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 $\lambda : E \rightarrow 2^N$ 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 $G = (G, \lambda)$ 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 can always be found 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.

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 [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 journey), both cases being carefully discussed in

*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 still possible.
this paper. Temporal connectivity was considered in an early paper by Awerbuch and Even [4] (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 [6]. More recently, it has been the subject of several algorithmic studies, such as [2] and [5] (discussed below), and [18] and [20], which consider algorithms for computing structures related to temporal connectivity. Broad reviews on these topics can be found, e.g., in [7, 12, 15], 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 [14] (conference version [13]), 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 [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 \( \left( \frac{n}{2} \right) - \lfloor n/4 \rfloor \) 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 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, 16, 17]) or the current network topology of an updated dynamic graph (see e.g., [3, 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 \( \Theta(n \log n) \)-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 [14] 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 $\lambda$ in some way to avoid such a 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-unicty assumption made in [2] (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 a $O(n \log n)$-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 $G$ with single, locally distinct labels, and computes a $O(n \log n)$-sparse temporal spanner of $G$ in polynomial time. This algorithm is based on a number of original techniques introduced in this paper, 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 [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).

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 [2] 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 limitations, some content is placed in Appendix.

# 2 Definitions and Basic Results

## 2.1 Model and definitions

Let $G = (V, E)$ be an undirected graph and $\lambda : E \rightarrow 2^N$ 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 $G = (G, \lambda)$ as a temporal graph (other models and terminologies exist, all being equivalent for the considered problem). If $\lambda$ is single-valued and locally injective (i.e., adjacent edges have different labels), then we say that $\lambda$ is simple, and by extension, a temporal graph is simple if its labeling is simple.

A temporal path in $G$ (also called journey), is a finite sequence of $k$ triplets $J = \{(u_i, u_{i+1}, t_i)\}$ 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 \( G \) 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 \( G = ((V, E), \lambda) \) as a temporal graph \( G' = ((V', E'), \lambda') \) such that \( V = V' \), \( E' \subseteq E \) and \( \lambda': E' \rightarrow 2^N \) with \( \lambda'(e) \subseteq \lambda(e) \) for all \( e \in E' \). We call \( G' \) a valid spanner if it is temporally connected. Observe that, if \( G \) is simple, then spanners are fully determined by the chosen subset of edges \( E' \subseteq E \) (as in the above citation from [14]). 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, 14], although they are not referred to as “spanners” in these works.

Finally, when the underlying graph \( G \) is a complete graph, we call \( G \) 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.](image)

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 \( G = (G, \lambda) \) with unrestricted labeling \( \lambda \) into a clique \( H = (G, \lambda_H) \) with simple labeling such that any valid temporal spanner of \( H \) induces a valid temporal spanner of \( G \). (As explained, the converse is false, but this is fine because our result is positive on \( H \).) 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 \( H \), such that if a journey exists in \( H \), then the same sequence of edges allows for a (possibly non-strict) journey in \( G \).

As for the second claim of the introduction, if strict journeys are the only ones allowed, then as explained, we assume the existence of a simple sub-labeling of \( \lambda \) (whose computation is not discussed). 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. [2], 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 Appendix A.2 we improve this technique up to removing a constant fraction of \( \left\lfloor n^2/12 \right\rfloor \) edges. However, we consider as unlikely that such purely structural techniques could sparsify a graph within \( o(n^2) \) edges. The techniques we develop in this paper are different in essence.

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 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 also proposed in Appendix A.5.1 on page 19.

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 \(G\) and then reduce the computation to finding a temporal spanner in the smaller clique \(G[V \setminus u]\). We state this more formally in the following theorem. (See Figure 6a on page 16 for an illustration.)

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

*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 \(G\) through \(v\) and the edges of \(S'\). A symmetrical argument implies that all vertices in \(G\) can reach \(u\) through \(w\) using only \(\{u, w\}\) and the edges of \(S'\). \(\Box\)

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.](image)

**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 Appendix A.5.2). 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 $G$ 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 $k$, as explained in Appendix A.5.2 on page 19. 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 $G = (G, \lambda)$ 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 1.** Directed paths in $G^-$ correspond to journeys in $G$.

**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 $G$ 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. \qed

By construction, $E^-$ induces a disjoint set of out-trees (one source, possibly several sinks). We transform $E^-$ into a disjoint set $T^- = (V, E^-)$ 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 $(v, u_1), ..., (v, u_\ell)$ be its out-arcs with $(v, u_\ell)$ 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 $T^- = (V, E^-_T)$ be the resulting set of in-trees $T^-_1, ..., T^-_k$ (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.](image-url)
Fact 4. The set of in-trees $T^- = (V, E^-_T)$ has the following properties:

1. Directed paths in $T^-$ correspond to journeys in $G$.
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 1 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 $(v, u_\ell)$, so $(u_i, v), (v, u_\ell)$ is a journey in $G$. Observe that some of the journeys induced by the arcs of $T^-$ 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 $T^-$ cannot exceed half of the total number of vertices.

