Maximizing Reachability via Shifting of Temporal Paths

Argyrios Deligkas
Royal Holloway, University of London, Egham, UK

Michelle Döring
Hasso Plattner Institute, University of Potsdam, Germany

Eduard Eiben
Royal Holloway, University of London, Egham, UK

George Skretas
Hasso Plattner Institute, University of Potsdam, Germany

Georg Tennigkeit
Hasso Plattner Institute, University of Potsdam, Germany

Abstract

We examine the problem of maximizing the reachability of a given source in temporal graphs that are given as the union of k temporal paths, i.e., every given path is a sequence of edges with strictly increasing labels that denote availability in time. This type of temporal graphs represent train networks. We consider shifting operations on the labels of the paths that maintain their temporal continuity. This means that we can move the availability of a temporal edge later or earlier in time, and propagate the shifts to all other affected edges of the path in order to preserve its temporal connectivity. We study the parameterized complexity of the problem with respect to the number of paths k, and the total budget b, where b is the maximum number of shifts we are allowed to perform. Our results reveal that fixed parameter tractability can be achieved (1) when parameterized both by k and b, and (2) when parameterized by k, and b is unlimited. In almost every other case, e.g., parameterized by a single parameter or parameterized by k, while having a bound on b, we establish intractability lower bounds that are matched by XP algorithms.

2012 ACM Subject Classification Mathematics of computing → Graph theory; Theory of computation → Graph algorithms analysis; Theory of computation → Fixed parameter tractability

Keywords and phrases temporal graphs, temporal path number, path graph, reachability, train system, network optimization, parameterized complexity

Digital Object Identifier 10.4230/LIPIcs.ESA.2026.122

Related Version *Full Version*: `https://arxiv.org/abs/2605.11873`

Funding *Argyrios Deligkas*: EPSRC Grant EP/X039862/1 "NAfANE: New Approaches for Approximate Nash Equilibria".
Michelle Döring: HPI Research School on Foundations of AI (FAI).
Eduard Eiben: Engineering and Physical Sciences Research Council (EPSRC) grant UKRI4530: Exploring Parameterized Complexity in Blockchain Systems.
Georg Tennigkeit: HPI Research School on Foundations of AI (FAI).

All formal proofs and many helpful illustrations of the statements can be found in the full version of the paper.

1 Introduction

Probably the simplest way to describe the problem of maximizing reachability on a temporal graph is via a train network. We are given the routes and corresponding timings for each train, along with a specific station s. The question is, how should we *modify* the network such that s can reach as many other stations as possible?

Formally, in a temporal graph, every edge has a set of *labels* that correspond to the specific timesteps at which the edge can be used. The modifications of the temporal graph proposed in the past ranged from the addition of (non-existing) edges [1] and labels, to *advancing* or *delaying*, i.e., *shifting*, the labels of the edges [6, 7, 15, 16]. In real-life networks, though, especially physical ones, it is almost impossible to add new edges. We cannot simply add one direct connection from Warsaw to L'Aquila because we have decided so. Hence, shifting seems a more suitable operation in applications.

Furthermore, in certain scenarios, like train networks, delaying or advancing a temporal edge cannot happen in isolation. Trains pass through many different stations, and we cannot simply change the time at which it goes between two consecutive stations without potentially affecting the schedule of the entire train line. Imagine, for example, a train that leaves from A to B at 9:00 and from B to C at 10:00. We cannot delay the A-B connection to 12:00 and maintain the B-C connection at 10:00; the delay *must propagate* and connection B-C *must* be after 13:00. Thus, changes on a train line that occur due to shifting must propagate until feasibility on the corresponding temporal path is maintained.

While maximum reachability via delaying operations had been studied in the past, the above-mentioned dimension of real life networks was overlooked. The goal of the paper is to remedy this situation and resolve the (parameterized) complexity of maximum reachability of a vertex via shifting on temporal k-path graphs, i.e., temporal graphs that are defined by a collection of k temporal base-paths, denoted MaxReach-ShiftPath (MR-SP for short).

MaxReach-ShiftPath (MR-SP)

Input: A k-path graph $\mathcal{G}$, a budget $b \in \mathbb{N}$, and a vertex s.
Problem: Find a sequence of shifting operations with a total cost of at most b, such that the number of vertices that s can reach in the resulting $\mathcal{G}'$ is maximized.

More formally, a shifting operation either increases the time label of a specified edge (delaying) or decreases it (advancing). When delaying an edge, the delay propagates to subsequent edges on the unique containing base-path insofar as the labels remain strictly increasing. Likewise, advancing an edge propagates to preceding edges on the containing base-path. If there was any waiting time between edges in the base-path, it reduces the propagating shift, as if the train can make up for its delay by waiting less at the station.

The *cost* of one shifting operation is equal to the absolute value by which the specified edge is modified, while propagated changes do not count towards the cost. This reflects that the budget measures only the *active intervention* by the operator, namely how much a single departure is intentionally rescheduled. The resulting propagation along the same base-path is then a forced consequence of preserving a feasible timetable for that line, rather than an additional independent modification.

Figure 1 demonstrates an instance of MR-SP alongside a solution.

1.1 Our Contribution

We begin our investigation by establishing some observations and deriving structural properties of optimal solutions in k-path graphs that help us design our algorithms. The most fundamental observation, that our algorithms heavily exploit, is the fact that in any k-path graph if vertex u reaches vertex v, then there is a temporal path from u to v that uses *at most one* contiguous segment from each base-path. This implies that there exists an optimal

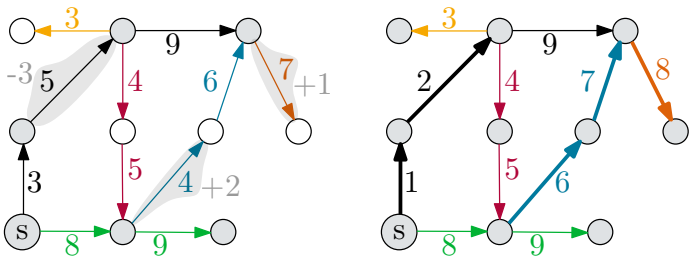

Figure 1 An example instance of MR-SP on a temporal 6-path graph, whose base-paths are depicted with different colours, along with a sequence of operations that allows s to reach all vertices for a cost of 6. Gray vertices are reached by s.

solution where each base-path *switches* to a different base-path at most once. This in turn implies the existence of a *switch-path-tree*, denoted SPT. This is a directed tree, rooted at the source, which describes how temporal paths starting from the source should switch in order to reach different base-paths. Crucially, we prove that we can enumerate all SPTs of a k-path graph in $\mathcal{O}(k^{k-2})$ time (Lemma 11).

Then, we continue by studying the complexity of our problem with respect to two dimensions: a) the parameters we consider and b) the types of shifting operations we are allowed to perform, i.e., delay, advance, or both. We begin our exploration from the most relaxed version of the problem where we allow both delays and advances, and the budget is unconstrained. Although the problem is NP-hard (Theorem 12), when parameterized by the number k of base-paths we prove that it becomes FPT (Theorem 14). At a high level, our algorithm guesses the SPT of an optimal solution and then executes a suitable modification of BFS to compute the shifts that maximize reachability.

