Optimizing Reachability Sets in Temporal Graphs by Delaying

Paper 5844

Abstract

A temporal graph is a dynamic graph, where every edge is assigned a set of integer time labels that indicate at which discrete time step the edge is available. In this paper, we study how changes of the time labels, corresponding to delays on the availability of the edges, affect the reachability sets from given sources. The questions about reachability sets are motivated by numerous applications of temporal graphs in network epidemiology, or scheduling problems in supply networks in manufacturing. We study the computational complexity of several optimization problems with different reachability objectives and in particular the control mechanism based on natural operations of delaying time events. The first one, termed merging, is global and batches together consecutive time labels in the whole network simultaneously, which corresponds to postponing all events until a particular time. The second one, imposes independent delays on the time labels of every edge of the graph. We provide a thorough investigation of the computational complexity of different objectives related to reachability sets when these operations are used. For the merging operation, we prove NP-hardness results for several minimization and maximization reachability objectives, even for very simple graph structures. For the second operation, we prove that the minimization problems are NP-hard when the number of allowed delays is bounded. We complement this with a polynomial-time algorithm for the case of unbounded delays.

Introduction

A plethora of real life scenarios can be modelled as a dynamic network that changes over time. These scenarios range from train, bus, and distribution schedules, to livestock movements between farms and virus spreading. Many of these dynamic networks consist of a fixed set of nodes and what changes over time is the connectivity between pairs of them; the locations of stations, distribution centers, and farms remain the same over time, while the connections between any two of them can change every few minutes, hours, or days. An equivalent way to see these networks is as a sequence of static networks that change according to a predetermined, known in advance, schedule. These type of networks, known as *temporal graphs*, were formalized in the seminal work of (Kempe, Kleinberg, and Kumar 2002). Since then, there is flourish of work on temporal graphs (Zschoche et al. 2018; Michail 2016; Casteigts et al. 2019; Chen et al. 2018; Michail and Spirakis 2016; Erlebach, Hoffmann, and Kammer 2015). In a temporal graph every edge is assigned a set of integer time labels that indicate at which discrete time steps the edge is available. In other words, a temporal graph is a schedule of edge sets $E_1, E_2, \ldots, E_t$ over a fixed set of vertices.

In this paper, we study questions related to *reachability sets* on temporal graphs. The reachability set of vertex v is the set of vertices that there exist *temporal paths* from v to them. Informally speaking, a temporal path is a path whose edges, additionally to the usual connectivity, use strictly increasing time labels (Whitbeck et al. 2012). However, in contrast to static graphs, temporal paths do not preserve transitivity. As a result, some of well known concepts from graph theory, like Menger's theorem, do not hold for temporal graphs and had to be redefined (Akrida et al. 2019; 2017; Mertzios et al. 2013; Erlebach and Spooner 2018).

Reachability sets emerge naturaly in several real life applications. One of them, is the minimization of spread of infectious diseases over contact networks. Data provide significant evidence that commuter patterns and airline flights (Colizza et al. 2006; Brockmann and Helbing 2013), and livestock movements between farms (Mitchell et al. 2005) could spread an infectious disease. The importance of these problems combined with their inherit temporal nature, made temporal graph theory a tool for analyzing epidemiology in (animal) networks (Braunstein and Ingrosso 2016; Enright and Meeks 2015; 2018; Enright et al. 2018; Enright and Kao 2018; Nöremark and Widgren 2014; Valdano et al. 2015a; 2015b). Another application is the minimization of the cost of distribution schedules. In this case, the goal to utilize in the best way the infrastructure of the network and thus *maximize* the reachability sets of the sources by utilizing the network infrastructure in the best possible way.

(Enright and Meeks 2018) and (Enright et al. 2018) studied how the reachability sets on temporal graphs change, when the edge schedule can change according to specific operations. More specifically they studied, MINMAXREACH

and MAXMINREACH problems where the goal, respectively, was to minimize the maximum, or to maximize the minimum size of reachability sets between a given set of sources. (Enright and Meeks 2018) studied these objectives under the operation of *reordering* of the edge sets and it was proven that both problems are NP-hard. However, in many real life scenarios, reordering can be difficult, costly, or even impossible to perform due to the physical constraints of the network, or due to the number of changes required in the existing infrastructure. For MINMAXREACH, (Enright et al. 2018) studied the operations of deletion of whole edge sets, or individual edges within an edge set. Under these operations, an upper bound on the number of allowed deletions is required. This is crucial, since the deletion of every edge trivially minimizes the reachability sets, but makes the existing network infrastructure useless. In addition, edge deletions can create a "bottleneck" problem in the network even if their number is bounded. More specifically, the deletion of an edge can create a sink to the network, or completely isolate some of its parts. It was proven that MINMAXREACH under both notions of deletion is NP-hard, but they admit an FPT algorithm when they are parameterized by the treewidth of the network and the number of non-reachable vertices.

Although optimization of reachability sets capture important real life problems, some of the proposed solutions are not completely satisfying due to big changes in infrastructure. Instead, we wish to study the following problem.

Given a temporal graph and a set of sources, optimize the size of the reachability set of the sources using only "natural" *and* "infrastructure-friendly" *operations.*

Natural operations are those that can be naturally applied on a given temporal graph. For example, the deletion, or the postponement, of a temporal edge can naturally happen. On the other hand, bringing forward a temporal edge cannot be always feasible; a bus cannot go faster than its top speed. Infrastructure-friendly operations should not be to difficult to perform and that do not require many changes to the given network and temporal schedule.

