On the Enumeration of Bicriteria Temporal Paths

Petra Mutzel and Lutz Oettershagen[✉]

Department of Computer Science, TU Dortmund University, Dortmund, Germany
{petra.mutzel,lutz.oettershagen}@tu-dortmund.de

Abstract. We discuss the complexity of path enumeration in weighted temporal graphs. In a weighted temporal graph, each edge has an availability time, a traversal time and some cost. We introduce two bicriteria temporal min-cost path problems in which we are interested in the set of all efficient paths with low costs and short duration or early arrival times, respectively. Unfortunately, the number of efficient paths can be exponential in the size of the input. For the case of strictly positive edge costs, however, we are able to provide algorithms that enumerate the set of efficient paths with polynomial time delay and polynomial space. If we are only interested in the set of Pareto-optimal solutions (not in the paths themselves), then we show that in the case of nonnegative edge costs these sets can be found in polynomial time. In addition, for each Pareto-optimal solution, we are able to find an efficient path in polynomial time.

1   Introduction

A weighted temporal graph $G = (V, E)$ consists of a set of vertices and a set of temporal edges. Each temporal edge $e \in E$ is associated with an edge cost and is only available (for departure) at a specific integral point in time. Traversing an edge takes a specified amount of traversal time. We can imagine a temporal graph as a sequence of $T \in \mathbb{N}$ graphs $G_1, G_2, \ldots, G_T$ sharing the common set of vertices V; each graph G_i has its own set of edges E_i. Therefore, G can change its structure over a finite sequence of integral time steps.

Given a directed weighted temporal graph $G = (V, E)$, a source $s \in V$ and a target z[1] $\in V$, we want to find *earliest arrival* or *fastest* (s, z)-paths with *minimal costs*. A motivation can be found in typical queries in (public) transportation networks. Here, each vertex represents a bus stop, metro stop or a transfer point and each edge a connection between two such points. In this model, the availability time of an edge is the departure time of the bus or metro, the traversal time is the time a vehicle takes between the two transfer points, and the edge cost provides the ticket price. Two natural questions are the following: (1) Minimize the costs and the arrival times, or (2) minimize the costs and the total travel time.

---

[1] We use t to denote *time steps* and z for the target vertex.

In general, there is no path that minimizes both objectives simultaneously, and therefore we are interested in the set of all efficient paths. A path is called *efficient* if there is no other path that is strictly better in one of the criteria and at least as good concerning both criteria. In other words, a path is efficient iff its cost vector is Pareto-optimal.

We denote by MCFENUM and MCEAENUM the enumeration problems, in which the task is to enumerate the set of all efficient paths w.r.t. cost and duration or cost and arrival time, respectively. Unfortunately, there can be an exponential number of efficient paths. So we cannot expect to find polynomial time algorithms for the above mentioned enumeration problems. However, Johnson, Yannakakis, and Papadimitriou [7] have defined complexity classes for enumeration problems, where the time complexity is expressed not only in terms of the input size but also in the output size. We use the output complexity model to analyze the proposed enumeration problems and show that the problems belong to the class of polynomial time delay with polynomial space (**PSDelayP**). If we are only interested in the sets of Pareto-optimal solutions and not in all the paths themselves, we show that then the problems can be solved in polynomial time for nonnegative edge weights. In these cases we can also provide an associated path with each solution.

Contribution – In this paper we show the following:

1. MCFENUM and MCEAENUM are in **PSDelayP** for weighted temporal graphs with strictly positive edge costs.
2. In case of nonnegative edge costs, finding the Pareto-optimal set of cost vectors is possible in polynomial time (thus the set of Pareto-optimal solutions is polynomially bounded in the size of the input), and for each Pareto-optimal solution we can find an efficient path in polynomial time.
3. The decision versions that ask if a path is efficient, are in **P**. Deciding if there exists an efficient path with given cost and duration or arrival time is possible in polynomial time.

In the remainder of this section we discuss the related work. In Sect. 2 we provide all necessary preliminaries. Next, in Sect. 3 structural results are presented. These are the foundation of the algorithms for MCFENUM and MCEAENUM in the following Sects. 4 and 5, respectively. Finally, in Sect. 6 conclusions are drawn.

Related Work – Temporal graphs and related problems are discussed in several recent works. A general overview is provided, e.g., in [1,6,8]. Xuan et al. [12] discuss communication in dynamic and unstable networks. They introduce algorithms for finding a fastest path, an earliest arrival path and a path that uses the least number of edges. Wu et al. [11] further discuss the fastest, shortest and earliest arrival path problems in temporal graphs and introduce the latest-departure path problem. Their algorithm for calculating an earliest arrival (s, v)-path has time complexity $\mathcal{O}(n + m)$ and space complexity $\mathcal{O}(n)$, where n denotes the number of vertices and m the number of edges in the given temporal graph. Furthermore, they present an algorithm that finds a fastest (s, v)-path in

$\mathcal{O}(n + m \log c)$ time and $\mathcal{O}(\min\{n \cdot \mathcal{S}, n + m\})$ space. Here, $\mathcal{S}$ is the number of distinct availability times of edges leaving vertex s, and c the minimum of $\mathcal{S}$ and the maximal in-degree over all vertices of G. The algorithm uses a label setting approach to find a fastest (s, v)-path for each $v \in V$. However, Xuan et al. [12] and Wu et al. [11] do not consider weighted temporal graphs.

Hansen [5] introduces bicriteria path problems in static graphs, and provides an example for a family of graphs for which the number of efficient paths grows exponentially with the number of vertices. Meggido shows that deciding if there is an (s, z)-path that respects an upper bound on both objective functions is **NP**-complete (Meggido 1977, private communication with Garey and Johnson [3]). Martins [9] presents a label setting algorithm based on the well known Dijkstra algorithm for the bicriteria shortest path problem, that finds the set of all efficient (s, v)-paths for all $v \in V$. Ehrgott and Gandibleux [2] provide an overview of the work on bi- and multicriteria shortest path problems. Hamacher et al. [4] propose an algorithm for the bicriteria time-dependent shortest path problem in networks in which edges have time dependent costs and traversal times. The traversal time of an edge is given as a function of the time upon entering the edge. Moreover, each edge has a two-dimensional time dependent cost vector. Waiting at a vertex may be penalized by additional bicriteria time dependent costs. They propose a label setting algorithm that starts from the target vertex and finds the set of all efficient paths to each possible start vertex.

We are not aware of any work discussing the enumeration of efficient paths in weighted temporal graphs.

