On Temporally Connected Graphs of small cost*

Eleni C. Akrida[1], Leszek Gąsieniec[1],

George B. Mertzios[2], and Paul G. Spirakis[1]

[1] Department of Computer Science, University of Liverpool, UK
{Eleni.Akrida,L.A.Gasieniec,P.Spirakis}@liverpool.ac.uk
[2] School of Engineering and Computing Sciences, Durham University, UK
George.Mertzios@durham.ac.uk

Abstract. We study the design of small cost temporally connected graphs, under various constraints. We mainly consider undirected graphs of n vertices, where each edge has an associated set of discrete availability instances (labels). A journey from vertex u to vertex v is a path from u to v where successive path edges have strictly increasing labels. A graph is temporally connected iff there is a (u, v)-journey for any pair of vertices $u, v,\ u \neq v$. We first give a simple polynomial-time algorithm to check whether a given temporal graph is temporally connected. We then consider the case in which a designer of temporal graphs can *freely choose* availability instances for all edges and aims for temporal connectivity with very small *cost*; the cost is the total number of availability instances used. We achieve this via a simple polynomial-time procedure which derives designs of cost linear in n, and at most the optimal cost plus 2. To show this, we prove a lower bound on the cost for any undirected graph. However, there are pragmatic cases where one is not free to design a temporally connected graph anew, but is instead *given* a temporal graph design with the claim that it is temporally connected, and wishes to make it more cost-efficient by removing labels without destroying temporal connectivity (redundant labels). Our main technical result is that computing the maximum number of redundant labels is APX-hard, i.e., there is no PTAS unless $P = NP$. On the positive side, we show that in dense graphs with random edge availabilities, all but $\Theta(n)$ labels are redundant whp. A temporal design may, however, be *minimal*, i.e., no redundant labels exist. We show the existence of minimal temporal designs with at least $n \log n$ labels.

1 Introduction and motivation

A temporal network is, roughly speaking, a network that changes with time. A great variety of modern and traditional networks are dynamic, e.g., social networks, wireless networks, transport networks. Dynamic networks have been attracting attention over the past years [3,4,7,9,21], exactly because they model

* Supported in part by (i) the School of EEE and CS and the NeST initiative of the Univeristy of Liverpool, (ii) the FET EU IP Project MULTIPLEX under contract No. 317532, and (ii) the EPSRC Grant EP/K022660/1.

real-life applications. Following the model of [1,13,20], we consider *discrete time* and restrict attention to systems in which the connections between the participating entities may change but the entities remain unchanged. This assumption is clearly natural when the dynamicity of the system is inherently discrete and gives a purely combinatorial flavor to the resulting models and problems.

In several such dynamic settings, maintaining connections may come at a cost; consider a transport network or an unstable chemical or physical structure, where energy is required to keep a link available. We define the cost as the total number of discrete time instances, e.g., days or hours, at which the network links become available, i.e., the sum over all edges of the number of the edge's availability instances. We focus on design issues of temporal networks that are temporally connected; a temporal network is temporally connected if information can travel over time from any node to any other node following *journeys*, i.e., paths whose successive edges have strictly increasing availability time instances. If one has absolute freedom to design a small cost temporally connected temporal network on an underlying static network, i.e, choose the edge availabilities, then a reasonable design would be to select a rooted spanning tree and choose appropriate availabilities to construct time-respecting paths from the leaves to the root and *then* from the root back to the leaves. However, in more complicated scenarios one might not be free to *choose* edge availabilities arbitrarily but instead *specific* link availabilities might pre-exist for the network. Imagine a hostile network on a complete graph where availability of a link means a break in its security, e.g., when the guards change shifts, and only then are we able to pass a message through the link. So, if we wish to send information through the network, we may only use the times when the shifts change and it is reasonable to try and do so by using as few of these breaks as possible. In such scenarios, we may need to first verify that the pre-existing edge availabilities indeed define a temporally connected temporal network. Then, we may try to reduce the cost of the design by *removing* unnecessary (redundant) edge availabilities if possible, without loosing temporal connectivity. Consider, again, the clique network of n vertices with one time availability per edge; it is clearly temporally connected with cost $\Theta(n^2)$. However, it is not straightforward if all these edge availabilities are necessary for temporal connectivity. We resolve here the complexity of finding the maximum number of redundant labels in any given temporal graph.

1.1 The model and definitions

It is generally accepted to describe a network topology using a graph, the vertices and edges of which represent the communicating entities and the communication opportunities between them respectively. We consider graphs whose edge availabilities are described by sets of positive integers (labels), one set per edge.

Definition 1 (Temporal Graph). *Let $G = (V, E)$ be a (di)graph. A temporal graph on G is an ordered triplet $G(L) = (V, E, L)$, where $L = \{L_e \subseteq \mathbb{N} : e \in E\}$ is an* assignment *of labels*[3] *to the edges (arcs) of G. L is called a* labeling *of G.*

[3] The labels of an edge (arc) are the *discrete time instances* at which it is available.