Our contribution. Our contribution is twofold. Firstly, we introduce and study two operations that are natural and infrastructure-friendly and were not studied in the past. We introduce the *merging* and the *delaying* operations. The idea behind both operations is the postponement of the edges of the graph. The merging operation is parameterized by λ and it batches together λ consecutive edge sets; a λ-merge on $E_1, \dots E_\lambda$ changes the first $\lambda - 1$ edge sets to the empty sets, and the λ-th edge set is the union of all λ edge sets. The delaying operation, independently delays a temporal edge; a δ-delay on the label i of edge uv changes it to $i + \delta$. Both operations are natural, easy to understand, and infrastructure-friendly. In addition, in contrast to deletion of temporal edges that can directly isolate vertices, our operations isolate some vertices *only temporarily*. Our second contribution is a thorough investigation of the computational complexity of reachability objectives under these operations. In particular, we study the MINMAXREACH, MINREACH, and MINAVGREACH, where in the last two cases the goal is to minimize the number, or the average number respectively, of reachable vertices from a given set of sources.With respect to maximization objectives, we study MAXREACH, MAXMINREACH, and MAXAVGREACH. We proved that these problems are NP-hard under the merging operation even for very restricted classes of graphs. Our results are depicted in Table 1. For the delaying operation, we studied the minimization problems. They remain NP-hard when we *bound* the number of times we are allowed to use this operation. We complement these results with a polynomial time algorithm for the case of the unbounded number of delays that works for any δ.

Problem	Graph Class	Sources	Labels/ Edge	Edges/ Step
MINREACH	Path	$O(n)$	1	3
MINREACH MINMAXREACH MINAVGREACH	Tree $\Delta = 3$	1	1	1
MAXREACH	Path	$O(n)$	1	4
MAXREACH MAXMINREACH MAXAVGREACH	Bipart. $\Delta = 3$	1	1	4
MAXREACH MAXMINREACH MAXAVGREACH	Tree $\Delta = 3$	1	1	10

Table 1: NP-hardness results for the λ-merge operation for any $\lambda \geq 2$. Δ denotes the maximum degree of the graph. The results are approximation preserving and they rule out PTASs and they hold even if the under lying graph is directed.

Preliminaries

Temporal Graphs. A *temporal graph* $\langle G, \mathcal{T} \rangle$ is a pair of a (directed) graph $G = (V, E)$ and a function $\mathcal{T}$ that maps every edge of G to a list of time steps at which the edge is available. The maximum time step, $t_{\max}$, on the edges of G defines the *lifetime* of the temporal graph. Another interpretation of a temporal graph, which is useful in our case, is to see it as a schedule of subgraphs, or edge-sets, $E_1, E_2, \dots, E_{t_{\max}}$ of G, known in advance and defined by the function $\mathcal{T}$; at time step t function $\mathcal{T}$ defines a set $E_t \subseteq E$ of edges available for this time step. We will say that an edge has the label i, if it is available at time step i. The size of a temporal graph $\langle G, \mathcal{T} \rangle$ is $|V| + \sum_{t \leq t_{\max}} |E_t|$. A *temporal path* in $\langle G, \mathcal{T} \rangle$ from v_1 to v_k is a sequence of edges $v_1v_2, v_2v_3, \dots, v_{k-1}v_k$ such that each edge v_iv_{i+1} is available at time step t_i and $t_i < t_{i+1}$ for every $i \in [k-1]$.

Reachability Sets. A vertex u is *reachable* from vertex v if there exists a temporal path from u to v. We assume that a vertex is reachable from itself. It is possible that u is reachable from v, but v is not reachable from u. The reachability set of v, denoted $\texttt{reach}(v, \langle G, \mathcal{T} \rangle)$, is the set of vertices reachable from v in $\langle G, \mathcal{T} \rangle$. Given a temporal graph with lifetime $t_{\max}$ and a time step $t < t_{\max}$, the set

$\texttt{reach}_t(v, \langle G, \mathcal{T}\rangle)$ contains all the vertices reachable from v up to time t. The set $\texttt{reach}(v, \langle G, \mathcal{T}\rangle)$ can be computed in polynomial time with respect to the size of $\langle G, \mathcal{T}\rangle$; it suffices to check whether there exists a temporal path from v to every vertex in V, which can be done efficiently (Wu et al. 2014).

Merging. A *merging* operation on $\mathcal{T}$ postpones some of the edge-sets in a particular way. Intuitively, a merging operation "batches together" a number of consecutive edge-sets.

Definition 1 (λ-merge) *For every positive integer λ, a λ-merge of $E_i, E_{i+1}, \ldots, E_{i+\lambda-1}$, replaces $E_j = \emptyset$ for every $i \leq j < i + \lambda - 1$ and $E_{i+\lambda-1} = \bigcup_{i \leq j \leq i+\lambda-1} E_j$.*

So, every merge corresponds to the global delay of events from times $i, i+1, \ldots, i+\lambda-1$ to the time $i+\lambda-1$. We say that two λ-merges are *feasible*, if they merge $E_i, \ldots E_{i+\lambda-1}$ and $E_j, \ldots E_{j+\lambda-1}$ respectively, and $i + \lambda - 1 < j$. When it is clear from the context, instead of writing that we merge E_i with E_{i+1} we will write that we merge i with $i + 1$.

Definition 2 ((λ, μ)-merging scheme) *For positive integers λ and μ, a (λ, μ)-*merging scheme *applies μ independent λ'-merges on $E_1, E_2, \ldots, E_{t_{\max}}$, where $\lambda' \leq \lambda$. A merging scheme is* maximal *if there is no other feasible λ-merge available.*

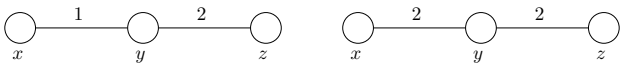

Figure 1: Left: a temporal graph where $E_1 = xy$ and $E_2 = yz$; vertex z is reachable from x. Right: the resulted graph after merging E_1 with E_2; vertex z is no longer reachable from x.

