Temporal Cliques Admit Sparse Spanners

Arnaud Casteigts, Joseph G. Peters, Jason Schoeters

Version of Record: https://www.sciencedirect.com/science/article/pii/S0022000021000428
Manuscript_28bbff010eac34cfcd2c3e0ef859aa65

Temporal Cliques Admit Sparse Spanners [*†]

Arnaud Casteigts
LaBRI, University of Bordeaux, France
arnaud.casteigts@labri.fr

Joseph G. Peters
School of Computing Science
Simon Fraser University, Burnaby, BC, Canada
peters@cs.sfu.ca

Jason Schoeters
LaBRI, University of Bordeaux, France
jason.schoeters@labri.fr

Abstract

Let $G = (V, E)$ be an undirected graph on n vertices and $\lambda : E \to 2^{\mathbb{N}}$ a mapping that assigns to every edge a non-empty set of integer labels (discrete times when the edge is present). Such a labelled graph $\mathcal{G} = (G, \lambda)$ is *temporally connected* if a path exists with non-decreasing times from every vertex to every other vertex. In a seminal paper, Kempe, Kleinberg, and Kumar [17] asked whether, given such a temporally connected graph, a *sparse* subset of edges always exists whose labels suffice to preserve temporal connectivity – a *temporal spanner*. Axiotis and Fotakis [5] answered negatively by exhibiting a family of $\Theta(n^2)$-dense temporal graphs which admit no temporal spanner of density $o(n^2)$. In this paper, we give the first positive answer as to the existence of $o(n^2)$-sparse spanners in a dense class of temporal graphs, by showing (constructively) that if G is a complete graph, then one can always find a temporal spanner with $O(n \log n)$ edges.

1 Introduction

The study of highly dynamic networks has gained interest lately, motivated by epidemics analysis and emerging technological contexts like vehicular networks, robots, and drones, where the entities move and communicate with each other. The communication links in these networks vary with time, leading to the definition of temporal graph models (also called time-varying graphs or evolving graphs) where temporality plays a key role. In these graphs, the properties of interest are often defined over time rather than at a given instant. For example, the graph may never be connected, and yet offer a form of connectivity over time. In [7], a dozen temporal properties were identified that have been effectively exploited in the distributed computing and networking literature. Perhaps the most basic property is that of *temporal connectivity*, which requires that every vertex can reach every other vertex through a temporal path (also called *journey* [6]), that is, a path whose edges are used over a non-decreasing sequence of times. The times may also be required to be strictly increasing (strict journeys), both cases being carefully discussed in this paper. Temporal connectivity was considered in an early paper by Awerbuch and Even [4] (1984), and studied from a graph-theoretical point of view in the early 2000's in a number of seminal works including Kempe, Kleinberg, and Kumar [17], and Bui-Xuan, Ferreira, and Jarry [6] (see also [21] for an early study of graphs with time-dependent delays on the edges). More recently, it has been the subject of algorithmic

[*]This article is the full version of an article presented at ICALP 2019 [9]. This paper is best read in colour. If this is not possible, the colour "green" appears with a lighter tone than the colour "red", making their interpretation possible.

[†]This research was supported by ANR project ESTATE (ANR-16-CE25-0009-03) and NSERC of Canada.

studies, such as [2] and [5] (discussed below), and [8, 11, 13, 18, 23, 25], which consider algorithms for computing structures related to temporal connectivity. Broad reviews of these topics can be found, *e.g.,* in [7, 16, 19], although the list is non-exhaustive and the literature is rapidly evolving.

1.1 Sparse Temporal Spanners and Related Work

In a seminal paper, Kempe, Kleinberg, and Kumar [17] asked whether every temporally connected graph on n vertices has a sparse spanning subgraph that is also temporally connected. In their model, each edge has a single label indicating a time when it is present, thus the edges are identified by their labels, but the discussion is more general. What they are asking, essentially, is whether an analogue of spanning tree exists for temporal graphs when the labels are *already fixed*. We call such a temporally connected spanning subgraph a *temporal spanner*.

Kempe *et al.* answer their own question immediately (and negatively) for the particular case of $\Theta(n)$ density, by showing that hypercubes labelled in a certain way need all of their edges to achieve temporal connectivity, thus some temporal graphs of density $\Theta(n \log n)$ cannot be sparsified. The more general question, asking whether $o(n^2)$-sparse spanners always exist in dense temporal graphs remained open for more than a decade, and was eventually settled, again negatively, by Axiotis and Fotakis [5]. The proof in [5] exhibits an infinite family of temporally connected graphs with $\Theta(n^2)$ edges that do not admit $o(n^2)$-sparse spanners. Their construction can be adapted for strict and non-strict journeys.

On the positive side, Akrida *et al.* [2] show that, if the underlying graph G is a complete graph and every edge is assigned a single globally-unique label, then it is always possible to find a temporal spanner of density $\binom{n}{2} - \lfloor n/4 \rfloor$ edges (leaving however the asymptotic density unchanged). Akrida *et al.* [2] also prove that if the label of every edge in G is chosen uniformly at random (from an appropriate interval), then almost surely the graph admits a temporal spanner with $O(n \log n)$ edges. Both [5] and [2] include further results related to the (in-)approximability of finding a *minimum* temporal spanner, which is out of the scope of this paper.

By its nature, the problem of finding a temporal spanner in a temporal graph seems to be significantly different from its classical (*i.e.,* non-temporal) version, whether this version considers a static graph (see *e.g.,* [10, 20, 22]) or the current topology of an updated dynamic graph (see *e.g.,* [3, 12, 14]). The essential difference is that spanning trees always exist in standard (connected) graphs, thus one typically focuses on the trade-off between the density of a solution and a quality parameter like the *stretch factor*, rather than on the very existence of a sparse spanner.

1.2 Contributions

In this paper, we establish that temporal graphs built on top of complete graphs *unconditionally* admit $O(n \log n)$-sparse temporal spanners when *non-strict* journeys are allowed. Furthermore, such a spanner can be computed in polynomial time. The case of strict journeys requires more discussion. Kempe *et al.* observed in [17] that if every edge of a complete graph is given the same label, then this graph is temporally connected, but no multi-hop *strict* journey exists, thus none of the edges can be removed, and the problem is trivially unsolvable. To make the problem interesting when only strict journeys are allowed, one should constrain the edge labelling to avoid such a pathological situation. In the present case, we require that a sub-labelling of one label per edge exists in which any two adjacent edges have different labels. This formulation slightly generalizes the single-label global-unicity assumption made in [2] (although essentially equivalent) and eliminates the distinction between strict and non-strict journeys. Under this restriction, we establish that every temporal graph whose underlying graph is complete admits an $O(n \log n)$-sparse temporal spanner. Moreover, if the restricted labelling is given, then the spanner can

be computed in polynomial time. (The problem of deciding whether a general labelling admits such a sub-labelling is not discussed here; it might be computationally hard.)

We start by observing that the above two settings one-way reduce to the setting where every edge has a single label and two adjacent edges have different labels. The reduction is "one-way" in the sense that the transformed instance may have fewer feasible journeys than the original instance, but all of these journeys correspond to valid journeys in the original instance, so a temporal spanner computed in the transformed instance is valid in the original instance. As a result, our main algorithm takes as input a complete graph $\mathcal{G}$ with single, locally distinct labels, and computes an $O(n \log n)$-sparse temporal spanner of $\mathcal{G}$ in polynomial time.

In summary, we give the first positive answer to the question of whether sparse temporal spanners always exist in a class of dense graphs, focusing here on the case of complete graphs. This answer complements the negative answer by Axiotis and Fotakis [5] and motivates more investigation to understand where the transition occurs between their negative result (no sparse spanners exist in some dense temporal graphs) and our positive result (they essentially always exist in complete graphs). Our algorithms are based on a number of original techniques, which we think may be of more general interest for problems related to temporal connectivity.

The paper is organized as follows. In Section 2, we define the model and notation, and describe the one-way reductions that allow us to concentrate subsequently on single and distinct labels. We also mention two basic techniques of interest for this problem, although they are not used subsequently in the paper, namely the *sub-clique* technique (from [2]) and the *pivoting* technique (introduced here). In Section 3 we introduce the main concepts and techniques used in our algorithms, namely *delegation*, *dismountability*, and k*-hop dismountability*, whose purpose is to recursively self-reduce the problem to smaller graph instances. These techniques are subsequently combined into a more sophisticated algorithm that successfully computes a temporal spanner of $O(n \log n)$ edges. The first step, presented in Section 4, is called *fireworks*. It is based on a generalization of delegation called *one-sided delegation* and results in a spanner of density (essentially) $\binom{n}{2}/2$. Then, in Section 5, we exploit a particular dichotomy in the structure of the residual instance, which allows us to sparsify the graph down to $O(n \log n)$ edges. In Section 6, we review the main components of the algorithm, showing that its running time complexity is polynomial. Finally, we present a few open questions in Section 7 and conclude with some remarks in Section 8.

2 Definitions and Basic Results

2.1 Model and Definitions

Let $G = (V, E)$ be an undirected graph on $n = |V|$ vertices and $\lambda : E \to 2^{\mathbb{N}}$ a mapping that assigns to every edge of E a non-empty set of integer labels. These labels can be seen as discrete times when the edge is present. In this paper, we refer to the resulting graph $\mathcal{G} = (G, \lambda)$ as a *temporal graph* (other models and terminologies exist, many of them being equivalent for the considered problem). If λ is single-valued and locally injective (*i.e.,* adjacent edges have different labels), then we say that λ is *simple*, and by extension, a temporal graph is simple if its labelling is simple.