In the general setting where we must adhere to a budget b, we prove that if we allow only delays, then the problem becomes W[1]-hard parameterized by k (Theorem 16). Conversely, if we allow advances, or both advances and delays, the complexity of the problem is partially answered: the problem is W[1]-hard if we ask to find the optimal shifts for a *fixed* SPT (Theorem 17), and its computational complexity is open otherwise. On the positive side though, we match all above-mentioned hardness results with upper bounds by designing an XP-algorithm that tackles all versions of the problem (Theorem 20). This algorithm guesses the essential switching points of an optimal solution and then uses an integer linear program (ILP) to find the minimum cost needed to realize them. The trick is to precompute the effects of propagation to keep the ILP sufficiently small.

Then, we consider the complexity of the problem when the budget b is a parameter. Unfortunately, in [6] they essentially provide W[2]-hardness for all types of shifts. We complement these lower bounds by observing that there exists a straightforward XP algorithm (Theorem 18). Hence, in order to establish tractability we have to augment our set of parameters. Indeed, as we show in Theorem 21 and Theorem 22 the problem becomes fixed parameter tractable by k and b for every allowed operation of shifts. This is our most technical result, which works at a very high level as follows. Again, we start by guessing the SPT of an optimal solution. This time, since b is a parameter, we can guess in fpt time the allocated budget for delaying and advancing at a switch per base-path as well as the interactions of propagation between switches. Then for each base-path, we "pseudo"-greedily pick the best switch that aligns with these guesses. The main difficulty stems from the fact that the switches we pick, both between base-paths and on the same base-path, are *not* independent. In particular, we have to pick some switches in batches to make sure that the

Table 1 Overview of our results. All settings are NP-hard. The rows show the parameterized complexity of the MaxReach problem variants with respect to the number of base-paths k, the budget b, and their combination; "fxd SPT" stands for "fixed switch-path-tree" which denotes the special case where the order in which we have to traverse the base-paths is additionally given. The $b = \infty$ column represents our results for delaying/advancing/shifting without a budget constraint, so there is no parameterization by b.

MaxReach	$b = \infty$	Delay	Advance	Shift
by k	FPT (Thm. 14)	W[1]-h (Thm. 16), XP (Thm. 20)	open, XP (Thm. 20)	
by k fxd SPT			W[1]-hard (Thm. 17), XP (Thm. 20)	
by b	—	W[2]-h (Thm. 15), XP (Thm. 18)		
by k+b	—	FPT (Thm. 21)	FPT (Thm. 22)	

placement of dependent switches displays the guessed interactions, which we achieve via an intricate sequence of arguments.

A concise overview of our results is depicted in Table 1.

1.2 Related work

We survey work on reachability-motivated modifications which aim to maximize/minimize reachability or satisfy specific travel demands, followed by other temporal graph modification problems and work on the k-path graph model.

Shifting and scheduling to maximize reachability. Two previous works aim to maximize reachability in some sense and are thus closely related to ours. Deligkas, Eiben and Skretas [6] study the ReachFast problem of minimizing the time for one or multiple sources to reach all vertices in a temporal graph using advancing, delaying, or both (shifting). Their setting differs from ours in two key ways: (1) they work on general temporal graphs (not k-path graphs), which removes propagating interactions, and (2) they aim to reach all vertices while minimizing travel time, while we aim to maximize the number of vertices reached.

Brunelli, Crescenzi and Viennot [2] study MaxReach Trip Temporalization: given a collection of walks in a static directed graph, assign a starting time to each walk that turns it into a temporal walk with no waiting at the vertices, such that the number of reachable pairs of nodes in the resulting temporal graph is maximized. They consider one-to-one, one-to-all, and all-to-all reachability variants.

Edge modification for other objectives. There are many previous works on restricting the reachability of one or multiple sources via edge modifications [7, 11, 12, 16]. Enright, Larios-Jones, Meeks and Pettersson [10] study reachability under uncertainty, where ζ edge labels may be perturbed by $\pm\delta$. Kutner and Larios-Jones [14] introduce the *Temporal Reachability Dominating Set* (TaRDiS) problem: can a population be fully reached from a small set of k initially infected individuals, and give parameterized complexity results. Kutner and Sommer [15] study the DelayBetter problem where a fixed set of passengers' travel demands must be satisfied, rather than a global reachability objective.

***k*-Path Graphs.** The k-path graph model was introduced by Deligkas, Döring, Eiben, Goldsmith, Skretas, Tennigkeit [4], who define the Exact Edge-Cover problem as a mathematical way of identifying temporal graphs as public transport networks. An exact

edge-cover is a decomposition of the temporal edge set into temporal-edge-disjoint trips. This naturally leads to the definition of a k-path graph as a temporal graph being given as a union of temporal paths, as well as the *temporal path number* as the minimum number of temporal paths that a temporal graph's edge set can be partitioned into. The same authors study the parameterized complexity of temporal connected components with respect to the temporal path number in [5].

2 Preliminaries

A *temporal graph* $\mathcal{G} = (V, E, \lambda)$ consists of a static graph $G = (V, E)$, called the *footprint*, along with a labeling function $\lambda\colon E \to 2^{\mathbb{Z}}$. Note that we also permit negative labels. All temporal graphs we consider in this paper are directed. A pair (e, t), where $e \in E$ and $t \in \lambda(e)$, is a *temporal edge* with *label* t. We denote the set of all temporal edges by $\mathcal{E}$. The range of λ is referred to as the *lifetime* Λ. The static graph $G_t = (V, E_t)$, where $E_t = \{e \in E\colon t \in \lambda(e)\}$, is called the *snapshot* at time t.

A *temporal path* is a sequence of temporal edges $\langle (e_i, t_i)\rangle$ where $\langle e_i\rangle$ forms a path in the footprint and the time labels $\langle t_i\rangle$ are strictly increasing. Given a temporal path P and vertices u, v on P, we write $P[u : v]$ for the temporal sub-path of P starting from u and ending at v. We further write $P[u :]$ for the suffix starting at u, and $P[: v]$ for the prefix ending at v. Because we never consider static paths and only use temporal graphs in this paper, we will often refer to them simply as *paths* for brevity. If there exists a temporal path from u to v, we say u *reaches* v. We refer to the set of vertices that v reaches in $\mathcal{G}$ as $R^{\mathcal{G}}(v)$.

Parameterized complexity. We refer to the standard books for a basic overview of parameterized complexity theory [3, 9, 13]. A problem is *fixed-parameter tractable* (FPT) by a parameter k if it can be solved in time $f(k) \cdot \mathtt{poly}(n)$, where f is a computable function. Showing that a problem is W[1]-hard parameterized by k rules out the existence of such an FPT algorithm under the assumption W[1] $\neq$ FPT (the same goes for W[2]-hardness). Problems admitting an algorithm with running time $\mathcal{O}(n^{f(k)})$ for some computable function f belong to the class XP.

3 k-Path Graphs and Path-Shifting

We consider the class of temporal graphs that are represented as a union of k temporal paths. In order to avoid any confusion between the k temporal paths that define the graph and other temporal paths in the graph, we will refer to the former as *base-paths*.