A (λ, μ)-merging scheme for a temporal graph $\langle G, \mathcal{T}\rangle$ essentially produces a new temporal graph by modifying the schedule $\mathcal{T}$ using μ times a λ-merge operation. We will use $\mathcal{T}^M_{(\lambda,\mu)}$ to denote the modified schedule and $\langle G, \mathcal{T}^M_{(\lambda,\mu)}\rangle$ the corresponding modified temporal graph.

Our goal is to compute merging schemes that optimise some objectives regarding reachability sets from a given set of vertices. The input to the problems we study consists of a temporal graph $\langle G, \mathcal{T}\rangle$, two positive integers λ, μ, and a subset of vertices $S \subseteq V$ which will term *sources*. The objectives we study are formally defined in Table 2.

Delaying. While merging operations affect globally the whole graph; *edge delays* affect only one label of an edge. We parameterize the delay operation by δ; the maximum delay we can impose on a label of an edge. Hence, a δ-delay on edge uv at time step i changes the label i to $i' \leq i + \delta$. We denote $\mathcal{T}^D_\delta$ the schedule produced after applying δ-delay operations on $\mathcal{T}$. When the number of allowed δ-delay operations is κ, we denote $\mathcal{T}^D_{(\delta,\kappa)}$ produced schedule. Under delaying operations we study MINREACH, MINMAXREACH, and MINAVGREACH.

$(\lambda, \mu) -$ MERGING	Objective
1. MINREACH	$\min \lvert \bigcup_{v \in S} \texttt{reach}(v, \langle G, \mathcal{T}^M_{(\lambda,\mu)}\rangle)\rvert$
2. MINMAXREACH	$\min \max_{v \in S} \lvert\texttt{reach}(v, \langle G, \mathcal{T}^M_{(\lambda,\mu)}\rangle)\rvert$
3. MINAVGREACH	$\min \sum_{v \in S} \lvert\texttt{reach}(v, \langle G, \mathcal{T}^M_{(\lambda,\mu)}\rangle)\rvert$
4. MAXREACH	$\max \lvert \bigcup_{v \in S} \texttt{reach}(v, \langle G, \mathcal{T}^M_{(\lambda,\mu)}\rangle)\rvert$
5. MAXMINREACH	$\max \min_{v \in S} \lvert\texttt{reach}(v, \langle G, \mathcal{T}^M_{(\lambda,\mu)}\rangle)\rvert$
6. MAXAVGREACH	$\max \sum_{v \in S} \lvert\texttt{reach}(v, \langle G, \mathcal{T}^M_{(\lambda,\mu)}\rangle)\rvert$

Table 2: Problems 1 - 3 are minimization problems, while Problems 4 - 6 are maximization problems. If $|S| = 1$, then the solution for all minimization problems is the same; similarly for the maximization problems.

MAX2SAT(3). To produce many of our results we use the problem MAX2SAT(3). An instance of MAX2SAT(3) is a CNF formula ϕ with n boolean variables and m clauses, where each clause involves exactly two variables and every variable appears in at most three clauses. The goal is to maximize the number of satisfied clauses. Without loss of generality we will assume that every variable in ϕ appears exactly one time as a positive literal and at most two times as a negative literal. In (Berman and Karpinski 1999) it was proven that MAX2SAT(3) is NP-hard and that it is even hard to approximate better than 1.0005.

Merging: Minimization problems

In this section we study minimization problems under merging operations. To begin with, we prove that MINREACH under $(2, \mu) -$ MERGING is NP-hard even when G is a path with many sources. Then, using this result, we explain how to get NP-hardness any $\lambda \geq 2$. Next, we show how to extend our construction and get a bipartite, planar graph of maximum degree 3 and only one source, and thus we prove NP-hardness for all Problems 1 - 3. Our next result is NP-hardness for the same set of problems on trees with one source. For this result, we derive a new construction. Although all of our results are presented for undirected graphs, they can be trivially extended for directed graphs. In addition, all of our reductions are approximation preserving hence we get that all Problems 1 - 3 are NP-hard even to approximate.

MINREACH on paths

Construction. We will reduce from MAX2SAT(3). The k-th clause of ϕ will be associated with the time steps $4k, 4k + 1$ and $4k + 2$, while the i-th variable will be associated with the time steps $M + 4i, M + 4i + 1$ and $M + 4i + 2$, where $M = 4m + 4$. For every clause of ϕ we construct a path with nine vertices, where the middle vertex is a source. Consider the path for the k-th clause, that involves the variables x_i and x_j. An example of such a path can be found on Figure 2. The middle piece of the path consists of the vertices $c_k, y^l_k, z^l_k, y^r_k, z^r_k$; where $c_k \in S$. Edge $c_k y^l_k$ has the label $4k$,

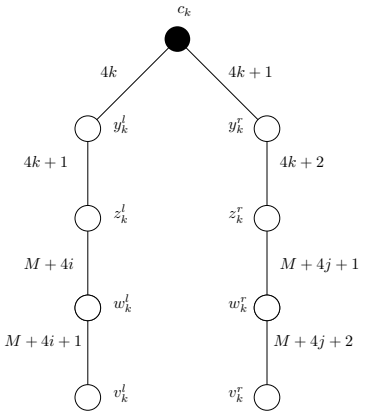

Figure 2: The gadget for the clause $(\bar{x}_i, x_j)$. The labels on the edges denote the time steps these edges are available; $M = 4m + 4$. The black vertex, c_k, is a source, i.e., it belongs to S.