A temporal path in $\mathcal{G}$ (also called *journey*), is a finite sequence of k triplets $\mathcal{J} = \{(u_i, u_{i+1}, t_i)\}$ such that $(u_1, \ldots, u_{k+1})$ is a path in G and for all $1 \leq i \leq k$, $\{u_i, u_{i+1}\} \in E$, $t_i \in \lambda(\{u_i, u_{i+1}\})$ and $t_{i+1} \geq t_i$. Strict temporal paths (strict journeys) are defined analogously by requiring that $t_{i+1} > t_i$. We say that a vertex u can *reach* a vertex v iff a journey exists from u to v (strict or non-strict, depending on the context). If every vertex can reach every other vertex, then $\mathcal{G}$ is *temporally connected*. Finally, observe that the distinction between strict and non-strict journeys does not exist in simple temporal graphs, as all the journeys are strict.

In general, one can define a *temporal spanner* of a temporally connected graph $\mathcal{G} = ((V, E), \lambda)$ as a temporally connected spanning subgraph $\mathcal{G}' = ((V', E'), \lambda')$ such that $V = V'$, $E' \subseteq E$ and $\lambda' : E' \to 2^{\mathbb{N}}$ with $\lambda'(e) \subseteq \lambda(e)$ for all $e \in E'$. Observe that, if $\mathcal{G}$ is simple, then the structure of a spanner of $\mathcal{G}$ is fully determined by the chosen subset of edges $E' \subseteq E$ and the temporal connectivity of the spanner is established by the labels on the edges in E'. Thus, in such cases, we say that E' itself *is* the spanner. Many of these notions are analogous to the ones considered in [2, 5, 17], although they are not referred to as "spanners" in these works.

Finally, when the graph G underlying a temporal graph $\mathcal{G} = (G, \lambda)$ is a complete graph, we call $\mathcal{G}$ a temporal clique. An example of a temporal spanner of a *simple temporal clique* is shown in Figure 1. The endpoints of the edge labelled 1 are temporally connected in the spanner by the journey with labels 4 and 6 in one direction and the journey with labels 3 and 7 in the other direction.

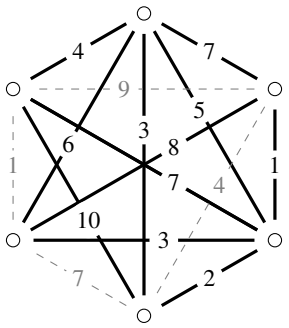


Figure 1: Example of a simple temporal clique and one of its temporal spanners (edges in bold). This spanner is not minimum (nor even minimal) and the reader may try to remove further edges.

2.2 Generality of Simple Labellings

We claimed in Section 1.2 that if non-strict journeys are allowed, then one can transform a temporal clique $\mathcal{G} = (G, \lambda)$ with *unrestricted* labelling λ into a clique $\mathcal{H} = (G, \lambda_H)$ with *simple* labelling λ_H such that any valid temporal spanner of $\mathcal{H}$ induces a valid temporal spanner of $\mathcal{G}$. (As explained, the converse is false, but this is not a problem.) The reduction is in two steps. First, for every edge e, restrict $\lambda(e)$ to a single label chosen arbitrarily from $\lambda(e)$. Then in the resulting temporal clique, whenever k edges incident to the same vertex have the same label l, these edges are relabelled with distinct values in the interval $[l, l+k-1]$ (arbitrarily) and the values of all the other labels in the graph that are larger than l are increased by k. It is not difficult to see that if a journey exists in $\mathcal{H}$, then the same sequence of edges induces a (possibly non-strict) journey in $\mathcal{G}$.

As for the second claim of the introduction, if strict journeys are the only ones allowed, then as explained, we require the existence of a *simple* sub-labelling λ' of λ (whose computation is not discussed). Here, it is even more direct that any journey based on the sub-labelling λ' is available in the original instance. Based on these arguments, the rest of the paper focuses on simple temporal cliques, and we sometimes drop the adjective "simple" when it is clear from the context.

2.3 Basic Techniques

We present here two basic sparsification techniques. The first (subcliques) is from previous works, and the second (pivotability) is original. Although these techniques are not directly used in the rest of the paper, they serve two main purposes. Firstly, both techniques are simple and can serve as a gentle warm-up for the reader. Secondly, the pivotability technique is a natural analog of Kosaraju's principle, which gives

linear-size spanners in static directed graphs. However, we show that pivotability does not always work in temporal graphs, by presenting an infinite family of graphs that are not pivotable. This negative result thus motivates (and legitimates) the more sophisticated techniques presented in the rest of the paper.

2.3.1 The Subclique Technique

So far, the only existing approach for sparsifying simple temporal cliques is that of Akrida *et al.* [2], who prove that one can always remove $\lfloor n/4 \rfloor$ edges without breaking temporal connectivity. Their approach is as follows. First, it is established that if $n = 4$, then it is always possible to remove at least *one* edge. Then, as $n \to \infty$, one can arbitrarily partition the input clique into (essentially) $n/4$ subcliques of 4 vertices each, and remove an edge from each subclique. The edges *between* subcliques are kept, thus the impact of each removal is limited to the corresponding subclique and the resulting graph is temporally connected. Before moving to other techniques, let us observe that this technique can be greatly improved as follows.

Observation 1 (Improving the subclique technique)**.** The type of partitioning used in [2] is *vertex-disjoint*, but it turns out that the same argument can be applied to *edge-disjoint* subcliques, with significant consequences. Indeed, by Wilson's Theorem [24], the number of edge-disjoint cliques on 4 vertices in a complete graph on n vertices is $\lfloor n^2/12 \rfloor$. (More generally, as $c \to 1$, graphs with minimum degree at least cn have $\lfloor \binom{n}{2} / \binom{k}{2} \rfloor$ disjoint copies of K_k for all k.) Therefore, one can remove $\lfloor n^2/12 \rfloor = \Theta(n^2)$ edges.

Although this technique allows us to remove $\Theta(n^2)$ edges, it seems unlikely that purely *structural* techniques like this one will lead to spanners of $o(n^2)$ edges. The techniques we develop in this paper are different in essence and consider the interplay of timestamps at a finer scale.

2.3.2 The Pivotability Technique

Another natural approach that one might think of is inspired by Kosaraju's principle for testing strong connectivity in a directed graph (see [1]). This principle relies on finding a vertex that all the other vertices can reach (through directed paths) and that can reach all these vertices in return. This condition is sufficient in standard graphs because paths are transitive. In the temporal setting, transitivity does not hold, but we can define a temporal analogue as follows. A *pivot vertex* p is a vertex such that all other vertices can reach p by some time t (through journeys) and p can reach all other vertices *after* time t. The union of the tree of (incoming) journeys towards p and the tree of (outgoing) journeys from p forms a temporal spanner with at most $2(n-1)$ edges. Such a spanner is illustrated in Figure 2. Experiments suggest that pivot vertices exist asymptotically almost surely in random temporal cliques (where an instance corresponds to a random permutation of the edge labels $\{0, 1, 2, \ldots, \binom{n}{2} - 1\}$. Unfortunately, arbitrarily large non-pivotable cliques exist, as shown next.

Non-pivotable Graphs. Here, we explain how to construct non-pivotable simple temporal cliques of arbitrary sizes. The construction ensures that there is a time t before which no vertex can be reached by all vertices, and after which no vertex can reach all vertices. The choice of t does not matter, as moving it forward or backward could only worsen one of the directions. Thus, the simple existence of such a t rules out the existence of a pivot vertex. The construction is first presented with respect to the 6-vertex clique of Figure 3; then we explain how to generalize it. Thus, let $n = 6$. In this case, let $t = 7$ and let us consider the two periods $[0, 7]$ and $[8, 14]$. Looking at the graph, observe that none of the vertices can be reached by all the others during the first period. This is true because (1) the only vertex that w can reach is v, making v the only candidate, and (2) none of the other vertices (except u) can reach v. Similarly, none of the vertices can reach all the others in the second period. This is true because (1) u can only be reached by w, making w the only candidate, and (2) w cannot reach v in the second period.

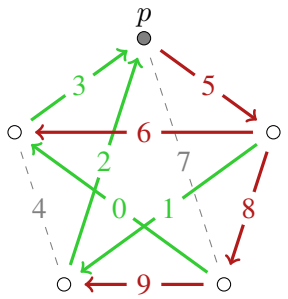


Figure 2: Example of a pivotable graph. The (light) green edges belong to the tree of incoming journeys to pivot vertex p (with $t = 4$); the (darker) red edges belong to the tree of outgoing journeys; the dashed edges belong to neither.

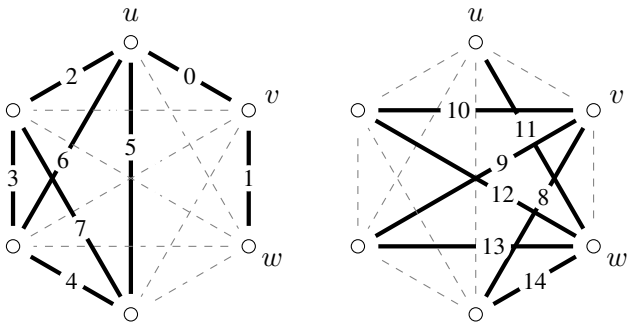


Figure 3: Example of a non-pivotable clique seen as the union of two specific subgraphs which represent the periods $[0, 7]$ (left) and $[8, 14]$ (right).