2 Preliminaries

A weighted temporal graph $G = (V, E)$ consists of a set V of $n \in \mathbb{N}$ vertices and a set E of $m \in \mathbb{N}$ weighted and directed temporal edges. A weighted and directed temporal edge $e = (u, v, t, \lambda, c) \in E$ consists of the starting vertex $u \in V$, the end vertex $v \in V$ (with $u \neq v$), availability time $t \in \mathbb{N}$, traversal time $\lambda \in \mathbb{N}$ and cost $c \in \mathbb{R}_{\geq 0}$. Each edge $e = (u, v, t, \lambda, c) \in E$ is only available for entering at its availability time t and traversing e takes λ time. For $T := \max\{t \mid (u, v, t, \lambda, c) \in E\}$, we can view G as a finite sequence $G_1, G_2, \ldots, G_T$ of static graphs over the common set of vertices V; each G_i with its own set of edges $E_i := \{(u, v, \lambda, c) \mid e = (u, v, i, \lambda, c) \in E\}$. Figure 1 shows an example. We denote the set of incoming (outgoing) temporal edges of a vertex $v \in V$ by $\delta^-(v)$ ($\delta^+(v)$). Note that in general for temporal graphs the number of edges is not bounded by a function in the number of vertices, i.e. we may have arbitrarily more temporal edges than vertices. In the following, we use a *stream representation* of temporal graphs. A temporal graph is given as a sequence of the m edges, which is ordered by the availability time of the edges in increasing order with ties being broken arbitrarily.

2.1   Temporal Path Problems

A *temporal* (u, v)-*walk* $P_{u,v}$ is a sequence $(e_1, \ldots, e_i = (v_i, v_{i+1}, t_i, \lambda_i, c_i), \ldots, e_k)$ of edges with $e_i \in E$ for $1 \leq i \leq k$, and with $v_1 = u$, $v_{k+1} = v$ and $t_i + \lambda_i \leq t_{i+1}$ for $1 \leq i < k$. If a temporal (u, v)-walk visits each $v \in V$ at most once, it is *simple* and we call it (u, v)-*path*. We denote by $s(P_{u,v}) := t_1$ the *starting time*, and by $a(P_{u,v}) := t_k + \lambda_k$ the *arrival time* of $P_{u,v}$. Furthermore, we define the *duration* as $d(P_{u,v}) := a(P_{u,v}) - s(P_{u,v})$. A path $P_{u,v}$ is *faster* than a path $Q_{u,v}$ if $d(P_{u,v}) < d(Q_{u,v})$. The *cost* of a path $P = (e_1, \ldots, e_k)$ is the sum of the edge cost, i.e. $c(P) := \sum_{i=1}^{k} c_i$. Finally, for a path $P = (e_1, \ldots, e_i, \ldots, e_k)$, we call $(e_1, \ldots, e_i)$ *prefix-path* and $(e_i, \ldots, e_k)$ *suffix-path* of P. In Fig. 1 the (s, z)-path $((s, b, 1, 1, 2), (b, z, 2, 1, 1))$ has arrival time 3, duration 2 and cost 3. The (s, z)-path $((s, z, 3, 1, 3))$ also has cost 3, but it has a later arrival time of 4 and is faster with a duration of only 1.
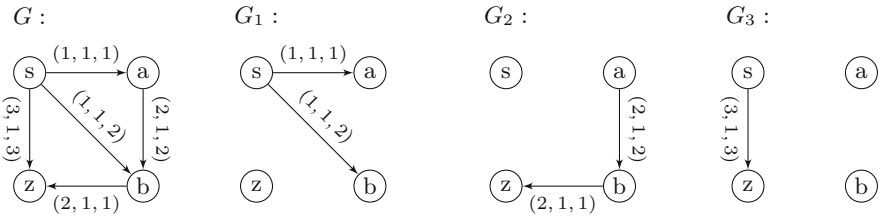

Fig. 1. Example for a weighted temporal graph G. Each edge label (t, λ, c) describes the time t when the edge is available, its traversal time λ and its cost c. For each time step $t \in \{1, 2, 3\}$, layer G_t is shown.

For the discussion of bicriteria path problems, we use the following definitions. Let $\mathcal{X}$ be the set of all feasible (s, z)-paths, and let $f(P)$ be the temporal value of P, i.e. either arrival time $f(P) := a(P)$ or duration $f(P) := d(P)$. We call a path $P \in \mathcal{X}$ *efficient* if there is no other path $Q \in \mathcal{X}$ with $c(Q) < c(P)$ and $f(Q) \leq f(P)$ or $c(Q) \leq c(P)$ and $f(Q) < f(P)$. We map each $P \in \mathcal{X}$ to a vector $(f(P), c(P))$ in the two-dimensional objective space which we denote by $\mathcal{Y}$. Complementary to efficiency in the decision space, we have the concept of *domination* in the objective space. We say $(f(P), c(P)) \in \mathcal{Y}$ *dominates* $(f(Q), c(Q)) \in \mathcal{Y}$ if either $c(P) < c(Q)$ and $f(P) \leq f(Q)$ or $c(P) \leq c(Q)$ and $f(P) < f(Q)$. We call $(f(P), c(P))$ *nondominated* if and only if P is efficient. We define the bicriteria enumeration problems MCFENUM and MCEAENUM as follows.

MIN-COST FASTEST PATHS ENUMERATION PROBLEM (MCFENUM)
Given: A weighted temporal graph $G = (V, E)$ and $s, z \in V$.
Task: Enumerate all and only (s, z)-paths that are efficient w.r.t. duration and costs.
MIN-COST EARLIEST ARRIVAL PATHS ENUMERATION PROBLEM (MCEAENUM)
Given: A weighted temporal graph $G = (V, E)$ and $s, z \in V$.

Task: Enumerate all and only (s, z)-paths that are efficient w.r.t. arrival time and costs.

We denote by MCF and MCEA the optimization versions, in which the task is to find a single efficient (s, z)-path.

2.2   Complexity Classes for Enumeration Problems

Bi- and multicriteria optimization problems are often not easily comparable using the traditional notion of worst-case complexity, due to their potentially exponential number of efficient solutions. We use the *output complexity* model as proposed by Johnson, Yannakakis, and Papadimitriou [7]. Here, the time complexity is stated as a function in the size of the input and the output.

Definition 1. *Let $\mathcal{E}$ be an enumeration problem. Then $\mathcal{E}$ is in*

1. **DelayP** (Polynomial Time Delay) *if the time delay until the output of the first and between the output of any two consecutive solutions is bounded by a polynomial in the input size.*
2. **PSDelayP** (Polynomial Time Delay with Polynomial Space) *if $\mathcal{E}$ is in* **DelayP** *and the used space is also bounded by a polynomial in the input size.*

In Sects. 4 and 5 we show that MCFENUM and MCEAENUM are both in **PSDelayP** if the input graph has strictly positive edge costs. We provide algorithms that enumerate the set of all efficient paths in polynomial time delay and use space bounded by a polynomial in the input size.

3   Structural Results

We show that it is possible to find an efficient (s, z)-path for the Min-Cost Earliest Arrival Path Problem (MCEA) in a graph G, if we are able to solve MCF. We use a transformed graph G', in which a new source vertex and a single edge is added. The reduction is from a search problem to another search problem. We show that it preserves the existence of solutions and we also provide a mapping between the solutions. This is also known as *Levin* reduction.

