Approximating Optimal Labelings for Temporal Connectivity

Daniele Carnevale[1], **Gianlorenzo D'Angelo**[1], **Martin Olsen**[2]

[1]Gran Sasso Science Institute, L'Aquila, Italy
[2]Aarhus University, Aarhus, Denmark
daniele.carnevale@gssi.it, gianlorenzo.dangelo@gssi.it, martino@btech.au.dk

Abstract

In a temporal graph the edge set dynamically changes over time according to a set of time-labels associated with each edge that indicates at which time-step the edge is available. Two vertices are connected if there is a path connecting them in which the edges are traversed in increasing order of their labels. We study the problem of scheduling the availability time of the edges of a temporal graph in such a way that all pairs of vertices are connected within a given maximum allowed time a and the overall number of labels is minimum.

The problem, called *Minimum Aged Labeling* (MAL), has several applications in logistics, distribution scheduling, and information spreading in social networks, where carefully choosing the time-labels can significantly reduce infrastructure costs, fuel consumption, or greenhouse gases.

Problem MAL has previously been proved to be NP-complete on undirected graphs and APX-hard on directed graphs. In this paper, we extend our knowledge on the complexity and approximability of MAL in several directions. We first show that the problem cannot be approximated within a factor better than $O(\log n)$ when $a \geq 2$, unless P $=$ NP, and a factor better than $2^{\log^{1-\epsilon} n}$ when $a \geq 3$, unless NP $\subseteq$ DTIME$(2^{\text{polylog}(n)})$, where n is the number of vertices in the graph. Then we give a set of approximation algorithms that, under some conditions, almost match these lower-bounds. In particular, we show that the approximation depends on a relation between a and the diameter of the input graph.

We further establish a connection with a foundational optimization problem on static graphs called *Diameter Constrained Spanning Subgraph* (DCSS) and show that our hardness results also apply to DCSS.

Introduction

We consider a scheduling problem on dynamic networks that is motivated by several applications in logistics, distribution scheduling, and information diffusion in social networks.

As a real-world example, consider a parcel-delivery scenario where there is a warehouse W serving three cities as the center of a star topology, see Figure 1 for an illustration. Each city has a bunch of parcels to be delivered to the other cities and for each pair of cities (A, B) there is at least one parcel that must be delivered from A to B. In each city A there is a vehicle that is used to deliver the parcels from its *home city* A to the warehouse and vice-versa. Each vehicle can possibly depart from its home city at every hour and, once the vehicle reaches W, it deposit the parcels and then comes back to its home city. A travel of a vehicle from its own town to W and back is called a *trip*. For simplicity, we assume that the travel time is negligible. When a vehicle V_1 moves from city A to the the warehouse W, it deposits in W all the parcels that departs from A and whose final destination is any of the other cities and then comes back to its home city A. When a vehicle V_2 returns to its home city B it brings to B all the parcels that have been deposited in W so far and whose final destination is B. If the trip of V_1 is scheduled earlier than that of V_2, the parcels directed from A to B are delivered. Otherwise, they must wait for the next trip. Carefully scheduling the trips of all vehicles might reduce at the same time the delivery time and the costs in terms of number of required trips, fuel consumption, pollutants, and emission of greenhouse gasses. For example, the 1st schedule in Figure 1, requires 6 trips to deliver all parcels and the last parcels are delivered at 9am. The 2nd schedule optimizes the latest arrival time and ensures that the last parcels are delivered two hours earlier than the previous solution, at 7am, but still requires 6 trips to deliver all parcels. The 3rd schedule, instead optimizes the required number of trips, reducing

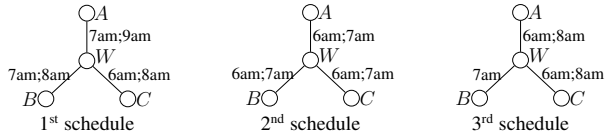

Figure 1: The scheduled time of trips are reported close to the edges. **1st schedule**: All the 6 scheduled trips are needed to deliver all parcels and the last parcel is delivered at 9am. **2nd schedule**: All the parcels are deposited in W with the 3 trips at 6am and then are delivered from W to the right cities at 7am; still 6 trips are needed but the last parcel is delivered at 7am. **3rd schedule**: The parcels leaving cities A and C are deposited in W with the trips at 6am; the single trip from B at 7am deposits the parcels from B in W and brings the parcels directed to B previously deposited at 6am; finally, the trips at 8am deliver the parcels directed to A and C. Only 5 trips are needed, but the last parcel is delivered at 8am.

them to 5, but the last parcels are delivered at 8am, one hour later than the previous case.

The problem of scheduling trips becomes much more complex if we consider a general network in which each vertex might serve both as a warehouse and as a city, and connections among vertices are given by an arbitrary underlying graph. Moreover, similar problems arise in other contexts such as distribution scheduling (Deligkas and Potapov 2020) and information spreading, where the aim is to schedule a small number of meetings among employees of a company in such a way that each employee can share its own information with any other by a given time, see (Deligkas, Eiben, and Skretas 2023).

Motivated by these applications, we consider the following question: *What is the minimum number of trips needed to deliver all parcels within a given time?*

We model a schedule of trips along edges of a network with a *temporal graph* (a.k.a. dynamic graph) in which the scheduling time of a vehicle is represented as an edge-label and a path in a graph is valid (or *temporal*) only if the edges are traversed in increasing order of their labels. We then consider the optimization problem of assigning the minimum number of labels to the edges of a graph in such a way that each pair of vertices is connected via a temporal path and the largest label is not greater than a given integer a, called the *maximum allowed age*. This problem, called *Minimum Aged Labeling* (MAL), has been introduced by Mertzios, Michail, and Spirakis (2019), who proved that it is APX-hard in *directed graphs*. Later, Klobas et al. (2024) showed that MAL is NP-complete also for *undirected graphs*. To the best of our knowledge, there are no hardness or algorithmic results on the approximation of MAL in undirected graphs. Moreover, the reduction used to prove the APX-hardness in directed graphs (Mertzios, Michail, and Spirakis 2019) cannot be easily adapted to undirected graphs as in the constructed graph the direction of edges is used to bound the reachability of vertices. In this paper, we study the complexity of approximating MAL in undirected graphs showing when, depending on a relation between a and the diameter D_G of the input graph, it is hard to approximate, or it can be approximated in polynomial time.

