is not in Change(l) for any $l \leq k$, we append it to Change($k+1$), together with a back reference to (e, t). Once Change(k) is empty, any edge not in Change(l) for any $l \leq k$ is unchangeable.

▶ **Lemma 8** (Change is computable in Polynomial-Time). *For a temporal graph $\mathcal{G} = (V, \mathcal{E})$ the above algorithm correctly computes* Change(k) *for all $0 \leq k \leq M$ in $\mathcal{O}(M^2)$.*

Proof. First, we show by induction that Change(k) contains all k-changeable edges for any $k \in \mathbb{N}$. As Tarjan's Algorithm correctly identifies all non-bridges, this holds for Change(0).

Suppose for some $i \in \mathbb{N}$ that Change(j) contains all j-changeable edges for all $j \leq i$. As we iterate through all edges (e', t') in Change(i) and append all edges (e, t) in Cross(e', t') which are not in Change(j) for any $j \leq i$ to Change($i+1$), it follows from Lemma 6 that Change($i+1$) will contain exactly the $(i+1)$-changeable edges.

When the algorithm terminates at some i, Lemma 7 implies that all edges for which no sequence to change them was found, are unchangeable. Since Change is correct, we can reconstruct a shortest reconfiguration sequence to make any edge into a non-bridge using the back references stored in it.

Regarding runtime, Cross can be computed in $\mathcal{O}(M^2)$, and for every edge (e, t), Cross(e, t) contains at most M entries. Computing Change(0) takes $\mathcal{O}(M)$. Since each edge (e, t) is added to some Change(k) at most once, we iterate through each entry of Cross(e, t) at most once. Therefore the total runtime is in $\mathcal{O}(M^2)$. ◀

In Lemma 8 we showed that, for each edge, a shortest reconfiguration sequence to turn it into a non-bridge can be computed in polynomial-time, if one exists. This allows us to identify which edges are changeable and which are unchangeable, completing the first part for proving Theorem 4.

Finding a reconfiguration sequence to decrease difference

Suppose we want to find a valid reconfiguration sequence from temporal graph $\mathcal{G}_1 = (V, \mathcal{E}_1)$ to $\mathcal{G}_2 = (V, \mathcal{E}_2)$. We define the *difference* between them, denoted by $\delta(\mathcal{G}_1, \mathcal{G}_2)$, as the number of edges in $\mathcal{G}_1$ that are not in $\mathcal{G}_2$. Recall our assumption that $\mathcal{G}_1$ and $\mathcal{G}_2$ have the same number of edges between each pair of vertices. Under this assumption, the graphs are equal if their difference is 0. Given the existence of a changeable edge in $\mathcal{G}_1$ that is not in $\mathcal{G}_2$, the following algorithm computes reconfiguration sequences $(\mathcal{G}_1 = \mathcal{G}_1^0, \mathcal{G}_1^1, \cdots \mathcal{G}_1^{k+1})$ and $(\mathcal{G}_2 = \mathcal{G}_2^0, \mathcal{G}_2^1, \cdots \mathcal{G}_2^k)$ for some $k \in \mathbb{N}$ such that $\delta(\mathcal{G}_1^{k+1}, \mathcal{G}_2^k) < \delta(\mathcal{G}_1, \mathcal{G}_2)$.

First, compute Change(k) for all $0 \leq k \leq M$. Let (e, t) be the edge in $\mathcal{E}_1 \setminus \mathcal{E}_2$ with the shortest valid reconfiguration sequence $(\mathcal{G}_1 = \mathcal{G}_1^0, \mathcal{G}_1^1, \cdots \mathcal{G}_1^k)$ turning it into a non-bridge. If (e, t) is already a non-bridge, then $k = 0$ and the sequence consists only of $\mathcal{G}_1$. Otherwise, apply the same edge relabeling operations from $(\mathcal{G}_1, \cdots, \mathcal{G}_1^k)$ to $\mathcal{G}_2$, obtaining $(\mathcal{G}_2 = \mathcal{G}_2^0, \mathcal{G}_2^1, \cdots \mathcal{G}_2^k)$. Finally, from $\mathcal{G}_1^k$ to $\mathcal{G}_1^{k+1}$, relabel (e, t) to an edge that exists in $\mathcal{G}_2^k$ but not in $\mathcal{G}_1^k$.