Lemma 1. *There is a Levin reduction from MCEA to MCF.*

Proof. Let $I = (G = (V, E), s, z)$ be an instance of MCEA. We construct the MCF instance $I' = (G' = (V \cup \{s'\}, E \cup \{e_0 = (s', s, 0, 0, 0)\}), s', z)$. Furthermore, let $\mathcal{X}_I$ be the sets of all (s, z)-paths for I, and $\mathcal{X}_{I'}$ be the sets of all (s', z)-paths for I'. We define $g : \mathcal{X}_I \to \mathcal{X}_{I'}$ as bijection that prepends edge e_0 to the paths in $\mathcal{X}_I$, i.e. $g((e_1, \ldots, e_k)) = (e_0, e_1, \ldots, e_k)$. We show that $P \in \mathcal{X}_I$ is efficient (for MCEA) iff $g(P) \in \mathcal{X}_{I'}$ is efficient (for MCF).

Let $P = (e_1, \ldots, e_k)$ be an efficient (s, z)-path in G w.r.t. costs and arrival time. Then $Q := g(P) = (e_0, e_1, \ldots, e_k)$ is an (s', z)-path in G' with $a(Q) = a(P)$ and $c(Q) = c(P)$. Now, assume Q is not efficient w.r.t. costs and duration in G'.

Then there is a path Q' with less costs and at most the duration of Q or with shorter duration and at most the same costs of Q. Path Q' also begins with edge e_0, and G contains a path P' that uses the same edges as Q' with exception of edge e_0. Then, at least one of the following two cases holds.

- Case $c(Q') \leq c(Q)$ and $d(Q') < d(Q)$: Since the costs of e_0 are 0, it follows that $c(P') = c(Q') \leq c(Q) = c(P)$. Because the paths start at time 0 and for each path $d(P) = a(P) - s(P)$, it follows $d(Q') = a(Q') = a(P') < a(P) = a(Q) = d(Q)$.
- Case $c(Q') < c(Q)$ and $d(Q') \leq d(Q)$: Analogously, here we have $c(P') = c(Q') < c(Q) = c(P)$. And $a(P') = a(Q') \leq a(Q) = a(P)$.

Either of these two cases leads to a contradiction to the assumption that P is efficient.

Let $Q = (e_0, e_1, \ldots, e_k)$ and assume it is an efficient (s', z)-path in G' w.r.t. to cost and duration. Then there exists an (s, z)-path $P = (e_1, \ldots, e_k)$ in G such that $g(P) = Q$, $a(Q) = a(P)$ and $c(Q) = c(P)$. Now, assume that P is not efficient. Then there is a path P' with less costs and not later arrival time than P or with earlier arrival time and at most the costs of P. In G' exists the path $Q' = g(P')$ that uses the same edges as P', and additionally the edge e_0 as prefix-path from s' to s. We have the cases $c(P') < c(P)$ and $a(P') \leq a(P)$ or $c(P') \leq c(P)$ and $a(P') < a(P)$. Again, either of them leads to a contradiction to the assumption that Q is efficient. □

Based on this result, we first present an algorithm for MCFENUM in Sect. 4 that we use in a modified version to solve MCEAENUM in Sect. 5. In the rest of this section, we focus on graphs with strictly positive edge costs.

Observation 1. *Let $G = (V, E)$ be a weighted temporal graph. If for all edges $e = (u, v, t, \lambda, c) \in E$ it holds that $c > 0$, then all efficient walks for MCEAENUM and MCFENUM are simple, i.e. are paths.*

Similar to the non-temporal static case, it would be possible to delete the edges of a cycle contained in the non-simple walk. We denote the two special cases for graphs with strictly positive edge costs by $(c{>}0)$-MCFENUM and $(c{>}0)$-MCEAENUM. Our enumeration algorithms use a label setting technique. A label $l = (b, a, c, p, v, r, \Pi)$ at vertex $v \in V$ corresponds to a (s, v)-path and consists of

- the starting time $b = s(P_{s,v})$,
- the arrival time $a = a(P_{s,v})$,
- the cost $c = c(P_{s,v})$,
- the predecessor label p,
- the current vertex v,
- the availability time r of the previous edge and
- a reference to a list of equivalent labels Π.

Moreover, each label is uniquely identifiable by an additional identifier and has a reference to the edge that lead to its creation (denoted by $l.edge$). The proposed algorithms process the edges in order of their availability time. When processing an edge $e = (u, v, t, \lambda, c_e)$, all paths that end at vertex u can be extended by pushing labels over edge e to vertex v. Pushing a label $l = (b, a, c, p, u, r, \Pi)$ over e means that we create a new label $l_{new} = (b, t + \lambda, c + c_e, l, v, t, \cdot)$ at vertex v.

If we would create and store a label for each efficient path, we possibly would need exponential space in the size of the input. The reason is that the number of efficient paths can be exponential in the size of the input.
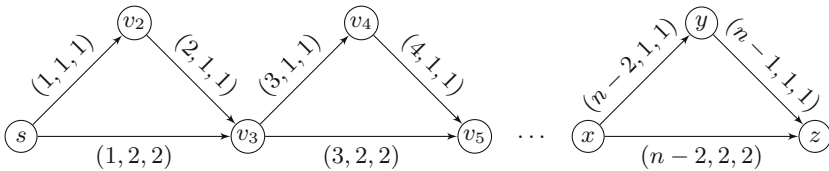

Fig. 2. Example for an exponential number of efficient paths for MCEAENUM and MCFENUM. All (s, v)-paths for $v \in V$ are efficient.

Figure 2 shows an example for MCFENUM and MCEAENUM, which is similar to the one provided by Hansen [5], but adapted to the weighted temporal case. G has m edges and $n = \frac{2m}{3} + 1$ vertices. There are two paths from s to v_3, four paths from s to v_5, eight paths from s to v_7 and so on. All (s, v)-paths for $v \in V$ are efficient. In total, there are $2^{\lfloor \frac{n}{2} \rfloor}$ efficient (s, z)-paths to be enumerated. However, the following lemma shows properties of the problems that help us to achieve polynomial time delay and a linear or polynomial space complexity. Let $\mathcal{Y}_A$ ($\mathcal{Y}_F$) denote the objective space for MCEAENUM (MCFENUM, respectively). Moreover, we define $\mathcal{S}$ to be the number of distinct availability times of edges leaving the source vertex s.

Lemma 2. *For* MCEAENUM, *the number of nondominated points in* $\mathcal{Y}_A$ *is in* $\mathcal{O}(m)$. *For* MCFENUM, *the number of nondominated points in* $\mathcal{Y}_F$ *is in* $\mathcal{O}(m^2)$.

Proof. Let $G = (V, E)$ be a weighted temporal graph and $s, z \in V$. First we show that the statement holds for MCEAENUM. The possibilities for different arrival times at vertex z is limited by the number of incoming edges at z. For each path $P_{s,z}$ we have

$$a(P_{s,z}) \in \{\alpha \mid \alpha := t_e + \lambda_e \text{ with } e = (u, z, t_e, \lambda_e, c) \in \delta^-(z)\}.$$

Consequently, there are at most $|\delta^-(z)| \in \mathcal{O}(m)$ different arrival times. For each arrival time a, there can only be one nondominated point $(a, c) \in \mathcal{Y}_A$ that has the minimum costs of c, and which represents exactly all efficient paths with arrival time a and costs c.