Lemma 2. The number of emitters in $T^-$ is at most $n/2$

Proof. After the transformation from $E^-$ to $E^-_T$, there is only one emitter in each in-tree $T^-_i \in T^-$ (Fact 4.4), and there are at most $n/2$ trees because each one contains at least 2 vertices (Fact 4.3).

We are now ready to define a temporal spanner based on $T^-$, 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 $S^-_T = \{\{u, v\} \in E : (u, v) \in T^-\} \cup \{\{u, v\} \in E : u$ is an emitter$\}$.

Theorem 2. $S^-_T$ is a temporal spanner of $G$.

Proof. By Fact 4 every vertex $v$ of $G$ that is a non-emitter in $T^-$ 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 $T^-$ 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.

![Forward fireworks cover and the resulting spanner](image-url)
Theorem 3. Forward fireworks covers have at most $3(n^2)/4 + O(n)$ edges.

Proof. Let $S_T$ be a forward fireworks cover based on a set of in-trees $T^-$. Each non-emitter in $T^-$ has only one out-arc which becomes one edge in $S_T$, thus overall $T^-$ contributes less than $n$ edges to $S_T$. Now, every emitter has an edge to every other vertex in $S_T$, and there are at most $n/2$ emitters in $T^-$ by Lemma 2. Note that the edges between emitters are selected twice but should be counted only once. Thus in the end, there are at most $n(n/2) - n^2/8$ edges, plus the edges of the in-trees (that is $O(n)$ 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 structure of the in-trees.

Lemma 3. Every non-emitter vertex $v$ can reach a vertex $v'$ in the same in-tree $T_i^-$ (emitter or not) using a journey of length at most two that arrives at $v'$ through $e^-(v')$.

Proof. Before the transformation from $E^-$ to $E_T^+$, all arcs $(u, v) \in E^-$ are such that $\{u, v\} = e^-(v)$, thus all the journeys in $E^-$ have the required property. Then, some arcs are flipped by the transformation (replaced by oppositely directed arcs). Now, if an arc $(v, u)$ is flipped, then its head $u$ is a sink in $E^-$ and its tail $v$ is not, thus no arc of the form $(w, v)$ is flipped. As a result, at least one arc in any two consecutive arcs in a journey in $T_i^-$ is the minimum edge of its head. □

4.2 Backward Fireworks

A symmetrical concept of fireworks can be defined based on the edges $\{u, v\} = e^+(v)$ of a temporal clique $G = (G, \lambda)$. 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 $G^+ = (V, E^+)$ which is a disjoint set of in-trees. By an analogous transformation as above, this set is then converted into a disjoint set $T^+ = (V, E_T^+)$ of out-trees each of which 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. The following lemmas are obtained by symmetrical arguments.

Lemma 4. The number of collectors in $T^+$ is at most $n/2$

Finally, we can build a temporal spanner $S_T^+ = \{\{u, v\} : (u, v) \in T^+\} \cup \{\{u, v\} : u$ 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 4. $S_T^+$ is a temporal spanner of the temporal clique $G$.

Theorem 5. Backward fireworks covers have at most $3(n^2)/4 + O(n)$ edges.

An example of a backward fireworks cover is given on Figure 7 on page 16. Finally, we establish a symmetrical property as in Lemma 3 to be used also in Section 5.

Lemma 5. For every non-collector vertex $v$ in an out-tree $T_i^+ \in T^+$, there exists a vertex $v'$ in $T_i^+$ (collector or not), such that a journey of length at most two from $v'$ to $v$ exists, leaving $v'$ 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 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 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 with 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 $T^{-}$ be the disjoint set of in-trees obtained during the construction of a forward fireworks cover (see Figure 4), and let $T^{+}$ 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 $T^{-}$) and let $X^{+}$ be the set of collectors (one per out-tree in $T^{+}$). The two sets can overlap, as a vertex may happen to be both an emitter in some tree in $T^{-}$ and a collector in some tree in $T^{+}$, 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 $S = \{(u, v) \in T^{-} \cup T^{+} \cup E_{H} \}$ be a bidirectional fireworks cover (or simply a fireworks cover). An illustration is given in Figure 8 on page 16.