Definition 2 (Time edge). *Let $e = \{u, v\}$ (resp. $e = (u, v)$) be an edge (resp. arc) of the underlying (di)graph of a temporal graph and consider a label $l \in L_e$. The ordered triplet (u, v, l) is called* time edge.[4]

Definition 3 (Cost of a labeling). *Let $G(L) = (V, E, L)$ be a temporal (di)graph and L be its labeling. The* cost *of L is defined as $c(L) = \sum_{e \in E} |L_e|$.*

A basic assumption that we follow here is that when a message or an entity passes through an available link at time t, then it can pass through a subsequent link only at some time $t' > t$ and only at a time at which that link is available.

Definition 4 (Journey). *A* temporal path *or* journey *j from a vertex u to a vertex v ((u, v)-*journey*) is a sequence of time edges (u, u_1, l_1), (u_1, u_2, l_2), ... , (u_{k-1}, v, l_k), such that $l_i < l_{i+1}$, for each $1 \leq i \leq k - 1$. We call the last time label, l_k,* arrival time *of the journey.*

Definition 5 (Foremost journey). *A (u, v)-journey j in a temporal graph is called* foremost journey *if its arrival time is the minimum arrival time of all (u, v)-journeys' arrival times, under the labels assigned to the underlying graph's edges. We call this arrival time the* temporal distance*, $\delta(u, v)$, of v from u.*

In this work, we focus on *temporally connected* temporal graphs:

Definition 6 (Property TC). *A temporal (di)graph $G(L) = (V, E, L)$ satisfies the property TC, or equivalently L satisfies TC on G, if for any pair of vertices $u, v \in V$, $u \neq v$, there is a (u, v)-journey* and *a (v, u)-journey in $G(L)$. A temporal (di)graph that satisfies the property TC is called* temporally connected*.*

Definition 7 (Minimal temporal graph). *A temporal graph $G(L) = (V, E, L)$ over a (strongly) connected (di)graph is* minimal *if $G(L)$ has the property TC, and the removal of any label from any L_e, $e \in E$, results in a $G(L')$ that* does not *have the property TC.*

Definition 8 (Removal profit). *Let $G(L) = (V, E, L)$ be a temporally connected temporal graph. The* removal profit *$r(G, L)$ is the largest total number of labels that can be removed from L without violating TC on G.*[5]

1.2 Previous work and our contribution

In recent years, there is a growing interest in distributed computing systems that are inherently dynamic. For example, temporal dynamics of network flow problems were considered in a set of pioneering papers [10,11,14,15]. The model we consider here is a direct extension of the one considered in the seminal paper of [13] and its sequel [20]. In [13], the authors consider the case of one label per edge and examines how basic graph properties change in the temporal setting.

[4] Note that an undirected edge $e = \{u, v\}$ is associated with $2 \cdot |L_e|$ time edges, namely both (u, v, l) and (v, u, l) for every $l \in L_e$.

[5] Here, removal of a label l from L refers to the removal of l only from a particular edge and not from all edges that are assigned label l, that is, if $l \in L_{e_1} \cap L_{e_2}$ and we remove l from both L_{e_1} and L_{e_2}, it counts as two labels removed from L.

In [20], this model is extended to many labels per edge and the number of labels needed for a temporal design of a network to guarantee several graph properties with certainty is examined. The latter also defined the cost notion and, amongst other results, gave an algorithm to compute foremost journeys which can be used to decide property TC. However, the time complexity of that algorithm was *pseudo-polynomial*, as it was dominated by the *cube of the maximum label* used in the given labeling. Random edge availabilities were first considered in [1] in order to study the Expected Temporal Diameter of temporal graphs.

Here, we show that if the designer of a temporal graph can select edge availabilities freely, then an almost optimal linear-cost (in the size of the graph) design that satisfies TC can be easily obtained (cf. Section 3). We give an almost matching lower bound to indicate optimality. However, there are pragmatic cases where one is not free to design a temporal graph anew, but is *given* a set of possible availabilities per edge with the claim that they satisfy TC and the constraint that she may only use them or a subset of them for her design. We show that we can verify TC in *low* polynomial time (cf. Section 2). The *given* design may also be minimal; we partially characterise minimal designs in Section 4. On the other hand, there may be some labels of the initial design that can be removed without violating TC (and also result in a lower cost). In this case, how many labels can we remove at best? Our main technical result is that this problem is APX-hard, i.e. it has no PTAS unless $P = NP$. On the positive side, we show that in the case of complete graphs and random graphs, if the labels are also assigned at random, we can remove all but $O(n)$ labels.

Stochastic aspects and/or survivability of network design were also considered in [12,18,19]. An extended report of related work [3–9,16,17,21–23] can be found in our full paper (cf. Appendix).

2 Property TC is decidable in low polynomial time

In this section, we give a simple polynomial-time algorithm which, given a temporal (di)graph $G(L)$ and a source vertex s, computes a *foremost* (s, v)-journey, for every $v \neq s$, if such a journey exists. Curiously enough, the previously known algorithm was pseudo-polynomial [20]. Our algorithm significantly improves the running time. In fact, we conjecture it is optimal.