Now consider the case for MCFENUM. The number of distinct availability times of edges leaving the source vertex $\mathcal{S}$ is bounded by $|\delta^+(s)| \in \mathcal{O}(m)$. Because the duration of any (s, z)-path P equals $a(P) - s(P)$, there are at most $\mathcal{S} \cdot |\delta^-(z)| \in \mathcal{O}(m^2)$ different durations possible at vertex z. For each duration, there can only be one nondominated point $(d, c) \in \mathcal{Y}_F$ that has minimum costs c.   □

Note that for general bicriteria optimization (path) problems there can be an exponential number of nondominated points in the objective space. Skriver and Andersen [10] give an example for a family of graphs with an exponential number of nondominated points for a bicriteria path problem. The fact that in our case the number of nondominated points in the objective space is polynomially bounded, allows us to achieve polynomial time delay and space complexity for our algorithms. The idea is to consider equivalence classes of labels at each vertex, such that we only have to proceed with a single representative for each class. First, we define the following relations between labels.

Definition 2. *Let* $l_1 = (b_1, a_1, c_1, p_1, v, r_1, \Pi_1)$ *and* $l_2 = (b_2, a_2, c_2, p_2, v, r_2, \Pi_2)$ *be two labels at vertex* v.

1. *Label* l_1 *is* equivalent *to* l_2 *iff* $c_1 = c_2$ *and* $b_1 = b_2$.
2. *Label* l_1 predominates l_2 *if* l_1 *and* l_2 *are not equivalent,* $b_1 \geq b_2$, $a_1 \leq a_2$ *and* $c_1 \leq c_2$ *with at least one of the inequalities being strict.*
3. *Finally, label* l_1 dominates l_2 *if* $a_1 - b_1 \leq a_2 - b_2$ *and* $c_1 \leq c_2$ *with at least one of the inequalities being strict.*

For each class of equivalent labels, we have a representative l and a list Π_l that contains all equivalent labels to l. For each vertex $v \in V$, we have a set R_v that contains all representatives. The algorithms consist of two consecutive phases:

– *Phase 1* calculates the set of non-equivalent representatives R_v for every vertex $v \in V$ such that every label in R_v represents a set of equivalent paths from s to v. For each of the nonequivalent labels $l \in R_v$ we store the list Π_l that contains all labels equivalent to l.
– *Phase 2* recombines the sets of equivalent labels in a backtracking fashion, such that we are able to enumerate exactly all efficient (s, z)-paths without holding the paths in memory.

A label in Π_l at vertex v represents all (s, v)-paths that are extension of all paths represented by its predecessor, and $l \in R_v$ is a representative for all labels in Π_l. The representative itself is in Π_l and has minimum arrival time among all labels in Π_l.

We have to take into account that a prefix-path $P_{s,w}$ of an efficient (s, z)-path may not be an efficient (s, w)-path. Figure 3(a) shows an example for a weighted temporal graph with a non-optimal prefix-path. Consider the following paths:
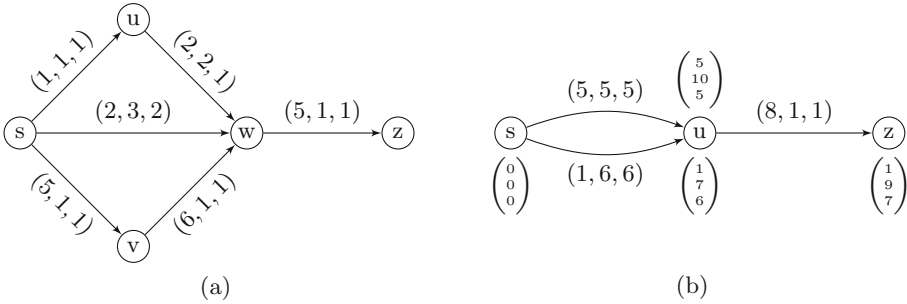
Fig. 3. (a) An example for non-efficient prefix-paths. (b) The vertices are annotated with labels that describe the starting time, arrival time and costs of the paths starting at s.

- $P_{s,z} = ((s, w, 2, 3, 2), (w, z, 5, 1, 1))$ with arrival time 6 and duration 4
- $P_{s,w}^1 = ((s, w, 2, 3, 2))$ with arrival time 5 and duration 3
- $P_{s,w}^2 = ((s, u, 1, 2, 1), (u, w, 2, 1, 1))$ with arrival time 3 and duration 3
- $P_{s,w}^3 = ((s, v, 5, 1, 1), (v, w, 6, 1, 1))$ with arrival time 7 and duration 2

All (s, z)-paths have cost 3 and all (s, w)-paths have cost 2. Path $P_{s,z}$ is efficient for MCFENUM and MCEAENUM. For MCEAENUM the prefix-path $P_{s,w}^1$ is not efficient, because $P_{s,w}^2$ arrives earlier. However, for MCFENUM the only efficient (s, w)-path is $P_{s,w}^3$. Consequently, we cannot discard a non-efficient path that possibly is a prefix-path of an efficient path. We use the predomination relation to remove labels that do not represent a prefix-path of an efficient path.

Lemma 3. *Let $l_1 = (b_1, a_1, c_1, p_1, v, r_1, \Pi_1)$ and $l_2 = (b_2, a_2, c_2, p_2, v, r_2, \Pi_2)$ be two distinct labels at vertex $v \in V$. If l_1 predominates l_2, then l_2 cannot be a label representing a prefix-path of any efficient path.*

Proof. There are two distinct paths $P_{s,v}^1$ and $P_{s,v}^2$ from s to v corresponding to l_1 and l_2. Due to the predomination of l_1 over l_2 it follows that $a_1 = a(P_{s,v}^1) \leq a(P_{s,v}^2) = a_2$, $b_1 = s(P_{s,v}^1) \geq s(P_{s,v}^2) = b_2$ and $c_1 = c(P_{s,v}^1) \leq c(P_{s,v}^2) = c_2$ with at least one of the later two relations being strict, due to the fact that the labels are not equivalent. Let $P_{s,w}$ be a path from s to some $w \in V$ such that $P_{s,v}^2$ is prefix-path of $P_{s,w}$, and assume that $P_{s,w}$ is efficient. Let $P'_{s,w}$ be the path where the prefix-path $P_{s,v}^2$ is replaced by $P_{s,v}^1$. This is possible because $a(P_{s,v}^1) \leq a(P_{s,v}^2)$. Now, since $s(P_{s,v}^1) \geq s(P_{s,v}^2)$ and $c(P_{s,v}^1) \leq c(P_{s,v}^2)$ with at least one of the inequalities being strict, it follows that $a(P'_{s,w}) - s(P'_{s,w}) \leq a(P_{s,w}) - s(P_{s,w})$ and $c(P'_{s,w}) \leq c(P_{s,w})$ also with one of the inequalities being strict. Therefore, $P'_{s,w}$ dominates $P_{s,w}$, a contradiction to the assumption that $P_{s,w}$ is efficient. □

Figure 3(b) shows an example for a non-predominated label of a prefix-path that we cannot discard. Although path $P_1 = ((s, u, 5, 5, 5))$ dominates path $P_2 = ((s, u, 1, 6, 6))$, we cannot discard P_2. The reason is that the arrival time of P_1 is later than the availability time of the only edge from u to z. Therefore, P_2 is the prefix-path of the only efficient path $((s, u, 1, 6, 6), (u, z, 8, 1, 1))$.