▶ **Definition 1** (k-Path Graphs). *A k-path graph $\mathcal{G} = (V, E, \lambda)$ is a temporal multigraph*[1] *for which there exists a collection $\mathcal{P} = \{P_1, \ldots, P_k\}$ of k temporal base-paths such that the multiset union of their edges is exactly $\mathcal{E}$. We may denote such a graph as $\mathcal{G} = \biguplus \mathcal{P} = \biguplus_{i\in[k]} P_i$.*

For vertices v, w on the same base-path P we say that $v <_P w$ iff v occurs on P before w.

[1] We define k-path graphs as multigraphs because shifting operations might cause an edge to have the same label as another edge between the same pair of vertices. Without multigraphs, such an edge would effectively be removed from the graph and it might no longer be a k-path graph.

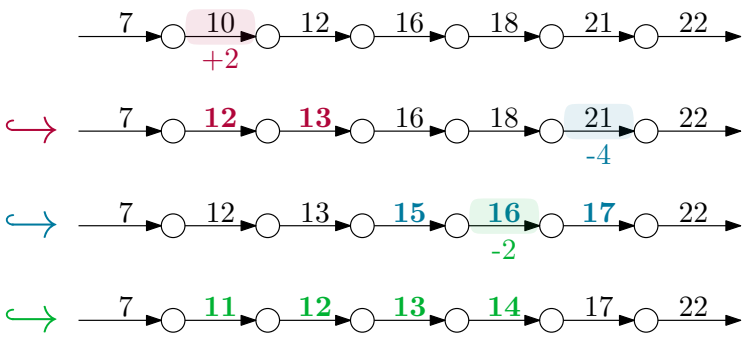

Figure 2 An example sequence of three shifting operations applied to a base-path.

Base-Path Shifting. A *shifting operation* on a k-path graph $\mathcal{G}$ is a pair $((e,t),\delta)$ where $(e,t) \in \mathcal{E}$ and $\delta \in \mathbb{Z}$. Applying $((e,t),\delta)$ to $\mathcal{G}$ results in a k-path graph $\mathcal{G}'$ where (e,t) is replaced with $(e,t+\delta)$. If $\delta > 0$, we call it a *delaying operation*, and the delay propagates to subsequent edges on the unique base-path P_i containing (e,t): if e' is the d-th edge after e on P_i, its label is increased to $t+\delta+d$ if it is currently lower. Likewise if $\delta < 0$, we call it an *advancing operation*, and it propagates to the preceding edges on the same base-path: if e' is the d-th edge before e on P_i, its label is reduced to $t+\delta-d$ if it is currently higher. Formally, applying the shifting operation $((e,t),\delta)$ with $(e,t) \in P_i$ to $\mathcal{G}$ creates a new k-path graph $\mathcal{G}' = (V, E, \lambda')$ with the labeling function λ' defined as follows:

$$\forall e' \in E\colon \lambda'(e') = \begin{cases} \lambda(e'), & \text{if } e' \notin P_i, \\ t+\delta, & \text{if } e' = e, \\ \max(\lambda(e'), t+\delta+d), & \text{if } e' \text{ is the } d\text{-th edge after } e \text{ on } P_i, \\ \min(\lambda(e'), t+\delta-d), & \text{if } e' \text{ is the } d\text{-th edge before } e \text{ on } P_i. \end{cases} \tag{1}$$

The *cost* of an operation $((e,t),\delta)$ is $|\delta|$. Propagation is not accounted in the cost. A sequence of shifting operations $\mathcal{S}$ is applied one after another, and its cost is the sum of costs of all operations. See Figure 2 for an example.

With these definitions in place, let us restate the problem definition:

MaxReach-ShiftPath (MR-SP)

Input: A k-path graph $\mathcal{G}$, a budget $b \in \mathbb{N}$, and a vertex s.

Problem: Find a sequence $\mathcal{S}$ of shifting operations with a total cost of at most b, such that $|R^{\mathcal{G}'}(s)|$ is maximized, where $\mathcal{G}'$ is the resulting k-path graph after applying $\mathcal{S}$ to $\mathcal{G}$.

If we allow only delaying or only advancing operations, we call the problem MaxReach-DelayPath (MR-DP) or MaxReach-AdvancePath (MR-AP) respectively.

Note that the order of shifting operations matters (for example, switching the order of the last two operations in Figure 2 would return a different labeling). However, it is generally optimal to apply delaying operations beginning-to-end along a base-path, and apply advancing operations in the reverse order; this way, operations are applied on top of the propagation of prior operations, maximizing the overall change caused. We further observe that it is never efficient to cause a delay that propagates into previously advanced edges as this would undo part of the previous advancing operation, and vice versa.

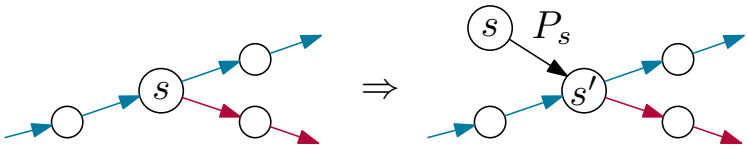

Figure 3 A k-path graph (left) without a distinct base-path P_s and its adjustment $k+1$-path graph (right) with an additional base-path P_s (black) of length 1.

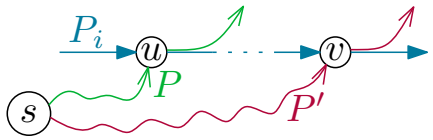

Figure 4 The two paths P and P' both start at s and use a segment of the base-path $P_i \in \mathcal{P}$, but P switches onto it at u and P' does so at v. Because u is before v along P_i we could replace $P'[:v]$ in P' with $P[:u] \circ P_i[u:v]$, making both switch onto P_i in the same vertex.

To simplify notation later, we slightly adjust our representation of the given source s. In the rest of this paper, we will assume that there is exactly one base-path that contains s, which we will call P_s. For k-path graphs where this is not the case, we rename s to s', add a new source vertex s, and a new base-path P_s with only the edge (s, s') that has a very small label (smaller than all other labels in the graph, even if they get advanced by a reasonable amount). See Figure 3. This does not affect the studied problem as any path that started in s before the adjustment is now simply preceded by this new edge.

Properties of k-Path Graphs. We begin with a fundamental observation about k-path graphs that we will use throughout the paper:

▶ **Lemma 2.** *In a k-path graph, for any two vertices u, v where u reaches v, there is a temporal path from u to v that uses at most one contiguous segment from each base-path.*

Furthermore, if there are two paths starting from a source s that switch onto base-path P_i at different vertices, both paths could switch onto P_i using the vertex that is earlier on P_i (see Figure 4). Crucially, this means that we can maintain the entire reachability of s even when we restrict paths to only switch onto a base-path at one designated switch-vertex.

Switch-Vertex-Sets and Switch-Path-Trees. Based on the previous insight, we can represent the vertices reachable from the source using solely the base-paths and switches between them. Since every path in $\mathcal{G}$ starting from s starts with the base-path P_s and then continues onto other base-paths, this yields a tree structure rooted in P_s. In the following, we define the mathematical objects that model this representation, beginning with a set of vertices that uniquely characterizes the points at which any path starting in s switches between base-paths. See Figure 5 for an example.