Theorem 1. *Algorithm 1 satisfies the following, for every vertex $v \neq s$:*

(a) If $arrival\_time[v] < +\infty$, then there exists a foremost journey from s to v, the arrival time of which is exactly $arrival\_time[v]$. This journey can be constructed by following the $parent[v]$ pointers in reverse order.
(b) If $arrival\_time[v] = +\infty$, then no (s, v)-journey exists.
(c) The time complexity of Algorithm 1 is dominated by the sorting time of the set of time edges (resp. time arcs).

Corollary 1. *The time complexity of Algorithm 1 is $O\big(c(L) \cdot \log c(L)\big)$.*

Conjecture *We conjecture that any algorithm that computes journeys out of a vertex s must sort the time edges (arcs) by their labels, i.e., we conjecture that Algorithm 1 is asymptotically optimal with respect to the running time.*

Note that Algorithm 1 can even compute foremost (s, v)-journeys, if they exist, that *start* from a given time $t_{start} > 0$ onward. Simply, one ignores the time edges (arcs) with labels smaller than the start time.

Algorithm 1 Foremost journey algorithm

Input: A temporal (di)graph $G(L) = (V, E, L)$ of n vertices, the set of all time edges (arcs) of which is denoted by $S(L)$; a designated source vertex $s \in V$

Output: A foremost (s, v)-journey from s to all $v \in V \setminus \{s\}$, where such a journey exists; if no (s, v)-journey exists, then the algorithm reports it.

Sort $S(L)$ in increasing order of labels; // Note that $|S(L)| = c(L)$
Let S' be the sorted array of time edges (resp. time arcs) according to time labels;
$R := \{s\}$; // The set of vertices to which s has a foremost journey
for each $v \in V \setminus \{s\}$ **do**
  $parent[v] := \emptyset$;
  $arrival\_time[v] := +\infty$;
Proceed sequentially in S', examining each time edge (resp. time arc) only once;
for the current time edge (resp. time arc) (a, b, l) **do**
  if $a \in R$ **and** $b \notin R$ **then**
    $parent[b] := a$;
    $arrival\_time[b] := l$;
    $R := R \cup \{b\}$;

3 Nearly cost-optimal design for TC in undirected graphs.

In this section, we study temporal design on connected undirected graphs, so that the resulting temporal graphs satisfy TC. In this scenario, the designer has absolute freedom to choose the edge availabilities of the underlying graph.

Theorem 2. *(a) Given a connected undirected graph $G = (V, E)$ of n vertices, we can design a labeling L of cost $c(L) = 2(n-1)$ that satisfies the property TC on G. L can be computed in polynomial time.*

(b) For any connected undirected graph $G = (V, E)$ of $n \geq 2$ vertices and for any labeling L that satisfies the property TC on G, the cost of L is $c(L) \geq 2n-4$.

4 Minimal Temporal Designs

Suppose now that a temporal graph on a (strongly) connected (di)graph $G = (V, E)$ is *given*[6] to a designer with the claim that it satisfies TC. If the given design is not minimal, she may wish to remove as many labels as possible, thus reducing the cost. Minimality of a design can be verified using Algorithm 1.

[6] In this scenario, the designer is allowed to only use the given set of edge availabilities, or a subset of them.

4.1 Minimal designs of non linear cost in the number of vertices

Notice that if many edges have the same label(s), we can encounter *trivial cases* of minimal temporal graphs. For example, the complete graph where all edges are assigned the same label is minimal, but there are no journeys of length larger than 1. Here, we focus on minimal temporal graphs, where minimality is not caused merely because of the use of the same labels on every edge. Consider graphs every edge of which only becomes available at most one moment in time and no two different edges become available at the same time. Are there minimal temporal graphs of the above scenario with non linear (in the size of the graph) cost? For example, any complete graph with a single label per edge, different labels to different edges, satisfies TC. Are all these $\Theta(n^2)$ labels needed for TC, i.e., are there minimal temporal complete graphs? As we prove in Theorem 4, the answer is negative. However, we give below a minimal temporal graph on n vertices with non-linear in n cost, namely with $O(n \log n)$ labels.

A minimal temporal design of $n \log n$ cost

Definition 9 (Hypercube graph). *The k-hypercube graph, commonly denoted Q_k, is the k-regular graph of 2^k vertices and $2^{k-1} \cdot k$ edges.*

Theorem 3. *There exists an infinite class of minimal temporal graphs on n vertices with $\Theta(n \cdot \log n)$ edges and $\Theta(n \cdot \log n)$ labels, such that different edges have different labels.*

Sketch of proof. We present a minimal temporal graph on the hypercube. Consider Protocol 2 for labeling the edges of $G = Q_k$. The temporal graph, $G(L)$, that this labeling procedure produces on the hypercube is minimal. □

Protocol 2 Labeling the hypercube graph, $G = Q_k$

Consider the k dimensions of the hypercube $G = Q_k$, $x_1, x_2, \ldots, x_k$;
for $i = 1 \ldots k$ **do**
  Let $X_i := \{e_{i1}, e_{i2}, \ldots, e_{i2^{k-1}}\}$ be the list of edges in dimension x_i, in an arbitrary order;
  Let L_i be the (sorted from smallest to largest) list of labels $L_i := \{(i-1) \cdot 2^{k-1} + 1, (i-1) \cdot 2^{k-1} + 2, \ldots, i \cdot 2^{k-1}\}$;
for $i = 1 \ldots k$ **do**
  for $j = 1 \ldots 2^{k-1}$ **do**
    Assign the (current) first label of L_i to the (current) first edge of X_i ;
    Remove the (current) first label of L_i from the list;
    Remove the (current) first edge of X_i from the list;
return the produced temporal graph, $G(L)$;

Cliques of at least 4 vertices are not minimal The complete graph on n vertices, K_n, with a labeling L that assigns a single, different for every edge, label per edge is an interesting case, since $K_n(L)$ always satisfies TC. However, it is not minimal as the theorem below shows.