MAL also has a theoretical motivation as it can be interpreted as a *dynamic version* of a foundational graph theoretical problem called the *Diameter Constrained Spanning Subgraph* (DCSS) problem, which asks to find a spanning subgraph H of a graph G such that the diameter of H is at most a given integer and its number of edges is minimum.

Our results. We provide both hardness of approximation lower-bounds and approximation algorithms for MAL.

Hardness of approximation. We first prove that, even when the maximum allowed age a of a labeling is a fixed value greater or equal to 2, MAL cannot be approximated within a factor better than $O(\log n)$, unless P = NP. Then, we show that, unless NP $\subseteq$ DTIME($2^{\text{polylog}(n)}$), we cannot find any $2^{\log^{1-\epsilon} n}$-approximation algorithm for MAL, even when a is a fixed value greater or equal to 3 and $\epsilon \in (0, 1)$.

These results advance our knowledge on the computational complexity of MAL in two directions. (1) From an exact computation point of view, the NP-hardness given in (Klobas et al. 2024) only holds for $a = D_G = 10$, while we show that MAL is NP-hard for any fixed $a \geq 2$ (still, we require $a = D_G$). This closes the characterization of the computational complexity of MAL with respect to the parameter a, as the case $a = 1$ is trivial. Moreover, we provide a considerably simpler reduction than the one in (Klobas et al. 2024). (2) From an approximation point of view, the reduction in (Mertzios, Michail, and Spirakis 2019) shows that MAL is APX-hard for $a = D_G = 9$ in directed graphs. We show two stronger lower-bounds on the approximation, namely that MAL is hard to approximate better than a logarithmic factor, under P $\neq$ NP, and a factor $2^{\log^{1-\epsilon} n}$, under a stronger complexity condition. The second lower-bound suggests that it is unlikely to approximate MAL to a factor better than a polynomial. Moreover, our lower-bounds hold even for any fixed $a \geq 2$ and for $a \geq 3$, resp., and for undirected graphs (again, we require $a = D_G$). Finally, we remark that our lower-bounds also apply to DCSS.

Approximation algorithms. Like in (Klobas et al. 2024) and (Mertzios, Michail, and Spirakis 2019), all our reductions require that $a = D_G$. Hence, we investigate the approximability of MAL when $a \geq D_G$, addressing an open question posed in (Klobas et al. 2024). We give three sets of results, which interestingly show how the approximation of MAL depends on a relation between a and D_G.

(1) We first consider the case in which a is sufficiently larger than R_G, the radius of G. If $a \geq 2R_G$ ($a \geq 2R_G + 1$, resp.), we can compute in polynomial time a solution that requires at most only 2 labels (1 label, resp.) more than the optimum. Observe that these additive approximation bounds correspond to asymptotic multiplicative bounds with approximation factors that arbitrarily approach 1 as the input size grows. Moreover, if $a \geq 2D_G + 2$, we can compute an optimal solution in polynomial time. As MAL does not admit any feasible solution when $a < D_G$, this first set of results leaves open the cases when $D_G \leq a < 2R_G$.

(2) We then consider the case in which a is slightly larger than D_G by exploiting a relation between MAL and DCSS. Specifically, we show that there is a gap of a factor a between the approximation of MAL and that of DCSS. We first prove that when $a = D_G = 2$, MAL can be approximated with a logarithmic factor, which asymptotically matches our first hardness lower-bound.[1] When $a \geq D_G + 2$, we achieve an approximation factor of $O(D_G \cdot n^{1/2})$, which is sublinear when D_G is sufficiently small. Moreover, if $a \geq D_G + 4$ or $a \geq D_G + 6$, we reduce the approximation factor to $O(D_G \cdot n^{2/5})$ and $O(D_G \cdot n^{1/3})$, respectively. Finally, for any value of $a \geq D_G$ we achieve an approximation factor of $O(D_G \cdot n^{3/5+\epsilon})$, for any $\epsilon > 0$. All these approximation factors linearly depend on D_G, due to the gap between the approximation of MAL and that of DCSS.

(3) Our main algorithmic contribution consists in approximating MAL when $D_G \leq a < 2R_G$ without passing through DCSS, thus avoiding a linear dependency on D_G in the approximation ratio. We show that when $a \geq \lceil 3/2 \cdot D_G \rceil$

[1] When $a = 2$ and $D_G = 1$ the graph is a clique. Hence $R_G = 1$ and we can solve MAL with 2 labels more than the optimum.

($a \geq \lceil 5/3 \cdot D_G \rceil$, resp.), then we can approximate MAL within a factor of $O(\sqrt{n \log n})$ ($O(\sqrt[3]{D_G n \log^2 n})$, resp.). Both bounds are sublinear, and the second algorithm outperforms the first one when $D_G = o(\sqrt{n/\log n})$ but requires greater values of a.

Due to space constraints, some proofs are deferred to the full version of the paper.

Related Work. Due to their versatility, temporal graphs have been considered from several perspectives and using different terminology, e.g. *dynamic*, *evolving*, and *time-varying* graphs or networks, see (Michail 2016). Mainly motivated by virus-spread minimization (see e.g. (Braunstein and Ingrosso 2016; Enright and Kao 2018)), an area that received considerable interest is the one related to the modification of a temporal network in such a way that some objective is optimized (see a recent survey (Meeks 2022)). Several operations were considered e.g. delaying labels and merging consecutive times (Deligkas and Potapov 2020), edge-deletion and label-deletion (Enright et al. 2021), and changing the relative order of times (Enright, Meeks, and Skerman 2021). Moreover, (Molter, Renken, and Zschoche 2024) studied how the choice between edge-deletion and delaying influences the parameterized complexity of the reachability-minimization objective. In (Deligkas, Eiben, and Skretas 2023), the authors studied a problem similar but orthogonal to MAL in which one wants to minimize the maximum time a subset of vertices requires to reach every vertex, by shifting labels. Klobas et al. (2024), also considered a generalization of MAL where only a subset of *terminal* vertices must be connected, subject to a constraint on the maximum allowed age, and show that the problem is $W[1]$-hard when parameterized by the number of labels.