4   Min-Cost Fastest Path Enumeration Problem

In this section, we present the algorithm for MCFENUM. Algorithm 1 expects as input a weighted temporal graph with strictly positive edge costs in the edge stream representation, the source vertex $s \in V$ and the target vertex $z \in V$. First, we insert an initial label l_{init} into R_s and $\Pi_{l_{init}}$. Next, the algorithm processes successively the m edges in order of their availability time. For each edge $e = (u, v, t, \lambda, c)$, we first determine the set $S \subseteq R_u$ of labels with distinct starting times, minimal costs and an arrival time less or equal to t at vertex u (line 4). Next, we push each label in S over e. We check for predomination and equivalence with the other labels in R_v and discard all predominated labels. In case the new label is predominated, we discard it and continue with the next label in S. In case that the new label l_{new} is equivalent to a label $l = (a, c, p, v, t_e, \Pi) \in R_v$, we add l_{new} to Π. If l_{new} arrives earlier at v than the arrival time of l, we replace the representative l with l_{new} in R_v. If the new label is not predominated and not equivalent to any label in R_v, we insert l_{new} into R_v and $\Pi_{l_{new}}$. In this case, l_{new} is a new representative and we initialize $\Pi_{l_{new}}$ (which contains only l_{new} at this point). For the following discussion, we define the set of all labels at vertex $v \in V$ as $L_v \coloneqq \bigcup_{l \in R_v} \Pi_l$.

Lemma 4. *Let $P_{s,v}$ be an efficient path and $P_{s,w}$ a prefix-path of $P_{s,v}$. At the end of Phase 1 of Algorithm 1, R_w contains a label representing $P_{s,w}$.*

Proof. We show that each prefix-path $P_0, P_1, \ldots, P_k$, with $P_0 = P_{s,s}$ and $P_k = P_{s,v}$ is represented by a label at the last vertex of each prefix-path by induction over the length h. Note that all prefix-paths have the same starting time $b = s(P_{s,v})$. For $h = 0$ we have $P_0 = P_{s,s}$ and since s does not have any incoming edges, the initial label $l_{init} = l_0$ representing P_0 is in L_s after Phase 1 finishes. Assume the hypothesis is true for $h = i-1$ and consider the case for $h = i$ and the prefix-path $P_i = P_{s,v_{i+1}} = (e_1, \ldots, e_i = (v_i, v_{i+1}, t_i, \lambda_i, c_i))$, which consists of the prefix-path $P_{i-1} = P_{s,v_i} = (e_1, \ldots, e_{i-1} = (v_{i-1}, v_i, t_{i-1}, \lambda_{i-1}, c_{i-1}))$ and edge $e_i = (v_i, v_{i+1}, t_i, \lambda_i, c_i)$. Due to the induction hypothesis, we conclude that L_{v_i} contains a label $l_{i-1} = (b, a_{i-1}, c_{i-1}, p_{i-1}, v_i, r_{i-1}, \Pi_{i-1})$ that represents P_{i-1}. Because P_{i-1} is a prefix-path of $P_{s,v}$ the representing label l_{i-1} must have the minimum cost in L_{v_i} under all labels with starting time b before edge e_i arrives. Else, it would have been predominated and replaced by a cheaper one (Lemma 3). The set S contains a label that represents l_{i-1}, because the representative of Π_{i-1} has an arrival time less or equal to a_{i-1}. Therefore, the algorithm pushes $l_{new} = (b, t_i + \lambda_i, c_{i-1} + c_i, l_{i-1}, v_{i+1}, t_i, \cdot)$ over edge e_i. If $R_{v_{i+1}}$ is empty the label l_{new} gets inserted into $R_{v_{i+1}}$ and $\Pi_{l_{new}}$. Otherwise we have to check for predomination and equivalence with every label $l' = (b', a', c', p', v_{i+1}, r', \Pi') \in R_{v_{i+1}}$. There are the following cases:

1. l_{new} predominates l': We can remove l' from $R_{v_{i+1}}$ because it will never be part of an efficient path (Lemma 3). The same is true for each label in Π' and therefore we delete Π'. However, we keep l_{new} and continue with the next label.

Algorithm 1. for McFEnum

Input: Graph G in edge stream representation, source $s \in V$ and target $z \in V$
Output: All efficient (s, z)-paths

    Phase 1
1: initialize R_v for each $v \in V$
2: insert label $l_{init} = (0, 0, 0, -, s, -, \Pi_{l_{init}})$ into R_s and $\Pi_{l_{init}}$
3: **for** each edge $e = (u, v, t_e, \lambda_e, c_e)$ **do**
4:     $S \leftarrow \{(b, a, c, p, v, r, \cdot) \in R_u \mid a \le t_e, c$ minimal and distinct starting times $b\}$
5:     **for** each $l = (b, a, c, p, v, r, \cdot) \in S$ with $a \le t_e$ **do**
6:         **if** $u = s$ **then**
7:             $l_{new} \leftarrow (t_e, t_e + \lambda_e, c_e, l, s, t_e, \cdot)$
8:         **else**
9:             $l_{new} \leftarrow (b, t_e + \lambda_e, c + c_e, l, u, t_e, \cdot)$
10:         **for** each $l' = (b', a', c', p', v, t', \Pi') \in R_v$ **do**
11:             **if** l_{new} predominates l' **then**
12:                 remove l' from R_v and delete Π'
13:             **else if** l' is equivalent to l_{new} **then**
14:                 insert l_{new} into Π'
15:                 set reference $\Pi \leftarrow \Pi'$
16:                 **if** $t_e + \lambda_e < a'$ **then**
17:                     replace l' in R_v by l_{new}
18:                 goto 5
19:             **else if** l' predominates l_{new} **then**
20:                 delete l_{new}
21:                 goto 5
22:         insert l_{new} into R_v and initialize $\Pi_{l_{new}}$ with l_{new}
    Phase 2
23: mark nondominated labels in R_z
24: **for** each marked label $l' = (b, a, c, r, z, p, \Pi) \in R_z$ **do**
25:     **for** each label $l \in \Pi$ with minimal arrival time **do**
26:         initialize empty path P
27:         call OutputPaths(l, P);

    Procedure for outputting paths
28: **procedure** OutputPaths(label $l = (b, a, c, p, cur, r, \Pi)$, path P)
29:     prepend edge $l.edge$ to P
30:     **if** l has predecessor $p = (b_p, a_p, c_p, p_p, v_p, r_p, \Pi_p)$ **then**
31:         **for** each label $l' = (b_{l'}, a_{l'}, c_{l'}, p_{l'}, v_{l'}, r_{l'}, \Pi_p)$ in Π_p **do**
32:             **if** $a_{l'} \le r$ **then**
33:                 call OutputPaths(l', P, *visited*)
34:         **return**
35:     output path P

---

2. l_{new} and l' are equivalent: We add l_{new} to Π'. In this case we represent the path P_i by the representative of Π'. Consequently, the path is represented by a label in $L_{v_{i+1}}$.