It may seem unintuitive to apply relabeling operations intended for $\mathcal{G}_1$ directly to $\mathcal{G}_2$. However, we will see that our choice of (e,t) guarantees that these operations are also valid when applied to $\mathcal{G}_2$.

▶ **Lemma 9** (Sequence for decreasing difference). *For temporal graphs $\mathcal{G}_1$ and $\mathcal{G}_2$, the above algorithm correctly computes valid reconfiguration sequences $(\mathcal{G}_1, \cdots, \mathcal{G}_1^{k+1})$ and $(\mathcal{G}_2, \cdots, \mathcal{G}_2^{k})$ for some $k \in \mathbb{N}$ with $\delta(\mathcal{G}_1^{k+1}, \mathcal{G}_2^{k}) < \delta(\mathcal{G}_1, \mathcal{G}_2)$ in $\mathcal{O}(M^2)$.*

Proof. Since Change is correct, the reconfiguration sequence $(\mathcal{G}_1, \cdots, \mathcal{G}_1^{k})$ is valid, and (e,t) is a non-bridge in $\mathcal{G}_1^k$. Under our assumption that $\mathcal{G}_1$ and $\mathcal{G}_2$ have the same number of edges between any pair of vertices, and since (e,t) is not in $\mathcal{G}_2$, there must exist an edge (e,t') in $\mathcal{G}_2^k$ which is not in $\mathcal{G}_1^k$. Therefore, the whole reconfiguration sequence $(\mathcal{G}_1, \cdots, \mathcal{G}_1^{k+1})$ is valid.

We show by induction that the reconfiguration sequence $(\mathcal{G}_2, \cdots, \mathcal{G}_2^{k})$ is valid. For the base case, the sequence consisting only of $\mathcal{G}_2$ is trivially valid. Suppose $(\mathcal{G}_2^0, \ldots, \mathcal{G}_2^j)$ is valid for some $j < k$. Let (e',t') be the edge changed from $\mathcal{G}_1^j$ to $\mathcal{G}_1^{j+1}$. Since (e,t) was chosen with the shortest reconfiguration sequence and $j < k$, all edges in $\mathcal{G}_1^j$ that are not in $\mathcal{G}_2^j$ are bridges. By contraposition, all non-bridges in $\mathcal{G}_1^j$ are also in $\mathcal{G}_2^j$. Every non-bridge in $\mathcal{G}_1^j$ must be part of a cycle consisting of non-bridges. Therefore a cycle including (e',t') in $\mathcal{G}_1^j$ must also be in $\mathcal{G}_2^j$, which shows that (e',t') is a non-bridge in $\mathcal{G}_2^j$. Therefore the reconfiguration sequence $(\mathcal{G}_2^0, \cdots, \mathcal{G}_2^{j+1})$ is valid. By induction, the whole reconfiguration sequence $(\mathcal{G}_2, \cdots \mathcal{G}_2^{k})$ is valid.

Since both sequences apply the same operations up to step k, it holds that $\delta(\mathcal{G}_1, \mathcal{G}_2) = \delta(\mathcal{G}_1^k, \mathcal{G}_2^k)$. As (e,t) is not in $\mathcal{G}_2^k$ and is relabeled in $\mathcal{G}_1^k$ to an edge contained in $\mathcal{G}_2^k$, it follows that $\delta(\mathcal{G}_1^{k+1}, \mathcal{G}_2^k) < \delta(\mathcal{G}_1, \mathcal{G}_2)$.

Finally, since computing Change takes $\mathcal{O}(M^2)$ by Lemma 8 and the sequences have length at most M, the total runtime is in $\mathcal{O}(M^2)$. ◀

We can now prove our main result Theorem 4, restated here for convenience.

▶ **Theorem 4** (Finding a reconfiguration sequence in polynomial-time). *For temporal graphs $\mathcal{G}_1 = (V, \mathcal{E}_1)$ and $\mathcal{G}_2 = (V, \mathcal{E}_2)$, there exists a valid reconfiguration sequence between $\mathcal{G}_1$ and $\mathcal{G}_2$ if and only if all edges in $\mathcal{E}_1 \setminus \mathcal{E}_2$ are changeable. If that is the case we can find a valid reconfiguration sequence of length at most $2M^2$ in $\mathcal{O}(M^3)$.*

Proof. If there exists an unchangeable edge in $\mathcal{G}_1$ that is not in $\mathcal{G}_2$, this edge remains a bridge in all graphs reachable from $\mathcal{G}_1$ through a valid reconfiguration sequence. In that case, reconfiguration is not possible.