Preliminaries

For an integer $k \in \mathbb{N}$, let $[k] := \{1, 2, \ldots, k\}$ and $[k]_0 := \{0\} \cup [k]$. We consider simple undirected graphs $G = (V, E)$ with $n = |V|$ vertices and $m = |E|$ edges. We also use $V(G)$ and $E(G)$ to refer to the vertex set and the edge set of the graph G, respectively. We denote an edge $e \in E$ between two vertices v and w with $e = \{v, w\}$, and say that v and w are the *endpoints* of e. A *subgraph* $H = (V', E')$ of a graph G, written as $H \subseteq G$, is a graph whose vertex set V' and edge set E' are subsets of $V(G)$ and $E(G)$ respectively, with the additional constraint that if $\{v, w\} \in E'$ then $v, w \in V'$. If $V' = V$ and it is connected, then the subgraph H is said to be a *spanning subgraph*.

A *path* P between vertices v and w in graph G is a sequence of distinct vertices $v_0, v_1, \ldots, v_k \in V$ and sequence of distinct edges $\{v_i, v_{i+1}\} \in E$, for each $i \in [k-1]_0$, where $v_0 = v$ and $v_k = w$. The *length* of P is equal to the number of its edges, i.e. k. A *shortest path* is a path with minimum length. The *distance* between two vertices v and w in the graph G is denoted $d_G(v, w)$ and is equal to the length of a shortest path between v and w. Let $S \subseteq V$, we denote by $d_G(v, S)$ the distance between v to its closest vertex in S, i.e. $d_G(v, S) = \min_{s \in S} d_G(v, s)$. A graph G is *connected* if there exists a path between any pair of vertices. A *Shortest Path Forest* (SPF) F rooted at some set of vertices $S \subseteq V$ is a forest spanning all vertices V such that $d_F(w, S) = d_G(w, S)$, for each $w \in V(G)$. If S is a singleton, $S = \{v\}$, a SPF rooted at S is called a *Shortest Path Tree* (SPT) rooted at v.

Given a vertex $v \in V$, the *eccentricity* $\text{ecc}_G(v)$ of v in G is the maximum distance between v and any other vertex, i.e. $\text{ecc}_G(v) = \max_{w \in V} d_G(v, w)$. The *diameter* and *radius* of G are defined as $D_G = \max_{v \in V} \text{ecc}_G(v)$ and $R_G = \min_{v \in V} \text{ecc}_G(v)$, respectively. We say that a vertex v is a *center* of a graph G if $\text{ecc}_G(v) = R_G$.

We will drop the subscript G in the notation when the graph in question is clear from the context.

A *temporal graph* is a pair (G, λ) where G is a graph and $\lambda : E \longrightarrow 2^{\mathbb{N} \setminus \{0\}}$ is a function assigning to every edge of G a set of discrete *labels* that we interpret as presence in time. The *lifetime* (a.k.a. *age*) L_λ of a temporal graph (G, λ) is the largest label in λ i.e. $L_\lambda = \max\{t : t \in \lambda(e), e \in E\}$. The total number of labels in a temporal graph is $|\lambda| = \sum_{e \in E} |\lambda(e)|$.

One central notion in temporal graphs is that of a *temporal path*, in this paper we will use a definition often called a *strict* temporal path. A *temporal path* is a sequence $(e_1, t_1), (e_2, t_2), \ldots, (e_k, t_k)$ where $e_1, e_2, \ldots, e_k$ is a path in G, $t_i \in \lambda(e_i)$ for every $i \in [k]$, and $t_1 < t_2 < \cdots < t_k$. We say that a vertex v is *temporally reachable* from a vertex w if there exists a temporal path from w to v. A temporal graph is called *temporally connected* if every vertex is temporally reachable from every other vertex.

We are ready to introduce the *Minimum Aged Labelling* (MAL) problem, which is the subject of this paper.

Definition 1. *Given a graph $G = (V, E)$ and an integer a, the Minimum Aged Labelling (MAL) problem asks to find a function $\lambda : E \to 2^{\mathbb{N}}$ such that (G, λ) is temporally connected, $L_\lambda \leq a$, and $|\lambda|$ is minimum.*

We assume that the graph G is connected as otherwise MAL does not admit any feasible solution. The same holds if $a < D_G$, as already observed in (Klobas et al. 2024). On the other hand, if $a \geq 2D_G + 2$, then MAL is solvable in polynomial time as we will prove in Theorem 7-(iii) by using a result from (Klobas et al. 2024). Hence, we will consider in the rest of the paper the case $D_G \leq a \leq 2D_G + 2$.

A problem that is strictly related to MAL is the *Diameter Constrained Spanning Subgraph* (DCSS) problem.

Definition 2. *Given a graph G and an integer $d \geq D_G$, the DCSS problem asks to find a spanning subgraph H such that $D_H \leq d$ and the number of edges of H is minimum.*

There is a close connection between problems DCSS and MAL. Roughly, MAL can be seen as a temporal version of the DCSS problem. The next theorem relates the approximations of MAL and DCSS.

Theorem 1. *Let G be a graph and b be an integer such that $b = O(|V(G)|)$.*

i) If there exists an α-approximation algorithm for MAL, then there exists an (αb)-approximation algorithm for DCSS, where (G, b) is the input to DCSS.

ii) If there exists an α-approximation algorithm for DCSS, then there exists an (αb)-approximation algorithm for MAL, where (G, b) is the input to MAL.

To the best of our knowledge the DCSS problem appeared, in a slightly different form, only in (Elkin and Peleg 2001). However, few other known optimization problems generalize DCSS and hence we can exploit the approximation algorithms known for these problems also for DCSS. In the following we summarize the approximation bounds that we can obtain by reducing DCSS to other problems; we defer to the full version for a detailed discussion of the reductions and how to obtain such bounds.

Corollary 1. *If $d = 2$, there exists a $O(\log n)$-approximation algorithm for DCSS.*

Corollary 2. *For any constant $\epsilon > 0$, there exists a $O(n^{3/5+\epsilon})$-approximation algorithm for the DCSS problem.*

Corollary 3. *Let (G, d) be an instance of DCSS.*

i) If $d \geq D_G + 2$ then there exists an $O(n^{1/2})$-approximation algorithm for DCSS.
ii) If $d \geq D_G + 4$ then there exists an $O(n^{2/5})$-approximation algorithm for DCSS.
iii) If $d \geq D_G + 6$ then there exists an $O(n^{1/3})$-approximation algorithm for DCSS.

Hardness of Approximation

In this section we show that MAL is hard to be approximated, even under some restrictions on the input instance.

We start by showing our logarithmic lower-bound for $a = 2$ in the next theorem.

