Temporal Cliques Admit Sparse Spanners

Arnaud Casteigts
LaBRI, Université de Bordeaux, CNRS, Bordeaux INP, France
arnaud.casteigts@labri.fr

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

Jason Schoeters
LaBRI, Université de Bordeaux, CNRS, Bordeaux INP, 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 positive integer labels. These labels can be seen as discrete times when the edge is present. Such a labeled graph $\mathcal{G} = (G, \lambda)$ is said to be *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 (STOC 2000) asked whether, given such a temporal graph, a *sparse* subset of edges can always be found whose labels suffice to preserve temporal connectivity – a *temporal spanner*. Axiotis and Fotakis (ICALP 2016) answered negatively by exhibiting a family of $\Theta(n^2)$-dense temporal graphs which admit no temporal spanner of density $o(n^2)$. The natural question is then whether sparse temporal spanners always exist in *some* classes of dense graphs.

In this paper, we answer this question affirmatively, by showing that if the underlying graph G is a complete graph, then one can always find temporal spanners of density $O(n \log n)$. The best known result for complete graphs so far was that spanners of density $\binom{n}{2} - \lfloor n/4 \rfloor = O(n^2)$ always exist. Our result is the first *positive* answer as to the existence of $o(n^2)$ sparse spanners in adversarial instances of temporal graphs since the original question by Kempe et al., focusing here on complete graphs. The proofs are constructive and directly adaptable as an algorithm.

2012 ACM Subject Classification Theory of computation → Sparsification and spanners; Theory of computation → Dynamic graph algorithms

Keywords and phrases Dynamic networks, Temporal graphs, Temporal connectivity, Sparse spanners

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.134

Category Track C: Foundations of Networks and Multi-Agent Systems: Models, Algorithms and Information Management

Related Version A full version of this paper is available at `https://arxiv.org/abs/1810.00104` [7].

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

Acknowledgements We thank Cyril Gavoille for advice on the presentation of these results.

1 Introduction

The study of highly dynamic networks has gained interest lately, motivated by emerging technological contexts (e.g. vehicular networks, wireless sensors, 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 central role. In these graphs, the properties of interest are often defined over the 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 [6], 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* [5]), 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 journey), both cases being carefully discussed in this paper. Temporal connectivity was considered in an early paper by Awerbuch and Even [3] (1984), and systematically studied from a graph-theoretical point of view in the early 2000's in a number of seminal works including Kempe, Kleinberg, and Kumar [13], and Bui-Xuan, Ferreira, and Jarry [5] (see also [16] for an early study of graphs with time-dependent delays on the edges). More recently, temporal connectivity has been the subject of several algorithmic studies, such as [1] and [4] (discussed below), and [18] and [19], which consider algorithms for computing structures related to temporal connectivity. Broad surveys on these topics can be found, e.g. in [6, 11, 14], although the list is non-exhaustive and the literature is rapidly evolving.

1.1 Sparse Temporal Spanners and Related Work

In the last section of [13] (conference version [12]), Kempe, Kleinberg, and Kumar ask

> "Given a temporally connected network $G = (V, E)$ on n nodes, is there a set $E' \subseteq E$ consisting of $O(n)$ edges so that the temporal network on the subgraph (V, E') is also temporally connected? In other words, do all temporal networks have sparse subgraphs preserving this basic connectivity property?"

Here, Kempe et al. consider a model where each edge has a single label, thus the edges are identified with their labels, but the discussion is more general. What they are asking, essentially, is whether an analogue of spanning tree exists for temporal networks when the labels are *already fixed.* They answer immediately (and negatively) for the particular case of $O(n)$ density, by showing that hypercubes labeled 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 [4]. The proof in [4] 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. [1] show that, if the underlying graph G is complete 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 the asymptotic density unchanged). Akrida et al. [1] 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 [4] and [1] 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 network seems to be significantly different from its classical (i.e., non-temporal) version, whether this version considers a static graph (see e.g. [8, 15, 17]) or the current network topology of an updated dynamic graph (see e.g. [2, 9, 10]). The essential difference is that spanning trees always exist in standard (connected) graphs, thus one typically focuses on the tradeoff between the density of a solution and a quality parameter like the *stretch factor*, rather than to 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 spanners can be computed in polynomial time. The case of strict journeys requires more discussion. Kempe et al. observed in [13] 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 can exist, 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 labeling to avoid this pathological situation. Thus, in this case, we require that a subset 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 [1] (although essentially equivalent) and eliminates the distinction between strict and non-strict journeys. Under this restriction, we establish that all temporal graphs whose underlying graph is complete admit an $O(n \log n)$-sparse temporal spanner. Moreover, if the restricted labeling is given, then the spanner can be computed in polynomial time. (The problem of deciding whether a general labeling admits such a sub-labeling is not discussed here; it might be an interesting problem on its own, possibly computationally hard.)