The construction can be generalized to any larger value of n by choosing three vertices to play the same role as u, v, and w. The graph of the first period corresponds to a subclique on $n - 2$ vertices (including u, but not v and w), plus the two edges $\{u, v\}$ and $\{v, w\}$. Thus $t = \binom{n-2}{2} + 1$. The labelling assigns 0 to $\{u, v\}$, 1 to $\{v, w\}$, and all values in $[2, t]$ to the edges of the subclique (arbitrarily). By the same argument as above, none of the vertices can be reached by all others during the first period. As for the second period, the labelling must ensure that all the edges of w have a larger label than all the edges of v, and that $\{u, w\}$ is assigned the smallest label among the edges incident to w, which results in direct applicability of the same argument as above. Thus, none of the vertices can reach all the others during the second period.

3 Delegation and Dismountability

This section introduces a number of basic techniques which are subsequently used (and extended) in Sections 4 and 5. Given a vertex v, we use $e^-(v)$ to denote the edge with smallest label incident to v, and $e^+(v)$ analogously for the largest label.

Lemma 1. *Given a temporal clique $\mathcal{G}$, if $\{u, v\} = e^-(v)$, then u can reach all other vertices through v. Symmetrically, if $\{u, w\} = e^+(w)$, then all vertices can reach u through w.*

Proof. If $\{u, v\} = e^-(v)$, then v has an edge with every other vertex after that time, thus a journey exists from u to every vertex through v. A symmetrical argument applies in the second case. □

Observe that Lemma 1 holds only when the underlying graph is complete. This property makes it possible for a vertex u to *delegate* its emissions to a vertex v, *i.e.,* exploit the fact that v can still reach all

the other vertices *after* interacting with u, thus none of u's other edges are required for reaching the other vertices. By a symmetrical argument, u can delegate its receptions to a vertex w if w can be reached by all the other vertices *before* interacting with u, so the other edges of u are not needed for being reached by other vertices.

3.1 Dismountability

The delegation concept suggests an interesting technique to construct temporal spanners. We say that a vertex u in a temporal clique $\mathcal{G}$ is *dismountable* if there exist two other vertices v and w such that $\{u, v\} = e^-(v)$ and $\{u, w\} = e^+(w)$, i.e., u can delegate both its emissions and its receptions. The existence of such a vertex enables a self-reduction of the spanner construction as follows: select $e^-(v)$ and $e^+(w)$ for future inclusion in the spanner, then recurse on the smaller clique $\mathcal{G}[V \setminus u]$, as illustrated in Figure 4. More precisely:

Theorem 1 (Dismountability)**.** *Let $\mathcal{G}$ be a temporal clique, and let u, v, w be three vertices in $\mathcal{G}$ such that $\{u, v\} = e^-(v)$ and $\{u, w\} = e^+(w)$. Let S' be a temporal spanner of $\mathcal{G}[V \setminus u]$. Then $S = S' \cup \{\{u, v\}, \{u, w\}\}$ is a temporal spanner of $\mathcal{G}$.*

Proof. Since $\{u, v\} = e^-(v)$, all edges incident to v in S' have a larger label than $\{u, v\}$, thus u can reach all the vertices through v using only the edges of S'. A symmetrical argument implies that all vertices in $\mathcal{G}$ can reach u through w using only $\{u, w\}$ and the edges of S'. □

We call a graph *dismountable* if it contains a dismountable vertex. It is said to be *fully dismountable* if one can find an ordering of V that allows for a recursive dismounting of the graph until the residual instance has two vertices. An example of fully dismountable graph is given in Figure 4.

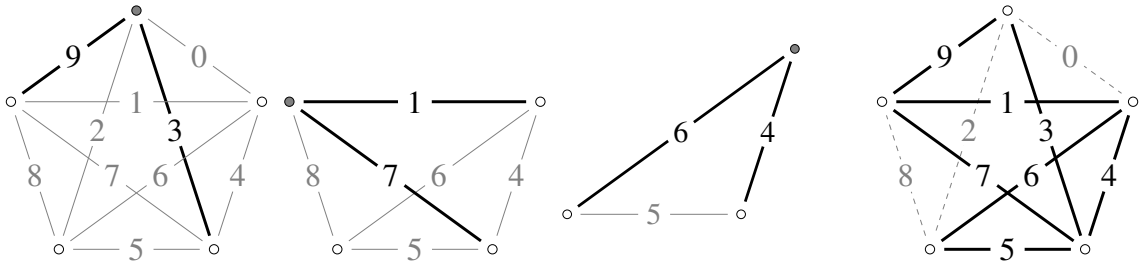


Figure 4: Example of a fully dismountable graph and the resulting spanner.

Lemma 2 (Spanners based on dismountability)**.** *If a graph can be fully dismounted, then the resulting spanner will have $2(n-2)+1 = 2n-3$ edges.*

Proof. Each dismounted vertex contributes two edges to the spanner and the residual instance contributes one edge. □

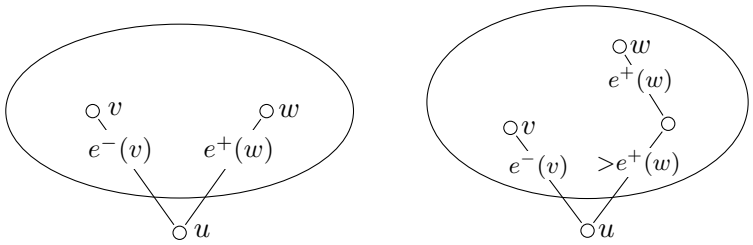
Unfortunately, there are arbitrarily large temporal cliques which are not fully dismountable as shown in Subsection 3.3.

3.2 k-hop Delegation and k-hop Dismountability

The dismountability technique can be generalized to multi-hop journeys. Let J be a journey from vertex u to vertex v through vertices $u = u_0, u_1, \ldots, u_k = v$ with $\{u_{k-1}, u_k\} = e^-(v)$. The key observation is that

u can delegate its emissions to v even if $\{u_{i-1}, u_i\} \neq e^-(u_i)$ for some i. Indeed, it is sufficient that the *last* edge of a journey from u to v is the minimum (at v) in order to delegate u's emissions to v. Symmetrically, it is sufficient that the *first* edge of a journey from w to u is $e^+(w)$ in order to delegate u's receptions to w. Accordingly, a vertex u is called *k-hop dismountable* if one can find two other vertices v and w (possibly identical if $k > 1$) such that there are journeys of *at most* k hops (1) from u to v that arrives at v through $e^-(v)$, and (2) from w to u that leaves w through $e^+(w)$. See Figure 5(b) for an illustration.

We call a graph *k-hop dismountable* if it contains a k-hop dismountable vertex. It is said to be *fully k-hop dismountable* if one can find an ordering of V that allows for a recursive k-hop dismounting of the graph until the residual instance has two vertices. Note that k-hop dismountability implies k'-hop dismountability for $k' > k$ by definition but the converse is not true.



(a) Dismountability principle. (b) Example of 2-hop dismountability.

Figure 5: Illustration of the principle of dismountability and k-hop dismountability.

Temporal spanners can be obtained in a similar way to 1-hop dismountability by selecting all of the edges involved in these journeys for inclusion in the spanner. However, only the edges adjacent to the dismounted vertex are removed in the recursion, thus some edges used in a multi-hop journey may be selected several times over the recursion (to our advantage). We can then state an analogous fact for k-hop dismountability as follows.

Lemma 3. *If a temporal graph $\mathcal{G}$ is fully k-hop dismountable, then this process yields a temporal spanner with at most $2k(n-2)+1 < 2kn$ edges.*

Proof. Each dismounted vertex contributes at most $2k$ edges to the spanner and the residual instance contributes one edge. □

Unfortunately, again there exist arbitrarily large graphs (in fact, even cliques) which are not k-hop dismountable (for any k), as discussed next. Nonetheless, k-hop dismountability is a core component in the more sophisticated techniques presented in this paper.

3.3 Non-dismountable Cliques

Here, we explain how to construct an infinite family of simple temporal cliques which are not k-hop dismountable for any k. Such a clique can be constructed for any even $n \geq 4$. We construct this family by first partitioning the vertices into two equal parts V^- and V^+, and then adding the edges (within and between these parts) using time labels in the set $\{0, 1, 2, ..., \binom{n}{2} - 1\}$, as follows (see Figure 6). First, create a subclique among all vertices of V^- and put the $\binom{n/2}{2}$ smallest labels on these edges. Similarly, create a subclique among all vertices of V^+ and put the $\binom{n/2}{2}$ largest labels on the edges. Then, create two edge-disjoint maximum matchings, denoted M^- and M^+, between the two subcliques (so each matching has $\frac{n}{2}$ edges). On the edges of M^-, put the $n/2$ smallest remaining labels, and on M^+ put the $n/2$ largest remaining labels. Add all of the missing edges between V^- and V^+ (call this set of edges E') to obtain

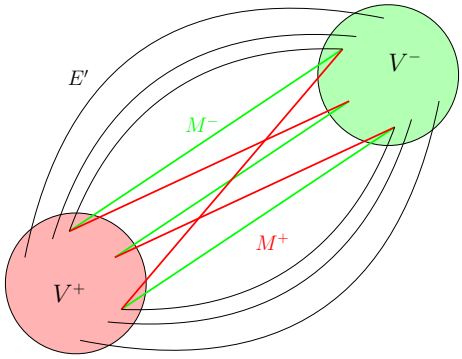


Figure 6: Construction of a non-dismountable graph.

a clique on n vertices, and label them with the remaining labels. Thus, in increasing order of labels, we have the edges among the vertices of V^-, edges of M^-, edges of E', edges of M^+, and finally the edges among the vertices of V^+.