Theorem 4. *Let $n \in \mathbb{N},\ n \geq 4$ and denote by K_n the complete graph on n vertices. Any labeling L that assigns a single label to every edge of K_n, different label per edge produces a temporal graph $K_n(L)$ that is* not minimal. *In fact, $\exists S \subseteq \cup_{e \in E(K_n)} L_e,\ |S| = \lfloor \frac{n}{4} \rfloor$, such that when we remove all the labels of S from L, the resulting temporal graph still satisfies TC. Note that by the union $\cup_{e \in E(K_n)} L_e$ we denote the* multiset *of all labels used in L.*

Proof. The proof is divided in two parts, as follows:

(a) We show that the theorem holds for K_4. Without loss of generality, we use labels 1 to 6, one label per edge, and show that we can always remove a label and still satisfy TC. The proof requires an exhaustive check of 720 permutations of the labels and is done via a *computer program* (code can be found online here: `http://cgi.csc.liv.ac.uk/~akridel/research-results.html`).

(b) Now, consider the complete graph on $n \geq 4$ vertices, $K_n = (V, E)$. Partition V arbitrarily into $\lceil \frac{n}{4} \rceil$ subsets $V_1, V_2, \ldots, V_{\lceil \frac{n}{4} \rceil}$, such that $|V_i| = 4, \forall i = 1, 2, \ldots, \lceil \frac{n}{4} \rceil - 1$ and $|V_{\lceil \frac{n}{4} \rceil}| \leq 4$. In each 4-clique defined by $V_i,\ i = 1, 2, \ldots, \lfloor \frac{n}{4} \rfloor$, we can remove a "redundant" label, as shown in (a). The resulting temporal graph on K_n still preserves TC since for every ordered pair of vertices $u, v \in V$:
   - if u, v are in the same V_i, $i = 1, 2, \ldots, \lfloor \frac{n}{4} \rfloor$, then there is a (u, v)-journey that uses time edges within the 4-clique on V_i, as proven in (a).
   - if $u \in V_i$ and $v \in V_j,\ i \neq j$, then there is a (u, v)-journey that uses the (direct) time edge on $\{u, v\}$. □

4.2 Computing the removal profit is APX-hard

Recall that the removal profit is the largest number of labels that can be removed from a temporally connected graph without destroying TC. We now show that it is hard to arbitrarily approximate the value of the removal profit for an arbitrary graph, i.e., there exists no PTAS[7] for this problem, unless P=NP. It is worth noting here that, in our hardness proof below, we consider undirected graphs.

We prove our hardness result by providing an approximation preserving polynomial reduction from a variant of the maximum satisfiability problem, namely from the *monotone Max-XOR(3)* problem. Consider a monotone XOR-boolean formula ϕ with variables $x_1, x_2, \ldots, x_n$.[8] The clause $\alpha = (x_i \oplus x_j)$ is XOR-satisfied by a truth assignment τ iff $x_i \neq x_j$ in τ. The number of clauses of ϕ that are XOR-satisfied in τ is denoted by $|\tau(\phi)|$. If every variable x_i appears in exactly k XOR-clauses in ϕ, then ϕ is called a *monotone XOR(k)* formula. The *monotone Max-XOR(k)* problem is, given a monotone XOR(k) formula ϕ, to compute a truth assignment τ of the variables $x_1, x_2, \ldots, x_n$ that XOR-satisfies the largest possible number of clauses, i.e., an assignment τ such that $|\tau(\phi)|$ is maximized. The monotone Max-XOR(3) problem essentially encodes the *Max-Cut* problem on 3-regular (cubic) graphs, which is known to be APX-hard [2].

[7] PTAS stands for Polynomial-Time Approximation Scheme.

[8] A monotone XOR-boolean formula is a conjunction of XOR-clauses of the form $(x_i \oplus x_j)$, where no variable is negated.

Lemma 1. *[2] The monotone Max-XOR(3) problem is APX-hard.*

Now we provide our reduction from the monotone Max-XOR(3) problem to the problem of computing $r(G, L)$. Let ϕ be an arbitrary instance of monotone Max-XOR(3) with n variables $x_1, x_2, \ldots, x_n$ and m clauses. Since every variable x_i appears in ϕ in exactly 3 clauses, it follows that $m = \frac{3}{2}n$. We will construct from ϕ a graph $G_\phi = (V_\phi, E_\phi)$ and a labeling L_ϕ of G_ϕ.