If neither of these two cases apply for any label in $R_{v_{i+1}}$, we add l_{new} to $R_{v_{i+1}}$ and to $\Pi_{l_{new}}$. The case that a label l is not equivalent to l_{new} and predominates l_{new} cannot be for the following reason. If l predominates l_{new}, there is a path P' from s to v_{i+1} with less costs or later starting time (because to l and l_{new} are not equivalent) and a not later arrival time. Replacing the prefix-path P_i with P' in the path $P_{s,v}$ would lead to a (s, v)-path with less costs and/or shorter duration. This contradicts our assumption that $P_{s,v}$ is efficient. Therefore, after Phase 1 finished, the label l_{new} representing the prefix-path P_i is in L_{v_i}. It follows that if $P_{s,v} = (e_1, \ldots, e_k)$ is an efficient path, then after Phase 1 the set R_v contains a label representing $P_{s,v}$ (possibly, such that a label in R_v represents a list of equivalent labels, that contains the label representing $P_{s,v}$). □

After all edges have been processed, the algorithm continues with Phase 2. First, the algorithm marks all nondominated labels in R_z. For each marked label l the algorithm iterates over the list of equivalent labels Π_l and calls the output procedure for each label in Π_l. We show that all and only efficient paths are enumerated.

Theorem 1. *Let $G = (V, E)$ be a weighted temporal graph with strictly positive edge costs and $s, z \in V$ an instance of McfEnum. Algorithm 1 outputs exactly the set of all efficient (s, z)-paths.*

Proof. Lemma 4 implies that for each efficient path $P_{s,z}$ there is a corresponding representative label in R_z after Phase 1 is finished. Note that there might also be labels in L_z that do not represent efficient paths. First, we mark all non-dominated labels in R_z. For every marked representative $l' = (b, a, c, p, z, r, \Pi_{l'})$ in R_z we proceed by calling the output procedure for all labels $l \in \Pi_{l'}$ with minimal arrival time. Each such label l represents at least one efficient (s, z)-path and we call the output procedure with l and the empty path P. Let path $Q = (e_1 \ldots, e_k)$ be an efficient (s, z)-paths represented by l. We show that the output procedure successively constructs the suffix-paths $P_i = (e_{k-i+1}, \ldots, e_k)$ of Q for $i \in \{1, \ldots, k\}$ and finally outputs $Q = P_k = (e_1, \ldots, e_k)$.

We use induction over the length $i \geq 1$ of the suffix-path. For $i = 1$ the statement is true. $P_1 = (e_k)$ is constructed by the first instruction which prepends the last edge of Q to the initially empty path P. Now, assume the statement holds for $i = j - 1 < k$, i.e. the suffix-path P_{j-1} of Q with length $j - 1$ has been constructed, by calling the output procedure with P_{j-2} and label $l_{j-1} = (b, a, c, p, v_{k-j+2}, r, \Pi)$. The suffix-path $P_j = (e_{k-j+1}, \ldots, e_k)$ equals P_{j-1} with the additional edge $e_{k-j+1} = (v_{k-j+1}, v_{k-j+2}, t, \lambda, c)$ with v_{k-j+2} being the first vertex of P_{j-1}. The predecessor of l_{j-1} is label $p = (b, a_p, c_p, p_p, v_{k-j+1}, r_p, \Pi')$. We recursively call the output procedure for each label in the list of equivalent labels Π' and verify that the arrival time of each of these labels is less or equal to t. Due to Lemma 4, we particularly call the output procedure for the label that represents the beginning of the suffix-path P_j and which has an arrival time less than t. Consequently, there is a call of the output procedure that constructs P_j. If P_j does not have a predecessor, we arrived at vertex s and the algorithm outputs the found path $Q = P_k$.

We still have to show that only efficient paths are enumerated. In order to enumerate a non-efficient (s, z)-path Q', there has to be a label l_q in L_z for which the output procedure is called and which represents Q'. For Q' to be non-efficient there has to be at least one label l_d in L_z that dominates l_q. In line 23 the algorithm marks all nondominated labels in R_z. This implies that l_d and l_q have the same cost and starting times and that they are in the same list, let this list be Π_x for some label $x \in R_z$. Because l_q is dominated by l_d the arrival time of l_d is strictly earlier than the arrival time of l_q. However, we call the output procedure only for the labels in Π_x with the minimal arrival time. Consequently, it is impossible that the non-efficient path Q' is enumerated.

Finally, because all edge costs are strictly positive and due to Observation 1 only paths are enumerated.   □

Example: Figure 4 shows an example for Algorithm 1 at the end of Phase 1. The edges are numbered according to their position in the sequence of the edge stream. The representative labels at the vertices only show the starting time, arrival time and cost. The lists Π of equivalent labels are not shown. All of them contain only the representative, with exception of Π_l represented by label l in R_w. The list Π_l contains label $l = (3, 7, 4)^T$ representing path $((s, w, 3, 4, 4))$ and the equivalent label $(3, 8, 4)^T$ representing path $((s, u, 3, 3, 3), (u, w, 6, 1, 1))$. There are three efficient paths. Starting the output procedure from vertex z with the label $(7, 10, 6)^T$ yields path (e_5, e_8), and starting with label $(3, 9, 5)^T$ yields the two paths (e_1, e_4, e_7) and (e_2, e_7). Notice that label $(7, 9, 2)^T$ in R_w which dominates label $(3, 7, 4)^T$, is not part of an efficient (s, z)-path, due to its late arrival time.

Lemma 5. *Phase 1 of Algorithm 1 has a time complexity of $\mathcal{O}(\mathcal{S} \cdot m^2)$.*

Proof. The outer loop iterates over m edges. For each edge $e = (u, v, t_e, \lambda_e, c_e)$ we have to find the set $S \subseteq R_u$ consisting of all labels with minimal cost, distinct starting times and arrival time less or equal to t_e (see line 4). This can be done in $\mathcal{O}(m)$ time. For each label in S we have to check for predominance or equivalence with each label in R_v in $\mathcal{O}(\mathcal{S} \cdot m)$ total time. Since we have $|S| \leq \mathcal{S}$, we get a total time of $\mathcal{O}(\mathcal{S} \cdot m^2)$.   □

The following lemma shows that the number of labels is polynomially bounded in the size of the input.