▶ **Definition 3** (Switch-vertex-set). *For a k-path graph $\mathcal{G} = \biguplus \mathcal{P}$ with vertex set V, let $\mathcal{V}$ be a subset of $V \times \mathcal{P} \times \mathcal{P}$. $\mathcal{V}$ is a* switch-vertex-set *(SVS) if the following holds:*

- ***Switch exists:*** *For each $(v, P_{from}, P_{to}) \in \mathcal{V}$, P_{from} contains an edge entering v, and P_{to} contains an edge leaving v.*
- ***At most one switch to each base-path:*** *For each $P \in \mathcal{P} \setminus \{P_s\}$, there is at most one $(v, P_{from}, P_{to}) \in \mathcal{V}$ with $P = P_{to}$. We refer to this v as $v^P_{\mathcal{V}}$. There is no switch onto P_s.*

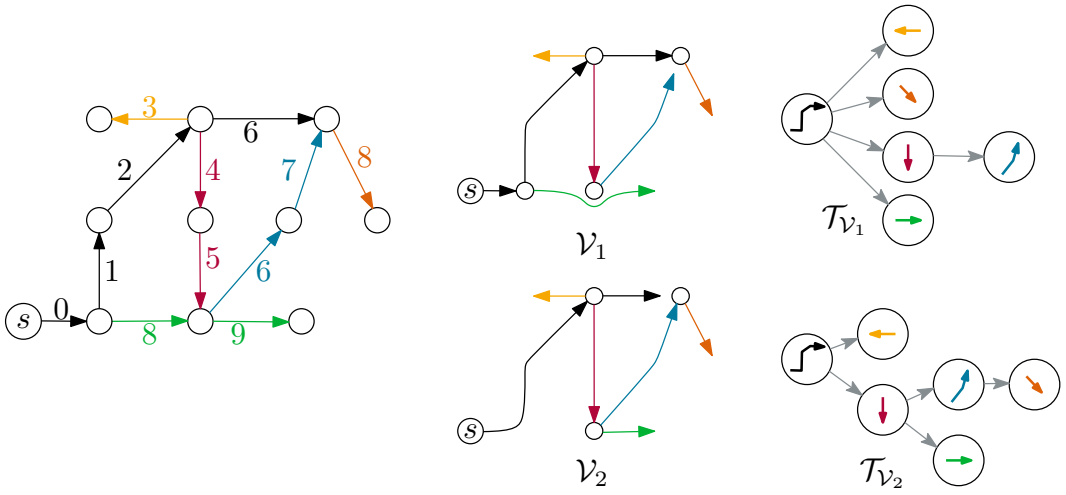

Figure 5 A 6-path graph $\mathcal{G}$ and two SVSs $\mathcal{V}_1, \mathcal{V}_2$ (see Definition 3) that are both temporal in $\mathcal{G}$, along with their respective implied SPTs (see Definition 9).

- ***Switch onto base-path before switching off:*** *For each* $(v, P_a, P_b) \in \mathcal{V}$*, all* v' *with* $(v', P_b, P_c) \in \mathcal{V}$ *are on* P_b *after* v*.*
- ***Base-path switches form a tree:*** *Let* $\mathcal{T}_{\mathcal{V}}$ *be the directed graph on* $\mathcal{P}$ *defined via the edge-set* $\{(P_{from}, P_{to}) : \exists (v, P_{from}, P_{to}) \in \mathcal{V}\}$*.* $\mathcal{T}_{\mathcal{V}}$ *is a directed tree rooted at* P_s*.*

Note that this definition disregards time labels. We will separately distinguish whether a specified switch or switch-vertex-set is temporal.

▶ **Definition 4** (temporal switch)**.** *We say that a switch* (v, P_{from}, P_{to}) *is* temporal in $\mathcal{G}$ *if the edge of* P_{from} *entering* v *has a smaller label than the edge of* P_{to} *leaving* v*. An* SVS $\mathcal{V}$ *is* temporal in $\mathcal{G}$ *if all contained switches are temporal in* $\mathcal{G}$*.*

We are interested in paths that only switch onto base-paths according to a given SVS, as per the following definition:

▶ **Definition 5** ($\mathcal{V}$-respecting, $\mathcal{V}$-reachability)**.** *We say that a path in* $\mathcal{G}$ *is* $\mathcal{V}$-respecting *if, for any two consecutive edges* $((u, v), t_1) \in P_a$ *and* $((v, w), t_2) \in P_b \neq P_a$*, we have* $(v, P_a, P_b) \in \mathcal{V}$*. Furthermore, the* $\mathcal{V}$-reachability $R^{\mathcal{G}}_{\mathcal{V}}(s)$ *of a vertex* s *is the set of all vertices* v *for which there is a* $\mathcal{V}$*-respecting path from* s *to* v*.*

If an SVS $\mathcal{V}$ is temporal, the $\mathcal{V}$-reachability is simply the combined suffixes of each base-path starting from their respective switch:

▶ **Lemma 6.** *Let* $\mathcal{G}$ *be a* k*-path graph with a source* s *and let* $\mathcal{V}$ *be an* SVS *that is temporal in* $\mathcal{G}$*. Then* $R^{\mathcal{G}}_{\mathcal{V}}(s) = \bigcup_{P \in \mathcal{P}} \{v \in P[v^P_{\mathcal{V}} :]\}$*.*

Next, we show that for maximizing the reachability of s, it is sufficient to only consider reachability with respect to a specific SVS.

▶ **Lemma 7.** *For a* k*-path graph* $\mathcal{G}$ *and a vertex* s*, there exists a temporal* SVS $\mathcal{V}$ *such that* $R^{\mathcal{G}}(s) = R^{\mathcal{G}}_{\mathcal{V}}(s)$*.*

An SVS further simplifies the shifting operations we might consider to optimize reachability. Given an SVS $\mathcal{V}$ as per the previous lemma, we find that reachability can be maximized by only applying operations to the edges referenced in some switch of $\mathcal{V}$.

▶ **Lemma 8.** *Given a k-path graph $\mathcal{G}$, there is an optimal solution $\mathcal{G}^*$ and an* SVS $\mathcal{V}$ *with $R^{\mathcal{G}^*}(s) = R^{\mathcal{G}^*}_{\mathcal{V}}(s)$ such that $\mathcal{G}^*$ can be constructed via a sequence of shifting operations with optimal cost where for each $(v^*_P, P_{from}, P) \in \mathcal{V}$ there is:*

- *at most one advance $\alpha_P \leq 0$ applied to the temporal edge of P_{from} entering v^*_P, and*
- *at most one delay $\delta_P \geq 0$ applied to the temporal edge of P leaving v^*_P.*

Lastly, we define *switch-path-trees* which are a simpler representation as they only specify which base-path switches onto each base-path but not the vertices used to switch. In particular, every switch-vertex-set $\mathcal{V}$ defines a switch-path-tree $\mathcal{T}_{\mathcal{V}}$, though not every switch-path-tree is defined by some switch-vertex-set. See Figure 5 for examples.