Theorem 2. *For every $\epsilon \in (0, 1/4)$, there is no polynomial time $(\epsilon \log |V|)$-approximation algorithm for MAL, unless P = NP. The hardness holds even when the maximum allowed age a is equal to 2.*

Observe that Theorem 1 together with Theorem 2 imply that a logarithmic lower-bound also holds for DCSS when $d = 2$.

To prove Theorem 2, we introduce the *Set Cover* (SC) problem. In SC we are given a universe $U = \{u_1, \ldots, u_\eta\}$, with $|U| = \eta$, and a collection of μ subsets $\mathcal{C} = \{s_1, \ldots, s_\mu\} \subseteq 2^U$. The goal is to find a minimum size subcollection $\mathcal{C}^* \subseteq \mathcal{C}$ such that $\bigcup_{s_i \in \mathcal{C}^*} s_i = U$.

The next theorem proves the intractability of approximating SC to a factor smaller than $\log N$, where N is the size of an instance.

Theorem 3 ((Dinur and Steurer 2014))**.** *For every $\alpha \in (0, 1)$, it is* NP*-hard to approximate* SC *to within $(1 - \alpha) \log N$, where N is the size of the instance. The reduction runs in time $N^{O(1/\alpha)}$.*

Let $(U, \mathcal{C})$ be an instance of SC. We construct a graph $G = (V, E)$ which we will use to prove Theorem 2. See Figure 2 for a reference of the construction. Graph G contains a vertex s_j for each set of $\mathcal{C}$, a vertex u_i for each element of the universe U, and two sets of new vertices $T = \{t_1, t_2, t_3\}$ and $W = \{w_1, w_2, \ldots, w_x\}$ where x is an integer to be defined later. Formally, $V(G) = \mathcal{C} \cup U \cup T \cup W$. The set of

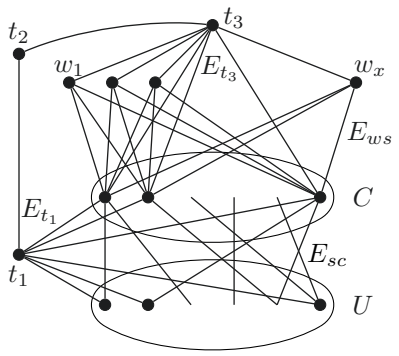

Figure 2: The graph G constructed starting from a SC instance $(U, \mathcal{C})$ in the proof of Theorem 2.

edges of G contains the edges induced by the instance of SC, that is $E_{\text{sc}} = \{\{u_i, s_j\} : u_i \in U \wedge s_j \in \mathcal{C} \wedge u_i \in s_j\}$. It also contains edges $E_{t_1} = \{\{t_1, z\} : z \in U \cup \mathcal{C} \cup \{t_2\}\}$ and edges $E_{t_3} = \{\{t_3, z\} : z \in W \cup \mathcal{C} \cup \{t_2\}\}$. Moreover, we add to G the edges between each $w_l \in W$ to all vertices in $\mathcal{C}$, more formally $E_{\text{ws}} = \{\{w_l, s_j\} : w_l \in W \wedge s_j \in \mathcal{C}\}$. Finally, the edge set of G is equal to $E(G) = E_{\text{sc}} \cup E_{t_1} \cup E_{t_3} \cup E_{\text{ws}}$. The reader can check that the diameter of G is equal to 2.

The idea of the reduction is that since the only paths of length 2 between any $w \in W$ and any $u \in U$ must cross C, therefore for each $w \in W$ we must select two, possibly identical, set covers – one to allow w to reach all vertices in U and the other one to allow all vertices in U to reach w. The vertices in T help us to connect the other pairs of vertices. However, the labels in the edges incident to the nodes in T might constitute a dominant term in the solution size. We solve this by inserting enough vertices in W.

Before proving Theorem 2, we need the following lemma.

Lemma 1. *Let $(U, \mathcal{C})$ be an instance of SC and let G be the graph constructed as above. Given an optimal set cover of size k^* for $(U, \mathcal{C})$, we can construct a feasible solution λ to the instance $(G, 2)$ of MAL with $|\lambda| \leq 6x + 2xk^*$.*

Now, we are ready to prove Theorem 2.

Proof of Theorem 2. Let $(U, \mathcal{C})$ be an instance of SC with $\eta = |U|$ and $\mu = |C|$, let $\mathcal{C}^*$ be an optimal set cover, and let $k^* = |\mathcal{C}^*|$. Let G be the graph constructed as previously described with $x = \eta + \mu + 1$. Let λ^* denote an optimal solution to MAL on instance $(G, 2)$. Assume there is an α-approximation algorithm for MAL and denote the solution given by such an algorithm with λ_{APX}. Denote with G_p where $p \in [2]$ the subgraph of G having only the edges that in λ_{APX} contain label p, that is $E(G_p) = \{e \in E(G) : p \in \lambda_{\text{APX}}(e)\}$. Observe that in G_p the existing edges going from w_l, $l \in [x]$, to the vertices in $\mathcal{C}$ must induce a set cover as the only paths of length 2 between w_l and any vertex in U must go through a vertex in $\mathcal{C}$. More formally, the set of vertices $C_{p,l} = \{s_j \in \mathcal{C} : p \in \lambda_{\text{APX}}(\{w_l, s_j\})\}$ is a set cover for every $p \in [2], l \in [x]$. Denote with k the minimum among the size of sets $C_{p,l}$, that is $k = \min_{p \in [2], l \in [x]} |C_{p,l}|$. Observe that for each $p \in [2]$ and $l \in [x]$, λ contains $|C_{p,l}|$ distinct

labels to connect vertex w_l to all nodes in $C_{p,l}$ with label p. Hence, $|\lambda_{\text{APX}}| \geq \sum_{p\in[2]} \sum_{l\in[x]} |C_{p,l}| \geq 2xk$.

By Lemma 1, we can compute a labeling λ such that $|\lambda| \leq 6x + 2xk^*$ and therefore, $|\lambda^*| \leq 6x + 2xk^*$.

As λ_{APX} is an α-approximation algorithm for $(G, 2)$, then $|\lambda_{\text{APX}}| \leq \alpha|\lambda^*|$, and therefore $2xk \leq |\lambda_{\text{APX}}| \leq \alpha|\lambda^*| \leq \alpha(6x + 2xk^*)$. Dividing by $2x$, we get that $k \leq \alpha(k^* + 3)$.