Our proofs are constructive. 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 less feasible journeys than the original instance, but all of these journeys correspond to valid journeys in the original instance, so that a temporal spanner computed in the transformed instance is valid in the original instance. As a result, the main algorithm takes as input a complete graph $\mathcal{G}$ with single, locally distinct labels, and computes an $O(n \log n)$-sparse spanner of $\mathcal{G}$ in polynomial time. This algorithm is based on several original techniques, which we think may be of interest for other problems related to temporal connectivity.

In summary, our results 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 [4] 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).

The paper is organized as follows. In Section 2, we define the model and notations, and describe the one-way reductions that allows us to concentrate subsequently on single (and distinct) labels. We also mention a technique from [1] and we introduce a basic technique called *pivoting* which is a natural analogue of Kosaraju's algorithm for temporal graphs. In Section 3 we introduce the main concepts used in the rest of the paper, namely *delegation*, *dismountability*, and *k-hop dismountability*, whose purpose is to recursively self-reduce the problem to smaller graph instances. While these technique alone fail in some cases, we extend them and combine them 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* and results in a spanner of density (essentially) $\binom{n}{2}/2$. It is sparsified further down to $O(n \log n)$ by exploiting a particular dichotomy in the structure of the residual instance. Due to space limitation, several proofs are omitted. They can be found in the full version [7].

2 Definitions and Basic Results

2.1 Model and definitions

Let $G = (V, E)$ be an undirected graph 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, all 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 labeling 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 $1 \leq i < k$, $\{u_i, u_{i+1}\} \in E$, $t_i \in \lambda(\{u_i, u_{i+1}\})$ and $t_{i+1} \geq t_i$. Strict temporal path (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*. Interestingly, 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 $\mathcal{G} = ((V, E), \lambda)$ as a temporal graph $\mathcal{G}' = ((V', E'), \lambda')$ such that $V = V'$, $E' \subseteq E$ and $\lambda' : E' \to 2^{\mathbb{N}}$ with $\lambda'(e) \subseteq \lambda(e)$ for all $e \in E'$. We call $\mathcal{G}'$ a *valid* spanner if it is temporally connected. Observe that, if $\mathcal{G}$ is simple, then spanners are fully determined by the chosen subset of edges $E' \subseteq E$ (as in the above citation from [13]). Thus, in such cases, we say that E' itself *is* the spanner. Many of these notions are analogous to the ones considered in [1, 4, 13], although they are not referred to as "spanners" in these works.

Finally, when the underlying graph G is a complete graph, we call $\mathcal{G}$ a temporal clique. An example of a (valid) temporal spanner of a *simple temporal clique* is shown in Figure 1.

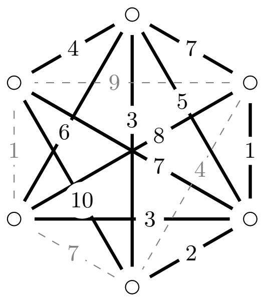

Figure 1 Example of a simple temporal clique and one of its temporal spanners (edges in bold). This spanner is not minimum (nor even minimal) and the reader may try to remove further edges.

2.2 Generality of simple labelings

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* labeling λ into a clique $\mathcal{H} = (G, \lambda_H)$ with *simple* labeling 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 fine because our result is *positive* on $\mathcal{H}$.) The reduction proceeds in two steps: (1) For every edge e, restrict $\lambda(e)$ to a single label chosen arbitrarily; and (2) Whenever k adjacent edges have identical label l, then all labels $l' > l$ are shifted to $l' + k$ and the k labels are assigned a unique value in the interval $[l, l + k]$

(again, arbitrarily). It should not be difficult to see that this reduction produces a simple clique $\mathcal{H}$, such that if a journey exists in $\mathcal{H}$, then the same sequence of edges allows for a (possibly non-strict) journey in $\mathcal{G}$.