edges $y_k^l z_k^l$ and $c_k y_k^r$ have the label $4k + 1$, and edge $y_k^r z_k^r$ has the label $4k + 2$. The labels on the left and the right pieces of the path depend on the literals of the variables of the clause. If variable x_i appears in the clause with a positive literal, then we pick an arbitrary side of the path, say the left, and add the label $M + 4i + 1$ to the edge $z_k^l w_k^l$ and the label $M+4i+2$ to the edge $w_k^l v_k^l$. If x_i appears in the clause with a negative literal, then we add the label $M + 4i$ to the edge $z_k^l w_k^l$ and the label $M + 4i + 1$ to the edge $w_k^l v_k^l$. In order to create a single path, we arbitrarily connect the endpoints of the paths we created for the clauses; every edge that connects two such paths has label 1. Observe that for the constructed temporal graph, $\langle G, \mathcal{T}\rangle$, the following hold: $\langle G, \mathcal{T}\rangle$ has $9m$ vertices and lifetime $4m + 4n + 6$; every vertex is reachable from a one of sources; at every time step there are at most three edges available; every edge has only one label, i.e., it is available only at one time step. Clearly, the size of $\langle G, \mathcal{T}\rangle$ is polynomial to the size of ϕ and $|S| = m$. We will ask for a $(2, m + n)$-merging scheme.

Intuition. The correctness of our reduction relies on two ideas. The first idea is that in every subpath, under any $(2, \mu)$-merging scheme, *at most* four vertices are not reachable from S. In order to make four vertices not reachable within a gadget, the following synergy must happen. Vertex c_k should choose which side of the path to "save"; merging $4k$ with $4k + 1$ make the vertices z_k^l, w_k^l, v_k^l unreachable; merging $4k+1$ with $4k+2$ make the vertices z_k^r, w_k^r, v_k^r unreachable. Observe that only one of the two merges can happen, since the two merges together are not feasible. Hence, such a merge makes three vertices of one side unreachable. In order to make the v-vertex of the other side unreachable, one extra merge has to happen. This merge will be translated to a truth assignment for a variable, which is the second idea of our reduction. The merge of $M + 4i$ with $M + 4i + 1$ corresponds to setting x_i to False and the merge of $M + 4i + 1$ with $M + 4i + 2$ to True.

Lemma 1 *If there exists an assignment that satisfies l clauses of ϕ, then there exists a $(2, n + m)$-merging scheme such that $3m + l$ vertices are not reachable from S.* [1]

Lemma 2 *If there exists an optimal $(2, n + m)$-merging scheme such that $3m + l$ vertices are not reachable from S, then there exists an assignment for ϕ that satisfies l clauses.*

The combination of Lemmas 1 and 2 already yield NP-hardness for MINREACH under $(2, \mu)$ − MERGING, when μ is part of the input. Furthermore, our construction allows us to restrict the search only to maximal merging schemes. In Lemma 1 we can arbitrarily extend the produced merging scheme to a maximal one without any issues in the correctness of the proof, while in Lemma 2 it is without loss of generality to assume that the merging scheme we consider is maximal.

We furthermore explain how to get NP-hardness for any $\lambda \geq 2$. Although our construction does not work when $\lambda > 2$, we can modify it and get the result for any $\lambda \geq 2$, by "spreading" the edges of $\langle G, \mathcal{T}\rangle$ over time. More formally, between any two time steps of $\langle G, \mathcal{T}\rangle$ we add $\lambda - 2$ dummy time steps where no edge is available. The crucial observation is that, after the spreading, any λ'-merge with $\lambda' < \lambda$ does not affect the set reachable from S. Hence, in the proofs of Lemmas 1 and 2 if we replace the 2-merges with λ-merges we get NP-hardness for any λ.

Theorem 1 MINREACH *under* (λ, μ) − MERGING *is* NP*-hard for any $\lambda \geq 2$ even when G is a path the following constraints hold:*

1. *every edge is available only at one time step;*
2. *at any time step there are at most three edges available;*
3. *the merging scheme is maximal.*

Graphs with a single source

Next, we explain how to get NP-hardness for the case where $|S| = 1$. Such a result immediately implies NP-hardness for MINMAXREACH and MINAVGREACH under (λ, μ) − MERGING, since when $|S| = 1$ these problems coincide with MINREACH. It is not hard to get this result, given the previous construction.We can simply add a vertex c_0 in the constructed graph that would be the only source and it is connected with every vertex that used to be a source, so c_0 is connected with all the c-vertices from the previous construction, with an edge available at time step 1. Since the next time step that an edge exists in is time step 4, this means that under *any* $(2, \mu)$-merging scheme, at time step 2 every vertex of S has been reached by c_0. Hence, the previous result follows. However, this is not the strongest result we can get, since this graph has a vertex with non constant degree. Instead, we modify $\langle G, \mathcal{T}\rangle$ as we describe next in order to get a tree of maximum degree 3.

The construction is depicted in Figure 6. We create a path with $m + 1$ vertices where c_0 is the only source. In addition, every vertex k of the path, different than c_0, is connected with vertex c_k from the construction of the previous section via an edge with label $2k + 2$. The labels in any two consecutive edges of this path differ by two. Finally, every edge

[1] Due to lack of space, all of our formal proofs are omitted.

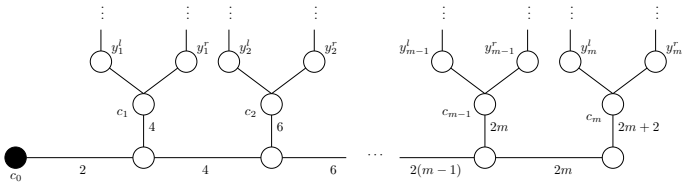

Figure 3: The construction for graphs with one source.

of the construction from the previous section is shifted by $2m + 4$, i.e., we add the number $2m + 4$ to its label. The crucial observation is that under any merging scheme that uses 2-merges at time step $2m + 2$ every c_k vertex for every $k \in [m]$ will be reached. Hence, at time step $2m + 2$, under any $(2, \mu)$-merging scheme, we get an instance that it is equivalent, with respect to reachability from this time step and on, to the instance of the previous section. Hence, we can get NP-hardness for $\lambda = 2$.Using the trick of "spreading" described before, we can get the result for any $\lambda \geq 2$.