Let $\alpha \leq \epsilon \log(|G|)$, where $|G| = O(N^2)$. We have $k \leq \epsilon' \log(N^2)k^* = 2\epsilon' \log(N)k^*$, for any $\epsilon' = \epsilon + o(1)$, which contradicts Theorem 3 for any $\epsilon' \in (0, 1/2)$. Hence, we have a contradiction for any $\alpha \leq \epsilon \log(|G|)$ and $\epsilon \in (0, 1/2)$. The theorem follows by observing that $|G| = O(|V(G)|^2)$. □

In the next theorem we extend our logarithmic lower-bound to the case in which a is any fixed value greater or equal to 3. Indeed, we prove this result for DCSS and then use Theorem 1 to show the same result for MAL.

Theorem 4. *For every $\epsilon \in (0, 1/6)$, there is no polynomial time $(\epsilon \log |V|)$-approximation algorithm for DCSS, unless P = NP. The hardness holds even when the required diameter d is a fixed parameter greater or equal to 3.*

Corollary 4. *For every $\epsilon \in (0, 1/(6a))$, there is no polynomial time $(\epsilon \log |V|)$-approximation algorithm for MAL, unless P=NP. The hardness holds even when the maximum allowed age a is a fixed parameter greater or equal to 3.*

The next theorem shows our second lower-bound, based on a stronger complexity hypothesis. Again, we prove the hardness for DCSS and then combine it with Theorem 1 to obtain a hardness bound for MAL. We highlight that in this last result, the interval for ϵ does not depend on a as ϵ appears in the exponent.

Theorem 5. *For any constant $\epsilon \in (0, 1)$, there is no polynomial time $2^{\log^{1-\epsilon} n}$-approximation algorithm for DCSS, unless NP $\subseteq$ DTIME$(2^{polylog(n)})$. The hardness holds even when the required diameter d is a fixed parameter greater or equal to 3.*

Corollary 5. *For any constant $\epsilon \in (0, 1)$, there is no polynomial time $2^{\log^{1-\epsilon} n}$-approximation algorithm for MAL, unless NP $\subseteq$ DTIME$(2^{polylog(n)})$. The hardness holds even when the maximum allowed age a is a fixed parameter greater or equal to 3.*

In order to prove Theorem 5, we will use reduction from a problem called MIN-REP, introduced in (Kortsarz 2001). In MIN-REP, we are given a bipartite graph $\tilde{G} = (A, B, \tilde{E})$ where A and B are disjoint set of vertices and $\tilde{E}$ are edges between vertices of A and B. Set A is partitioned into r groups $A_1, A_2, \ldots, A_r$ and set B is partitioned into r groups $B_1, B_2, \ldots, B_r$, with the additional property that every set A_i and every set B_j has the same size denoted with σ. Graph $\tilde{G}$ induces a bipartite *condensed graph* $G' = (U, W, E')$ defined as follows, $U = \{a_i : i \in [r]\}$, $W = \{b_j : j \in [r]\}$, and E' contains an edge between vertices $a_i \in U$ and $b_j \in W$ if and only if there is an edge in $\tilde{E}$ between some vertex in A_i and some vertex in B_j. The vertices in $V(G')$ are called *condensed vertices* and the edges in $E(G')$ are called *condensed edges*.

A *REP-cover* is a set $C \subseteq A \cup B$ of vertices with the property that it "covers" every condensed edge, that is for all condensed edges $\{a_i, b_j\}$ there are vertices $a \in C \cap A_i$ and $b \in C \cap B_j$ such that $\{a, b\} \in \tilde{E}$. The objective of MIN-REP is to construct a REP-cover of minimum size. Let us denote with C^* an optimal solution to MIN-REP.

We will say that an instance of MIN-REP is a YES-instance if $|C^*| = 2r$ (one vertex for each group) and is a NO-instance if $|C^*| \geq 2^{\log^{1-\epsilon'} n} r$. The following theorem is due to Kortsarz (2001).

Theorem 6 ((Kortsarz 2001))**.** *For any constant $\epsilon' \in (0, 1)$, there is no polynomial time algorithm that can distinguish between YES and NO instances of MIN-REP, unless NP $\subseteq$ DTIME $(2^{polylog(n)})$.*

Let $\tilde{G} = (A, B, \tilde{E})$ be an instance of MIN-REP with associated condensed graph $G' = (U, W, E')$. We denote with $\Gamma(w)$ the group in $\tilde{G}$ corresponding to condensed vertex $w \in U \cup W$. Therefore, for $u \in U$ we have that $\Gamma(u) = A_i$ for some $i \in [r]$, and for $w \in W$ we have that $\Gamma(w) = B_j$ for some $j \in [r]$. Let x be a parameter which we will set later.

Before we go into the details of the reduction from MIN-REP to DCSS we give some intuition of the construction. We start with a MIN-REP instance $\tilde{G} = (A, B, \tilde{E})$, we construct an instance $(G, 3)$ of DCSS, where G contains all vertices and edges of $\tilde{G}$ plus a vertex for every group (corresponding to the condensed vertex), which is connected to all vertices in its group. The idea is that in the DCSS instance we want two condensed vertices to be connected through the edges in $\tilde{G}$ if they are connected by a condensed edge in G', and to have some other path in the case they are not connected by a condensed edge. This will ensure that the selected edges will correspond to a REP-cover. For this purpose, we add an edge between two condensed vertices if and only if they are not connected by a condensed edge in G'.

To connect all vertices in G that are not condensed vertices, we add a vertex t that will act as the center of a star to connect all such vertices. This new vertex will be adjacent to all the vertices in A and B, and it will be connected to the condensed vertices through disjoint paths of length two. In this way, the condensed vertices will be the only vertices that are not connected through a path of length at most 3 passing through the star.

The labels of edges incident to t might induce a dominant term in the size of the solution. To fix this, we will replicate the condensed vertices into $x+1$ copies, where x copies have the same role described before, i.e., trying to reach the other condensed vertices through the MIN-REP instance (therefore selecting their own vertices for the REP-cover), and the other one having the role to connect its copies to the other vertices.