Clearly, no vertex $u \in V^-$ can reach a vertex v using $e^-(v)$, and no vertex $u \in V^+$ can be reached by a vertex w using $e^+(w)$, regardless of how many hops are allowed. Thus, this construction creates a (k-hop) non-dismountable temporal graph.

Remark 1. *The fireworks technique (presented in Section 4) always finds a 2-hop dismountable node if n is odd. Thus the restriction that n is even in the above construction is necessary (at least for $k \geq 2$).*

Finally, observe that the non-dismountable cliques obtained by this construction are pivotable. However, some graphs (and even some cliques) exist which are neither pivotable nor dismountable, as illustrated in Figure 7. Experiments that we have conducted suggest that there exist arbitrarily large graphs which are non-pivotable and non-dismountable, however, we leave the characterization of an infinite family with these properties as an open question.

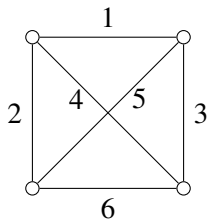


Figure 7: Example of a temporal clique that is neither dismountable nor pivotable.

4 The Fireworks Technique

In this section, we present an algorithm called *fireworks*, which exploits a system of multi-hop delegations among vertices. In particular, we take advantage of *one-sided delegations*, in which a vertex delegates only its emissions, or only its receptions. The combination of many such delegations is shown to lead to the removal of essentially half of the edges of the input clique. The residual instance has a particular structure that is exploited in Section 5 to obtain the final $O(n \log n)$-sparse spanners.

4.1 Forward Fireworks

The purpose of fireworks is to mutualize several one-sided delegations in a transitive way, so that many vertices do not need to reach the other vertices directly. Given a temporal clique $\mathcal{G} = (G, \lambda)$ with $G = (V, E)$, define the directed graph $G^- = (V, E^-)$ such that $(u, v) \in E^-$ iff $\{u, v\} = e^-(v)$, except that, if $e^-(u) = e^-(v)$ for some u and v, only one of the arcs is included (chosen arbitrarily).

Lemma 4. *The sequences of labelled edges in $\mathcal{G}$ corresponding to directed paths in G^- are journeys in $\mathcal{G}$.*

Proof (by contradiction). Let $(u_0, u_1), (u_1, u_2), ..., (u_{k-1}, u_k)$ be a directed path in G^- and suppose that the corresponding path in $\mathcal{G}$ is *not* a journey. Then it must be the case that the label of an edge (u_{i-1}, u_i) is greater than the label on the adjacent edge (u_i, u_{i+1}) for some i. Then $\{u_{i-1}, u_i\} \neq e^-(u_i)$ which is impossible. □

By construction, E^- induces a disjoint set of *out-trees* (one source, possibly several sinks). We transform G^- into a disjoint set $\mathcal{T}^- = (V, E_T^-)$ of *in-trees* (one sink, possibly several sources) as follows (see also Figure 8 for an illustration). Let E_T^- be initialized as a copy of E^-. For every v with outdegree at least 2 in G^-, let $(v, u_1), ..., (v, u_\ell)$ be its out-arcs with (v, u_ℓ) being the one with the *largest* label. For every $i < \ell$, if u_i is a sink vertex, then flip the direction of (v, u_i) in E_T^- (*i.e.,* replace (v, u_i) by (u_i, v) in E_T^-); otherwise remove (v, u_i) from E_T^-. Let $\mathcal{T}^- = (V, E_T^-)$ be the resulting set of in-trees $\mathcal{T}_1^-, ..., \mathcal{T}_k^-$ (containing possibly more in-trees than the number of initial out-trees).

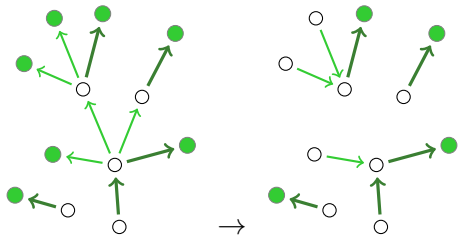


Figure 8: Example of transformation from a disjoint set of out-trees (V, E^-) to a disjoint set of in-trees (V, E_T^-). The coloured vertices represent sink vertices, and the darker arrows represent out-arcs with the largest label (for the vertex on the tail side).

Lemma 5. *The set of in-trees $\mathcal{T}^- = (V, E_T^-)$ has the following properties:*

1. *Directed paths in $\mathcal{T}^-$ correspond to journeys in $\mathcal{G}$.*
2. *Every vertex belongs to exactly one tree.*
3. *Every tree contains at least two vertices.*
4. *There is a unique sink in each tree.*
5. *The unique arc incident to a sink s corresponds to $e^-(s)$.*

Proof. Each property is proved as follows.

1. This follows from Lemma 4 because an arc (v, u_i) is only replaced by (u_i, v) if the label of (v, u_i) is less than the label of another arc (v, u_ℓ), so $(u_i, v), (v, u_\ell)$ is a journey in $\mathcal{G}$.
2. The transformation from out-trees to in-trees does not add arcs between trees.

3. Every out-tree contains at least one arc (the arc of its source with minimum label). An out-tree with a single arc is not changed by the transformation. Arcs to sinks are not removed (but they may be flipped). A vertex v that loses arcs during the transformation retains at least its arc with minimum label. In summary, no vertex loses all of its arcs.

4. Each vertex in an in-tree has at most one out-arc.

5. The unique arc to a sink is an arc that was not flipped by the transformation. □

Observe that some of the journeys induced by the arcs of $\mathcal{T}^-$ may include intermediate hops where the arc's label is not locally minimum for its head endpoint. However, as already discussed in Section 3.2, a delegation only requires that the label of the last hop of a journey be locally minimum, and that is the case here (Lemma 5.5).

The motivation behind this construction is that all the vertices in each in-tree are able to delegate their emissions to the corresponding sink vertex. For this reason, the sink vertex will be called an *emitter* in the rest of the paper. An important consequence of our construction is that the number of emitters in $\mathcal{T}^-$ cannot exceed half of the total number of vertices.

Lemma 6. *The number of emitters in $\mathcal{T}^-$ is at most $n/2$.*

Proof. After the transformation from E^- to E^-_T, there is only one emitter in each in-tree $\mathcal{T}^-_i \in \mathcal{T}^-$ (Lemma 5.4), and there are at most $n/2$ trees because each one contains at least 2 vertices (Lemma 5.3). □

We are now ready to define a temporal spanner based on $\mathcal{T}^-$, which consists of the union of all edges corresponding to arcs in the in-trees and all edges incident to at least one emitter. More formally, let $S^-_T = \{\{u,v\} \in E : (u,v) \in \mathcal{T}^-\} \cup \{\{u,v\} \in E : u \text{ is an emitter}\}$.

Theorem 2. *S^-_T is a temporal spanner of $\mathcal{G}$.*

Proof. By Lemma 5, every vertex v of $\mathcal{G}$ that is a non-emitter in $\mathcal{T}^-$ can reach an emitter s through an edge $e^-(s)$. Furthermore, the inclusion of all edges incident to a vertex s that is an emitter in $\mathcal{T}^-$ ensures that v can still reach all other vertices afterwards and so can s. Therefore, every vertex can reach all other vertices by using only edges from S^-_T. □

We call this type of spanner a *forward fireworks spanner*. An example is given in Figure 9, the corresponding journeys being depicted on the right side. Note that the orientations of the arcs with labels 0 and 1 in this example can be chosen arbitrarily when constructing E^-. If the orientations are chosen as shown in the right part of the figure, then $E^- = E^-_T$.

Theorem 3. *Forward fireworks spanners have at most $3\binom{n}{2}/4 + O(n)$ edges.*

Proof. Let S^-_T be a forward fireworks spanner based on a set of in-trees $\mathcal{T}^-$. Each non-emitter in $\mathcal{T}^-$ has only one out-arc which becomes one edge in S^-_T, thus overall $\mathcal{T}^-$ contributes less than n edges to S^-_T. Now, every emitter has an edge to every other vertex that is included in S^-_T, and there are at most $n/2$ emitters in $\mathcal{T}^-$ by Lemma 6. This contributes at most $(n/2)(n-1)$ edges. Note that the edges between emitters are selected twice but should be counted only once. Thus in the end, there are at most $(n/2)(n-1) - \binom{(n/2)}{2} = 3n^2/8 - n/4 = 3\binom{n}{2}/4 + n/8$ edges, plus the $O(n)$ edges contributed by the in-trees. □

Before moving to Section 4.2, we establish a small technical lemma about the structure of the in-trees, which will be used in Section 5.

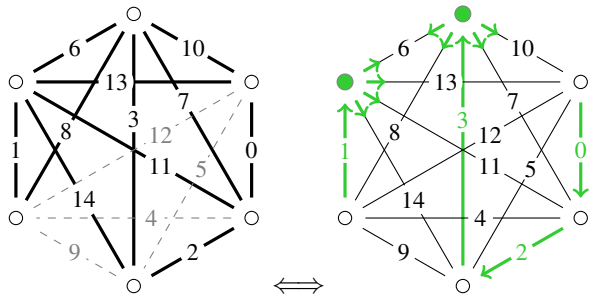


Figure 9: Example of a forward fireworks spanner (left) and the journeys from which it is constructed (right).