For every $i = 1, 2, \ldots, n$ we construct the graph $G_{\phi,i}$ and the labeling $L_{\phi,i}$ of Figure 1. In this figure, the labels of every edge in $L_{\phi,i}$ are drawn next to the edge. We call the induced subgraph of $G_{\phi,i}$ on the 4 vertices $\{s^{x_i}, u_0^{x_i}, w_0^{x_i}, v_0^{x_i}\}$ the *base* of $G_{\phi,i}$. Also, for every $p \in \{1, 2, 3\}$, we call the induced subgraph of $G_{\phi,i}$ on the 4 vertices $\{t_p^{x_i}, u_p^{x_i}, w_p^{x_i}, v_p^{x_i}\}$ the *pth branch* of $G_{\phi,i}$. Finally, we call the edges $\{u_0^{x_i}, w_0^{x_i}\}$ and $\{w_0^{x_i}, v_0^{x_i}\}$ the *transition edges* of the base of $G_{\phi,i}$ and, for every $p \in \{1, 2, 3\}$, we call the edges $\{u_p^{x_i}, w_p^{x_i}\}$ and $\{w_p^{x_i}, v_p^{x_i}\}$ the *transition edges* of the pth branch of $G_{\phi,i}$. For every $p \in \{1, 2, 3\}$ we associate the pth appearance of the variable x_i in a clause of ϕ with the pth branch of $G_{\phi,i}$.

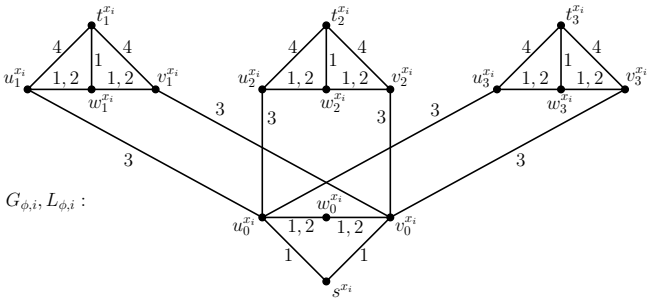

Fig. 1. The gadget $G_{\phi,i}$ for the variable x_i.

We continue the construction of $G_{\phi,i}$ and $L_{\phi,i}$ as follows. First, we add an edge between any possible pair of vertices $w_p^{x_i}, w_q^{x_j}$, where $p, q \in \{0, 1, 2, 3\}$ and $i, j \in \{1, 2, \ldots, n\}$, and we assign to this new edge $e = \{w_p^{x_i}, w_q^{x_j}\}$ the unique label $L_\phi(e) = \{7\}$. Note here that we add this edge $\{w_p^{x_i}, w_q^{x_j}\}$ also in the case where $i = j$ (and $p \neq q$). Moreover, we add an edge between any possible pair of vertices $t_p^{x_i}, t_q^{x_j}$, where $i \neq j$, $i, j \in \{1, 2, \ldots, n\}$, and $p, q \in \{1, 2, 3\}$. We assign to this new edge $e = \{t_p^{x_i}, t_q^{x_j}\}$ the unique label $L_\phi(e) = \{7\}$.

Furthermore we add a new vertex t_0 which is adjacent to vertex $w_0^{x_n}$ and to all vertices in the set $\{s^{x_i}, t_1^{x_i}, t_2^{x_i}, t_3^{x_i}, u_p^{x_i}, v_p^{x_i} : 1 \leq i \leq n,\ 0 \leq p \leq 3\}$. First we assign to the edge $\{t_0, w_0^{x_n}\}$ the unique label $L_\phi(\{t_0, w_0^{x_n}\}) = \{5\}$. Furthermore, for every vertex $t_p^{x_i}$, where $1 \leq i \leq n$ and $1 \leq p \leq 3$, we assign to the edge $\{t_0, t_p^{x_i}\}$ the unique label $L_\phi(\{t_0, t_p^{x_i}\}) = \{5\}$. Finally, for each of the vertices $z \in \{s^{x_i}, u_p^{x_i}, v_p^{x_i} : 1 \leq i \leq n,\ 0 \leq p \leq 3\}$ we assign to the edge $\{t_0, z\}$ the unique label $L_\phi(\{t_0, z\}) = \{6\}$. The addition of the vertex t_0 and the labels of the (dashed) edges incident to t_0 are illustrated in Figure 2(a).

Consider now a clause $\alpha = (x_i \oplus x_j)$ of ϕ. Assume that the variable x_i (resp. x_j) of the clause α corresponds to the pth (resp. qth) appearance of x_i

(resp. x_j) in ϕ. Then we identify the vertices $u_p^{x_i}, v_p^{x_i}, w_p^{x_i}, t_p^{x_i}$ of the pth branch of $G_{\phi,i}$ with the vertices $v_q^{x_i}, u_q^{x_i}, w_q^{x_i}, t_q^{x_i}$ of the qth branch of $G_{\phi,j}$, respectively (cf. Figure 2(b)). This completes the construction of G_ϕ and its labeling L_ϕ.