We now construct our DCSS instance. See Figure 3 for an illustration. For simplicity, we describe a reduction for the case of diameter 3, and then discuss how to extend it to any fixed $d \geq 3$. We start by defining the vertex set $V = A \cup B \cup V^L \cup V^R \cup S \cup \{t\}$, where $V^L = U \times [x]$, $V^R = W \times [x]$, and $S = \{s_u : u \in U \cup W\}$. In other words, the vertices of V^L

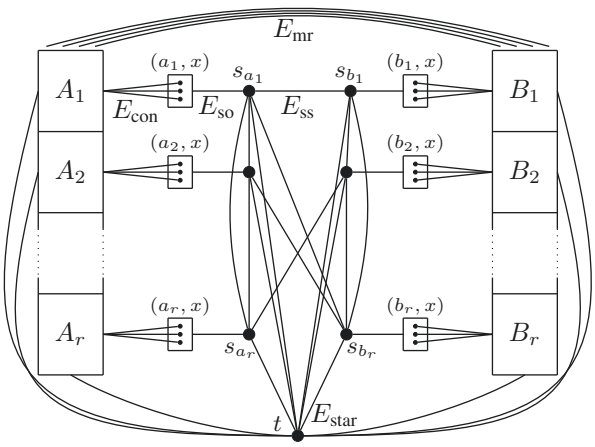

Figure 3: The graph used in the proof of Theorem 5. Edges between a vertex and a set indicate that the vertex is connected to all vertices in the set.

and V^R are x copies of the condensed vertices, S contains one vertex for each condensed vertex, and we add a special vertex t that will help us to easily connect the majority of the vertices in the instance.

Now, we define the edge set of G. We start with the edges of the MIN-REP instance, here denoted with $E_{\text{mr}} = \tilde{E}$. We next add the edges connecting the vertices of V^L and V^R to their corresponding group. More formally, $E_{\text{con}} = \{\{(u,i),a\} : u \in U \cup W \wedge a \in \Gamma(u) \wedge i \in [x]\}$. We next connect vertices in S using two edge sets: one connecting $s_u \in S$ to the vertices $(u,i) \in V^L \cup V^R$, $E_{\text{so}} = \{\{(u,i), s_u\} : u \in U \cup W \wedge i \in [x]\}$; and one connecting the vertices inside S in such a way that there is an edge between two vertices in S if and only if there is no condensed edge connecting the two corresponding condensed vertices in G'. Formally, $E_{\text{ss}} = \{\{s_u, s_{u'}\} : u, u' \in U \cup W \wedge \{u, u'\} \notin E(G')\}$ Observe that if we take two vertices in U the corresponding vertices of S will be connected by an edge, the same hold for the vertices in W. Finally, we add the edges having t as an endpoint, $E_{\text{star}} = \{\{t, y\} : y \in A \cup B \cup S\}$. The final edge set of G will be the union of the previous edge sets: $E(G) = E_{\text{mr}} \cup E_{\text{con}} \cup E_{\text{so}} \cup E_{\text{ss}} \cup E_{\text{star}}$.

The next lemma shows that D_G is equal to 3.

Lemma 2. *The graph G constructed above has diameter 3.*

We will show that if there exists a $2^{\log^{1-\epsilon} n}$-approximation algorithm for an instance $(G, 3)$ of DCSS, then Theorem 6 is contradicted. To this aim, we need two lemmas.

Let H be an arbitrary spanning subgraph of G with $D_H = 3$. The first lemma proves that the size of $E(H)$ is lower bounded by x times the size of an optimal solution to the MIN-REP instance i.e. $|E(H)| > x \cdot |C^*|$, where x is the number of copies in G of each condensed vertex.

Lemma 3. *For any feasible solution H to the instance $(G, 3)$ of DCSS we have $|E(H)| > x \cdot |C^*|$.*

The second lemma shows that there exists a spanning subgraph of diameter 3 which does not cost much more than C^* in a YES-instance of MIN-REP.

Lemma 4. *Assume that $\tilde{G}$ is a YES-instance of MIN-REP, then there exists a feasible solution H to the instance $(G, 3)$ of DCSS such that $|E(H)| \leq 4rx + 5r^2\sigma$.*

We are now ready to prove Theorem 5.

Proof of Theorem 5. Fix $\epsilon \in (0, 1)$. Let $\tilde{G}$ be an instance of MIN-REP of size $\tilde{n}$ such that it holds $\tilde{n} \geq 2^{2^{\frac{2}{\epsilon(1-\epsilon)}}}$. Let us set $x = r\sigma$. Clearly, the graph G created as previously described has size n that is polynomial in the size $\tilde{n} = 2r\sigma$ of $\tilde{G}$. In particular $n \leq (2r\sigma)^2$.

Assume there is a polynomial time $(2^{\log^{1-\epsilon} n})$-approximation algorithm for the DCSS problem, which we apply to the instance $(G, 3)$ previously defined, and let us denote the approximate solution with H_{APX}. Let H^* denote an optimal solution of DCSS on instance $(G, 3)$.

Theorem 6 implies that under the assumption that NP $\not\subseteq$ DTIME $(2^{\text{polylog}(n)})$, there is no polynomial time algorithm that can distinguish between the YES case where $|C^*| = 2r$ and the NO case where $|C^*| \geq r2^{\log^{1-\epsilon'}(2r\sigma)}$. As Theorem 6 holds for any choice of ϵ', we fix $\epsilon' = \epsilon^2$ and we have $|C^*| \geq r2^{\log^{1-\epsilon^2}(2r\sigma)}$ when $\tilde{G}$ is a NO-instance for MIN-REP.

Since $x = r\sigma$, Lemma 4 implies that if $\tilde{G}$ is a YES-instance of MIN-REP, then $|H^*| \leq 9xr$. Therefore, when $\tilde{G}$ is a YES-instance, the solution of our approximation algorithm on G has size $|H_{\text{APX}}| \leq 9xr \cdot 2^{\log^{1-\epsilon} n} \leq 9xr2^{2\log^{1-\epsilon}(2r\sigma)}$. While, when $\tilde{G}$ is a NO-instance of MIN-REP, by Lemma 3 we have that $|H^*| > x \cdot |C^*| \geq xr2^{\log^{1-\epsilon^2}(2r\sigma)} = xr2^{(\log^{1-\epsilon}(2r\sigma))^{1+\epsilon}}$. But now using the facts that $2r\sigma = n' \geq 2^{2^{\frac{2}{\epsilon(1-\epsilon)}}}$ and $0 < \epsilon < 1$ we get $9xr2^{2\log^{1-\epsilon}(n')} < xr2^{(\log^{1-\epsilon}(n'))^{1+\epsilon}}$, which implies that we can use our approximation algorithm to distinguish between YES and NO instances of MIN-REP, contradicting Theorem 6. Hence, for any ϵ, no such polynomial time $(2^{\log^{1-\epsilon} n})$-approximation algorithm can exist and the theorem follows for $d = 3$.