Otherwise, all edges in $\mathcal{G}_1$ that are not in $\mathcal{G}_2$ are changeable. We can then repeatedly apply Lemma 9 to compute valid reconfiguration sequences $(\mathcal{G}_1, \cdots \mathcal{G}_c)$, and $(\mathcal{G}_2, \cdots \mathcal{G}_c)$, as the graphs will be equal once their difference is 0. Since every edge relabeling operation can be reversed, while preserving the always-connected property, we reverse the sequence $(\mathcal{G}_2, \cdots, \mathcal{G}_c)$ to obtain a valid reconfiguration sequence $(\mathcal{G}_c, \cdots, \mathcal{G}_2)$. Together with the sequence $(\mathcal{G}_1, \cdots, \mathcal{G}_c)$, we get a valid reconfiguration sequence from $\mathcal{G}_1$ to $\mathcal{G}_2$.

Since there are at most M edges in $\mathcal{G}_1$ that are not in $\mathcal{G}_2$, there are at most M applications of Lemma 9. As each application results in reconfiguration sequences of length at most M, both sequences $(\mathcal{G}_1, \cdots, \mathcal{G}_c)$ and $(\mathcal{G}_c, \cdots, \mathcal{G}_2)$ have at most length M^2. Therefore the total reconfiguration sequence has length at most $2M^2$.

Since each application takes $\mathcal{O}(M^2)$ time, the total runtime is in $\mathcal{O}(M^3)$ and therefore in polynomial-time. ◀

5 NP-hardness of Shortest Reconfiguration

The natural next question is whether we can find a shortest reconfiguration sequence between two given graphs in polynomial-time. We show that such an algorithm does not exist if P ≠ NP, even for graphs that have lifetime $T = 2$. To do so, we show NP-hardness of the following decision problem: Given two always-connected temporal graphs $\mathcal{G}_1, \mathcal{G}_2$ and $\ell \in \mathbb{N}$, determine if there is a valid reconfiguration sequence from $\mathcal{G}_1$ to $\mathcal{G}_2$ of length at most ℓ. We call this problem Layered Connectivity Shortest Reconfiguration.

▶ **Theorem 10.** *Layered Connectivity Shortest Reconfiguration is NP-hard.*

We present a polynomial-time reduction from Vertex Cover which, given a graph $G = (V, E)$ and a $k \in \mathbb{N}$, asks whether there is a set of vertices $C \subseteq V$ of size at most k such that every edge has an endpoint in C.

Construction.