Denote the vertex sets $A = \{s^{x_i}, u_p^{x_i}, v_p^{x_i} : 1 \leq i \leq n,\ 0 \leq p \leq 3\}$, $B = \{w_p^{x_i} : 1 \leq i \leq n,\ 0 \leq p \leq 3\}$, and $C = \{t_p^{x_i} : 1 \leq i \leq n,\ 1 \leq p \leq 3\}$. Note that $V_\phi = A \cup B \cup C \cup \{t_0\}$. Furthermore, for every $i \in \{1, 2, \ldots, n\}$ and every $p \in \{1, 2, 3\}$ we define for simplicity of notation the temporal paths $P_{i,p} = (s^{x_i}, u_0^{x_i}, u_p^{x_i}, t_p^{x_i})$ and $Q_{i,p} = (s^{x_i}, v_0^{x_i}, v_p^{x_i}, t_p^{x_i})$. For every $i \in \{1, 2, \ldots, n\}$ the graph $G_{\phi,i}$ has 16 vertices. Furthermore, for every $p \in \{1, 2, 3\}$, the 4 vertices of the pth branch of $G_{\phi,i}$ also belong to a branch of $G_{\phi,j}$, for some $j \neq i$. Therefore, together with the vertex t_0, the graph G_ϕ has in total $10n+1$ vertices.

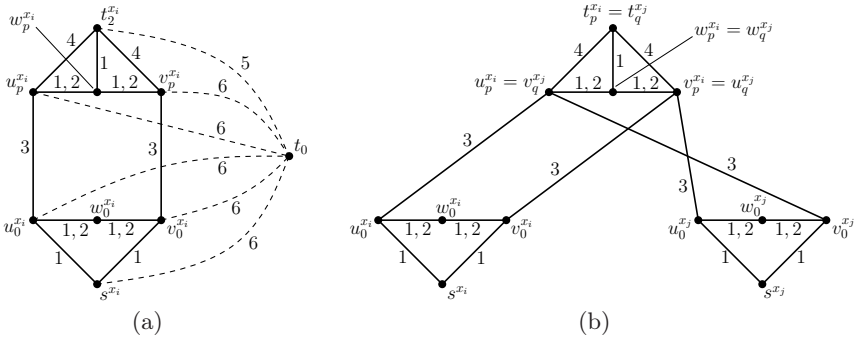

Fig. 2. (a) The addition of vertex t_0. There exists in G_ϕ also the edge $\{t_0, w_0^{x_n}\}$ with label $L_\phi(\{t_0, w_0^{x_n}\}) = \{5\}$. (b) The gadget for the clause $(x_i \oplus x_j)$.

To provide some intuition about the correctness of Theorem 5, we now briefly describe how we can construct a labeling L of G_ϕ by removing $9n+k$ labels from L_ϕ, given a truth assignment τ of ϕ with $|\tau(\phi)| \geq k$. First we keep in L all labels of L_ϕ on the edges incident to t_0. Furthermore we keep in L the label $\{7\}$ of all the edges $\{t_p^{x_i}, t_q^{x_j}\}$ and the label $\{7\}$ of all the edges $w_p^{x_i} w_q^{x_j}$. Moreover we keep in L the label $\{1\}$ of all the edges $\{t_p^{x_i}, w_p^{x_i}\}$. Let now $i = 1, 2, \ldots, n$. If $x_i = 0$ in τ, we keep in L the labels of the edges of the paths $P_{i,1}, P_{i,2}, P_{i,3}$, as well as the label 1 of the edge $\{v_0^{x_i}, w_0^{x_i}\}$ and the label 2 of the edge $\{w_0^{x_i}, u_0^{x_i}\}$. Otherwise, if $x_i = 1$ in τ, we keep in L the labels of the edges of the paths $Q_{i,1}, Q_{i,2}, Q_{i,3}$, as well as the label 1 of the edge $\{u_0^{x_i}, w_0^{x_i}\}$ and the label 2 of the edge $\{w_0^{x_i}, v_0^{x_i}\}$.

We now continue the labeling L as follows. Consider an arbitrary clause $\alpha = (x_i \oplus x_j)$ of ϕ. Assume that the variable x_i (resp. x_j) of α corresponds to the pth (resp. to the qth) appearance of variable x_i (resp. x_j) in ϕ. Then, by the construction of G_ϕ, the pth branch of $G_{\phi,i}$ coincides with the qth branch of $G_{\phi,j}$, i.e., $u_p^{x_i} = v_q^{x_j}$, $v_p^{x_i} = u_q^{x_j}$, $w_p^{x_i} = w_q^{x_j}$, and $t_p^{x_i} = t_q^{x_j}$. Let α be XOR-satisfied in τ, i.e., $x_i = \overline{x_j}$. If $x_i = \overline{x_j} = 0$ (i.e., $x_i = 0$ and $x_j = 1$) then we keep in L the label 1 of the edge $\{v_p^{x_i}, w_p^{x_i}\}$ and the label 2 of the edge $\{w_p^{x_i}, u_p^{x_i}\}$. In the symmetric case, where $x_i = \overline{x_j} = 1$ (i.e., $x_i = 1$ and $x_j = 0$), we keep in L the label 1 of the edge $\{u_p^{x_i}, w_p^{x_i}\}$ and the label 2 of the edge $\{w_p^{x_i}, v_p^{x_i}\}$. Let now α be

XOR-unsatisfied in τ, i.e., $x_i = x_j$. Then, in both cases where $x_i = x_j = 0$ and $x_i = x_j = 1$, we keep in L the label 1 of both edges $\{v_p^{x_i}, w_p^{x_i}\}$ and $\{w_p^{x_i}, u_p^{x_i}\}$. This finalizes the construction of L from the truth assignment τ of ϕ.