Lemma 6. *The total number of labels generated and hold at the vertices in Algorithm 1 is less than or equal than $\mathcal{S} \cdot m + 1$.*

Proof. We need one initial label l_{init}. For each incoming edge $e = (u, v, t, \lambda, c)$ in the edge stream we generate at most $|S| \leq \mathcal{S}$ new labels which we push over e to vertex v. Therefore, we generate at most $\mathcal{S} \cdot m + 1$ labels in total.   □
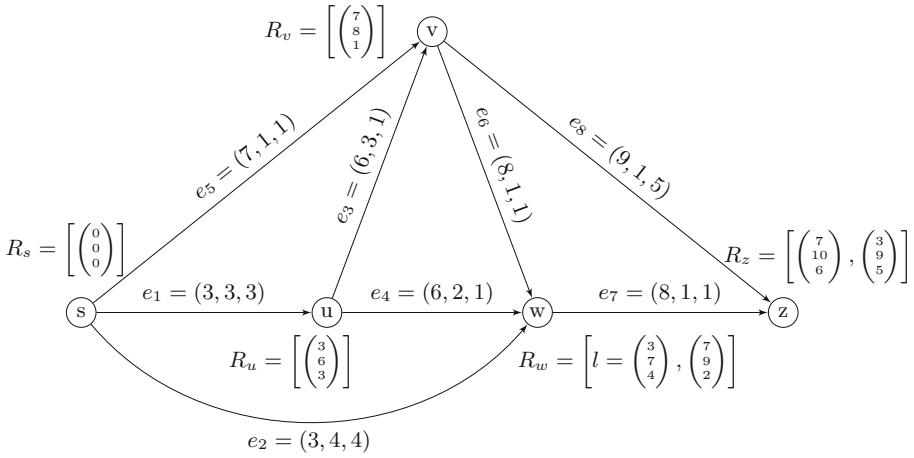
Fig. 4. Example for Algorithm 1. Each vertex is annotated with the representatives after Phase 1 finished.

Theorem 2. $(c>0)$-McFEnum $\in$ ***PSDelayP***.

Proof. Phase 1 takes only polynomial time in size of the input, i.e. number of edges (Lemma 5). In Phase 2 of Algorithm 1, we first find and mark all nondominated labels in R_z in $\mathcal{O}(m^2)$ time. For each nondominated label, we call the output procedure which visits at most $\mathcal{O}(m^2)$ labels and outputs at least one path. It follows that the time between outputting two consecutively processed paths is also bounded by $\mathcal{O}(m^2)$. Therefore, $(c>0)$-McFEnum is in **DelayP**. The space complexity is dominated by the number of labels we have to manage throughout the algorithm. Due to Lemma 6, the number of labels is in $\mathcal{O}(m^2)$. Consequently, $(c>0)$-McFEnum is in **PSDelayP**. $\square$

    The following problems are easy to decide, even if we allow zero weighted edges.

Theorem 3. *Given a weighted temporal graph $G = (V, E)$, $s, z \in V$ and*

1. *an (s, z)-path P, deciding if P is efficient for* McFEnum*, or*
2. *$c \in \mathbb{R}_{\geq 0}$ and $d \in \mathbb{N}$, deciding if there exists a (s, z)-path P with $d(P) \leq d$ and $c(P) \leq c$ is possible in polynomial time.*

Proof. We use Phase 1 of Algorithm 1 and calculate the set $\mathcal{N} \subseteq \mathcal{Y}$ of nondominated points. Due to the possibility of edges with cost 0, there may be non-simple paths, i.e. walks, that have zero-weighted cycles. Nonetheless, Phase 1 terminates after processing the m edges. If there exists an efficient (s, z)-walk W with $(c(W), a(W))$, then there also exists a simple and efficient (s, z)-path Q with the same cost vector. Q is the same path as W but without the zero-weighted cycles. In order to decide if the given path P is efficient, we first calculate the

cost vector $(c(P), a(P))$, and then validate if $(c(P), a(P)) \in \mathcal{N}$. For 2., we only need to compare (c, d) to the points in $\mathcal{N}$. The size of $\mathcal{N}$ is polynomially bounded (Lemma 2). Phase 1 and calculating the cost of P takes polynomial time. □

We can find a maximal set of efficient paths with pairwise different cost vectors in polynomial time.

Corollary 1. *Given a temporal graph $G = (V, E)$ and $s, z \in V$, a maximal set of efficient (s, z)-paths with pairwise different cost vectors for* MCFENUM *can be found in $\mathcal{O}(\mathcal{S} \cdot m^2)$.*

Proof. We use Phase 1 of Algorithm 1 and calculate the set $\mathcal{N} \subseteq \mathcal{Y}$ of non-dominated points in $\mathcal{O}(\mathcal{S} \cdot m^2)$ time. Furthermore, we use a modified output procedure, that stops after outputting the first path. We call the procedure for each nondominated label in R_z, and if a walk is found we additionally remove all zero-weighted cycles. Finding the walk and removing the cycles is possible in linear time, since the length of a walk is bounded by m. □

5   Min-Cost Earliest Arrival Path Enumeration Problem

Based on the reduction given in the beginning of Sect. 3, we modify Algorithm 1 to solve $(c{>}0)$-MCEAENUM. The modified algorithm only needs a linear amount of space and less time for Phase 1. Let $(G = (V, E), s, z)$ be the instance for $(c{>}0)$-MCEAENUM and $(G' = (V', E'), s', z)$ be the transformed instance in which all paths start at time 0 at the new source s'. Although edge $(s, s', 0, 0, 0)$ has costs 0, because s' has no incoming edges any efficient walk in the transformed instance is simple, i.e. a path. With all paths starting at 0, there are the following consequences for the earlier defined relations between labels. First, consider the equivalence from Definition 2 and let $l_1 = (0, a_1, c_1, p_1, v, r_1, \Pi_1)$ and $l_2 = (0, a_2, c_2, p_2, v, r_2, \Pi_2)$ be two labels at vertex v. Because the starting time of both labels is 0, the labels are equivalent if $c_1 = c_2$. It follows, that label l_1 predominates l_2 if $a_1 \leq a_2$ and $c_1 < c_2$, hence there is no distinction between domination and predomination.

Algorithm 2 shows a modified version of Algorithm 1, that sets all starting times to 0. In line 4 we only need to find a single label with the minimum costs, instead of the set S. At each vertex v we only have one representative l in R_v with minimal costs (w.r.t. the other labels in R_v), due to the equivalence of labels that have the same costs. In Phase 2 we do not need to explicitly find the nondominated labels in R_z. Because each label l in R_z has a unique cost value, we consider each represented class Π_l and call the output procedure with the labels that have the minimum arrival time in Π_l.

