Marche sans escale dans un graphe temporel

Juan Villacis-Llobet, Binh-Minh Bui-Xuan, Maria Potop-Butucaru

Marche sans escale dans un graphe temporel

Juan Villacis-Llobet[1] et Binh-Minh Bui-Xuan[1] et Maria Potop-Butucaru[1]

[1]*LIP6 (CNRS – Sorbonne Université), Paris, France*

Une marche temporelle est une suite d'arêtes adjacentes où chacune est munie d'une date prise dans un ordre croissant en suivant le sens de la marche. La marche est sans escale si les dates de toute paire d'arêtes consécutives se succèdent sans temps d'arrêt. Un chemin temporel est une marche temporelle où tout sommet apparaît au plus une fois. Dans un graphe temporel avec n sommets et m arêtes temporelles, il est possible de décider en temps $O(n + m \log m)$ s'il existe une marche temporelle sans escale partant d'un sommet s et arrivant à un sommet t avant une date d [Himmel, Bentert, Nichterlein and Niedermeier, CNA 2019]. En revanche, décider s'il existe un chemin temporel sans escale partant de s et arrivant à t avant la date d est NP-complet [Casteigts, Himmel, Molter and Zschoche, *Algorithmica*, 2021]. Nous présentons un algorithme linéaire pour décider le premier cas [†].

Mots-clefs : temporal graph, shortest path, foremost journey, non-stop journey.

1 Introduction

A temporal (di)graph is a collection of dated arcs over a vertex set. When studying the foremost arrival date for connecting two vertices in a temporal graph, there is no need to make a distinction between the notions of a walk and a path, in the following sense. Let us call the temporal version of a walk a journey, and define it as a sequence of dated arcs fulfilling two properties: firstly, the arcs are dated increasingly in the sequence (temporal); secondly, the target vertex of every preceding arc is the source vertex of its immediate successor (walk). This helps modelling both ground traffic and TCP/IP transmission [SKK19], where a vehicle or a TCP/IP package need to be at successive checkpoints in increasing arrival dates. Here, the journey's arrival date is the arrival date of the last arc in the sequence. A temporal path is a journey which never visits a vertex twice. Then, as long as the foremost arrival date is concerned, there is no need to circle around a stopover vertex. In other words, the foremost arrival date of a journey from a vertex s to a vertex t is the foremost arrival date of a temporal path from s to t.

Furthermore, the following hereditary property can be observed. There exists among the foremost journeys from s to t a journey where every prefix is itself a foremost journey (from s to the stopover vertex where the prefix ends). This hereditary property of journeys is fundamental for devising polynomial algorithms computing foremost journeys following a greedy approach similar to Dijkstra algorithm [BFJ03]. Similar prefix preservation properties are also proved to be important for classifying polynomial path-related problems on temporal graphs [CFMS15, RMNN21].

In this paper, we study journeys with the non-stop property, when the dates of every consecutive pair of arcs in the journey are without delay. This helps modeling cases with physical constraints when the traversal is performed by an aircraft or a boat: while a TCP/IP package can be retained at a vertex for an unlimited delay, an aircraft can not perform a stationary flight at a vertex waiting for a better wind condition. Another use case of the non-stop property is about disease/gossip spreading where a viral infection/information is supposed not to stay on any individual forever. Solving non-stop connectivity here would let us know if the destination vertex would be at risk of contamination whenever the source vertex is infected/informed.

Computationally, the situation for non-stop connection is more challenging because the two above mentioned properties for the with-stops case do not necessarily hold. On the one hand there can be non-stop journeys from s to t arriving at a date which is strictly earlier than the arrival date of any non-stop temporal path from s to t. There could also be instances where non-stop journeys exist while no non-stop temporal

†. An extended abstract of our results can be found in [VBP22].

path can be found, for instance when the dated arcs describe a non-stop looping over some stopover vertex. Furthermore, it does not seem obvious to retrieve an equivalent of the hereditary property for non-stop journeys. Thus, it seems difficult to adopt the greedy approach in order to compute the foremost arrival date from s to t via a non-stop journey, as with [BFJ03]. Luckily enough, such a journey can be computed in polynomial time, and as low as $O(n+m\log m)$ for an n-vertex m-dated arc temporal graph [HBNN19]. The situation is however much worse with non-stop temporal paths, where it is NP-complete to decide whether there exists a non-stop temporal path from s to t arriving at date d [CHMZ21].