Theorem 5. *There is an assignment τ of ϕ with $|\tau(\phi)| \geq k$ iff $r(G, L) \geq 9n+k$.*

Theorem 6. *The problem of computing $r(G, L)$ on a graph G is APX-hard.*

Proof. Denote now by $\mathrm{OPT}_{\text{mon-Max-XOR(3)}}(\phi)$ the greatest number of clauses that can be simultaneously XOR-satisfied by a truth assignment of ϕ. Then Theorem 5 implies that $r(G_\phi, L_\phi) \geq 9n + \mathrm{OPT}_{\text{mon-Max-XOR(3)}}(\phi)$. Note that a random truth assignment XOR-satisfies each clause of ϕ with probability $\frac{1}{2}$, and thus there exists an assignment τ that XOR-satisfies at least $\frac{m}{2}$ clauses of ϕ. Therefore $\mathrm{OPT}_{\text{mon-Max-XOR(3)}}(\phi) \geq \frac{m}{2} = \frac{3}{4}n$, and thus, $n \leq \frac{4}{3}\mathrm{OPT}_{\text{mon-Max-XOR(3)}}(\phi)$. Assume that there is a PTAS for computing $r(G, L)$. Then, for every $\varepsilon > 0$ we can compute in polynomial time a labeling $L \subseteq L_\phi$ for the graph G_ϕ, such that $|L_\phi \setminus L| \geq (1-\varepsilon) \cdot r(G_\phi, L_\phi)$. Given such a labeling $L \subseteq L_\phi$ we can compute by the sufficiency part ($\Leftarrow$) of the proof of Theorem 5 a truth assignment τ of ϕ so that $|L_\phi \setminus L| \leq 9n + |\tau(\phi)|$, i.e., $|\tau(\phi)| \geq |L_\phi \setminus L| - 9n$. Therefore it follows that:

$$\begin{aligned}
|\tau(\phi)| &\geq (1-\varepsilon) \cdot r(G_\phi, L_\phi) - 9n \\
&\geq (1-\varepsilon) \cdot \big(9n + \mathrm{OPT}_{\text{mon-Max-XOR(3)}}(\phi)\big) - 9n \\
&\geq (1-\varepsilon) \cdot \big(\mathrm{OPT}_{\text{mon-Max-XOR(3)}}(\phi)\big) - 9\varepsilon \cdot \frac{4}{3}\mathrm{OPT}_{\text{mon-Max-XOR(3)}}(\phi) \\
&= (1-13\varepsilon) \cdot \big(\mathrm{OPT}_{\text{mon-Max-XOR(3)}}(\phi)\big)
\end{aligned}$$

That is, assuming a PTAS for computing $r(G, L)$, we obtain a PTAS for the monotone Max-XOR(3) problem, which is a contradiction by Lemma 1, unless $P = NP$. So, computing $r(G, L)$ on an undirected graph G is APX-hard. □

4.3 Random labelings on dense graphs have high removal profit

We show here that dense graphs with random labels satisfy TC and have a very high removal profit with high probability (whp).

Definition 10. *We call* normalized uniform random temporal graph *any graph on $n \in \mathbb{N}$ vertices, each edge of which receives exactly one label uniformly at random, and independently of other edges, from the set $\{1, \ldots, n\}$.*

Theorem 7. *(a) In the normalized uniform random temporal clique of n vertices, we can delete all but $2n + O(\log n)$ labels without violating TC whp.*

(b) Let $G = (V, E)$ be an instance of the Erdös-Renyi model of random graphs, $G_{n,p}$, with $p \geq \frac{a\sqrt{n}\log n}{n}$, where a is constant, and consider a normalized uniform random temporal graph, $G(L)$, on G. We can delete all but $2n + O(\sqrt{n})$ labels of $G(L)$ without violating TC whp.

Sketch of proof. We provide a sketch of the proof of (a). Partition the set of available labels $\{1, 2, \ldots, n\}$ into 4 consecutive equisized subsets $A_1, \ldots, A_4$. Each edge receives a single random label l, with $Pr[l \in A_i] = \frac{1}{4},\ \forall i = 1, 2, 3, 4$. Color *green*$(g)$, *yellow*$(y)$, *blue*$(b)$ and *red*(r) the edges that are assigned a label in A_1, A_2, A_3 and A_4 respectively. A *temporal router* is a subgraph of the clique consisting of a central vertex with a number of yellow incident edges and a number of blue incident edges. Fix a vertex u of the graph. By use of Chernoff bounds, we show the following:

Lemma 2. *There is a set S_1 of at least $\frac{n}{4}$ yellow edges incident to u and a set S_2 of at least $\frac{n}{4}$ blue edges incident to u, with probability at least $1 - 2e^{-\frac{n}{16}}$.*

Conditioning on the above property of u, we arbitrarily select a subset D_i of S_i with $|D_i| = \alpha \log n,\ i = 1, 2$. $R = D_1 \cup D_2 \cup \{u\}$ is then a $O(\log n)$-size temporal router.