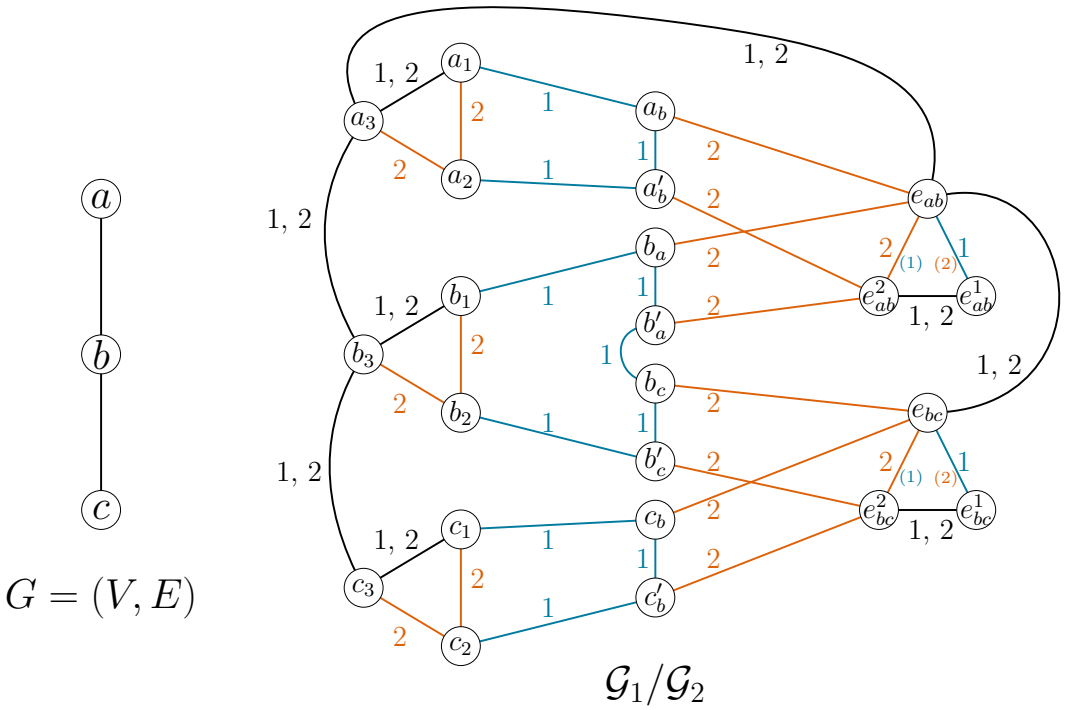

Figure 6 An example construction for a given graph G with 3 vertices a, b, c. All labels are given for $\mathcal{G}_1$. Labels in parentheses indicate differences in $\mathcal{G}_2$, and otherwise the labels are identical.

Given an instance $(G = (V, E), k)$ of Vertex Cover, we construct a temporal graph $\mathcal{G}_1$ with lifetime $T = 2$. Whenever we define an edge with label $*$, we imply that the edge exists once in each snapshot.

1. For each $v \in V$, create vertices v_1, v_2, v_3 with edges $(\{v_1, v_2\}, 2), (\{v_2, v_3\}, 2), (\{v_3, v_1\}, *)$. We refer to the 2-labeled edges incident to v_2 as *activation-edges of* v.
2. For each edge $\{u, v\} \in E$, create an *edge-gadget* with vertices $e_{uv}, e^1_{uv}, e^2_{uv}$ and edges $(\{e_{uv}, e^1_{uv}\}, 1), (\{e_{uv}, e^2_{uv}\}, 2), (\{e^1_{uv}, e^2_{uv}\}, *)$.

3. For each $v \in V$, for each edge $\{u, v\} \in E$ incident to v, create vertices v_u and v'_u. Construct a path between v_1 and v_2 through all these vertices such that any v'_u directly follows v_u. Assign label 1 to its edges. We refer to these edges as *1-transition-edges of* v.
4. For each edge $\{u, v\} \in E$, add edges $(\{e_{uv}, u_v\}, 2)$ and $(\{e_{uv}, v_u\}, 2)$, as well as $(\{e^2_{ab}, u'_v\}, 2)$ and $(\{e^2_{ab}, v'_u\}, 2)$. We refer to these edges as *2-transition-edges of* v.
5. Connect all v_3 for $v \in V$ and all e_{uv} for $\{u, v\} \in E$ in a path with label $*$.

Refer to Figure 6 for an example. $\mathcal{G}_2$ is identical to $\mathcal{G}_1$, except that the temporal edges in each edge-gadget are flipped – for brevity, we will refer to changing the label of an edge to the respective other label as *flipping* it.

$\mathcal{G}_1$ and $\mathcal{G}_2$ contain $\mathcal{O}(|V| + |E|)$ vertices edges and can be constructed in polynomial time. Let $\ell = 2k + 4|E|$. We show that G admits a vertex cover of size k if and only if there is a valid reconfiguration sequence between $\mathcal{G}_1$ and $\mathcal{G}_2$ of length at most ℓ.

Vertex Cover $\Longrightarrow$ Layered Connectivity Shortest Reconfiguration

Let $C \subseteq V$ be a vertex cover for $G = (V, E)$ of size at most k. We reconfigure $\mathcal{G}_1$ into $\mathcal{G}_2$ as follows (see Figure 7):

- For each $v \in C$:
  - Flip the activation-edge $(\{v_1, v_2\}, 2)$. This closes a 1-labeled cycle with the 1-transition-edges of v meaning all of them become non-bridges.
  - For each incident edge $\{u, v\}$ for which the edge-gadget does not yet have the labels assigned in $\mathcal{G}_2$:
    - Flip $(\{v_u, v'_u\}, 1)$. This closes a 2-labeled cycle containing $(\{e_{uv}, e^2_{uv}\}, 2)$.
    - Flip $(\{e_{uv}, e^2_{uv}\}, 2)$, and then flip $(\{e_{uv}, e^1_{uv}\}, 1)$. This edge-gadget now has the same labels as in $\mathcal{G}_2$.
    - Flip $(\{v_u, v'_u\}, 2)$ back.
  - Flip $(\{v_1, v_2\}, 1)$ back.