As for the case that strict journeys are the only ones allowed, as explained above, we consider that a *simple* sub-labeling of λ exists and is identified prior to the computation. Here, it is even more direct that any journey based on the sub-labeling only is a fortiori available in the complete instance. Based on these arguments, the rest of the paper focuses on simple temporal cliques, sometimes dropping the adjective "simple".

2.3 Preliminary techniques

The best approach so far for sparsifying simple temporal cliques is that of Akrida et al. [1], who prove that one can always remove $\lfloor n/4 \rfloor$ edges without breaking temporal connectivity as follows. (Their paper has other significant contributions.) First, they show 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 removals is confined to each subclique. In the full version of the present paper [7], we improve this technique to remove a constant fraction of $\lfloor n^2/12 \rfloor$ edges. However, we consider as unlikely that such purely *structural* techniques could sparsify a graph to $o(n^2)$ edges. The techniques that we develop here are completely different.

Another natural approach that one might think of is discussed in the full version of this paper, inspired by Kosaraju's principle for testing strong connectivity in a directed graph. This principle relies on finding a vertex that all of 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 back *after* time t. The union of the tree of (incoming) journeys towards p and the tree of (outgoing) journeys from p is a temporal spanner with at most $2(n-1)$ edges. Unfortunately, pivot vertices may not exist, even in temporal cliques. Both possibilities (positive and negative) are shown in Figure 2. A generic construction to build arbitrarily large non-pivotable graphs is proposed in the full version of this paper.

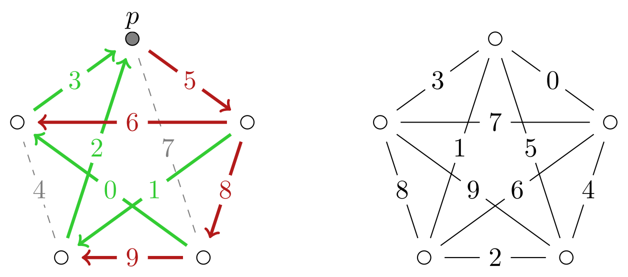

Figure 2 Examples of pivotable graph (left) and non-pivotable graph (right). The (light) green edges in the pivotable graph belong to the tree of incoming journeys to pivot vertex p (with $t = 4$); the (darker) red edges belong to the tree of outgoing journeys; the dashed edges belong to neither.

3 Delegation and Dismountability

This section introduces a number of basic techniques which are subsequently refined and adapted in Sections 4 and 5 in the main algorithm. Given a vertex v, write $e^-(v)$ for the edge with smallest label incident with v, and $e^+(v)$ analogously for the largest label.

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

Fact 1 applies because the underlying graph G is complete. This fact 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 (if need be, by a direct edge) *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 (collections) to a vertex w if w can be reached by all the other vertices (if need be, by a direct edge) *before* interacting with u, so u does not need its other edges to be reached by the other vertices.

This type of delegation suggests an interesting technique to obtain 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 $\{u, w\} = e^+(w)$. The idea of dismountability is to select $e^-(v)$ and $e^+(w)$ for inclusion in a temporal spanner for $\mathcal{G}$ and then reduce the computation to finding a temporal spanner in the smaller clique $\mathcal{G}[V \setminus u]$. We state this more formally in the following theorem.

▶ **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 with v in S' have a larger label than $\{u, v\}$, thus u can reach all the vertices in $\mathcal{G}$ through v and 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 is a two-vertex graph with a single edge. An example of such dismountable graph is given in Figure 3.