The main idea for dealing with non-stop journeys proposed in Himmel et al. [HBNN19] is to collect all arcs dated with a given date d, plus some selected arcs just preceding d, into a graph named G_d. Then, when scanning every G_d from the first to the last possible values of d, one can use an incremental computation of journeys ending before d in order to solve the foremost non-stop journey problem within the desired $O(n+m\log m)$ time by a careful revision of Dijkstra algorithm on G_d. Since the use of Dijkstra algorithm hints at the existence of a greedoid structure, it would be very interesting to study how the outer incremental scan over the values of d combines with the inner calls of the revised Dijkstra algorithm on each G_d.

Nevertheless, we below rather put the greedy approach aside and propose a new approach for non-stop journeys, while aiming at improving the complexity downto linear time. Our approach relies on a transformation of the problem over journeys to another problem over arc sets, *cf.* Section 2. A particular case of our algorithm is simple to implement and described in Section 3. Among other use, it can serve for unit testing more optimised implementations. We also give in [VBP22] a more general version of our algorithm whose time complexity remains linear. We close this paper by giving concluding remarks and propose some direction for future works.

2 Reducing a path problem to a set problem

We denote $V \otimes V = V \times V \setminus \{(v,v) : v \in V\}$. A *temporal digraph* is a tuple $G = (\tau, V, A, c)$ where: $\tau \in \mathbb{N}$ is an integer called the *timespan* of G from which we define interval $T = [\![0, \tau-1]\!]$ as the set of time instants used in G; V is a finite set called the *vertex set* of G; $A \subseteq T \times V \otimes V$ is called the *arc set* of G; and $c : A \to \mathbb{N}$ is a *cost function* representing the traversal time of every arc in G.

For every arc $a = (d, s, t) \in A$, we denote $s(a) = s$ the *source vertex* of the arc, $t(a) = t$ its *target vertex*, and $d(a) = d$ its *departure time*. The traversal of arc a departs from s towards t at departure time d and arrives to t at arrival time $d + c(a)$.

We define journeys with waiting time constraints following the formalism given in [HBNN19]. Let $s, t \in V$ be two distinct vertices of G. Let $\alpha, \beta : V \to \mathbb{N}$ be two functions representing the minimum and maximum waiting time at every vertex. An (α,β)-*journey* from s to t is a sequence of arcs $J = (a_1, a_2, \ldots, a_p) \in A^p$, where $s(a_1) = s$, $t(a_p) = t$, and for every $1 \le i < p$ we have both $t(a_i) = s(a_{i+1})$ and $d(a_i) + c(a_i) + \alpha(t(a_i)) \le d(a_{i+1}) \le d(a_i) + c(a_i) + \beta(t(a_i))$. For $1 \le i < p$, the traversal of arc a_i begins from source vertex $s(a_i)$ at departure time $d(a_i)$, it takes $c(a_i)$ time steps to arrive at target vertex $t(a_i)$, where the journey has to be delayed for at least $\alpha(t(a_i))$ and at most $\beta(t(a_i))$ time steps before pursuing with the traversal of arc a_{i+1}. The *arrival date* of J is defined as $d(a_p) + c(a_p)$. A journey is called *foremost* when its arrival date is minimum.