Lemma 7. *Every non-emitter vertex u in $\mathcal{T}^-$ can reach another vertex v in the same in-tree using a journey of length* at most two *that arrives at v through $e^-(v)$.*

Proof. For each non-emitter vertex u, there exists a w such that $(u, w) \in E_T^-$. Either $(u, w) \in E^-$ and then $\{u, w\} = e^-(w)$, or (u, w) has been obtained by flipping $(w, u) \in E^-$. In the latter case, another vertex v must exist such that $(w, v) \in E^- \cap E_T^-$ and, hence, $\{w, v\} = e^-(v)$. □

4.2 Backward Fireworks

A symmetrical concept of fireworks can be defined based on the edges $\{u, v\} = e^+(u)$ of a temporal clique $\mathcal{G} = (G, \lambda)$. All arguments developed in the context of forward fireworks can be adapted in a symmetrical way, so we will omit most of the details. First, we build a directed graph $G^+ = (V, E^+)$ such that $(u, v) \in E^+$ iff $\{u, v\} = e^+(u)$, except that, if $e^+(u) = e^+(v)$ for some u and v, only one of the arcs is included (chosen arbitrarily). E^+ induces a disjoint set of *in-trees*.

We transform G^+ into a disjoint set $\mathcal{T}^+ = (V, E_T^+)$ of *out-trees* as follows. Let E_T^+ be initialized as a copy of E^+. For every v with indegree at least 2 in G^+, let $(u_1, v), ..., (u_\ell, v)$ be its in-arcs with (u_ℓ, v) being the one with the *smallest* label. For every $i < \ell$, if u_i is a source vertex, then flip the direction of (u_i, v) in E_T^+ (*i.e.,* replace (u_i, v) by (v, u_i) in E_T^+); otherwise remove (u_i, v) from E_T^+. Let $\mathcal{T}^+ = (V, E_T^+)$ be the resulting set of out-trees $\mathcal{T}_1^+, ..., \mathcal{T}_k^+$.

Each out-tree in $\mathcal{T}^+$ contains only one source which we call a *collector*. The collector s of an out-tree can reach all of the other vertices in this tree by journeys that leave s through its edge $e^+(s)$, thereby guaranteeing that every other vertex that reaches s can subsequently reach all other vertices in the tree. Finally, we can build a temporal spanner $S_T^+ = \{\{u, v\} \in E : (u, v) \in \mathcal{T}^+\} \cup \{\{u, v\} \in E : u \text{ is a collector}\}$ which we call a *backward fireworks spanner*. An example of a backward fireworks spanner is given in Figure 10. The proofs of the following four lemmas and theorems are symmetrical to the arguments in Section 4.1.

Lemma 8. *The number of collectors in $\mathcal{T}^+$ is at most $n/2$*

Theorem 4. *S_T^+ is a temporal spanner of the temporal clique $\mathcal{G}$.*

Theorem 5. *Backward fireworks spanners have at most $3\binom{n}{2}/4 + O(n)$ edges.*

Lemma 9. *Every non-collector vertex v in $\mathcal{T}^+$ can reach another vertex v' in the same out-tree using a journey of length* at most two *that leaves v' through $e^+(v')$.*

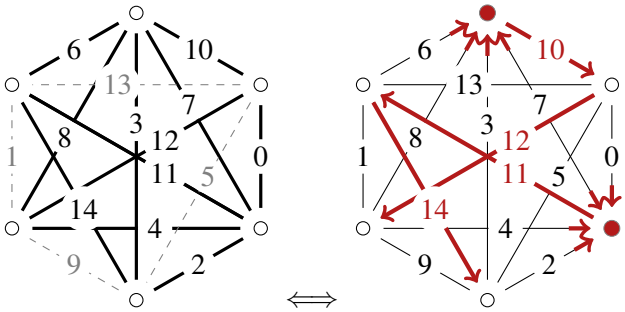


Figure 10: Example of a backward fireworks spanner (left) and the journeys from which it is constructed (right).

4.3 Bidirectional Fireworks

A forward fireworks spanner makes it possible to identify a subset of vertices, the *emitters*, such that every vertex can reach at least one emitter u through $e^-(u)$ and u can reach every other vertex afterwards *through a single edge*. Similarly, a backward fireworks spanner makes it possible to identify a subset of vertices, the *collectors*, such that every vertex can be reached by at least one collector v through $e^+(v)$ and v can be reached by every other vertex before this *through a single edge*. Combining both ideas, we can define a sparser spanner in which we do not need to include *all* of the edges incident to emitters and collectors, but only the edges *between* emitters and collectors (plus, of course, the edges used for reaching an emitter and for being reached by a collector).

Precisely, let $\mathcal{T}^-$ be the disjoint set of in-trees obtained during the construction of a forward fireworks spanner (see Figure 8), and let $\mathcal{T}^+$ be the disjoint set of out-trees obtained during the construction of a backward fireworks spanner. Let X^- be the set of emitters (one per in-tree in $\mathcal{T}^-$) and let X^+ be the set of collectors (one per out-tree in $\mathcal{T}^+$). The two sets can overlap, as a vertex may happen to be both an emitter in some tree in $\mathcal{T}^-$ and a collector in some tree in $\mathcal{T}^+$. Let $H = (X^- \cup X^+, E_H)$ be the graph such that $E_H = \{\{u, v\} \in E : u \in X^-, v \in X^+\}$; in other words, H is the subgraph of G that connects all emitters with all collectors. Finally, let $S = \{\{u, v\} : (u, v) \in \mathcal{T}^- \cup \mathcal{T}^+\} \cup E_H$. We call S a bidirectional fireworks spanner (or simply a *fireworks spanner*). An illustration is given in Figure 11.

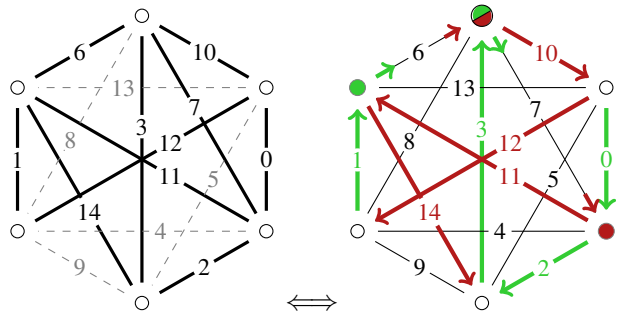


Figure 11: A bidirectional fireworks spanner (left) and the journeys from which it is constructed (right). The forward component and the emitters are depicted in (light) green; the backward component and the collectors are depicted in (darker) red. The top vertex is both an emitter and a collector.

Theorem 6. *S is a temporal spanner of $\mathcal{G}$.*

Proof. Every non-emitter vertex can reach at least one emitter u through $e^-(u)$. Every emitter can reach *all* collectors afterwards. Every vertex can be reached by a collector v through $e^+(v)$. □

Theorem 7. *Bidirectional fireworks spanners have at most* $\binom{n}{2}/2 + O(n)$ *edges.*

Proof. The number of edges in $\mathcal{T}^-$ and $\mathcal{T}^+$ is linear in n. By Lemma 6, the number of emitters is at most $n/2$, and so is the number of collectors by Lemma 8. Some vertices may be both emitter and collector; however, the number of edges is maximized when X^- and X^+ are disjoint, *i.e.,* H is a complete bipartite graph with $n/2$ vertices in each part. Thus, the spanner contains at most $n^2/4 = \binom{n}{2}/2 + n/4$ edges plus the $O(n)$ edges of $\mathcal{T}^-$ and $\mathcal{T}^+$. □

5 Recursing or Sparsifying

After applying the fireworks technique, one is left with a residual instance (a spanner of the input clique $\mathcal{G}$ on vertices V) made of all the edges between emitters X^- and collectors X^+, together with all the edges corresponding to the arcs of $\mathcal{T}^-$ and $\mathcal{T}^+$, these edges being denoted S^- and S^+. Depending on the properties of this residual instance, the algorithm may dismount it (and recurse on a subset of vertices), or it may continue to sparsify it using different techniques. The global algorithm is summarized in Figure 14. To simplify the notation in the following, we use variables $\mathcal{G}$ and V to refer to the residual instance in the current step of the recursion.

The following two cases are considered: Either $X^- \cup X^+ \neq V$ (Case 1) or $X^- \cup X^+ = V$ (Case 2).

Case 1 ($X^- \cup X^+ \neq V$)**.** In this configuration, at least one vertex v is neither emitter nor collector. By Lemma 7, there exists a journey of length at most two from v that arrives at some vertex $u \neq v$ through $e^-(u)$. Similarly, by Lemma 9, there is a journey of length at most two from some vertex $w \neq v$ to v, leaving w through $e^+(w)$. As a result, v is 2-hop dismountable (see Section 3). One can thus select the corresponding edges (at most four) for future inclusion in the spanner and recurse on $\mathcal{G}[V \setminus v]$; that is, re-apply the fireworks technique from scratch to this smaller graph. Repeating this strategy, either the recursion keeps entering Case 1 and dismounts the graph entirely, or it eventually enters Case 2.