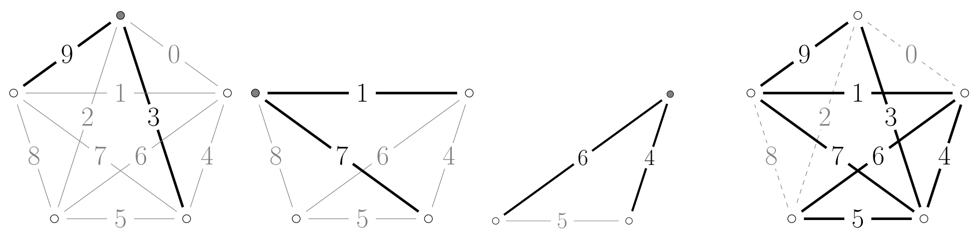

Figure 3 Example of a fully dismountable graph and the resulting spanner.

▶ **Fact 2** (Spanners based on dismountability). *If a graph can be fully dismounted, then the union of the pairs of edges involved in all steps of the recursion, plus the last edge forms a temporal spanner. There are $n - 2$ steps, so this spanner has $2(n - 2) + 1 = 2n - 3$ edges.*

Unfortunately, one can design arbitrarily large temporal cliques which are not fully dismountable (we give a generic construction in the full version of this paper). Yet, techniques derived from dismountability are at the core of our algorithm. To start, the concept of dismountability can be generalized to *multi-hop* journeys. The key observation is that a multi-hop journey may exist from a vertex u to another vertex v, say through vertices $u = u_0, u_1, \ldots, u_k = v$ such that $\{u_{k-1}, u_k\} = e^-(v)$, despite the fact that $\{u_{i-1}, u_i\} \neq e^-(u_i)$ for some i. Indeed, it is sufficient that the *last* edge of a journey from u to v is $e^-(v)$ 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. Thus, 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 $e^+(w)$.

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 (with positive impact). We can then extend Fact 2 to k-hop dismountability as follows.

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

Unfortunately, again, there exist arbitrarily large graphs which are not k-hop dismountable for any k (see the full version of this paper [7]). Nonetheless, k-hop dismountability is one of the components of the more sophisticated techniques in Sections 4 and 5.

4 The Fireworks Technique

In this section, we present an algorithm called *fireworks*, which exploits delegations among vertices in a more subtle way than dismountability. In particular, we take advantage of *one-sided delegations*, in which a vertex may be able to delegate 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 we exploit in Section 5 to obtain $O(n \log n)$-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 others vertices directly, most of their edges being consequently eliminated. 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 2.** *Directed paths in G^- correspond to journeys in $\mathcal{G}$.*

By construction, E^- induces a disjoint set of *out-trees* (one source, possibly several sinks). We transform E^- into a disjoint set $\mathcal{T}^- = (V, E^-_T)$ of *in-trees* (one sink, possibly several sources) as follows, see also Figure 4 for an illustration. Let E^-_T be initialized as a copy of E^-. For every v with outdegree at least 2 in E^-, let $(v, u_1), \ldots, (v, u_\ell)$ 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 $\mathcal{T}^-_1, \ldots, \mathcal{T}^-_k$ (containing possibly more in-trees than the number of initial out-trees).

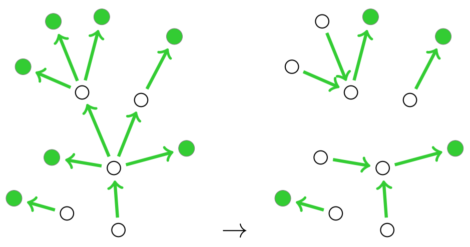

Figure 4 Example of transformation from a disjoint set of out-trees (V, E^-) to a disjoint set of in-trees (V, E^-_T). The colored vertices represent sink vertices.

▶ **Fact 4.** *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 with a sink s corresponds to $e^-(s)$.*

Fact 4.1 follows from Lemma 2 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 $\mathcal{G}$. 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, a delegation only requires that the label of the last hop of a journey be locally minimum, and that is the case here (Fact 4.5).

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 3.** *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 $\mathcal{T}^-_i \in \mathcal{T}^-$ (Fact 4.4), and at most $n/2$ in-trees, each having at least 2 vertices (Fact 4.3). ◀

We are now ready to define a temporal spanner based on $\mathcal{T}^-$, which consists of the union of all edges involved in an in-tree and all edges incident with 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 4.** *S^-_T is a temporal spanner of $\mathcal{G}$.*

Proof. By Fact 4, 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 cover*. An example is given in Figure 5, the corresponding journeys being depicted on the left side.