**Theorem 6.** $S$ is a temporal spanner of $G$.

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

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

**Proof.** The number of edges in $T^{-}$ and $T^{+}$ is linear in $n$. By Lemma 2, the number of emitters is at most $n/2$, and so is the number of collectors by Lemma 3. 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 $n^2/4$ edges plus the edges of $T^{-}$ and $T^{+}$. \hfill \Box

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 $T^{-}$ and $T^{+}$, 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 $G$ 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 3 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 5 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 $G[V \setminus v]$, 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 2 and 4), 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 6. 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 smaller label exists between $v$ and another emitter $u'$, which creates a 2-hop journey from $u'$ to $v$ 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 direct edge), thus $u'$ is 2-hop dismountable. □

Lemma 6 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 Figure 9 on page 17.

In summary, either the algorithm recurses until the input clique is fully dismounted (through Case 1 or Case 2), resulting in a $O(n)$-dense spanner (Fact 4), 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 $G$. 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 6). 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 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 \( k = n/2 \) be 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 \( e^+(v) = e^-(v) \) and \( e^k(v) = e^+(v) \).

The \( k \) ranks are partitioned into subintervals of doubling size \( I_j = [2^j+2 - 7, 2^{j+3} - 8] \), 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 \( I_1 = [1, 8], I_2 = [9, 24], I_3 = [25, 56], \) and \( I_4 = [57, 120] \). Computation step \( j \) is made with respect to the subgraph \( B_j = (X_j^-, X^+, E_j) \) where \( E_j = \{ e^i(v) \in E_B : i \in I_j, v \in X_j^- \} \), namely the edges of the currently alive emitters, whose ranks are in the interval \( I_j \).

**Lemma 7.** 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 Appendix [A.3 on page 17](#) (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 1.** 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 \( |X_{j+1}^-| = k \) 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 \( i \in I_j \). 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 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 8.** 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 \( I_j \) 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 $S_{\text{last}}$) 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., $S = (\bigcup_j S_j) \cup S_{\text{last}} \cup S^- \cup S^+$.

Theorem 8. $S$ is a temporal spanner of the complete bipartite graph $B$ and it is made of $\Theta(n \log n)$ 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 $j$, $\frac{k}{3^j}$ 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 $S_{\text{last}}, S^-, S^+$ each contain $\Theta(n)$ edges (and $S^+$ is actually included in $S_{\text{last}}$).

Corollary 1. 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.

Additional comments regarding the overall running time of the algorithm (establishing its polynomiality) are given in Appendix A.4, and some observations as to the optimality of $O(n \log n)$-sparse spanners in temporal cliques are made in Appendix A.6.

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. 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.

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 *Proceedings of 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] L. 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 *Proceedings of 19th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 591–600, 2008.

[11] H.A. Harutyunyan, A.L. Liestman, J.G. Peters, and D. Richards. Broadcasting and gossiping in communication networks. In P. Zhang J.L. Gross, J. Yellen, editor, *Handbook of Graph Theory, Second Edition*, chapter 12.2, pages 1477–1494. CRC Press, 2013.
[12] Petter Holme and Jari Saramäki. *Temporal Networks*. Springer, 2013.

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

[14] 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.

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

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

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

[20] 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.
\section*{A Appendix}

Some additional content is presented in this section, including four illustrations (Subsection A.1), a technique which is not central in the paper (Subsection A.2), the full proof of Lemma 14 (Subsection A.3), a discussion of the complexity of our algorithm (Subsection A.4), the construction of adversarial families of instances (Subsection A.5), and finally an open question about the optimal density of the temporal spanners (Subsection A.6).

\subsection*{A.1 Additional illustrations}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig6}
\caption{Illustration of the principle of dismountability (p.\pageref{fig6}).}
\end{figure}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig7}
\caption{Example of a backward fireworks cover and the resulting spanner (p.\pageref{fig7}).}
\end{figure}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig8}
\caption{A bidirectional fireworks cover and the corresponding spanner. 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. (p.\pageref{fig8})}
\end{figure}
A.2 Improvement of the structural technique in [2]

Recall that the technique in [2] consists of partitioning the input clique into (essentially) \( n/4 \) subcliques of 4 vertices each, then remove an edge from each subclique, resulting in \( \lfloor n/4 \rfloor \) edges removed. The key observation is that the type of partitioning used in [2] is node-disjoint, but the same argument actually holds when applied to edge-disjoint partitions, with significant consequences. Indeed, by Wilson’s Theorem [19], the number of edge-disjoint cliques on 4 vertices in a complete graph on \( n \) vertices is \( \lfloor n^2/12 \rfloor \) (possibly with a few vertices remaining). (More generally, as \( c \to 1 \), graphs with minimum degree at least \( cn \) have \( \lfloor (n^2)/3 \rfloor \) disjoint copies of \( K_k \) for all \( k \).) The immediate consequence is that one can remove \( \lfloor n^2/12 \rfloor = \Theta(n^2) \) edges by a similar technique. As discussed in Section 2.3, we consider as unlikely that purely structural techniques could suffice to sparsify a graph down to \( o(n^2) \) edges. The techniques developed in the rest of this paper are orthogonal to this one.