▶ **Definition 9** (switch-path tree)**.** *For a k-path graph $\mathcal{G} = \biguplus \mathcal{P}$, let $\mathcal{T} = (V_{\mathcal{T}}, E_{\mathcal{T}})$ be a directed graph with $V_{\mathcal{T}} \subseteq \mathcal{P}$ (meaning every vertex in $\mathcal{T}$ represents a base-path). $\mathcal{T}$ is a* switch-path-tree *(or* SPT*) if it is a directed tree rooted at P_s.*

Akin to Definition 5, a path in $\mathcal{G}$ is $\mathcal{T}$-respecting if, for any two consecutively edges $((u,v),t_1) \in P_a$ and $((v,w),t_2) \in P_b \neq P_a$, we have $(P_a, P_b) \in E_{\mathcal{T}}$. The $\mathcal{T}$-reachability $R^{\mathcal{G}}_{\mathcal{T}}(s)$ of a vertex s is the set of all vertices v for which there is a $\mathcal{T}$-respecting path from s to v.

We conclude by providing parameterized algorithms for enumerating all SVSs and SPTs.

▶ **Lemma 10.** *All* SVSs *of a k-path graph with n vertices can be enumerated in $\mathcal{O}\big((kn)^{k-1}\big)$ time.*

▶ **Lemma 11.** *All* SPTs *of a k-path graph can be enumerated in $\mathcal{O}\big(k^{k-2}\big)$ time.*

4 Unconstrained Budget: Path Temporalization

To get familiar with the problem, we will first consider a restricted setting without a budget constraint, i. e., the budget is infinite.

For this setting we observe that the permitted type of operations and even the original base-path labels are irrelevant: Without a limited budget, we can delay or advance each edge on each base-path by an arbitrary amount. The propagation of these shifts will keep the labels on each path strictly increasing, thereby retaining the temporal path property. This way, we can create any relative temporal order of the edges, so long as edges along a base-path are strictly increasing. Because temporal reachability only requires the traversed edges to have increasing labels, only the relative temporal order of edges is relevant for our problem. Thus neither the initial labels nor the type of allowed operation have an effect. To simplify, we define the following equivalent *temporalization problem*:

MaxReach-PathTemporalization (MR-PT)

Input: A collection of static paths $\mathbb{P}$ and a vertex s.
Problem: Find a labeling λ for the edges of each path in $\mathbb{P}$ such that the labels along each paths are strictly increasing and the number of vertices that s can reach in the resulting k-path graph $(\biguplus \mathbb{P}, \lambda)$ is maximized.

This problem is closely related to the TripTemporalization problem introduced by [2]. Here, we are given a collection of walks and have to assign a starting time to each walk. This starting time becomes the label of the first edge of the walk, and all following edges are labeled incrementally[2]. In this way, the collection of walks is turned into a temporal graph, and the goal is to maximize some notion of reachability in that resulting graph.

[2] More precisely, each following edge is labeled with the accumulated travel time of all preceding edges. However, the authors show that it suffices to consider instances in which every edge has travel time 1.

The two problems are related in the following way: On the one hand, [2] allow walks instead of paths and are thus slightly more general. On the other hand, their model is also slightly more restrictive, since the labels along a walk are determined entirely by its starting time, and no additional waiting times can be inserted.

Since their input consists of a collection of walks, we will refer to this operation as *walk temporalization* (instead of trip temporalization). The specific variant in which the goal is to maximize the reachability of a given source is then called MaxReach-WalkTemporalization (MR-WT), and is proven to be NP-hard. We show that this hardness transfers to MR-PT.

▶ **Theorem 12.** *MR-PT is* NP*-hard.*

Despite this hardness result, we can show that there exists an FPT algorithm parameterized by k. For this, we first show that given a switch-path-tree $\mathcal{T}$, there is an algorithm to determine in polynomial time a switch-vertex-set $\mathcal{V}$ from among all switch-vertex-sets that imply $\mathcal{T}$, such that the reachability of s with respect to $\mathcal{V}$ is maximized.

▶ **Lemma 13.** *Given a k-path graph $\mathcal{G}$, a source s, and an* SPT *$\mathcal{T}$, one can determine in polynomial time an* SVS *$\mathcal{V}$ with $\mathcal{T}_{\mathcal{V}} = \mathcal{T}$ such that for all other* SVSs *$\mathcal{V}'$ with $\mathcal{T}_{\mathcal{V}'} = \mathcal{T}$ we have $|R^{\mathcal{G}}_{\mathcal{V}'}(s)| \leq |R^{\mathcal{G}}_{\mathcal{V}}(s)|$, if any such* SVSs *exist.*

With the previous lemma, we can efficiently find an optimal switch-vertex-set for a given switch-path-tree. We can combine this with the fact that there are only $f(k)$ many switch-path-trees, and then describe how to make any switch-vertex-set temporal using infinite budget. This yields the following FPT result.

▶ **Theorem 14.** *MR-PT is* FPT *when parameterized by k.*

5 Limited Budget

We now turn to the originally defined problem where we have to adhere to a limited budget b. This means that a solution must indeed be sufficiently similar to the given labeling and we cannot construct a labeling from scratch. We show additional hardness results for this setting, followed by an XP algorithm by k and FPT algorithms by $b + k$.

5.1 Negative Results

It is already known that this version of the problem is NP-hard, as well as W[2]-hard by b, for individual edges [6]. As a k-path graph may consist exclusively of base-paths of length 1 (i. e., individual edges), this hardness directly transfers to our problem.

▶ **Theorem 15** ([6])**.** *MR-D/A/SP is* NP*-hard and* W[2]*-hard when parameterized by b.*

The natural next question given the previous results is whether we can hope for an FPT algorithm parameterized by k, as existed for the relabeling setting (Theorem 14). However, we can show W[1]-hardness via a reduction from Multi-Colored Independent Set (MCIS), which is known to be W[1]-hard [3, Theorem 13.25].

For MCIS, we are given a $k \in \mathbb{N}$ and an undirected static graph $G = (\bigcup_{C \in \mathcal{C}} C, E)$, where $\mathcal{C}$ contains k disjoint node sets (colors). The question is whether there is a colorful independent set of size k, that is, a selection of one node of each color such that no two selected nodes share an edge.

▶ **Theorem 16.** *MR-DP is* W[1]*-hard when parameterized by k.*

Given an instance (G, k) of MCIS, we construct a $2k+1$-path graph $\mathcal{G}$ and a budget b such that a multicolored independent set of size k exists in G if and only if it is possible to make s reach *all* vertices in $\mathcal{G}$ using delaying operations within budget b. We give an example of the construction below, along with some intuition.

Construction. We will first describe the color-gadget in which we represent choosing a node as specific delaying operations. Afterwards we show how to represent the edges E of G to restrict the available choices.

For this construction we use time labels that are multiples of some large number ω, sometimes with some additive terms to make the edges along a base-path strictly increasing. We want to choose ω large enough that the additive terms in our arguments never add up to ω itself, so that reachability can usually be gleaned from the multiplier of ω while the additive terms can be disregarded. Choosing $\omega = |V(G)|^4$ is sufficient for this purpose.