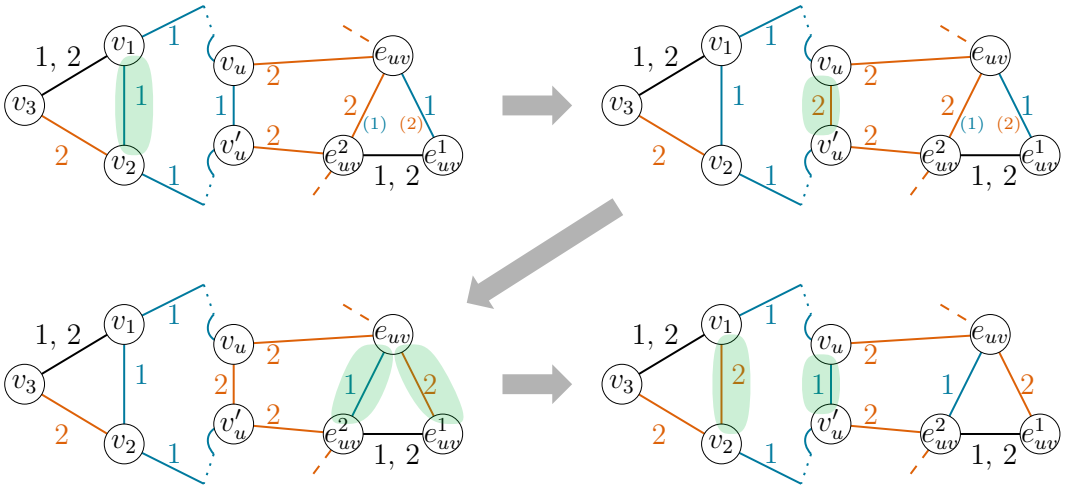

Figure 7 The reconfiguration steps to reconfigure the differing edges in the edge-gadget of $\{u, v\}$ using an activation-edge of v.

Because C is a vertex cover, all edge-gadgets will eventually be correctly reconfigured. Because these hold the only differences between $\mathcal{G}_1$ and $\mathcal{G}_2$, all other changes are reverted, and we only ever flip non-bridges, this is a valid reconfiguration sequence from $\mathcal{G}_1$ to $\mathcal{G}_2$. It

contains 4 steps for each edge in E and 2 steps for each element of C, thus it has at most length $2k + 4|E| = \ell$.

Layered Connectivity Shortest Reconfiguration $\Longrightarrow$ Vertex Cover

Let $(\mathcal{G}_1 = \mathcal{G}^0, \mathcal{G}^1, \ldots, \mathcal{G}^{\ell'} = \mathcal{G}_2)$ be a valid reconfiguration sequence from $\mathcal{G}_1$ to $\mathcal{G}_2$ of length $\ell' \leq \ell$. Let F be the set of static edges that are changed at least once during this reconfiguration. Based on this, let C be the set of vertices $v \in V$ for which an activation edge of v is in F. We show that C is a vertex cover in G of size at most k.

C has size at most k. As $\mathcal{G}_1$ and $\mathcal{G}_2$ differ in $2|E|$ many edges (those in the edge-gadgets), at least $2|E|$ steps of the reconfiguration are used to directly resolve those differences. In the remaining $\leq 2k + 2|E|$ steps, at most $k + |E|$ edges can be flipped and later un-flipped, thus F contains at most $k + |E|$ edges outside of edge-gadgets. We will argue that at least $|E|$ of them are not activation-edges, and therefore at most k edges can be activation-edges.

For every $\{u, v\} \in E$, the edges $(\{e_{uv}, e^1_{uv}\}, 1)$ and $(\{e_{uv}, e^2_{uv}\}, 2)$ of the edge-gadget are initially bridges. By Lemma 2, to turn either of them into non-bridges, a crossing edge in the reachability partition of that edge must be flipped beforehand. Note that $(\{e_{uv}, e^1_{uv}\}, 1)$ and $(\{e_{uv}, e^2_{uv}\}, 2)$ are each crossing the others reachability partition, so once one of them is flipped, the other can immediately be flipped as well. Still, some crossing edge outside of the edge-gadget must be flipped at least for one of them. For $(\{e_{uv}, e^1_{uv}\}, 1)$, this includes $(\{u'_v, e^2_{uv}\}, 2)$ and $(\{v'_u, e^2_{uv}\}, 2)$. For $(\{e_{uv}, e^2_{uv}\}, 2)$, this includes all edges incident to u'_v that have label 1, and all edges incident to v'_u that have label 1. Collect these prerequisite edges for $\{u, v\}$ in the set P_{uv} (see Figure 8).