Theorem 4. *Algorithm 2 outputs exactly all efficient (s, z)-paths w.r.t. arrival time and costs.*

Proof. Lemma 4 implies that for each efficient path $P_{s,z}$ there is a corresponding representative label in R_z after Phase 1 is finished. For every representative

---

Algorithm 2. for MCEAENUM

Input: Graph G in edge stream representation, source $s \in V$ and target $z \in V$
Output: All efficient (s, z)-paths

    Phase 1
1: initialize R_v for each $v \in V$
2: insert label $l_{init} = (0, 0, 0, 0, -, s, \Pi_{l_{init}})$ into R_s and $\Pi_{l_{init}}$
3: **for** each edge $e = (u, v, t_e, \lambda_e, c_e)$ **do**
4:     $l \leftarrow (0, a, c, p, u, \cdot, \cdot) \in R_u$ with $a \leq t_e$ and c minimal
5:     $l_{new} \leftarrow (0, t_e + \lambda_e, c + c_e, l, v, t_e, \Pi)$
6:     **for** each $l' = (0, a', c', p', v, r', \Pi') \in R_v$ **do**
7:         **if** l_{new} dominates l' **then**
8:             remove l' from R_v and delete Π'
9:         **else if** l' dominates l_{new} **then**
10:            delete l_{new}
11:            **goto** line 4
12:         **else if** l' is equivalent to l_{new} **then**
13:            set reference $\Pi \leftarrow \Pi'$
14:            insert p_{new} into Π'
15:            **if** $t_e + \lambda_e < a'$ **then**
16:                replace l' in R_v by l_{new}
17:            **goto** line 4
18:     $\Pi \leftarrow \Pi_{l_{new}}$
19:     insert l_{new} into R_v and u into $\Pi_{l_{new}}$
    Phase 2
20: **for** each label $l' = (0, a, c, p, z, r, \Pi_{l'}) \in R_z$ **do**
21:     **for** each label $l \in \Pi_{l'}$ with minimal arrival time **do**
22:         initialize empty path P
23:         call OUTPUTPATHS(l, P)

$l' = (0, a, c, p, z, r, \Pi)$ in R_z, it holds by construction that all labels in $\Pi_{l'}$ have the same costs. Therefore, we only need to consider the nondominated labels in $\Pi_{l'}$ with minimal arrival time $a_{min} := \min\{a \mid l = (0, a, c, p, w, r, \Pi) \in \Pi_{l'}\}$. Hence, for each $l' \in R_z$ we call the output procedure for every label in $l \in \Pi_{l'}$ if l has minimal arrival time a_{min}.   □

Algorithm 2 uses a linear number of labels.

Lemma 7. *The total number of labels generated and hold at the vertices in Algorithm 2 is at most $m + 1$.*

Proof. We need one initial label l_{init} at the source vertex s. For each incoming edge $e = (u, v, t, \lambda, c)$ in the edge stream, in line 4 we choose the label l with minimal costs and arrival time at most t. We only push l and generate at most one new label l_{new} at vertex v. Therefore, we generate at most $m + 1$ labels in total.   □

Lemma 8. *Phase 1 of Algorithm 2 has a time complexity of $\mathcal{O}(m^2)$.*

Proof. The outer loop iterates over m edges. In each iteration we have to find the representative label $l \in R_u$ with minimum costs and arrival time $a \leq t_e$. This is possible in constant time, since we always keep the label with the earliest arrival time of each equivalence class as representative in R_u. Next we have to check the domination and equivalence between l_{new} and each label $l' \in R_v$. Each of the cases takes constant time, and there are $\mathcal{O}(m)$ labels in R_v. Altogether, a time complexity of $\mathcal{O}(m^2)$ follows.   □

Algorithm 2 lists all efficient paths in polynomial delay and uses only linear space.

Theorem 5. $(c{>}0)$-MCEAENUM $\in$ ***PSDelayP***.   □

Using Algorithm 2, also the results of Theorem 3 and Corollary 1 can be adapted for the earliest arrival case.

Theorem 6. *Given a weighted temporal graph $G = (V, E)$, $s, z \in V$ and*

1. *an (s, z)-path P, deciding if P is efficient for* MCEAENUM*, or*
2. *$c \in \mathbb{R}_{\geq 0}$ and $a \in \mathbb{N}$, deciding if there exists a (s, z)-path P with $a(P) \leq a$ and $c(P) \leq c$ is possible in polynomial time.*   □

Corollary 2. *Given a temporal graph $G = (V, E)$ and $s, z \in V$, a maximal set of efficient (s, z)-paths with pairwise different cost vectors for* MCEAENUM *can be found in time $\mathcal{O}(m^2)$.*   □

6   Conclusion

We discussed the bicriteria optimization problems MIN-COST EARLIEST ARRIVAL PATHS ENUMERATION PROBLEM (MCEAENUM) and MIN-COST FASTEST PATHS ENUMERATION (MCFENUM). We have shown that enumerating the sets of all efficient paths with low costs and early arrival time or short duration is possible in polynomial time delay and linear or polynomial space if the input graph has strictly positive edge costs. In case of nonnegative edge costs, it is possible to determine a maximal set of efficient paths with pairwise different cost vectors in $\mathcal{O}(m^2)$ time or $\mathcal{O}(\mathcal{S} \cdot m^2)$ time, respectively, where $\mathcal{S}$ is the number of distinct availability times of edges leaving the source vertex s. We can find an efficient path for each nondominated point in polynomial time.

For the cases of zero-weighted or even negative edge weights, we cannot guarantee polynomial time delay for our algorithms to solve MCFENUM or MCEAENUM. However, the proposed algorithms can be used to determine all efficient (s, z)-walks in polynomial time delay. Because of each edge in a temporal graph can only be used for departure at a certain time, the number of different walks is finite and the algorithms terminate. So far, we are not aware of a way to ensure that only simple paths are enumerated without loosing the property that the delay between the output of two paths stays polynomially bounded.

References

1. Casteigts, A., Flocchini, P., Quattrociocchi, W., Santoro, N.: Time-varying graphs and dynamic networks. Int. J. Parallel Emergent Distrib. Syst. **27**(5), 387–408 (2012)
2. Ehrgott, M., Gandibleux, X.: A survey and annotated bibliography of multiobjective combinatorial optimization. OR-Spektrum **22**(4), 425–460 (2000)
3. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York (1979)
4. Hamacher, H.W., Ruzika, S., Tjandra, S.A.: Algorithms for time-dependent bicriteria shortest path problems. Discrete Optim. **3**(3), 238–254 (2006)
5. Hansen, P.: Bicriterion path problems. In: Fandel, G., Gal, T. (eds.) Multiple Criteria Decision Making Theory and Application, pp. 109–127. Springer, Berlin (1980). https://doi.org/10.1007/978-3-642-48782-8_9
6. Holme, P., Saramäki, J.: Temporal networks. Phys. Rep. **519**(3), 97–125 (2012)
7. Johnson, D.S., Yannakakis, M., Papadimitriou, C.H.: On generating all maximal independent sets. Inf. Process. Lett. **27**(3), 119–123 (1988)
8. Kostakos, V.: Temporal graphs. Phys. A: Stat. Mech. Appl. **388**(6), 1007–1023 (2009)
9. Martins, E.Q.V.: On a multicriteria shortest path problem. Eur. J. Oper. Res. **16**(2), 236–245 (1984)
10. Skriver, A., Andersen, K.: A label correcting approach for solving bicriterion shortest-path problems. Comput. Oper. Res. **27**(6), 507–524 (2000)
11. Wu, H., Cheng, J., Huang, S., Ke, Y., Lu, Y., Xu, Y.: Path problems in temporal graphs. Proc. VLDB Endow. **7**(9), 721–732 (2014)
12. Xuan, B.B., Ferreira, A., Jarry, A.: Computing shortest, fastest, and foremost journeys in dynamic networks. Int. J. Found. Comput. Sci. **14**(02), 267–285 (2003)