Theorem 2 *Problems 1 - 3 are* NP*-hard for any $\lambda \geq 2$ even when there exists only one source, G is a tree of maximum degree three, and the Constraints 1- 3 from Theorem 1 hold.*

Merging: Maximization problems

In this section we prove NP-hardness for maximization problems. Before we proceed with the exposition of the results though, we should discuss some issues about maximization reachability problems and merging. Any merge weakly decreases the reachability set from the sources. Hence. while in the minimization problems in principle we would like to perform as many merges as possible, for maximization problems we would like to perform the minimum number of merges that are allowed. In addition, the reachability set weakly decreases with λ, i.e., the size of the merge. Hence, for maximization problems it is better to do the smallest merge possible, i.e., perform only 2-merges. For this reason, if we get NP-hardness for $(2, \mu)$-merging schemes, then we can immediately conclude that finding the optimal (λ, μ)-merging scheme is NP-hard, for any $\lambda \geq 2$. So, for maximization problems, we *require* that *at least* μ λ-merges need to happen. This is motivated by distribution networks the use of the network comes at a cost and thus where we would like to use the networks as fewer times as possible.

Again, we prove our results by reducing to MAX2SAT(3). This time though we need to be more careful; in order to make our reduction valid, we should not allow for "dummy" merges, i.e., merges that do not change the reachable vertices from the sources. As before, we first prove NP-hardness for paths with multiple sources. Then, we modify our reduction to get NP-hardness for graphs with one source and we provide a reduction for trees with just one source.

MAXREACH on paths

Construction. We will reduce from MAX2SAT(3). Every variable x_i of ϕ will be associated with the time steps $3i, 3i+1$, and $3i + 2$. For every variable we create a path of length 5 with ends the vertices y_i and z_i; this path is depicted at the bottom of Figure 4. In this path, the edges at the ends of the paths have labels $3i + 2$ and the rest of the edges have label $3i + 3$. The paths that are created from variables, we will call them as variable-paths. Both ends of every variable-path are sources. For every clause we create a path with five vertices. So, if the k-th clause involves the variables x_i and x_j, we construct a path with ends the vertices v_k^i and v_k^i, and middle the vertex c_k, which will be called as c-vertex. If x_i appears with a positive literal in the clause, then the labels on the two edges that connect v_k^i and c_k are $3i$ and $3i+1$ respectively; else the labels are $3i+1$ and $3i+2$. The top side of Figure 4 depicts the path for the clause $(\bar{x}_i, x_j)$. Both ends of the path are sources. These paths will be termed clause-paths. To create one path, we arbitrarily connect the constructed paths. Observe that and temporal graph $\langle G, \mathcal{T} \rangle$ we constructed has $6n+5m$ vertices, lifetime $3n+2$, at any time step at most four edges are available, and furthermore all the vertices are reachable from S.

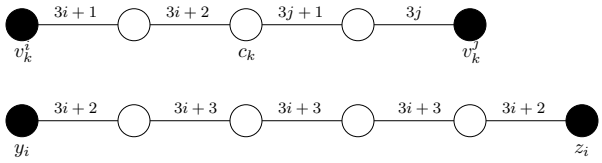

Figure 4: Gadgets for MAXREACH on paths. Top: the gadget for the clause $(\bar{x}_i, x_j)$. Bottom: the gadget for variable x_i.

Intuition. The labels on the variable-paths guarantee that there exists a $(2, n)$-merging scheme that maximizes the number of reachable vertices such that it does not apply a 2-merge that merges time step $3i+2$ with $3i+3$ for any $i \in [n]$. This guarantees two things. Firstly, there exists an optimal $(2, n)$-merging scheme where all the vertices, except the c-vertices, are reachable from S. Hence, any optimal merging scheme has to maximize the number of reachable c-vertices. Second, given the subset of c-vertices that are reachable under the produced merging scheme, we can easily deduce a truth assignment that satisfies the clauses that correspond to the reachable c-vertices.

In what follows, we will assume that we do not allow any merge that involves time steps 1 and 2. This is because such merges would be "dummy", since there are no edges available at these time steps. However, this assumption is without loss of generality; we can simply decrease all the labels by 2, hence an edge with label i will get label $i - 2$.

Lemma 3 *If there exists an assignment that satisfies l clauses of ϕ, then there exists a $(2, n)$-merging scheme such that $6n + 4m + l$ vertices are reachable from S.*

Lemma 4 *If there exists an optimal $(2, n)$-merging scheme such that $6n + 4m + l$ vertices are reachable from S,then there exists an assignment for ϕ that satisfies l clauses.*

The combination of Lemmas 3 and 4 imply that MAXREACH under $(2, \mu)-$MERGING is NP-hard. However,

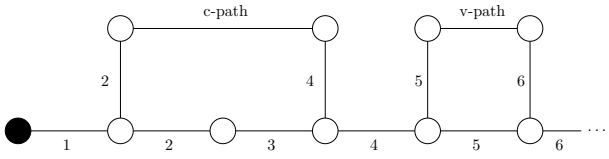

Figure 5: The construction used to prove NP-hardness for MAXREACH with one source. c-path and v-path stand for a clause-path and variable-path respectively from the previous section where we add $n+m+1$ to the labels of these paths. The filled vertex is the only source of the graph.

as we have already explained this means that the problem is NP-hard for any $\lambda \geq 2$.

Theorem 3 MAXREACH *under* (λ, μ) $-$ MERGING *is* NP*-hard for any* $\lambda \geq 2$*, even when the underlying graph is a path, every edge has one label, and at any time step there exist at most four edges available.*

Graphs with a single source

In this section we explain how to modify the construction of Section 6 in order to prove NP-hardness for the case where there is only one source. Thus, we can get as a corollary the NP-hardness for MAXMINREACH and MAXAVGREACH.