By Lemma 3, P_{uv} does not change through other reconfiguration steps that leave $(\{e_{uv}, e^1_{uv}\}, 1)$ and $(\{e_{uv}, e^2_{uv}\}, 2)$ as bridges. Because at least one edge of P_{uv} must be flipped, and P_{uv} is disjoint from any other $P_{u'v'}$ with $\{u, v\} \neq \{u', v'\} \in E$, at least $|E|$ edges (one from each P-set) must be contained in F. Thus at most k edges in F could be activation-edges, and therefore C contains at most k elements by definition.

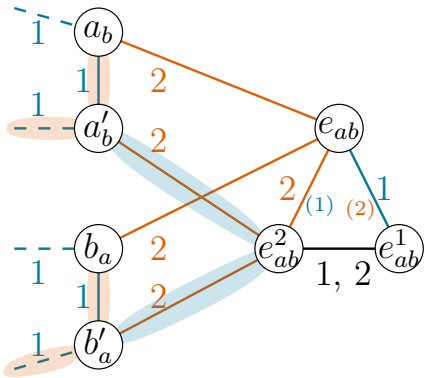

Figure 8 The set of prerequisite edges P_{uv} for the edge-gadget of $\{u, v\}$. Blue-highlighted edges are crossing edges in the reachability partition of $(\{e_{uv}, e^1_{uv}\}, 1)$, and orange-highlighted edges are the same for $(\{e_{uv}, e^2_{uv}\}, 2)$.

C is a vertex cover. We first show that for any $v \in V$, if no activation edge of v is in F, no 1-transition-edges of v and no 2-transition-edges of v can be in F: All these edges are initially bridges, so by Lemma 2 a crossing edge in their reachability partition must be flipped beforehand. For edges of the 1-transition path of v, this includes some 2-transition-edges of v and the activation-edges of v (which we are ruling out). For the mentioned 2-transition-edges,

this includes 1-transition-edges of v. We observe a cyclic dependency: Without changing an activation edge of v, the 1-transition-path and the incident 2-transition-edges depend on one another to be changed first. Thus none of them can actually be changed, and none of them can be contained in F assuming a valid reconfiguration sequence.

Now assume towards contradiction that C is not a vertex cover, i.e. there is an edge $\{u, v\}$ that is not covered by C. Then no activation-edge of u and no activation-edge of v is in F, and thus no 1- or 2-transition-edges of u or v can be in F. As explained before, at least one of the prerequisite edges P_{uv} must be flipped to reconfigure the edge-gadget of $\{u, v\}$. But because P_{uv} only contains 1- and 2-transition-edges of u and v (see Figure 8), none of them can be in F, so the edge-gadget of $\{u, v\}$ cannot be reconfigured. This contradicts the given reconfiguration sequence being valid and arriving at $\mathcal{G}_2$. Thus the assumption is wrong and C must be a vertex cover.

6 Conclusion

In this paper, we started the research into temporal graph reconfiguration problems. As a first step, we introduced the Layered Connectivity Reconfiguration problem, which asks whether two temporal graphs can be transformed into each other through a sequence of edge relabeling operations while maintaining the always-connected property at every step.

We established a necessary and sufficient condition for the existence of such a reconfiguration sequence: Every different edge must be changeable, i.e. it can eventually be transformed into a non-bridge. We then showed how to determine changeable edges in polynomial-time and, building on this, developed a polynomial-time algorithm that computes a valid reconfiguration sequence between two temporal graphs. A natural next question after showing the decision and search problems to be in P is whether the same is true for the corresponding optimization problem of finding the shortest reconfiguration sequence. We explored this question and obtained first results on the NP-completeness of deciding whether there is a reconfiguration sequence of length at most k between two temporal graphs.