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

Before moving to Section 4.2, we establish a small technical lemma that will be used in Section 5, but is worth being stated here as it pertains to the in-trees.

▶ **Lemma 6.** *Every non-emitter vertex v can reach a vertex v' in the same in-tree $\mathcal{T}^-_i$ (emitter or not) using a journey of length* at most two *that arrives at v' through $e^-(v')$.*

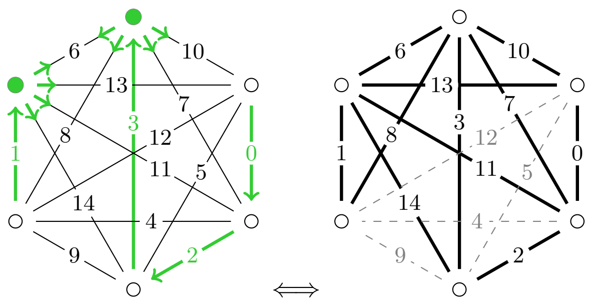

Figure 5 Example of forward fireworks cover and the resulting spanner.

4.2 Backward Fireworks

A symmetrical concept of fireworks can be defined based on the edges $\{u, v\} = e^+(v)$ of a temporal clique $\mathcal{G} = (G, \lambda)$. The above arguments can be adapted in a symmetrical way. First, we build a directed graph $G^+ = (V, E^+)$, which is a disjoint set of *in-trees*. A symmetrical transformation to the above converts this set into a disjoint set $\mathcal{T}^+ = (V, E^+_T)$ of *out-trees*, each with 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 vertex that reaches s can reach all other vertices in the tree.

▶ **Lemma 7.** *The number of collectors in $\mathcal{T}^+$ is at most $n/2$*

We build a temporal spanner $S^+_T = \{\{u, v\} : (u, v) \in \mathcal{T}^+\} \cup \{\{u, v\} : u \text{ is a collector}\}$ which we call a *backward fireworks cover*, and prove the following results by symmetrical arguments to the ones in Subsection 4.1.

▶ **Theorem 8.** *S^+_T is a temporal spanner of the temporal clique $\mathcal{G}$.*

▶ **Theorem 9.** *Backward fireworks covers have at most $3\binom{n}{2}/4 + O(n)$ edges.*

An example of a backward fireworks cover is given in the full version. Finally, we establish a symmetrical property to the one in Lemma 6, to be used later in Section 5.

▶ **Lemma 10.** *Every non-collector vertex v can be reached by a vertex v' in the same out-tree $\mathcal{T}^+_i$ (collector or not) using a journey of length* at most two *that leaves v' through $e^+(v')$.*

4.3 Bidirectional Fireworks

A forward fireworks cover 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 cover 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 that only includes 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 cover (see Figure 4), and let $\mathcal{T}^+$ be the disjoint set of out-trees obtained during the construction of a backward fireworks cover. 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 $\mathcal{T}^+$, which is not a problem. 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 cover (or simply a *fireworks cover*).

▶ **Theorem 11.** *S is a temporal spanner of $\mathcal{G}$.*

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

▶ **Theorem 12.** *Bidirectional fireworks covers have at most $\binom{n}{2}/2 + O(n)$ edges.*

5 Recursing or Sparsifying Further

After applying the fireworks technique, one is left with a residual instance (or spanner) 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^+ for simplicity. As we will see, the algorithm may recurse several times due to dismountability, thus it is worth mentioning that variables $\mathcal{G}$ and V refer to the instance of the current recursion. The algorithm considers two cases, depending on the outcome of the fireworks procedure. 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 6, 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 10, 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 then recurse on $\mathcal{G}[V \setminus v]$, i.e., re-apply the fireworks technique from scratch. Then, either the recursion keeps entering Case 1 and dismounting the graph completely, or it eventually enters Case 2.

▶ **Case 2 ($X^- \cup X^+ = V$).** Both X^- and X^+ have size at most $n/2$ (Lemma 3 and 7), 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 all vertices in X^- with all vertices in X^+ (called H in Section 4) is a complete bipartite graph. In fact, H possesses even more structure. Firstly, both S^- and S^+ are perfect matchings – by contradiction, if this is not the case, then at least one of the in-tree (out-tree) contains more than one edge, resulting in strictly less emitters (collectors) than $n/2$. Furthermore, every vertex is either an emitter or a collector, thus each of these edges connects an emitter with a collector, implying that the residual instance actually is H itself. Now, recall that every edge in S^- is locally minimum for the corresponding emitter (based on Fact 4.5), and every edge in S^+ is locally maximum for the corresponding collector. We then have the following stronger property.