Case 2 ($X^- \cup X^+ = V$)**.** Both X^- and X^+ have size at most $n/2$ (Lemma 6 and 8), thus if their union is V, then both sets must be disjoint and of size exactly $n/2$. As a result, the graph which connects vertices of X^- to vertices of X^+ (called H in Section 4) is a complete bipartite graph. In fact, H possesses even more structure; in particular, both S^- and S^+ are perfect matchings – by contradiction, if this is not the case, then at least one of the in-trees (out-trees) contains more than one edge, resulting in strictly less than $n/2$ emitters (collectors). Furthermore, every vertex is either an emitter or a collector, thus each of these edges connects an emitter with a collector, implying that the current residual instance $\mathcal{G}$ is H itself. Now, recall that every edge in S^- is locally minimum for the corresponding emitter (Lemma 5.5), and every edge in S^+ is locally maximum for the corresponding collector. We thus have the following stronger property.

Lemma 10. *If the minimum edge of an emitter is not also the minimum edge of the corresponding collector in* H*, then the residual instance is* 2*-hop dismountable. The same holds if the maximum edge of a collector is not also the maximum edge of the corresponding emitter in* H*.*

Proof. Let us prove this for minimum edges (a symmetrical argument applies for maximum edges). Consider an emitter u whose minimum edge $\{u, v\} = e^-(u)$ leads to collector v such that $\{u, v\} \neq e^-(v)$ in the bipartite graph. Then an edge with a smaller label exists between v and another emitter u' such that $\{u', v\} = e^-(v)$, which creates a 2-hop journey from u' to u arriving through $e^-(u)$, implying that u' can delegate its emissions to u. Moreover, emitters and collectors are disjoint, thus u' is not a collector. As a result, u' already delegates its receptions to a collector (through a single edge), thus u' is 2-hop dismountable. □

There are two subcases of Case 2 depending on whether or not the hypothesis of Lemma 10 holds.

Subcase 2.1 (H contains a 2-hop dismountable vertex v)**.** The algorithm 2-hop dismounts v, the corresponding edges (at most four) are selected for future inclusion in the spanner, and the algorithm recurses on $\mathcal{G}[V \setminus v]$.

Subcase 2.2 (The edges of the matchings S^- and S^+ are minimum (resp. maximum) *on both sides*)**.** An example of this subcase is given in Figure 12.

In summary, either the algorithm recurses until the input clique is fully dismounted (through Case 1 and Subcase 2.1), resulting in a spanner of size $O(n)$ (Lemma 3), or the *recursion stops* and the residual instance is sparsified further by a dedicated procedure for Subcase 2.2, described now.

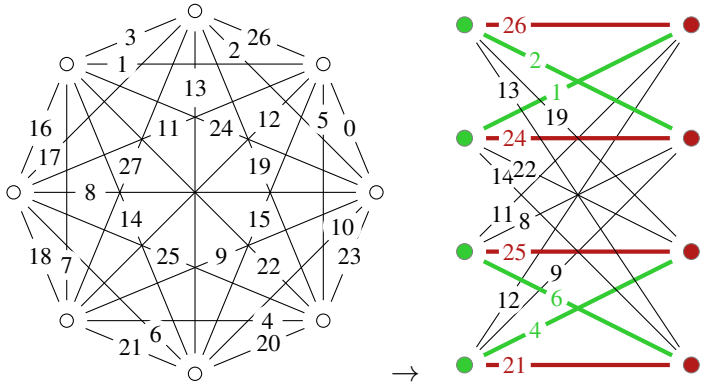


Figure 12: Temporal clique for which the output of the fireworks technique results in Subcase 2.2 in which every vertex is either an emitter or a collector and the edges of the matchings are minimum (*resp.* maximum) on both sides. The minimum edges are depicted in light green and maximum edges in dark red.

5.1 Sparsifying the Residual Instance

For simplicity, the sparsification of the residual instance is considered as a separate problem. The input is a labelled complete bipartite graph $B = (X^-, X^+, E_B)$ where X^- is the set of emitters, X^+ is the set of collectors, and the labels are inherited from $\mathcal{G}$. There are two perfect matchings S^- and S^+ in B such that the labels of the edges in S^- (resp. S^+) are minimum (resp. maximum) locally to both of their endpoints (Lemma 10). The objective is to remove as many edges as possible from E_B, while preserving S^-, S^+, and the fact that every emitter can reach *all* collectors by a journey. Indeed, these three properties ensure temporal connectivity of the graph (using the same arguments as in Theorem 6).

While both S^- and S^+ are matchings, our algorithm only exploits this property with respect to S^+ as follows.

Lemma 11. *In B, if an emitter u can reach another emitter v, then it can reach the collector w corresponding to v by adding the edge $(v, w) \in S^+$ to the journey from u to v.*

Proof. Edge $(v, w) \in S^+$, thus its label is locally maximum to v in B (Lemma 10). This implies that edge (v, w) may be added to any journey to v, resulting in a journey to w. Observe that, in the particular case that (u, w) is in S^-, then this edge is already in the journey from u to v (with the consequence that an edge is saved). □

This property makes it possible to reduce the task of reaching some collectors to that of reaching their corresponding emitters in S^+. It is however impossible for an emitter u to make a complete delegation to another emitter v, because the existence of a journey from u to v arriving through $e^-(v)$ would contradict the fact that S^- is also a matching. For this reason, when a journey from emitter u arrives at emitter v, some of v's edges have already disappeared. Nevertheless, the algorithm exploits such *partial* delegations, while paying extra edges for the missed opportunities (contained within a logarithmic factor). This is done by means of an *iterative* procedure called *layered delegations*, described in the remainder of this section. Note the term iterative, not recursive; from now on, the instance has a fixed vertex set and it is sparsified until the final bound is reached.

5.1.1 Layered Delegations

The algorithm proceeds by *eliminating* half of the emitters in each step j, while selecting a set S_j of edges for inclusion in the spanner, such that the eliminated emitters can reach all collectors using either single edges or indirect journeys through other emitters (partial delegations). The set of non-eliminated emitters at step j (called *alive*) is denoted by X_j^-, with $X_1^- = X^-$. The set of collectors X^+ is invariant over the execution. We denote by $k = n/2$ the initial degree of the emitters in B (one edge shared with each collector), and by $e^i(v)$ the edge with the i^{th} smallest label (label of *rank* i) locally to a vertex v, in particular $e^1(v) = e^-(v)$ and $e^k(v) = e^+(v)$.

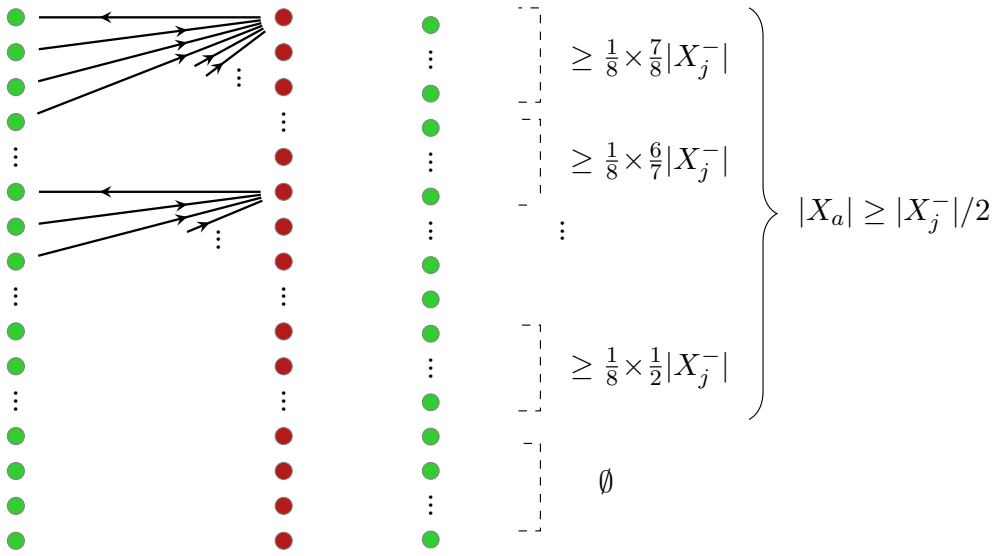
The k ranks are partitioned into subintervals of doubling size $\mathcal{I}_j = [2^{j+2} - 7, 2^{j+3} - 8]$, where j denotes the current step of the iteration, ranging from 1 to $\log_2 k - 3$. For simplicity, assume that k is a power of two; we explain below how to adapt the algorithm when this is not the case (see Remark 2). For example, if $k = 128$, then $\mathcal{I}_1 = [1, 8], \mathcal{I}_2 = [9, 24], \mathcal{I}_3 = [25, 56]$, and $\mathcal{I}_4 = [57, 120]$. Computation step j is made with respect to the subgraph $B_j = (X_j^-, X^+, E_j)$ where $E_j = \{e^i(v) \in E_B : i \in \mathcal{I}_j, v \in X_j^-\}$, namely the edges of the currently alive emitters, whose ranks are in the interval $\mathcal{I}_j$.

Lemma 12. *In each step j, X_j^- can be split into two sets X_a and X_b such that $|X_a| \geq |X_b|$ and every vertex in X_a can reach a vertex in X_b through a 2-hop journey (within B_j).*

Proof. This proof is illustrated in Figure 13 for the particular case that $j = 1$ (first step). We will show that the average degrees of collectors in B_j forces the existence of sufficiently many two-hop journeys among emitters. To start, observe that if a collector v shares an edge with d emitters in B_j (we say that these emitters *meet* at v), then the emitter whose edge with v has the largest label can be reached by the $d - 1$ other emitters through two-hop journeys. Thus, the proof proceeds by showing that, in each step j, the distribution of degrees over collectors forces the existence of sufficiently many such meetings among emitters. Here, the size of the first interval $\mathcal{I}_j$ matters, because if one starts with intervals of size only 2 or 4 (say), then the density of edges remains insufficient for the argument to apply, and starting with 8 (or any larger constant power of two) suffices, while not impacting the asymptotic cost, as we will see. Also observe that the doubling size of the rank intervals cancels out the halving size of X_j^- over the steps, leading to an average degree for collectors (in the considered range of ranks) that remains constant over the steps (namely 8).