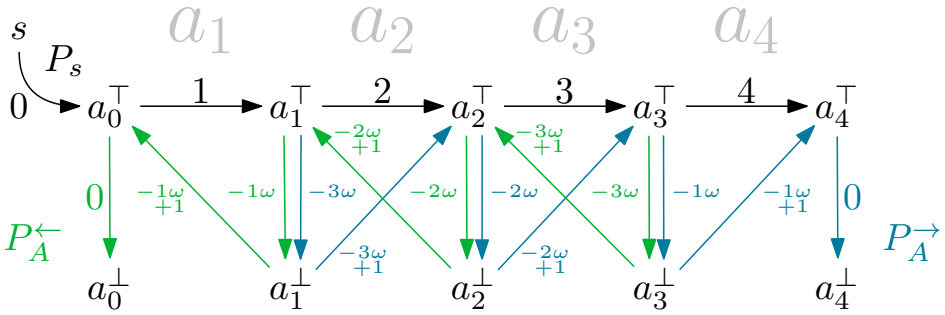

Figure 6 The color-gadget for a color $A \in \mathcal{C}$ where $A = \{a_1, a_2, a_3, a_4\}$. Each node a_i corresponds to the gap between the vertex-pairs $(a_{i-1}^\top, a_{i-1}^\bot)$ and $(a_i^\top, a_i^\bot)$. In order to reach all $a^\bot$-vertices, we need to delay $P_A^\rightarrow$ (blue) and $P_A^\leftarrow$ (green) at adjacent vertex-pairs in order to switch onto them from P_s (black). Such delaying operations cost at least $(|A|-1)\cdot\omega$.

We begin with a base-path P_s that starts in a vertex s. Now construct a gadget for each color $A = \{a_1, \dots a_n\} \in \mathcal{C}$ as seen in Figure 6. Intuitively, a node $a_i \in A$ corresponds to the *gap* between the vertex-pairs $\{a_{i-1}^\top, a_{i-1}^\bot\}$ and $\{a_i^\top, a_i^\bot\}$. Choosing a_i then corresponds to delaying $P_A^\leftarrow$ left of the gap, and delaying $P_A^\rightarrow$ right of the gap.

Now to represent the edges of G, we will create additional vertices and insert them into existing base-paths. For each $e = \{a_i, b_j\} \in E$, create a new vertex e_{a_i,b_j} and insert it into $P_A^\leftarrow$ and $P_A^\rightarrow$ such that they are in the gap associated with a_i. Likewise, insert them into $P_B^\leftarrow$ and $P_B^\rightarrow$. See Figure 7 for an example.

We choose the budget of the MR-DP instance as $b = \sum_{C\in\mathcal{C}}((|C|-1)\cdot\omega) + (\omega - 1)$.

MCIS ⇒ MR-DP. Given an independent set IS in G of size k, let $IS(A)$ be the index of the vertex chosen for color A. We can delay $P_A^\rightarrow$ after $a_{IS(A)}^\top$, and delay $P_A^\leftarrow$ after $a_{IS(A)-1}^\top$, by so much that switching to them from P_s in that vertex is possible. This costs $(|A|-1)\cdot\omega$ plus some additive non-ω-term, which is within the budget. Also, all vertices become reachable, which for e-vertices relies on IS being an independent set.

MR-DP ⇒ MCIS. Given a solution to MR-DP where s can reach all vertices, we can show that for each color A, $P_A^\rightarrow$ and $P_A^\leftarrow$ must be delayed to have at least some positive labels. Let $(a_\ell^\top, a_\ell^\bot)$ (resp $(a_r^\top, a_r^\bot)$) be the first edge along $P_A^\leftarrow$ (resp $P_A^\rightarrow$) to receive a positive label. We prove that $\ell = r - 1$ due to the budget constraint. We can then show that choosing a_r for A and doing likewise for every other color yields an independent set, which relies on s being able to reach all e-vertices.

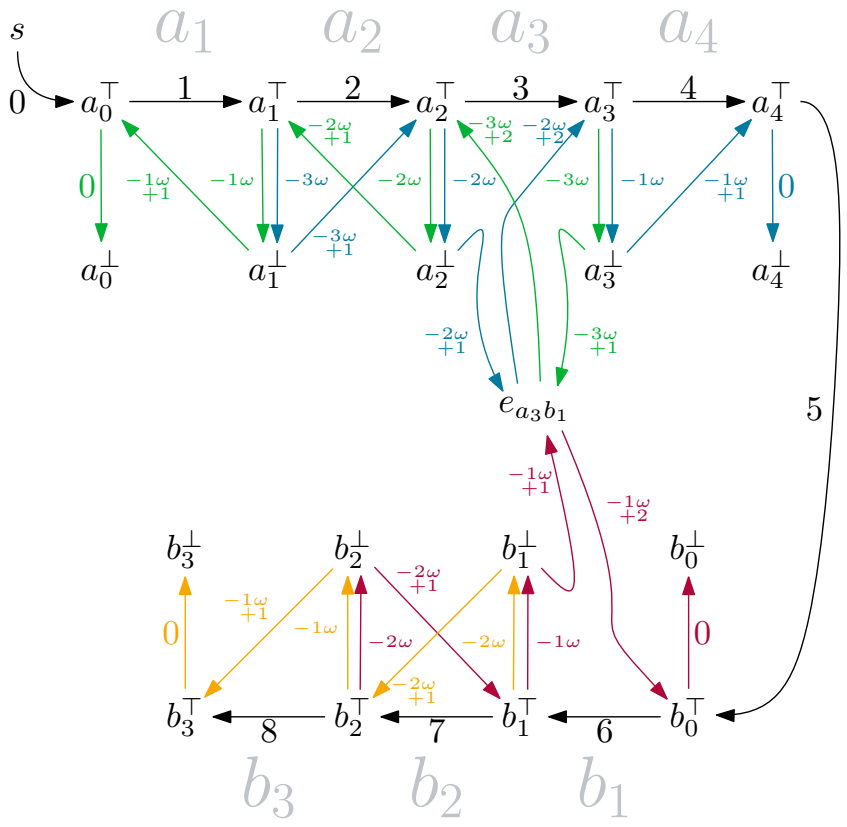

Figure 7 The representation of an edge between $a_3 \in A$ and $b_1 \in B$ in $\mathcal{G}$.

While delaying a path makes it easier to switch onto that path, advancing a path makes it easier to switch from that path onto other paths. This changes the strategy for making a specific SVS temporal: Where delaying a path can only make the switch onto that path temporal, advancing a path can make many switches from that path to other paths temporal.

In turn, the reduction used in Theorem 16 does not transfer to the case of advancing. We leave MR-AP parameterized by k as an open question, though we do give the following starting point: The strategy of our other positive results (guessing an SPT $\mathcal{T}$ and finding shifting operations to maximize the reachability respecting $\mathcal{T}$) would likely not work for this problem, as solving MR-AP *for a specific* SPT is W[1]-hard.

▶ **Theorem 17.** *Given an* SPT $\mathcal{T}$*, MR-D/A/SP for maximizing the* $\mathcal{T}$*-reachability of the source is* W[1]*-hard when parameterized by* k*.*

This reduction is again from Multi-Colored Independent Set and functions very similarly to the one for Theorem 16. Now we adapt the construction in a way where any switch can be temporalized both via advancing an edge or delaying an edge, for the same cost. See Figure 8 for the overall construction and the chosen $\mathcal{T}$. The budget is $b = \sum_{C \in \mathcal{C}} (|C| - 1) \cdot \omega$.