A.3 Proof of Lemma 7 (p.12)

**Lemma 7**. 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 some vertex in \( X_b \) through a 2-hop journey (within \( B_j \)).

**Proof.** 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. 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 \( I_j \) matters, as 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 (in fact, any constant power of two) does not impact 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 that remains constant over the steps (namely, 8).

The generic calculation relative to step \( j \) is itself based on an iterative argument that one should be careful not to mistake with the outer loop varying \( j \). Thus, keeping \( j \) fixed for the rest of the proof, \( X_a \) and \( X_b \) are built iteratively as follows: identify the collector \( c \) with highest degree and add all the corresponding emitters to \( X_a \) except for 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 work (and
always terminates), observe that the average degree of 8 for collectors forces at least one collector to be of degree 8. In fact, by the pigeon hole principle, this property remains true as long as the number of emitters not being processed yet (i.e., in \(X_j^-(X_a \cup X_b)\)) is larger than \(7/8 \cdot |X_j^-|\), which guarantees that \(X_a\) has size at least \(1/8 \cdot 7/8 \cdot |X_j^-|\) when the number of non processed emitters goes below that threshold. An analogue argument forces at least one collector to be of degree 7 so long as the number of non processed emitters is above \(6/8 \cdot |X_j^-|\), resulting in at least \(1/8 \cdot 6/7 \cdot |X_j^-|\) more emitters in \(X_a\) when the next threshold is attained (indeed, six emitters out of the seven considered enter \(X_a\)). By iterating this argument, the size of \(X_a\) ends up being at least a fraction of \(|X_j^-|\) equal to \(1 \cdot (7/8+6/7+5/6+4/5+3/4+2/3+1/2) \approx 0.66 \geq 0.5\). □

This proof is illustrated on Figure 10 for the particular case that \(j = 1\).

![Diagram](image)

(a) Emitters “meeting” at collectors.  
(b) Calculation of the size of \(X_a\) with respect to the set of emitters \(X_j^-\).

Figure 10: Illustration of the method used in the proof of Lemma 7.

### A.4 Time Complexity (p.13)

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. To start, observe that whenever the algorithm recurses, the number of vertices is reduced by one, and in each recursion the fireworks 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.
A.5 Adversarial Families

A.5.1 Non-pivotable Graphs (p.4)

We explain how to construct non-pivotable graphs 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 direction. 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 11, 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.

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 $t = \left(\frac{n-2}{2}\right) + 1$. The labeling assigns 0 to $\{u, v\}$, 1 to $\{v, w\}$, and all values in $[3, t]$ 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 labeling 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.

Remark 2. One may be tempted to make both periods overlap by one unit, so that the edge labeled $t$ can be used for both directions. However, this would have no decisive effect on the construction.

A.5.2 Non-dismountable Graphs (p.5-7)

We explain how to construct arbitrarily large simple temporal cliques which are not $k$-hop dismountable for all $k$ (non-dismountable, for short). To start, consider a 4-vertex clique in which the local relations among labels is the same as in Figure 12. Such a clique is non-dismountable, because none of the vertices can (1) reach a vertex $u$ through $e^-(u)$ and (2) be reached from a vertex $v$ through $e^+(v)$. An arbitrary large clique can be built by making copies of this clique and assigning labels so that the local minima and maxima in each copy behave as in the original clique. An example with 8 vertices (thus 28 edges) is shown in Figure 13. All the other edges are assigned an intermediate label, i.e., whose value is between the local
minima and maxima, which prevents journeys between different cliques that arrive through a minimum edge (or leave through a maximum edge). This construction can be generalized to any number of vertices which is a multiple of 4.

Remark 3. The given construction for non-dismountable graphs makes it possible to find a pivot vertex. However, the experiments that we conducted showed that instances exist which are neither pivotable nor dismountable, and their numbers do not seem to diminish as \( n \) tends to \( \infty \).

### A.6 Open Questions

Despite extensive computer search, we were not able to find an instance (out of hundred of millions of different sizes) which does not admit a spanner of either \( 2n - 3 \) or \( 2n - 4 \) edges. The latter bound is actually not surprising, as a classical result in gossip theory (see e.g., Facts F29 through F32 in [11]) 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 such a spanner, which suggests further investigation to understand the structure of simple temporal cliques. These observations motivate the following questions.

**Open question 1.** Do simple temporal cliques always admit \( \Theta(n) \)-sparse temporal spanners?

**Open question 2** (audacious). Do simple temporal cliques always admit temporal spanners with at most \( 2n - 3 \) edges?

Another natural question is whether any other class of dense graph can always be sparsified within \( o(n^2) \) edges.

**Open question 3** (other classes). Is there a larger class of dense graphs that always admit \( o(n^2) \)-sparse temporal spanners?