Lemma 3. *Any vertex $w \in V \setminus R$ has an incident g edge to a vertex in D_1 and an incident r edge to a vertex in D_2 with probability at least $1 - 2e^{-\frac{\alpha \log n}{4}}$.*

Using Lemma 3, we show that whp, we can remove all the labels from the random labeling on the graph except for the labels on the edges of the "router" and the two incident edges of any $w \in V$, one g connecting it to a vertex in D_1 and one r connecting it to a vertex in D_2, and still satisfy the property TC. □

Acknowledgments *We wish to thank Thomas Gorry for co-implementing the code used in the proof of Theorem 4.*

References

1. E. C. Akrida, L. Gąsieniec, G. B. Mertzios, and P. G. Spirakis. Ephemeral networks with random availability of links: Diameter and connectivity. In *Proceedings of the 26th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2014.
2. P. Alimonti and V. Kann. Hardness of approximating problems on cubic graphs. In *Proceedings of the Third Italian Conference on Algorithms and Complexity*, CIAC '97, pages 288–298, London, UK, UK, 1997. Springer-Verlag.
3. C. Avin, M. Koucký, and Z. Lotker. How to explore a fast-changing world (cover time of a simple random walk on evolving graphs). In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 121–132, 2008.
4. B.-M. Bui-Xuan, A. Ferreira, and A. Jarry. Computing shortest, fastest, and foremost journeys in dynamic networks. *International Journal of Foundations of Computer Science*, 14(2):267–285, 2003.
5. A. Casteigts and P. Flocchini. Deterministic Algorithms in Dynamic Networks: Formal Models and Metrics. Technical report, Defence R&D Canada, April 2013.
6. A. Casteigts and P. Flocchini. Deterministic Algorithms in Dynamic Networks: Problems, Analysis, and Algorithmic Tools. Technical report, Defence R&D Canada, April 2013.

7. A. Casteigts, P. Flocchini, W. Quattrociocchi, and N. Santoro. Time-varying graphs and dynamic networks. *International Journal of Parallel, Emergent and Distributed Systems (IJPEDS)*, 27(5):387–408, 2012.
8. A. E. F. Clementi, C. Macci, A. Monti, F. Pasquale, and R. Silvestri. Flooding time of edge-markovian evolving graphs. *SIAM Journal on Discrete Mathematics (SIDMA)*, 24(4):1694–1712, 2010.
9. C. Dutta, G. Pandurangan, R. Rajaraman, Z. Sun, and E. Viola. On the complexity of information spreading in dynamic networks. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 717–736, 2013.
10. L. Fleischer and M. Skutella. Quickest flows over time. *SIAM J. Comput.*, 36(6):1600–1630, 2007.
11. L. Fleischer and É. Tardos. Efficient continuous-time dynamic network flow algorithms. *Operations Research Letters*, 23(3-5):71–80, 1998.
12. A. Gupta, R. Krishnaswamy, and R. Ravi. Online and stochastic survivable network design. *SIAM J. Comput.*, 41(6):1649–1672, 2012.
13. D. Kempe, J. M. Kleinberg, and A. Kumar. Connectivity and inference problems for temporal networks. In *Proceedings of the 32nd annual ACM symposium on Theory of computing (STOC)*, pages 504–513, 2000.
14. B. Klinz and G. J. Woeginger. One, two, three, many, or: complexity aspects of dynamic network flows with dedicated arcs. *Oper. Res. Lett.*, 22(4-5):119–127, 1998.
15. R. Koch, E. Nasrabadi, and M. Skutella. Continuous and discrete flows over time - A general model based on measure theory. *Math. Meth. of OR*, 73(3):301–337, 2011.
16. S. C. Kontogiannis and C. D. Zaroliagis. Distance oracles for time-dependent networks. In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, pages 713–725, 2014.
17. F. Kuhn, N. A. Lynch, and R. Oshman. Distributed computation in dynamic networks. In *Proceedings of the 42nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 513–522, 2010.
18. L. C. Lau, J. Naor, M. R. Salavatipour, and M. Singh. Survivable network design with degree or order constraints. *SIAM J. Comput.*, 39(3):1062–1087, 2009.
19. L. C. Lau and M. Singh. Additive approximation for bounded degree survivable network design. *SIAM J. Comput.*, 42(6):2217–2242, 2013.
20. G. B. Mertzios, O. Michail, I. Chatzigiannakis, and P. G. Spirakis. Temporal network optimization subject to connectivity constraints. In *Proceedings of the 40th International Colloquium on Automata, Languages and Programming (ICALP), Part II*, pages 657–668, 2013.
21. O. Michail, I. Chatzigiannakis, and P. G. Spirakis. Causality, influence, and computation in possibly disconnected synchronous dynamic networks. In *OPODIS*, pages 269–283, 2012.
22. R. O'Dell and R. Wattenhofer. Information dissemination in highly dynamic graphs. In *Proceedings of the 2005 joint workshop on Foundations of Mobile Computing (DIALM-POMC)*, pages 104–110, 2005.
23. C. Scheideler. Models and techniques for communication in dynamic networks. In *Proceedings of the 19th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 2285, pages 27–49, 2002.