To extend the hardness to the case in which the required diameter d is any fixed value $d \geq 3$, we need to slightly adjust the reduction. For simplicity assume that $d = 2k + 1$ with integer $k \geq 2$, we change the reduction by adding for each vertex $(u,i) \in V^L \cup V^R$ a path on $k-1$ copies of vertex (u,i) i.e. we add vertices $(u,i,1), (u,i,2), \ldots, (u,i,k-1)$ and edges $\{(u,i),(u,i,1)\}, \{(u,i,j),(u,i,j+1)\}$ with $j \in [k-2]$. In this graph, vertices $(u,i,k-1)$ have to select a REP-cover in order to reach every other vertices with a path of length at most d. As before, by choosing ϵ small enough and an instance of MIN-REP of size big enough we can prove that d-DCSS cannot be approximated with a factor better than $2^{\log^{1-\epsilon} n}$ for any $d \geq 3$. A similar construction can be done when d is even, with some more technicalities introduced by the fact that the paths length will be different based on $(u,i) \in V^L$ or $(u,i) \in V^R$. □

Approximation Algorithms

In this section we introduce our approximation algorithms for MAL. We recall that $D_G \leq a \leq 2D_G + 2$. Our first

set of results for the cases in which a is sufficiently large is given in the following theorem.

Theorem 7. *Let (G, a) denote an instance of MAL.*

i) If $a \geq 2R_G$ then we can compute in polynomial time a solution λ for MAL such that $|\lambda| \leq |\lambda^*| + 2$.
ii) If $a \geq 2R_G + 1$ then we can compute in polynomial time a solution λ for MAL such that $|\lambda| \leq |\lambda^*| + 1$.
iii) If $a \geq 2D_G + 2$ then we can compute in polynomial time a solution λ for MAL such that $|\lambda| = |\lambda^*|$.

The next corollary collects the approximation results for the cases in which a is slightly larger than D_G, which are derived from the relation between MAL and DCSS established in Theorem 1.

Corollary 6. *Let (G, a) denote an instance of MAL.*

i) For any $\epsilon > 0$, there exists an $O(D_G \cdot n^{3/5+\epsilon})$-approximation algorithm for MAL.
ii) If $a = 2$, then there exists an $O(\log(n))$-approximation algorithm for MAL.
iii) If $a \geq D_G + 2$, then there exists an $O(D_G \cdot n^{1/2})$-approximation algorithm for MAL.
iv) If $a \geq D_G + 4$, then there exists an $O(D_G \cdot n^{2/5})$-approximation algorithm for MAL.
v) If $a \geq D_G + 6$, then there exists an $O(D_G \cdot n^{1/3})$-approximation algorithm for MAL.

Proof. The statement follows by combining Theorem 1 with the approximations for DCSS given in Corollary 2, Corollary 1, and Corollary 3 and observing that $a = O(D_G)$. □

In the next theorem, we give our main algorithmic results, which consist of approximation algorithms for MAL that do not exploit the relation with DCSS and whose approximation ratios do not depend linearly on D_G.

Theorem 8. *Let (G, a) be an instance of MAL.*

i) If $a \geq \lceil 3/2 \cdot D_G \rceil$, then there exists an $O(\sqrt{n \log n})$-approximation algorithm for MAL.
ii) If $a \geq \lceil 5/3 \cdot D_G \rceil$, then there exists an $O(\sqrt[3]{D_G n \log^2 n})$-approximation algorithm for MAL.

The bounds in Theorem 8 hold when a is sufficiently larger than D_G. Moreover, the approximation bounds of Theorem 8 outperform those of Corollary 6 when D_G is large enough.

In order to prove Theorem 8 we need to introduce some new concepts. The following definition is an adaptation to undirected graphs of the one contained in (Choudhary and Gold 2020).

Definition 3 ((Choudhary and Gold 2020))**.** *For a graph $G = (V, E)$ and a set-pair (S_1, S_2), where $S_1, S_2 \subseteq V$, we say that (S_1, S_2) is a $\langle h_1, h_2 \rangle$-dominating set-pair of size-bound $\langle n_1, n_2 \rangle$ if $|S_1| = O(n_1)$, $|S_2| = O(n_2)$, and one of the following conditions holds.*

1. For each $v \in V(G)$, $d_G(v, S_1) \leq h_1$,
2. For each $v \in V(G)$, $d_G(v, S_2) \leq h_2$.

The next lemma follows by the results in (Choudhary and Gold 2020).

Lemma 5. *Let $G = (V, E)$ be an unweighted graph on n vertices. For any $\delta \in (0, 1)$ and any integer $n_2 \in [n]$, we can compute in polynomial time a $\langle \lfloor \delta D_G \rfloor, \lceil (1-\delta) D_G \rceil \rangle$-dominating set-pair of size-bound $\langle n_1, n_2 \rangle$, where $n_1 \leq \frac{n \log n}{n_2}$.*

Due to space constraints, we give a sketch of the proof of Theorem 8. The complete proof will be given in the full version of the paper.

Proof sketch of Theorem 8. i) We apply the algorithm in Lemma 5 with $n_2 = \lceil \sqrt{n \log n} \rceil$, and $\delta = 1/2$. With this choice of parameters we obtain a $\langle \lfloor D_G/2 \rfloor, \lceil D_G/2 \rceil \rangle$-dominating set-pair (S_1, S_2) where both S_1 and S_2 have size $O(\sqrt{n \log n})$ and one of them can reach every other vertex with a path of length bounded by $\lceil 1/2 \cdot D_G \rceil$. Let us denote with S the set satisfying one of the two conditions in Definition 3. The labeling λ is then constructed as follows.

For every vertex $v \in S$ we compute a SPT T_v rooted at v. Let $e = \{u, w\}$ be an edge of T_v with $d_{T_v}(v, u) = d_{T_v}(v, w) + 1$, we add to $\lambda(e)$ the label $D_G - d_{T_v}(v, u) + 1$. Observe that, with this assignment of labels, each vertex $w \in V(T_v)$ can temporally reach the root v through a temporal path ending with label D_G. We then compute a single SPF F rooted in set S. Let $e = \{u, w\}$ be an edge of F with the vertex u being the one farther from S. We add to $\lambda(e)$ the label $D_G + \min_{v \in S} d_F(v, u)$.