The reason the above reduction does not work in the general case is because other switches than those allowed by $\mathcal{T}$ may be more useful without corresponding to an independent set in G. For example, in Figure 8, advancing the entire $P_A^{\rightarrow}$ by 6 allows switching from $P_A^{\rightarrow}$ to $P_A^{\leftarrow}$ at a much lower cost that the intended switch from $P_A^{\top}$.

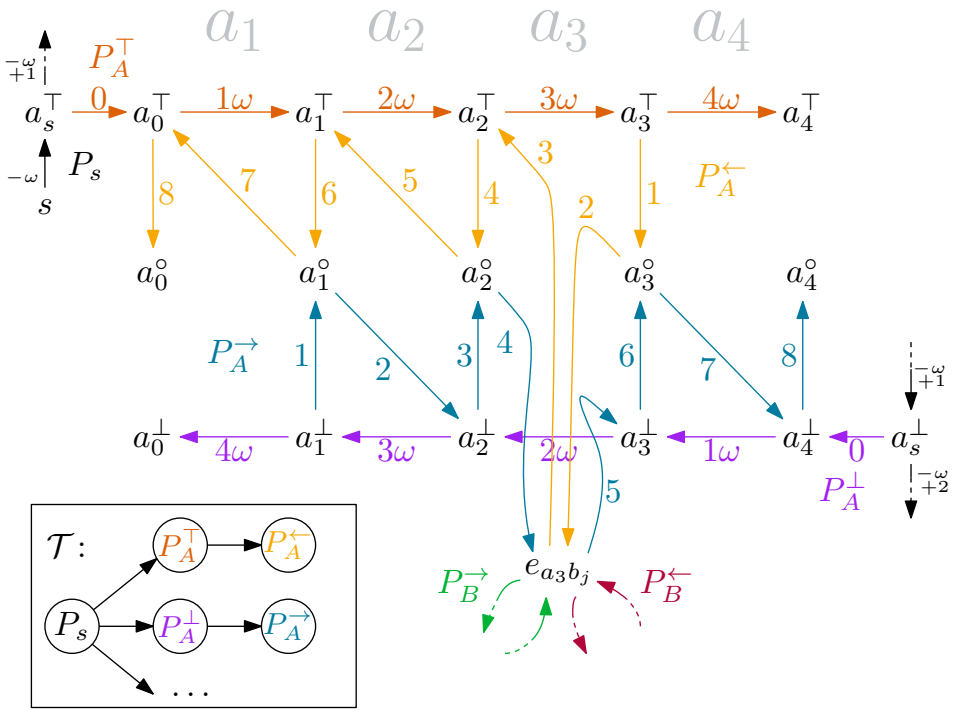

Figure 8 The color-gadget for color A and the representation of an edge between $a_3 \in A$ and some $b_j \in B$. Below is the SPT $\mathcal{T}$ for which we are to optimize the reachability.

5.2 Positive Results

In this section, we present our positive results. Specifically, we give two XP algorithms for the three modification variants, parameterized by b and by k, respectively, and two FPT algorithms parameterized by $b+k$: a faster and simpler algorithm for delay-only modifications, and a second, more involved algorithm for all three modification variants.

When parameterized by the budget b, an XP algorithm follows readily: for each unit of the budget, one guesses which edge to modify with it, yielding a bounded enumeration.

▶ **Theorem 18.** *MR-D/A/SP is in* XP *when parameterized by b, and can be solved in $\mathcal{O}(|\mathcal{G}|^b)$ time.*

When parameterized by the number of base-paths k, the algorithm is more involved. We enumerate candidate SVSs and, for each, check whether its switches can all be made temporal within budget using an integer linear program (ILP). By Lemma 8 we only need to consider operations applied to the edges referenced by a switch. The key ingredient is a compact description of how delays and advances propagate along each base-path. This relies heavily on existing waiting times:

▶ **Definition 19** (slack). *We denote the waiting time between two consecutive edges (e_i, t_i) and (e_{i+1}, t_{i+1}) on a base-path by* $\mathsf{slack}((e_i, t_i), (e_{i+1}, t_{i+1})) := t_{i+1} - t_i - 1$.

More generally, for two vertices $u \leq_P v$ on a base-path P, the cumulative slack between them is the total waiting time along P from u to v:

$$\mathsf{slack}(u, v) := \sum_{\substack{(e_i,t_i),\,(e_{i+1},t_{i+1}) \\ \text{consecutive on } P[u:v]}} \mathsf{slack}\big((e_i, t_i), (e_{i+1}, t_{i+1})\big).$$

While a shifting operation propagates forwards or backwards along a base-path, the propagation amount is effectively absorbed by slack, or passes though completely if there is no slack. The propagating amount applied to some edge through an operation on another edge of the same base-path thus only depends on the slack between them. Precomputing this slack allows us to abstract away all non-switch vertices and describe all propagating effects in an ILP of a size only dependent on k.

▶ **Theorem 20.** *MR-D/A/SP is in* XP *when parameterized by k, and can be solved in $\mathcal{O}(|V|^k \cdot (\log k)^{\mathcal{O}(k)} \cdot \mathtt{poly}(|\mathcal{G}|))$ time.*

We have seen earlier that under any operation, our problem is W[2]-hard when parameterized by the budget b (Theorem 15), and that for delaying, it is W[1]-hard when parameterized by the number of base-paths k (Theorem 16). In the following, we show that combining both parameters yields FPT algorithms for all three modification variants. Note that an FPT run-time by $b + k$ only allows for guessing the SPT, but not the SVS as in the XP algorithm for parameter k above.

We present two algorithms: a simpler and faster one for delaying only (Theorem 21), and a more complicated one that works for all three modification types (Theorem 22). Both follow the same two-phase structure:

Enumeration phase. Guess the SPT $\mathcal{T}$ and the budget-related values around the switch onto each base-path, that includes particularly the delay or advance that is to be applied.

Greedy phase. Given the guessed values, compute the optimal SVS by processing $\mathcal{T}$ root-to-leaves and assigning each path the earliest *valid* switch vertex.

For *only delaying*, it suffices to guess $\mathcal{T}$ and a single delay amount $\delta_P \in [b]$ per non-source base-path P (yielding $\mathcal{O}(k^k)$ and $\mathcal{O}(b^{k-1})$ choices, respectively). The greedy phase then exploits that delaying an edge propagates only *forward* on that base-path. Consequently, the *earliest* switch-vertex onto the base-path P simultaneously maximises the reachable suffix on P and minimises the propagated delay impacting later switches on $\mathcal{T}$. Both objectives point in the same direction (away from the source-path), so choosing the earliest switch that fits the guessed delay is optimal. The resulting running time is $\mathcal{O}\big(b^k \cdot k^k \cdot |\mathcal{G}|^{\mathcal{O}(1)}\big)$.

▶ **Theorem 21.** *MR-DP is in* FPT *when parameterized by $b + k$, and can be solved in $\mathcal{O}\big(b^k \cdot k^k \cdot |\mathcal{G}|^{\mathcal{O}(1)}\big)$ time.*

The previous algorithm relies on the fact that the earliest viable switch along a path both maximizes the reachability of s and minimizes the additional cost resulting from propagation. This simple argument fails when *advances* are allowed, because an advance propagates *backward* along the base-path P and the advances of multiple switches off P can interact in multiple ways. As a result, the two objectives no longer align: whether a given switch vertex is valid may depend on advances caused by switches that are assigned only later, creating a circular dependency.