The calculation relative to step j is itself based on an iterative argument that one should be careful not to confuse with the outer loop where j varies. Thus, keeping j fixed for the rest of the proof, X_a and X_b are built iteratively as follows: find the collector c with highest degree and add all the corresponding emitters to X_a except the one whose edge with c has largest label, which is added to X_b; eliminate these emitters and repeat until $X_a \geq X_j^-/2$; then add the remaining emitters to X_b. To see why this works, observe that the average degree of 8 for collectors implies that at least one collector has degree at least 8. In fact, this is true as long as the number of unprocessed emitters in X_j^- (i.e. not yet in X_a or X_b) is larger than $7/8 \cdot |X_j^-|$

(by the pigeon-hole principle). When $1/8$ of the emitters have been processed, we thus have at least $7/8$ of this $1/8$ fraction in X_a, i.e. X_a contains at least $7/8 \cdot 1/8 \cdot |X_j^-|$ (note that the emitters are not removed from X_j^-, just marked as processed and no longer counted for the degrees of collectors). An analogous argument implies that at least one collector has degree at least 7 as long as the fraction of unprocessed emitters in X_j^- remains above $6/8$, resulting in at least $6/7 \cdot 1/8 \cdot |X_j^-|$ more emitters in X_a when the next threshold is attained, and resulting (again, for the sake of illustration) in at least $5/6 \cdot 1/8 \cdot |X_j^-|$ more emitters when $5/8$ unprocessed emitters remain. By iterating this argument, the size of X_a ends up being a fraction of X_j^- at least equal to $1/8 \cdot (7/8 + 6/7 + 5/6 + 4/5 + 3/4 + 2/3 + 1/2) \simeq 0.66 \geq 0.5$. □



(a) Emitters "meeting" at collectors.

(b) Calculation of the size of X_a with respect to the set of emitters X_j^-.

Figure 13: Illustration of the method used in the proof of Lemma 12.

Remark 2. *The computation of X_a (described in the proof) proceeds by repeatedly considering the largest degree d of a collector and assigning $d-1$ of the corresponding emitters to X_a and one to X_b; it is therefore a greedy algorithm. The process is to be stopped whenever X_a reaches half the size of X_j^-. If X_a exceeds this threshold during step j, then some emitters can be arbitrarily transferred from X_a to X_b to preserve the fact that $|X_{j+1}^-|$ is a power of two. The case that $|X_1^-| = k$ is not a power of two is addressed similarly after the first iteration, in order to set the size of X_b to the highest power of two below k.*

How X_a and X_b are then used: When an emitter u in X_a reaches another emitter v in X_b, the corresponding journey arrives at v through some edge $e^i(v)$ with $i \in \mathcal{I}_j$. We say that u *partially delegates* its emissions to v in the sense that all collectors that v can reach after this time can be reached from u (Lemma 11), the other collectors being possibly no longer reachable from v after this time.

Lemma 13. *If an emitter u makes a partial delegation to v in step j, then the number of collectors that u may no longer reach through v is at most $2^{j+3} - 8$.*

Proof. This number is the largest value in the current interval; it corresponds to the largest rank of the edge through which the journey from u may have arrived at v. Every edge whose rank locally to v is larger than $2^{j+3} - 8$ can still be used and thus the corresponding collectors are still reachable. This also implies that all other edges incident to v (so with rank lower than $2^{j+3} - 8$) may have already disappeared. □

Lemma 13 implies the following.

Lemma 14. *In each step j, at most 2^{j+3} edges are selected relative to every eliminated emitter.*

Proof. Every eliminated emitter u delegates to some emitter v. This is done through a 2-hop journey (composed of 2 edges). Additionally, through this delegation, u may not be able to reach at most $2^{j+3}-8$ collectors, for which at most $2^{j+3}-8$ direct edges from u are selected. The total amount of edges used for this partial delegation is thus at most $2^{j+3}-6 \leq 2^{j+3}$. □

More globally, let J_j be the edges corresponding to all of the journeys used for partial delegation from vertices in X_a to vertices in X_b in step j, and D_j the union of edges between the emitters in X_a and the missed collectors. Let $S_j = J_j \cup D_j$. The algorithm thus consists of selecting all the edges in S_j for inclusion into the spanner. Then X^-_{j+1} is set to X_b and the iteration proceeds with the next step. The computation goes for j ranging from 1 to $\log_2 k - 3$, which leaves exactly *eight* final emitters alive. All the remaining edges of these emitters (call them S_{last}) are finally selected. Overall, the final spanner is the union of all selected edges, plus the edges corresponding to the two initial matchings, *i.e.,* $S = (\cup_j S_j) \cup S_{last} \cup S^- \cup S^+$.

Theorem 8. *S is a temporal spanner of the complete bipartite graph B and it has $\Theta(n \log n)$ edges.*

Proof. The key observation for establishing *validity* of the spanner is that eliminated emitters reach all collectors either directly or through an emitter that can still reach this collector *afterwards*. This property applies transitively (thanks to the disjoint and increasing intervals) until eight emitters remain, all the edges of which are selected for simplicity. Therefore, every initial emitter can reach all collectors. The rest of the arguments are the same as in the proof of Theorem 6: all vertices in the input clique can reach at least one emitter u through $e^-(u)$, and be reached by at least one collector v through $e^+(v)$.

Regarding the size of the spanner, in step j, $\frac{k}{2^j}$ emitters are eliminated and at most 2^{j+3} edges are selected for each of them (Lemma 14), amounting to at most $8k = 4n$ edges. The number of iterations is $\Theta(\log k) = \Theta(\log n)$. Finally, the sets S_{last}, S^-, and S^+ each contain $\Theta(n)$ edges (and S^+ is actually included in S_{last}). □

Theorem 9. *Simple temporal cliques always admit $O(n \log n)$-sparse spanners.*

Proof. In each recursion of the global algorithm, either the residual instance of the fireworks procedure is 2-hop dismountable and the algorithm recurses on a smaller instance induced by a removed vertex, after selecting a *constant* number of edges, or the algorithm computes a $\Theta(n \log n)$-sparse spanner of the residual instance through the layered delegation process. Let n_1 be the number of times the graph is 2-hop dismounted and $n_2 = n - n_1$ be the number of vertices of the residual instance when the layered delegation process begins (if applicable, 0 otherwise). The 2-hop dismounting contributes $\Theta(n_1)$ edges, and layered delegation contributes $\Theta(n_2 \log n_2)$ edges if it is performed, so the resulting spanner has $O(n \log n)$ edges. □

6 Time Complexity and Summary of the Algorithm

This short section reviews the cost in time of the main components involved in the algorithm. This discussion is by no means a detailed analysis, its purpose is rather to sustain the claim that the overall algorithm runs in polynomial time. The input to the algorithm is a temporal clique $\mathcal{G} = (G, \lambda)$, where $G = K_n$ is the complete graph on n vertices, and λ is a simple labelling of the edges, which may be represented as a permutation π of $\{0, 1, 2, ..., \binom{n}{2} - 1\}$. The global algorithm is portrayed in Figure 14. Observe that whenever the algorithm recurses, the number of vertices is reduced by one, and in each recursion the fireworks

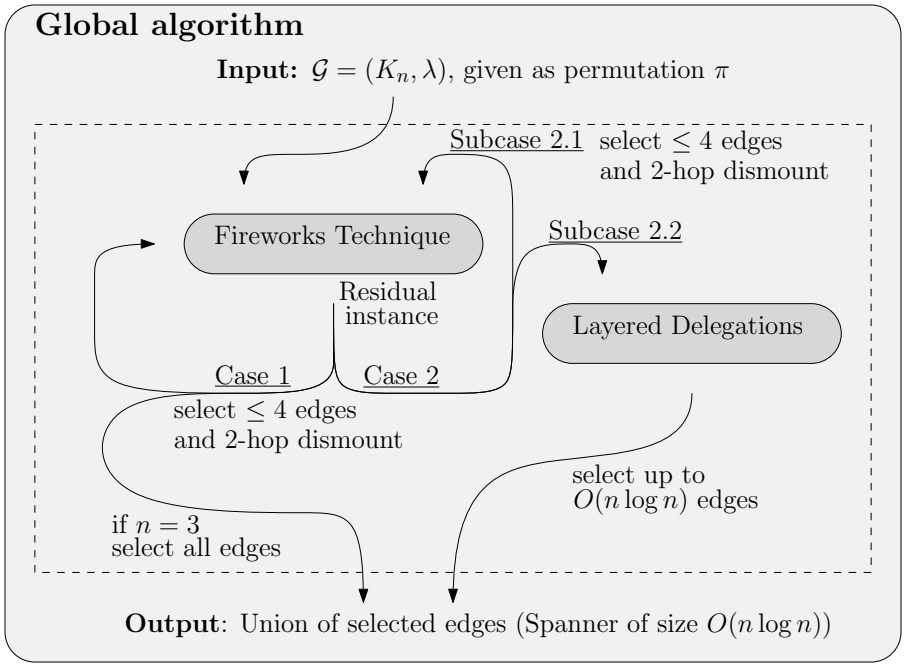


Figure 14: The global algorithm using the fireworks technique, dismounting, and layered delegations.

process is run twice (forward and backward), possibly followed by the layered delegation processes, in which case the algorithm subsequently terminates. The fireworks process will thus run less than $2n$ times and the delegation process at most one time. The fireworks process first identifies the edges which are minimum (maximum) for at least one vertex. Then, it transforms this structure by means of a set of local operations consisting of flipping edges (at most once) or discarding them. As for layered delegations, the main operation is the composition of the delegation sets X_a and X_b, which is done a logarithmic number of times by a greedy procedure whose main operation is to examine the degrees of all collectors to detect the local maximum among their labels. In light of these observations, we hope that it is clear to the reader that the overall running time is polynomial.