Beyond these results, several open questions remain. It would be interesting to analyze how well our algorithm approximates the length of the shortest reconfiguration sequence. Another direction is to investigate the problem in restricted graph classes. Finally, exploring parameterized variants could give further insight into the problem's complexity. For example, can we get efficient algorithms by parameterizing by the number of reconfiguration steps?

References

1 Hugo A. Akitaya, Matias Korman, Oliver Korten, Diane L. Souvaine, and Csaba D. Tóth. Reconfiguration of connected graph partitions via recombination. *Theor. Comput. Sci.*, 923:13–26, 2022. URL: https://doi.org/10.1016/j.tcs.2022.04.049, doi:10.1016/J.TCS.2022.04.049.

2 Marthe Bonamy, Nicolas Bousquet, Marc Heinrich, Takehiro Ito, Yusuke Kobayashi, Arnaud Mary, Moritz Mühlenthaler, and Kunihiro Wasa. The perfect matching reconfiguration problem. In Peter Rossmanith, Pinar Heggernes, and Joost-Pieter Katoen, editors, *44th International Symposium on Mathematical Foundations of Computer Science, MFCS 2019, August 26-30, 2019, Aachen, Germany*, volume 138 of *LIPIcs*, pages 80:1–80:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. URL: https://doi.org/10.4230/LIPIcs.MFCS.2019.80, doi:10.4230/LIPICS.MFCS.2019.80.

3 Nicolas Bousquet, Takehiro Ito, Yusuke Kobayashi, Haruka Mizuta, Paul Ouvrard, Akira Suzuki, and Kunihiro Wasa. Reconfiguration of spanning trees with many or few leaves. In Fabrizio Grandoni, Grzegorz Herman, and Peter Sanders, editors, *28th Annual European Symposium*

on Algorithms, ESA 2020, September 7-9, 2020, Pisa, Italy (Virtual Conference), volume 173 of *LIPIcs*, pages 24:1–24:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. URL: https://doi.org/10.4230/LIPIcs.ESA.2020.24, doi:10.4230/LIPICS.ESA.2020.24.

4 Nicolas Bousquet, Takehiro Ito, Yusuke Kobayashi, Haruka Mizuta, Paul Ouvrard, Akira Suzuki, and Kunihiro Wasa. Reconfiguration of spanning trees with degree constraint or diameter constraint. In Petra Berenbrink and Benjamin Monmege, editors, *39th International Symposium on Theoretical Aspects of Computer Science, STACS 2022, March 15-18, 2022, Marseille, France (Virtual Conference)*, volume 219 of *LIPIcs*, pages 15:1–15:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. URL: https://doi.org/10.4230/LIPIcs.STACS.2022.15, doi:10.4230/LIPICS.STACS.2022.15.

5 Joshua J. Daymude, Andréa W. Richa, and Christian Scheideler. The canonical amoebot model: algorithms and concurrency control. *Distributed Comput.*, 36(2):159–192, 2023. URL: https://doi.org/10.1007/s00446-023-00443-3, doi:10.1007/S00446-023-00443-3.

6 Argyrios Deligkas, Eduard Eiben, and George Skretas. Minimizing reachability times on temporal graphs via shifting labels. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China*, pages 5333–5340. ijcai.org, 2023. doi:10.24963/ijcai.2023/592.

7 Argyrios Deligkas and Igor Potapov. Optimizing reachability sets in temporal graphs by delaying. *Inf. Comput.*, 285(Part):104890, 2022. URL: https://doi.org/10.1016/j.ic.2022.104890, doi:10.1016/J.IC.2022.104890.

8 Riccardo Dondi and Manuel Lafond. On the complexity of temporal arborescence reconfiguration. In Arnaud Casteigts and Fabian Kuhn, editors, *3rd Symposium on Algorithmic Foundations of Dynamic Networks, SAND 2024, June 5-7, 2024, Patras, Greece*, volume 292 of *LIPIcs*, pages 10:1–10:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. URL: https://doi.org/10.4230/LIPIcs.SAND.2024.10, doi:10.4230/LIPICS.SAND.2024.10.

9 Jessica A. Enright, Kitty Meeks, George B. Mertzios, and Viktor Zamaraev. Deleting edges to restrict the size of an epidemic in temporal networks. *J. Comput. Syst. Sci.*, 119:60–77, 2021. doi:10.1016/j.jcss.2021.01.007.