On input a temporal graph $G = (\tau, V, A, c)$ with transit functions (α, β) and two vertices s, t in G, the problem of computing the minimum value of arrival date $d(a_p) + c(a_p)$ taken over every (α,β)-journey $J = (a_1, a_2, \ldots, a_p)$ from s to t is called the ForemostJourneyArrival under (α,β)-transit problem. When both α and β are constantly equal to 0, such a $(0,0)$-journey J is called a *non-stop* journey. Lemma 1 below is crucial because it helps us reduce a path-like problem down to a set problem. For any source vertex $s \in V$, we define the set of (α,β)-*reachable arcs* from s as

$$R(s) = \{a_p \in A : \exists\ (\alpha,\beta)\text{-journey } J = (a_1, a_2, \ldots, a_p) \in A^p \wedge s(a_1) = s\}.$$

Lemma 1 $\min\{d(a)+c(a) : a \in R(s) \wedge t(a) = t\}$ *is equal to the minimum arrival date of an* (α,β)*-journey from* s *to* t.

Lemma 1 is proper to temporal digraphs in the sense that we can from input $G = (\tau, V, A, c)$ filter the set A to a smaller subset $R(s) \subseteq A$, then filter further to the set of arcs whose target vertex is t, and finally reduce the stream to find the minimum value $d(a) + c(a)$, as with filter-map-reduce programming.

3 Computing the foremost arrival date

We show how to compute $R(s)$, using some intermediary steps. We claim that it is possible to compute $R(s)$ from what we call the set of *valid transit departures* in an (α,β)-journey from s, that is
$D(s) = \{(d,v) \in T \times V : \exists\ (\alpha,\beta)\text{-journey } J = (a_1, a_2, \ldots, a_p) \in A^p \wedge s(a_1) = s \wedge s(a_p) = v \wedge d(a_p) = d\}$.

Indeed, we can output $R(s)$ from $D(s)$ using a bucket sort (a.k.a. radix sort) approach as follows. We initialize a boolean table `R` indexed by the elements of A. For any $a \in A$ with $a = (d,u,v)$, we scan $D(s)$ and check if $(d,u) \in D(s)$. If this is the case we set `R[a]` to `true`. At the end of the process, we scan `R` and output every index a where `R[a]` has value `true`. With a data structure representing $D(s)$ with $O(1)$ access to membership testing, the computation of $R(s)$ from $D(s)$ is linear in $|A|$.

We use the following graph in Lemma 2 below to compute $D(s)$. The (α,β)-*transit departure digraph* $G_D = (V_D, A_D)$ is a static graph, defined as follows. First, $V_D = \{(d,v) : \exists a \in A, s(a) = v \wedge d(a) = d\}$ is the set of all possible transit departures, including those not necessarily valid w.r.t. any (α,β)-journey from s. For any pair of vertices $x = (d,u)$ and $y = (d',v)$ of V_D, we define $(x,y) \in A_D$ if and only if we have both that $a = (d,u,v)$ belongs to A and that $d + c(a) + \alpha(v) \leq d' \leq d + c(a) + \beta(v)$.

To prevent exceeding the linear complexity, we must ensure that the size of G_D is not too large. Let $\gamma = \max\{\beta(v) - \alpha(v) + 1 : v \in V\}$. We claim that $|V_D| \leq |A|$ and $|A_D| \leq \gamma \times |A|$. The former inequality follows from definition of V_D, which ensures there will be at most one vertex in V_D for every arc in A. Let us examine $(x,y) \in A_D$ with $x = (d,u)$ and $y = (d',v)$. By definition, $a = (d,u,v)$ must belong to A, and d' must satisfy the waiting time constraints $d + c(a) + \alpha(v) \leq d' \leq d + c(a) + \beta(v)$. Let us define function $f : A_D \to A \times [\![0, \gamma - 1]\!]$ as $f((d,u),(d',v)) = (d,u,v,d' - d - c(a))$. Then, we can check that f is a well-defined injective function, and therefore, deduce that $|A_D| \leq \gamma \times |A|$.