Construction. The idea is similar to the one we used before: we construct a tree whose leaves are the sources of the previous construction, so then we can use the result of the previous section. To achieve this, we will use the gadget depicted in Figure 5. This gadget consists of a line of length $2m+n$. The left end of the line has degree 1 and it is the only source, while the rest of the vertices of the line have degree 2 or 3. On the degree-3 vertices we add the clause-paths and the variable-paths, depicted as c-paths and v-paths in the figure, by adding $m+n+1$ to their labels. The end points of c-paths are connected with the j-th and the $j+2$-th vertices of the gadget where the $j+1$-th vertex has degree 2; this is done in order to get a bipartite graph. The end points of c-paths are connected with the j-th and the $j+1$-th vertices. The crucial observation is that in an optimal $(2, n)$-merging scheme we do not merge any time step lower than $2m+n+1$. Any such merge will make unreachable at least 5 or 6 vertices, the vertices of a clause-path or a variable-path, while a merge at a time step higher than $2m+n+1$ will make unreachable at most 2 vertices. Hence, at time step $2m+n+1$ all the vertices that correspond to the endpoints of the paths from the previous section have been reached by the source, so we can use the proof from the previous section and get our result.

Theorem 4 *Problems 4 - 6 are* NP*-hard even when there is one source, the underlying graph is bipartite, it has maximum degree 3, every edge has one label, and at any time step there exist at most four edges available.*

Trees with one source

In this section we prove that MAXREACH is NP-hard even on trees with only one source. Again, our reduction is from MAX2SAT(3). First, we will explain how to get the NP-hardness result for forests where every connected component of the forest has only one source. Then, using the idea from the previous section we can connect the components of the forest and create a single tree with only one source.

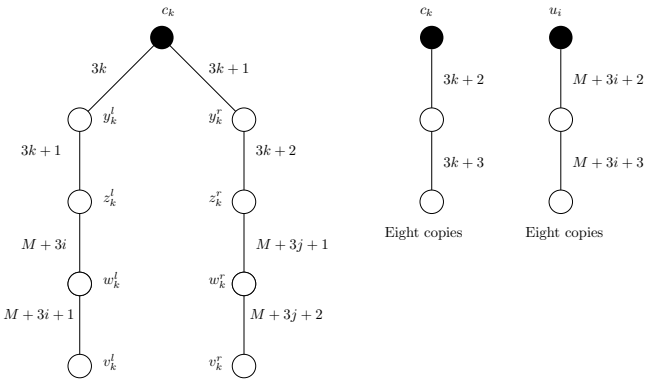

Figure 6: The gadgets we use in Section 6, where $M = 3m+2$. The left gadget corresponds to the k-th clause of ϕ equal to $(x_i, \bar{x}_j)$; the middle gadget is for the k-th clause; the right gadget is for the i-th variable

Construction. We will use a similar approach as in Section 6 and we will associate each variable with three consecutive time steps. This time though we will associate every clause with three consecutive time steps as well. For every clause we create a tree with 9 vertices where only one is a source; an example of this tree is depicted in the left part of Figure 6. Each such tree consists of three pieces: the middle piece, the left piece, and the right piece. For the k-th clause, the middle piece consists of the vertices $c_k, y_k^l, z_k^l, y_k^r, z_k^r$; c_k is a source. Edge $c_k y_k^l$ has the label $3k$, edges $y_k^l z_k^l$ and $c_k y_k^r$ have the label $3k+1$, and edge $y_k^r z_k^r$ has the label $3k+2$. The labels on the left and the right pieces of the path depend on the literals of the variables of the clause. If variable x_i appears in the clause with a positive literal, then we pick an arbitrary side of the path, say the left, and add the label $M+3i$ to the edge $z_k^l w_k^l$ and the label $M+3i+1$ to the edge $w_k^l v_k^l$, where $M = 3m+2$. If x_i appears in the clause with a negative literal, then we add the label $M+3i+1$ to the edge $z_k^l w_k^l$ and the label $M+3i+2$ to the edge $w_k^l v_k^l$. In addition, we create the following. For every $k \in [m]$ we create eight paths of length 2; the middle part of Figure 6. The one end point of every such path is a source. The labels of the edges for every path, from the source to the other end, are $3k+2$ and $3k+3$. For every $i \in [n-1]$ we create eight paths of length 2; the right part of Figure 6. The one end point of every path is a source. The labels of the edges for every path, from the source to the other end, are $M+3i+2$ and $M+3i+3$. The constructed temporal graph $\langle G, \mathcal{T} \rangle$ has $33m+24n$ vertices and lifetime $3m+3n+4$. Observe that from all the vertices of the graph are reachable from S. We will ask for a $(2, m+n)$-merging scheme. In addition, observe that at any time step there exist at most 10 edges available. Again, we will assume without loss of generality that no merging that involves labels 1 and 2 is allowed.

Intuition. The high level idea is that in any optimal $(2, m+n)$-merging scheme all the vertices of the paths of length 2 are reachable; this is guaranteed by the number of

copies of the paths and the choice of the labels in these paths. Hence, an optimal merging scheme maximizes the number of reachable vertices in the gadgets for the clauses. In every gadget though, at most 6 vertices are reachable under any optimal merging scheme. If 6 vertices are reachable, then we can easily deduce an assignment that satisfies this clause.