We resolve this by refining the idea behind the ILP in Theorem 20 and guessing more values witnessed by an optimal solution: We guess for each non-root base-path Q six values capturing the delay and advance propagations at its switch vertex on the parent path, the active delay applied there, the advance arriving from Q's children, and the label difference of the two incident temporal edges. We additionally guess a child-ordering σ for each base-path. Once these values are fixed, the guessed propagations impose *slack-distance conditions* between consecutive switch vertices on each parent path: either an exact condition when propagation is expected from another switch vertex (fixing both a lower and upper bound on the slack, forcing the switch vertex onto a short zero-slack subpath) or a minimum condition

when no propagation is expected (lower bound only). Maximal runs of siblings linked by exact conditions form *batches* whose relative positions are determined once the first member is placed. We then run a greedy construction root-to-leaves, placing each batch-head at the earliest position where 1) the minimum distance condition from the previous switch is satisfied and 2) the rest of the batch can be aligned from there such that slack-distance-conditions are satisfied. An exchange argument shows the first such match never destroys the existence of an optimal realization consistent with the guessed values. This argument utilizes an observation that all possible switches between two base-paths for which the incident edges have a specific label difference must have the same order along both base-paths; hence earlier switches along Q are also earlier along the base-path that we want to switch to. Putting this all together, we get an FPT algorithm with running time $\mathcal{O}\big(b^{6k} \cdot k^{2k} \cdot |\mathcal{G}|^{\mathcal{O}(1)}\big)$.

▶ **Theorem 22.** *MR-D/A/SP is in* FPT *when parameterized by* $b + k$*, and can be solved in* $\mathcal{O}\big(b^{6k} \cdot k^{2k} \cdot |\mathcal{G}|^{\mathcal{O}(1)}\big)$ *time.*

6 Discussion

The most obvious open question of our work is to settle the complexity of MR-SP and MR-AP when parameterized by k. Is the problem fixed-parameter-tractable, or is it W[1]-hard? An algorithm that utilizes switch-path trees would need to be able to discard hard SPT-choices as sub-optimal, or use a completely different strategy. Towards a hardness result, a novel reduction idea would be needed. A different direction would be to consider the problem with several sources. In this case, one can define several different objectives to optimize, obvious examples being maximization of the union of reachable vertices from all sources and the maximization of the minimum number of reachable vertices from any source. In the former case, it is easy to observe that we can add an auxiliary source s^* that is connected via one base-path to all sources and thus reduce the problem to MR-SP.

From a broader perspective, our results can be seen as an extension of the recent positive results on temporal k-path graphs [8], this time for the problem of maximum reachability. Thus, they serve as an indication that there exist "natural" classes of temporal graphs that can enable fixed-parameter tractability. This contrasts the vast of the literature on temporal graphs where "everything becomes immediately hard". Are there other natural classes that admit fixed parameter algorithms? Is there a subclass of temporal k-path graphs that lets us overcome the current intractability results for MR-SP and other problems? Answering these questions, even partially, can provide fertile ground for future research.

References

1 Thomas Bellitto, Jules Bouton Popper, and Bruno Escoffier. Temporal Connectivity Augmentation. In Kitty Meeks and Christian Scheideler, editors, *4th Symposium on Algorithmic Foundations of Dynamic Networks (SAND 2025)*, volume 330 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 3:1–3:16, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SAND.2025.3.

2 Filippo Brunelli, Pierluigi Crescenzi, and Laurent Viennot. Maximizing Reachability in a Temporal Graph Obtained by Assigning Starting Times to a Collection of Walks. *Networks*, 81(2):177–203, March 2023. doi:10.1002/net.22123.

3 Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Daniel Marx, Marcin Pilipczuk, and Michał Pilipczuk. *Parameterized Algorithms*, volume 5. Springer, a edition, 2015.

4 Argyrios Deligkas, Michelle Döring, Eduard Eiben, Tiger-Lily Goldsmith, George Skretas, and Georg Tennigkeit. How Many Lines to Paint the City: Exact Edge-Cover in Temporal Graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 26498–26506, April 2025. doi:10.1609/aaai.v39i25.34850.

5 Argyrios Deligkas, Michelle Döring, Eduard Eiben, Tiger-Lily Goldsmith, George Skretas, and Georg Tennigkeit. Parameterized Complexity of Temporal Connected Components: Treewidth and Temporal Path Number, October 2025. doi:10.48550/arXiv.2510.05806.

6 Argyrios Deligkas, Eduard Eiben, and George Skretas. Minimizing Reachability Times on Temporal Graphs via Shifting Labels. In *Thirty-Second International Joint Conference on Artificial Intelligence*, volume 5, pages 5333–5340, August 2023. doi:10.24963/ijcai.2023/592.

7 Argyrios Deligkas and Igor Potapov. Optimizing Reachability Sets in Temporal Graphs by Delaying. *Information and Computation*, 285:104890, May 2022. doi:10.1016/j.ic.2022.104890.

8 Michelle Döring. Temporal Connected Components. https://www.notion.so/Temporal-Connected-Components-057cae72f648434e96fbde5705a544b4, September 2025.

9 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer London, London, 2013.

10 Jessica Enright, Laura Larios-Jones, Kitty Meeks, and William Pettersson. Reachability in Temporal Graphs Under Perturbation. In Rastislav Královič and Věra Kůrková, editors, *SOFSEM 2025: Theory and Practice of Computer Science*, pages 255–269, Cham, 2025. Springer Nature Switzerland. doi:10.1007/978-3-031-82670-2_19.

11 Jessica Enright, Kitty Meeks, George B. Mertzios, and Viktor Zamaraev. Deleting Edges to Restrict the Size of an Epidemic in Temporal Networks. *Journal of Computer and System Sciences*, 119:60–77, August 2021. doi:10.1016/j.jcss.2021.01.007.

12 Jessica Enright, Kitty Meeks, and Fiona Skerman. Assigning Times to Minimise Reachability in Temporal Graphs. *Journal of Computer and System Sciences*, 115:169–186, February 2021. doi:10.1016/j.jcss.2020.08.001.

13 Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization: Theory of Parameterized Preprocessing*. Cambridge University Press, January 2019.

14 David C. Kutner and Laura Larios-Jones. Temporal Reachability Dominating Sets: Contagion in Temporal Graphs. *Journal of Computer and System Sciences*, 155:101–116, 2023. doi:10.1007/978-3-031-48882-5_8.

15 David C. Kutner and Anouk Sommer. Better Late, Then? The Hardness of Choosing Delays to Meet Passenger Demands in Temporal Graphs. *4th Symposium on Algorithmic Foundations of Dynamic Networks (SAND 2025)*, 330:7:1–7:18, June 2025. doi:10.4230/LIPIcs.SAND.2025.7.

16 Hendrik Molter, Malte Renken, and Philipp Zschoche. Temporal Reachability Minimization: Delaying vs. Deleting. *Journal of Computer and System Sciences*, 144:103549, 2024. doi:10.1016/j.jcss.2024.103549.