10 Thorsten Götte, Kristian Hinnenthal, Christian Scheideler, and Julian Werthmann. Time-optimal construction of overlay networks. *Distributed Comput.*, 36(3):313–347, 2023. URL: https://doi.org/10.1007/s00446-023-00442-4, doi:10.1007/S00446-023-00442-4.

11 Tesshu Hanaka, Takehiro Ito, Haruka Mizuta, Benjamin R. Moore, Naomi Nishimura, Vijay Subramanya, Akira Suzuki, and Krishna Vaidyanathan. Reconfiguring spanning and induced subgraphs. *Theor. Comput. Sci.*, 806:553–566, 2020. URL: https://doi.org/10.1016/j.tcs.2019.09.018, doi:10.1016/J.TCS.2019.09.018.

12 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. *Theor. Comput. Sci.*, 412(12-14):1054–1065, 2011. URL: https://doi.org/10.1016/j.tcs.2010.12.005, doi:10.1016/J.TCS.2010.12.005.

13 Takehiro Ito, Yuni Iwamasa, Naoyuki Kamiyama, Yasuaki Kobayashi, Yusuke Kobayashi, Shun-ichi Maezawa, and Akira Suzuki. Reconfiguration of time-respecting arborescences. In Pat Morin and Subhash Suri, editors, *Algorithms and Data Structures - 18th International Symposium, WADS 2023, Montreal, QC, Canada, July 31 - August 2, 2023, Proceedings*, volume 14079 of *Lecture Notes in Computer Science*, pages 521–532. Springer, 2023. doi:10.1007/978-3-031-38906-1_34.

14 Nina Klobas, George B. Mertzios, Hendrik Molter, and Paul G. Spirakis. The complexity of computing optimum labelings for temporal connectivity. *J. Comput. Syst. Sci.*, 146:103564, 2024. URL: https://doi.org/10.1016/j.jcss.2024.103564, doi:10.1016/J.JCSS.2024.103564.

15 Irina Kostitsyna, David Liedtke, and Christian Scheideler. Universal coating by 3d hybrid programmable matter. In Yuval Emek, editor, *Structural Information and Communication Complexity - 31st International Colloquium, SIROCCO 2024, Vietri sul Mare, Italy, May*

27-29, 2024, Proceedings, volume 14662 of *Lecture Notes in Computer Science*, pages 384–401. Springer, 2024. doi:10.1007/978-3-031-60603-8_21.

16 Othon Michail, George Skretas, and Paul G. Spirakis. On the transformation capability of feasible mechanisms for programmable matter. *J. Comput. Syst. Sci.*, 102:18–39, 2019. URL: https://doi.org/10.1016/j.jcss.2018.12.001, doi:10.1016/J.JCSS.2018.12.001.

17 Othon Michail, George Skretas, and Paul G. Spirakis. Distributed computation and reconfiguration in actively dynamic networks. *Distributed Comput.*, 35(2):185–206, 2022. URL: https://doi.org/10.1007/s00446-021-00415-5, doi:10.1007/S00446-021-00415-5.

18 Alfredo Navarra, Francesco Piselli, and Giuseppe Prencipe. Line formation and scattering in silent programmable matter. *J. Parallel Distributed Comput.*, 204:105129, 2025. URL: https://doi.org/10.1016/j.jpdc.2025.105129, doi:10.1016/J.JPDC.2025.105129.

19 Noam Solomon and Shay Solomon. A generalized matching reconfiguration problem. In James R. Lee, editor, *12th Innovations in Theoretical Computer Science Conference, ITCS 2021, January 6-8, 2021, Virtual Conference*, volume 185 of *LIPIcs*, pages 57:1–57:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. URL: https://doi.org/10.4230/LIPIcs.ITCS.2021.57, doi:10.4230/LIPICS.ITCS.2021.57.

20 R. Endre Tarjan. A note on finding the bridges of a graph. 2(6):160–161. URL: https://www.sciencedirect.com/science/article/pii/0020019074900039, doi:10.1016/0020-0190(74)90003-9.

21 Kunihiro Wasa, Katsuhisa Yamanaka, and Hiroki Arimura. The complexity of induced tree reconfiguration problems. *IEICE Trans. Inf. Syst.*, 102-D(3):464–469, 2019. URL: https://doi.org/10.1587/transinf.2018FCP0010, doi:10.1587/TRANSINF.2018FCP0010.