Lemma 5 *If there exists an assignment for the variables of ϕ that satisfies l clauses, then there exists a $(2, m+n)$-merging scheme for $\langle G, \mathcal{T}\rangle$ such that $29m+24n+l$ vertices are reachable from S.*

Lemma 6 *If there exists a reachability maximizing $(2, m+n)$-merging scheme such that $29m+24n+l$ vertices are reachable from S in $\langle G, \mathcal{T}\rangle$, then there exists a truth assignment that satisfies l clauses of ϕ.*

The combination of the two lemmas above already yield NP-hardness for MAXREACH on forests. However, we can use the gadget from Figure 5to get a tree of maximum degree 3, with only one source. Hence, we immediately get NP-hardness for MAXMINREACH and MAXAVGREACH on trees with a single source.

Theorem 5 *Problems 4 - 6 are NP-hard even when there exists only one source, G is a tree, has maximum degree three, and every edge has only one label.*

Delaying

In this section we study *edge-independent* delays. Firstly, we show that when the number of allowed delays is bounded, then the minimization problems are NP-hard. Then, we study the case where the number of δ-delaying operations is unbounded. In contrary to unbounded edge deletions where the solution to the problems become trivial by essentially isolating every source, an unbounded number of δ-delays does not trivialize the problems and most importantly does not destroy the underlying network.

Theorem 6 MINREACH, MINMAXREACH, *and* MINAVGREACH *are NP-hard under $\delta-$* DELAYING, *for any $\delta \geq 1$, when the number of operations is bounded by κ. In addition, they are $W[1]$-hard, when parameterized by κ.*

Next, we provide a polynomial-time algorithm for the minimization problems under $\delta-$DELAYING, for any $\delta \geq 1$. Let us define the *reachability network* which will be used by our algorithm. Given a temporal graph $\langle G, \mathcal{T}\rangle$ with lifetime $t_{\max}$ and a set of sources S, for every $t \leq t_{\max}$, we define $RV_t(\langle G, \mathcal{T}\rangle, S)$ to be the set of vertices that are reached at time t *for first time* from a vertex of S. Put formally, $v \in RV_t(\langle G, \mathcal{T}\rangle, S)$, if there exists $s \in S$ such that the *earliest arrival* path from s to v arrives at time t and moreover for every $t' < t$ there is no path from any $s \in S$ to v that arrives at time t'. Additionally, we define $RE_t(\langle G, \mathcal{T}\rangle, S)$ to be the set of temporal edges with label t that are adjacent to vertices in $RV_t(\langle G, \mathcal{T}\rangle, S)$. Observe that we can decide if $v \in RV_t(\langle G, \mathcal{T}\rangle, S)$ can be computed via computing the earliest arrival paths between every $s \in S$ and v, which can be done in polynomial time with respect to the size of $\langle G, \mathcal{T}\rangle$ (Wu et al. 2014). Similarly, we can efficiently compute $RE_t(\langle G, \mathcal{T}\rangle, S)$. Finally, we say that it is δ-possible to change a label of an edge from t to $t+1$ if the difference between $t+1$ and the original label of the edge is at most δ.

$\mathcal{T}_\delta^D \leftarrow \mathcal{T}$;
for $1 \leq t < t_{\max}$ **do**
  Compute $RE_{t+1}(\langle G, \mathcal{T}_\delta^D\rangle, S)$
  foreach *edge* $e \in RE_{t+1}(\langle G, \mathcal{T}_\delta^D\rangle, S)$ **do**
    Change the label i of e to $t+1$, if this is δ-possible;
    Update $\mathcal{T}_\delta^D$;
  end
end

Algorithm 1: Algorithm for $\delta-$ DELAYING.

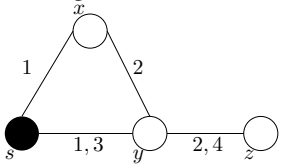

Figure 7: In the temporal graph $\langle G, \mathcal{T}\rangle$ above with $S = s$, we have $RV_1(\langle G, \mathcal{T}\rangle, s) = \{x, y\}$, $RV_2(\langle G, \mathcal{T}\rangle, s) = \{z\}$, $RE_1(\langle G, \mathcal{T}\rangle, s) = \{sx, sy\}$, and $RE_2(\langle G, \mathcal{T}\rangle, s) = \{yz\}$. Any other set is empty.

The next observation follows from the fact that a delay on any edge in $RE_t(\langle G, \mathcal{T}_\delta^D\rangle, S)$ weakly decreases $RV_{t+1}(\langle G, \mathcal{T}_\delta^D\rangle, S)$. Hence, delaying as many as possible from these edges minimizes the number of vertices that are reached by S at time step $t+1$.

Observation 1 *For any $1 \leq t < t_{\max}$, if we can delay only temporal edges with label t, Algorithm 1 minimizes $RV_{t+1}(\langle G, \mathcal{T}_\delta^D\rangle, S)$.*

Observation 1 can be used as an intermediate step in an induction argument to prove optimality of Algorithm 1.

Theorem 7 *Algorithm 1 is optimal for* MINREACH, MINMAXREACH, MINAVGREACH *under $\delta-$* DELAYING.

Discussion

Our hardness results immediately imply, or can be easily extended to prove, several other interesting results. Firstly, we observe that all of our reductions are approximation preserving, thus since we use MAX2SAT(3), we get that there are no approximation schemes for these problems, unless P = NP. In addition, we can add directions to the edges of the gadgets without breaking the correctness of our proofs. Another, less trivial, observation is that in our reductions we create unit disk graphs. This is important since this type of graphs are prominently used in epidemics.

Our work creates many interesting directions for future research. The most obvious ones are to get approximation algorithms for the problems we study. In another, broader front, our paper introduces a new conceptual direction in temporal graphs. Given a temporal network with an existing solution for a problem, can we utilize the current infrastructure in a better way and improve the solution without significantly changing the network?

References

Akrida, E. C.; Czyzowicz, J.; Gasieniec, L.; Kuszner, L.; and Spirakis, P. G. 2017. Temporal flows in temporal networks. In Fotakis, D.; Pagourtzis, A.; and Paschos, V. T., eds., *Algorithms and Complexity*, 43–54. Cham: Springer International Publishing.