Let us first consider the case of non-stop journeys. It follows from definition that $(x,y) \in A_D \wedge x \in D(s)$ implies $y \in D(s)$. Hence, $D(s)$ encompasses the set of vertices in G_D reachable from any vertex of the set $V_D \cap \{(d,s) : 0 \leq d < \tau\}$. We prove in Lemma 2 that $D(s)$ is exactly the latter set. Hence, $D(s)$ can be computed by any classical graph search on G_D, which takes linear time in $|A| = |V_D| = |A_D|$ since $\gamma = 1$ for non-stop journeys.

Lemma 2 *Let $G = (\tau, V, A, c)$ be a temporal digraph, $\alpha, \beta : V \to \mathbb{N}$ two functions representing the minimum and maximum waiting time constraints, $s \in V$, and $D(s)$ the set of valid transit departures in an (α,β)-journey from s. Let $G_D = (V_D, A_D)$ be the (α,β)-transit departure digraph of G. Then, $D(s)$ is exactly the set of vertices in G_D which are reachable from a (directed) path beginning at any vertex of the set $V_D \cap \{(d,s) : 0 \leq d < \tau\}$.*

Proof. We denote by $R_D(s)$ the set of vertices in G_D which are reachable from a (directed) path beginning at any vertex of the set $V_D \cap \{(d,s) : 0 \leq d < \tau\}$. By definition of G_D, we have the following closure property: if $(x,y) \in A_D$ and $x \in D(s)$ then $y \in D(s)$. Besides, whenever $(d,s) \in V_D$ for any $0 \leq d < \tau$, that is, whenever there exists $a \in A$ such that $s(a) = s$ and $d(a) = d$, we also have that $(d,s) \in D(s)$ by using the single-arc (α,β)-journey $J = (a)$ in the definition of $D(s)$. Now, we use the above mentioned closure property in order to deduce that $R_D(s) \subseteq D(s)$. Hence, the only thing left for us to show is that $D(s) \subseteq R_D(s)$.

Let $(d,v) \in D(s)$. We would like to prove that $(d,v) \in R_D(s)$. By definition of $D(s)$, there exists an (α,β)-journey $J = (a_1, a_2, \ldots, a_p) \in A^p$ such that $s(a_1) = s$, $s(a_p) = v$, and $d(a_p) = d$. Let us consider $J_q = (a_1, a_2, \ldots, a_q)$, for any $1 \leq q \leq p$. For convenience, we denote $d_q = d(a_q)$, $v_q = s(a_q)$, and $x_q = (d_q, v_q)$. Since $a_q \in A$ we have from definition of V_D that $x_q \in V_D$, for any $1 \leq q \leq p$. We claim that $(x_1, x_2, \ldots, x_p)$ is a directed walk in the static digraph G_D, with $x_1 \in V_D \cap \{(d,s) : 0 \leq d < \tau\}$ and $x_p = (d,v)$.

Indeed, by definition of $D(s)$ we have for any $1 \leq q \leq p$ that $(d_q, v_q) \in D(s)$. When $q = 1$, this implies $v_1 = s$, and therefore $x_1 = (d_1, v_1) = (d_1, s)$ belongs to $V_D \cap \{(d,s) : 0 \leq d < \tau\}$. Since the original J is an (α,β)-journey, it must satisfy the waiting time constraints, that is, we have $d_q + c(a_q) + \alpha(t(a_q)) \leq d_{q+1} \leq d_q + c(a_q) + \beta(t(a_q))$, for every $1 \leq q < p$. Besides, since $t(a_q) = s(a_{q+1}) = v_{q+1}$, we have both $(d_q, v_q, v_{q+1}) = a_q \in A$ and $d_q + c(a_q) + \alpha(v_{q+1}) \leq d_{q+1} \leq d_q + c(a_q) + \beta(v_{q+1})$. This implies (x_q, x_{q+1}) belongs to A_D for every $1 \leq q < p$. In other words, $(x_1, x_2, \ldots, x_p)$ is a directed walk in G_D. Since $d_p = d(a_p) = d$ and $s_p = s(a_p) = v$, we also have $x_p = (d,v)$. We have shown a directed walk in