▶ **Lemma 13.** *If the minimum edge of an emitter is not also minimum for 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 maximum for the corresponding emitter in H.*

Lemma 13 implies that either a vertex v is 2-hop dismountable and the algorithm can recurse as in Case 1, or the edges of the matchings are minimum (resp. maximum) *on both sides.* (An example of the latter case is given in the long version [7].)

In summary, either the algorithm recurses until the input clique is fully dismounted (through Case 1 or Case 2), resulting in an $O(n)$-dense spanner (Fact 3), or the *recursion stops* and the residual instance is sparsified further by a dedicated procedure, described now.

5.1 Sparsifying the Residual Instance

For simplicity, the sparsification of the residual instance is considered as a separate problem. The input is a labeled 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 13). 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 11).

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

▶ **Fact 5.** *If an emitter can reach another emitter, then it can reach the corresponding collector by adding to its journey the corresponding edge of S^+.*

This property makes it possible to reduce the task of reaching some collectors to that of reaching the corresponding emitter 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 over the remaining 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.

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, so that the eliminated emitters can reach all collectors by a mixture of direct edges and 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. 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 14.** *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. The proof is in the full version [7] (together with an illustration). The main idea is to show that the average degrees of collectors in B_j forces the existence of sufficiently many two-hop journeys among emitters. ◀

▶ Remark 15. The computation of X_a 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 can reach 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 *de facto* be reached from u (through v), the other collectors being possibly no longer reachable from v after this time. Thus, the delegation is *partial.*

▶ **Lemma 16.** *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. All the edges whose rank locally to v is larger than $2^{j+1} - 2$ can still be used and thus the corresponding collectors are still reachable. (In fact, the collector corresponding to the edge with last index in $\mathcal{I}_j$ locally to v can also be considered as reached, but this is a detail.) ◀

A partial delegation from u to v in step j implies the removal of u from the set of emitters, the selection of the two edges of the journey from u to v, and the selection of at most $2^{j+3} - 8$ direct edges between u and the missed collectors. This implies the following fact.

▶ **Fact 6.** *In each step j, at most 2^{j+3} edges are selected relative to every eliminated emitter.*

More globally, let J_j be the edges used in all the delegation journeys from vertices in X_a to vertices in X_b in step j, and D_j the union of direct edges towards 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 17.** *S is a temporal spanner of the complete bipartite graph B and it is made of $O(n \log n)$ 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 11: 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 (Fact 6), 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}). ◀

▶ **Corollary 18.** *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 resulting spanner has $\Theta(n_1) + \Theta(n_2 \log n_2) = O(n \log n)$ many edges. ◀

▶ Remark 19. The running time of the algorithm is polynomial (see the full version).

6 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 (pivoting, delegation, dismounting and k-hop dismounting, 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.

An open question is whether sparse spanners always exist in more general classes of dense graphs, keeping in mind that some dense graphs are unsparsifiable. Another question is whether a better density than $O(n \log n)$ could be obtained in the particular case of temporal cliques, in particular $O(n)$-dense spanners. 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.

References

1 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.

2 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.

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

4 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.

5 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.

6 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.

7 Arnaud Casteigts, Joseph G Peters, and Jason Schoeters. Temporal Cliques admit Sparse Spanners. *arXiv preprint*, 2019. `arXiv:1810.00104`.

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

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

10 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.

11 Petter Holme and Jari Saramäki. *Temporal Networks*. Springer, 2013.

12 D. Kempe, J. Kleinberg, and A. Kumar. Connectivity and inference problems for temporal networks. In *32nd ACM Symposium on Theory of Computing (STOC)*, pages 504–513, 2000.

13 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.

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

15 Giri Narasimhan and Michiel Smid. *Geometric Spanner Networks*. Cambridge Univ. Press, 2007.

16 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.

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

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

19 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.