Overall, we added $O(n\sqrt{n \log n})$ labels and since the minimum number of labels to temporally connect a graph is $2n - 4$ (see (Klobas et al. 2024)), we have $|\lambda^*| \geq 2n - 4$ and therefore $|\lambda| = O(\sqrt{n \log n}) \cdot |\lambda^*|$. In conclusion, it can be proved that (G, λ) is temporally connected.

ii) Let (S_1, S_2) be computed by the algorithm in Lemma 5 with parameters $n_2 = \left\lceil \sqrt[3]{D_G n \log^2 n} \right\rceil$ and $\delta = 1/3$. We obtain a $\langle \lfloor 1/3 D_G \rfloor, \lceil 2/3 D_G \rceil \rangle$-dominating set-pair (S_1, S_2) of size-bound $\langle n_1, n_2 \rangle$, where $n_1 \leq \sqrt[3]{D_G^{-1} n^2 \log n}$. If Condition 2 of Definition 3 holds for (S_1, S_2), then λ is constructed as in i) with S being S_2. In this case, the size of λ is upper-bounded by $O(n \sqrt[3]{D_G n \log^2 n})$ which is $O(\sqrt[3]{D_G n \log^2 n}) \cdot |\lambda^*|$.

Therefore, we assume in the remainder that S_1 is such that for all $v \in V(G)$, $d_G(v, S_1) \leq \lfloor 1/3 D_G \rfloor \leq 1/3 D_G$. The labeling λ is then constructed as follows.

Let F be a SPF rooted in set S_1. Let $e = \{u, w\}$ be an edge of F with the vertex u being the one further from S_1. We add to $\lambda(e)$ the two labels $1/3 D_G - \min_{v \in S} d_F(v, u) + 1$ and $4/3 D_G + \min_{v \in S} d_F(v, u)$.

Let $s, s' \in S_1$ be two distinct vertices, let $P_{s,s'} = e_1, e_2, \ldots, e_k$ be a shortest path between s and s' in G. We add to $\lambda(e_i)$ the labels $1/3 D_G + i$ and $1/3 D_G + k - i + 1$ for $i \in [k]$. Notice that with these labels we have created a temporal path from s to s', and a temporal path from s' to s, both using labels in the range $1/3 D_G + 1, 1/3 D_G + 2, \ldots, 4/3 D_G$. The overall number of labels is $|\lambda| = O(n \sqrt[3]{D_G n \log^2 n}) = O(\sqrt[3]{D_G n \log^2 n}) \cdot |\lambda^*|$. Moreover, the largest label used is at most $5/3 D_G$ and it can be proved that (G, λ) is temporally connected. □

Acknowledgements

This work was partially supported by: the European Union - NextGenerationEU under the Italian Ministry of University and Research National Innovation Ecosystem grant ECS00000041 - VITALITY - CUP: D13C21000430001; PNRR MUR project GAMING "Graph Algorithms and MinINg for Green agents" (PE0000013, CUP D13C24000430001); the European Union - Next Generation EU under the Italian National Recovery and Resilience Plan (NRRP), Mission 4, Component 2, Investment 1.3, CUP J33C22002880001, partnership on "Telecommunications of the Future"(PE00000001 - program "RESTART"), project MoVeOver/SCHEDULE ("Smart interseCtions witH connEcteD and aUtonomous vehicLEs", CUP J33C22002880001); ARS01_00540 - RASTA project, funded by the Italian Ministry of Research PNR 2015-2020. The first and second authors acknowledge the support of the MUR (Italy) Department of Excellence 2023–2027.

References

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

Choudhary, K.; and Gold, O. 2020. Extremal distances in directed graphs: tight spanners and near-optimal approximation algorithms. In *Proceedings of the Thirty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '20, 495–514. USA: Society for Industrial and Applied Mathematics.

Deligkas, A.; Eiben, E.; and Skretas, G. 2023. Minimizing Reachability Times on Temporal Graphs via Shifting Labels. In Elkind, E., ed., *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, 5333–5340. International Joint Conferences on Artificial Intelligence Organization. Main Track.

Deligkas, A.; and Potapov, I. 2020. Optimizing Reachability Sets in Temporal Graphs by Delaying. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, 9810–9817. AAAI Press.

Dinur, I.; and Steurer, D. 2014. Analytical approach to parallel repetition. In *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '14, 624–633. New York, NY, USA: Association for Computing Machinery. ISBN 9781450327107.

Elkin, M.; and Peleg, D. 2001. Approximating k-Spanner Problems for $k > 2$. In Aardal, K.; and Gerards, B., eds., *Integer Programming and Combinatorial Optimization*, 90–104. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-540-45535-6.

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

Enright, J.; Meeks, K.; Mertzios, G. B.; and Zamaraev, V. 2021. Deleting edges to restrict the size of an epidemic in temporal networks. *Journal of Computer and System Sciences*, 119: 60–77.

Enright, J. A.; Meeks, K.; and Skerman, F. 2021. Assigning times to minimise reachability in temporal graphs. *J. Comput. Syst. Sci.*, 115: 169–186.

Klobas, N.; Mertzios, G. B.; Molter, H.; and Spirakis, P. G. 2024. The complexity of computing optimum labelings for temporal connectivity. *Journal of Computer and System Sciences*, 146: 103564.

Kortsarz, G. 2001. On the Hardness of Approximating Spanners. *Algorithmica*, 30: 432–450.

Meeks, K. 2022. Reducing Reachability in Temporal Graphs: Towards a More Realistic Model of Real-World Spreading Processes. In Berger, U.; Franklin, J. N. Y.; Manea, F.; and Pauly, A., eds., *Revolutions and Revelations in Computability*, 186–195. Cham: Springer International Publishing.

Mertzios, G.; Michail, O.; and Spirakis, P. 2019. Temporal network optimization subject to connectivity constraints. *Algorithmica*, 81: 1416–1449. EPrint Processing Status: Full text deposited in DRO.

Michail, O. 2016. An Introduction to Temporal Graphs: An Algorithmic Perspective*. *Internet Mathematics*, 12(4).

Molter, H.; Renken, M.; and Zschoche, P. 2024. Temporal reachability minimization: Delaying vs. deleting. *Journal of Computer and System Sciences*, 144: 103549.