G_D beginning from vertex $x_1 \in V_D \cap \{(d, s) : 0 \leq d < \tau\}$, and ending at vertex $x_p = (d, v)$. This also implies there exists a directed path in G_D from x_1 to $x_p = (d, v)$. Hence, $(d, v) \in R_D(s)$. We have proved for every $(d, v) \in D(s)$ that $(d, v) \in R_D(s)$. In other words, $D(s) \subseteq R_D(s)$. □

The general case of arbitrary $\gamma \geq 1$ is considered in [VBP22]. Here, both bounds $|V_D| \leq |A|$ and $|A_D| \leq \gamma \times |A|$ are sharp: the size of G_D is no more linear in $|A|$, although it is not far from it. We showed that it is possible to implicitly represent A_D with a number of arcs linear in $|A|$. Furthermore, we also proved that V_D can be constructed in linear time in $|A|$, as well as the implicit representation of A_D. However, this did not allow us to conclude yet because a classical graph search on G_D is not necessarily possible when using the implicit representation of A_D. We then devised a marking algorithm for finding $D(s)$ in linear time in $|A|$, see [VBP22, Section 4, Stage 3]. All in all, our results combine for a linear time computation of the foremost arrival date of a (α, β)-journey from s to t.

4 Conclusion and perspectives

The non-stop transit condition adds new challenges to the computation of the foremost arrival date of connecting through vertices of a temporal graph. Firstly it makes a singular difference between journeys (temporal version of walks) and temporal paths, the former leading to a linear solution while the latter leads to an NP-complete problem. Then, it also blurs the possibilities of defining a greedoid after the loss of the hereditary property for non-stop foremost journeys. The hereditary property states that there always exists a foremost journey where every prefix is itself a foremost journey.

For future works, a first direction would be to explore polynomial cases of computing the foremost arrival date of a non-stop temporal path, either by restricting the input temporal graph, or by exploring the possibility of approximation algorithms. A more challenging direction would be to explore the possibilities of defining a greedoid greedy algorithm solving these restricted cases. As for journeys, although we described in [VBP22] how to compute the foremost arrival date of a non-stop journey in linear time, it could be necessary to find a simpler substitution for the procedure described in [VBP22, Section 4, Stage 3].

Acknowledgements: We are grateful to the anonymous reviewers for their helpful comments which greatly improved the paper.

References

[BFJ03] B.M. Bui-Xuan, A. Ferreira, and A. Jarry. Computing shortest, fastest, and foremost journeys in dynamic networks. *International Journal of Foundations of Computer Science*, 14(2):267–285, 2003.

[CFMS15] A. Casteigts, P. Flocchini, B. Mans, and N. Santoro. Shortest, fastest, and foremost broadcast in dynamic networks. *International Journal of Foundations of Computer Science*, 26(4):499–522, 2015.

[CHMZ21] A. Casteigts, A.S. Himmel, H. Molter, and P. Zschoche. Finding temporal paths under waiting time constraints. *Algorithmica*, 83:2754–2802, 2021.

[HBNN19] A.S. Himmel, M. Bentert, A. Nichterlein, and R. Niedermeier. Efficient computation of optimal temporal walks under waiting-time constraints. In *8th International Conference on Complex Networks and Their Applications*, volume 882 of *SCI*, pages 494–506, 2019.

[RMNN21] M. Rymar, H. Molter, A. Nichterlein, and R. Niedermeier. Towards classifying the polynomial-time solvability of temporal betweenness centrality. In *47th International Workshop on Graph-Theoretic Concepts in Computer Science*, volume 12911 of *LNCS*, pages 219–231, 2021.

[SKK19] J. Saramäki, M. Kivelä, and M. Karsai. Weighted temporal event graphs. *Temporal Network Theory*, pages 107–128, 2019.

[VBP22] J. Villacis-Llobet, B.M. Bui-Xuan, and M. Potop-Butucaru. Foremost non-stop journey arrival in linear time. In *29th International Colloquium on Structural Information and Communication Complexity*, volume 13298 of *LNCS*, pages 283–301, 2022.