7 Open Questions

Despite extensive computer search, we were not able to find an instance (out of hundred of millions of instances of different sizes) which does not admit a spanner of either $2n - 3$ or $2n - 4$ edges. The latter is actually a lower bound, as a classical result in gossip theory (see *e.g.,* Facts F29 through F32 in [15]) can be adapted to rule out the possibility that a graph with single labels is temporally connected with less than $2n - 4$ edges. Interestingly, we found many instances which are neither pivotable (see Section 2.3) nor k-hop dismountable (see Section 3), and yet admit a spanner of size $2n - 3$ or $2n - 4$. This suggests further investigation to understand the structure of simple temporal cliques. In particular:

Open question 1. *Do simple temporal cliques always admit temporal spanners of size* $\Theta(n)$*?*

Open question 2. *Do simple temporal cliques always admit temporal spanners of size* $2n - 3$*?*

A first step towards answering these questions might be to show that this is true in a probabilistic sense, which seems to be the case. More precisely, let a random simple temporal clique be built by assigning to

every edge e_i the label $\pi[i]$, where π is a random permutation of $\{0, 1, \ldots, \binom{n}{2} - 1\}$. Preliminary experiments suggest that random temporal cliques may asymptotically almost surely (*a.a.s.*) be both pivotable and fully-dismountable (*i.e.*, the probability that this is the case tends to 1 as n goes to infinity). However, our experiments also suggest that there exist instances of arbitrary sizes which are neither pivotable nor dismountable.

Open question 3. *Do random simple temporal cliques a.a.s. admit spanners of at most $2n - 3$ edges?*

Open question 4. *Are random simple temporal cliques a.a.s. fully (k-hop) dismountable?*

Open question 5. *Are random simple temporal cliques a.a.s. pivotable?*

On the deterministic side again, a natural question is whether a more general class than complete graphs deterministically admits sparse spanners, and what is the role of density. The family of counter-examples from Axiotis and Fotakis [5] have asymptotic density $1/9$, which leaves a significant gap between this family and complete graphs.

Open question 6. *Is there a larger class of dense graphs than complete graphs that always admit $o(n^2)$-sparse temporal spanners?*

Finally, we know that some graphs exist which are neither pivotable nor dismountable (see Figure 7 in Section 3.3), and some experiments that we have conducted suggest that arbitrarily large graphs that are both non-pivotable and non-dismountable may exist. However, the characterization of an infinite family with these properties is left open in this paper.

Open question 7. *Is there a canonical way to construct graphs of arbitrary size which are neither pivotable nor dismountable?*

8 Concluding remarks

In this paper, we established that sparse temporal spanners always exist in temporal cliques, proving constructively that one can find $O(n \log n)$ edges that suffice to preserve temporal connectivity. Our results hold for non-strict journeys with single or multiple labels on each edge, and strict journeys with single or multiple labels on each edge with the property that there is a subset of locally exclusive single labels. Our results give the first positive answer to the question of whether any class of dense graphs always has sparse temporal spanners.

To prove our results, we introduced several techniques (pivotability, delegation, dismountability and k-hop dismountability, forward and backward fireworks, partial delegation, and layered delegations), all of which are original and some of which might be of independent interest. Whether some of these techniques can be used for more general classes of graphs is an open question. Delegation and dismounting rely explicitly on the graph being complete; however, refined versions of these techniques like partial delegation might have wider applicability.

One of the main open questions is whether sparse spanners exist in more general classes of dense graphs, keeping in mind that some dense graphs are unsparsifiable. Another is whether a better density than $O(n \log n)$ can be achieved in the particular case of temporal cliques, and more precisely can spanners of size $O(n)$ always be found in this case. Experiments that we have conducted suggest that spanners with $O(n)$ edges might always exist. At a deeper level, all these questions pertain to identifying and studying analogues of spanning trees in temporal graphs, which do not enjoy the same matroid structure as in standard graphs and seem much more complex.

Acknowledgements. We thank the referees for their careful reading and constructive comments which significantly improved the presentation of these results.

References

[1] Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *Data Structures and Algorithms*. Addison-Wesley, 1983.

[2] Eleni C. Akrida, Leszek Gasieniec, George B. Mertzios, and Paul G. Spirakis. The complexity of optimal design of temporally connected graphs. *Theory of Computing Systems*, 61(3):907–944, 2017.

[3] Sunil Arya, David M Mount, and Michiel Smid. Dynamic algorithms for geometric spanners of small diameter: Randomized solutions. *Computational Geometry*, 13(2):91–107, 1999.

[4] B. Awerbuch and S. Even. Efficient and reliable broadcast is achievable in an eventually connected network. In *3rd Symposium on Principles of Distributed Computing (PODC)*, pages 278–281, 1984.

[5] Kyriakos Axiotis and Dimitris Fotakis. On the size and the approximability of minimum temporally connected subgraphs. In *43rd International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 149:1–149:14, 2016.

[6] Bin-Minh Bui-Xuan, Afonso Ferreira, and Aubin Jarry. Computing shortest, fastest, and foremost journeys in dynamic networks. *International Journal of Foundations of Computer Science*, 14(2):267–285, 2003.

[7] Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. Time-varying graphs and dynamic networks. *International Journal of Parallel, Emergent and Distributed Systems*, 27(5):387–408, 2012.

[8] Arnaud Casteigts, Anne-Sophie Himmel, Hendrik Molter, and Philipp Zschoche. The computational complexity of finding temporal paths under waiting time constraints. *CoRR*, abs/1909.06437, 2019.

[9] Arnaud Casteigts, Joseph G. Peters, and Jason Schoeters. Temporal cliques admit sparse spanners. In *46th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 134:1–134:14, 2019.

[10] Paul Chew. There is a planar graph almost as good as the complete graph. In *2nd Symposium on Computational Geometry*, pages 169–177, 1986.

[11] Giuseppe Antonio Di Luna, Stefan Dobrev, Paola Flocchini, and Nicola Santoro. Distributed exploration of dynamic rings. *Distributed Computing*, 33(1):41–67, 2020.

[12] Michael Elkin. A near-optimal distributed fully dynamic algorithm for maintaining sparse spanners. In *26th ACM Symposium on Principles of Distributed Computing*, pages 185–194, 2007.

[13] Thomas Erlebach, Michael Hoffmann, and Frank Kammer. On temporal graph exploration. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *42nd International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 9134 of *LNCS*, pages 444–455. Springer, 2015.

[14] Lee-Ad Gottlieb and Liam Roditty. Improved algorithms for fully dynamic geometric spanners and geometric routing. In *19th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 591–600, 2008.

[15] H.A. Harutyunyan, A.L. Liestman, J.G. Peters, and D. Richards. Broadcasting and gossiping in communication networks. In J.L. Gross, J. Yellen, and P. Zhang, editors, *Handbook of Graph Theory, Second Edition*, chapter 12.2, pages 1477–1494. CRC Press, 2013.

[16] Petter Holme and Jari Saramäki. *Temporal Networks*. Springer, 2013.

[17] David Kempe, Jon Kleinberg, and Amit Kumar. Connectivity and inference problems for temporal networks. *Journal of Computer and System Sciences*, 64(4):820–842, 2002.

[18] George B. Mertzios, Hendrik Molter, Rolf Niedermeier, Viktor Zamaraev, and Philipp Zschoche. Computing maximum matchings in temporal graphs. In *37th International Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 154 of *LIPIcs*, pages 27:1–27:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[19] Othon Michail and Paul G Spirakis. Elements of the theory of dynamic networks. *Communications of the ACM*, 61(2):72–72, 2018.

[20] Giri Narasimhan and Michiel Smid. *Geometric Spanner Networks*. Cambridge Univ. Press, 2007.

[21] Ariel Orda and Raphael Rom. Shortest-path and minimum-delay algorithms in networks with time-dependent edge-length. *Journal of the ACM (JACM)*, 37(3):607–625, 1990.

[22] David Peleg and Alejandro A Schäffer. Graph spanners. *Journal of Graph Theory*, 13(1):99–116, 1989.

[23] Tiphaine Viard, Matthieu Latapy, and Clémence Magnien. Computing maximal cliques in link streams. *Theoretical Computer Science*, 609:245–252, 2016.

[24] Richard M. Wilson. Decompositions of complete graphs into subgraphs isomorphic to a given graph. In *5th British Combinatorial Conference*, pages 647–659. Congressus Numerantium XV, 1975.

[25] Philipp Zschoche, Till Fluschnik, Hendrik Molter, and Rolf Niedermeier. The complexity of finding small separators in temporal graphs. In *43rd International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 45:1–45:17, 2018.