Akrida, E. C.; Czyzowicz, J.; Gasieniec, L.; Kuszner, L.; and Spirakis, P. G. 2019. Temporal flows in temporal networks. *Journal of Computer and System Sciences* 103:46 – 60.

Berman, P., and Karpinski, M. 1999. On some tighter inapproximability results. In *International Colloquium on Automata, Languages, and Programming*, 200–209. Springer.

Braunstein, A., and Ingrosso, A. 2016. Inference of causality in epidemics on temporal contact networks. *Scientific reports* 6:27538.

Brockmann, D., and Helbing, D. 2013. The hidden geometry of complex, network-driven contagion phenomena. *science* 342(6164):1337–1342.

Casteigts, A.; Klasing, R.; Neggaz, Y. M.; and Peters, J. G. 2019. Computing parameters of sequence-based dynamic graphs. *Theory of Computing Systems* 63(3):394–417.

Chen, J.; Molter, H.; Sorge, M.; and Suchý, O. 2018. Cluster Editing in Multi-Layer and Temporal Graphs. In Hsu, W.-L.; Lee, D.-T.; and Liao, C.-S., eds., *29th International Symposium on Algorithms and Computation (ISAAC 2018)*, volume 123 of *Leibniz International Proceedings in Informatics (LIPIcs)*, 24:1–24:13. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

Colizza, V.; Barrat, A.; Barthélemy, M.; and Vespignani, A. 2006. The role of the airline transportation network in the prediction and predictability of global epidemics. *Proceedings of the National Academy of Sciences* 103(7):2015–2020.

Enright, J., and Kao, R. R. 2018. Epidemics on dynamic networks. *Epidemics* 24:88 – 97.

Enright, J., and Meeks, K. 2015. Deleting edges to restrict the size of an epidemic: A new application for treewidth. In Lu, Z.; Kim, D.; Wu, W.; Li, W.; and Du, D.-Z., eds., *Combinatorial Optimization and Applications*, 574–585. Cham: Springer International Publishing.

Enright, J., and Meeks, K. 2018. Changing times to optimise reachability in temporal graphs. *CoRR* abs/1802.05905.

Enright, J.; Meeks, K.; Mertzios, G. B.; and Zamaraev, V. 2018. Deleting edges to restrict the size of an epidemic in temporal networks. *CoRR* abs/1805.06836.

Erlebach, T., and Spooner, J. T. 2018. Faster Exploration of Degree-Bounded Temporal Graphs. In Potapov, I.; Spirakis, P.; and Worrell, J., eds., *43rd International Symposium on Mathematical Foundations of Computer Science (MFCS 2018)*, volume 117 of *Leibniz International Proceedings in Informatics (LIPIcs)*, 36:1–36:13. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

Erlebach, T.; Hoffmann, M.; and Kammer, F. 2015. On temporal graph exploration. In Halldórsson, M. M.; Iwama, K.; Kobayashi, N.; and Speckmann, B., eds., *Automata, Languages, and Programming*, 444–455. Berlin, Heidelberg: Springer Berlin Heidelberg.

Kempe, D.; Kleinberg, J.; and Kumar, A. 2002. Connectivity and inference problems for temporal networks. *Journal of Computer and System Sciences* 64(4):820–842.

Mertzios, G. B.; Michail, O.; Chatzigiannakis, I.; and Spirakis, P. G. 2013. Temporal network optimization subject to connectivity constraints. In Fomin, F. V.; Freivalds, R.; Kwiatkowska, M.; and Peleg, D., eds., *Automata, Languages, and Programming*, 657–668. Berlin, Heidelberg: Springer Berlin Heidelberg.

Michail, O., and Spirakis, P. G. 2016. Traveling salesman problems in temporal graphs. *Theoretical Computer Science* 634:1 – 23.

Michail, O. 2016. An introduction to temporal graphs: An algorithmic perspective. *Internet Mathematics* 12(4):239–280.

Mitchell, A.; Bourn, D.; Mawdsley, J.; Wint, W.; Clifton-Hadley, R.; and Gilbert, M. 2005. Characteristics of cattle movements in britain–an analysis of records from the cattle tracing system. *Animal Science* 80(3):265–273.

Nöremark, M., and Widgren, S. 2014. Epicontacttrace: an r-package for contact tracing during livestock disease outbreaks and for risk-based surveillance. *BMC veterinary research* 10(1):71.

Valdano, E.; Ferreri, L.; Poletto, C.; and Colizza, V. 2015a. Analytical computation of the epidemic threshold on temporal networks. *Physical Review X* 5(2):021005.

Valdano, E.; Poletto, C.; Giovannini, A.; Palma, D.; Savini, L.; and Colizza, V. 2015b. Predicting epidemic risk from past temporal contact data. *PLoS computational biology* 11(3):e1004152.

Whitbeck, J.; Dias de Amorim, M.; Conan, V.; and Guillaume, J.-L. 2012. Temporal reachability graphs. In *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking*, Mobicom '12, 377–388. New York, NY, USA: ACM.

Wu, H.; Cheng, J.; Huang, S.; Ke, Y.; Lu, Y.; and Xu, Y. 2014. Path problems in temporal graphs. *Proceedings of the VLDB Endowment* 7(9):721–732.

Zschoche, P.; Fluschnik, T.; Molter, H.; and Niedermeier, R. 2018. The Complexity of Finding Small Separators in Temporal Graphs. In Potapov, I.; Spirakis, P.; and Worrell, J., eds., *43rd International Symposium on Mathematical Foundations of Computer Science (MFCS 2018)*, volume 117 of *Leibniz International Proceedings in Informatics (LIPIcs)*, 45:1–45:17. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.