Computing reachability under waiting-time constraints in linear time

Filippo Brunelli, Laurent Viennot

Temporal Workshop

5 July 2023
Temporal graph model
Temporal graph model

A temporal graph is a list of quadruples $e = (u, v, \tau, \lambda)$, called temporal edges, where:

- $u \in V$ is the tail of $e$,
- $v \in V$ is the head of $e$,
- $\tau \in \mathbb{R}$ is the departure time of $e$,
- $\lambda \in \mathbb{R}_{>0}$ is the travel time of $e$.

We also define the arrival time of $e$ as $\tau + \lambda$. 
Maximum waiting time constraints

\[ \beta_w = 3 \]

\[ \beta_u = 3 \]

\[ \beta_v = 3 \]

\[ \beta_s = 3 \]

\[ \beta_t = 3 \]
Temporal walks

Theorem (Casteigts, Himmel, Molter, Zschoche 2021)

In a temporal graph subject to waiting constraints, deciding whether there exists a temporal path between two nodes is NP-hard.
Temporal walks

Theorem (Casteigts, Himmel, Molter, Zschoche 2021)

In a temporal graph subject to waiting constraints, deciding whether there exists a temporal path between two nodes is NP-hard.

This moves the focus on studying temporal walks.
A *temporal walk* is a sequence of temporal edges, such that for any two consecutive:

\[ \text{arr}(e_i) \leq \text{dep}(e_{i+1}) \]

- Temporal paths: A *temporal path* \( P \) is a walk in which all vertices are pairwise distinct.
Temporal graph model

- A *temporal walk* is a sequence of temporal edges, such that for any two consecutive:

  ![Diagram of a temporal walk](image)

  \[ \text{arr}(e_i) \leq \text{dep}(e_{i+1}) \leq \text{arr}(e_i) + \beta_v \]

- Temporal paths: A *temporal path* \( P \) is a walk in which all vertices are pairwise distinct.
A *temporal walk* is a sequence of temporal edges, such that for any two consecutive:

\[ \text{arr}(e_i) \leq \text{dep}(e_{i+1}) \leq \text{arr}(e_i) + \beta_v \]

Temporal paths: A *temporal path* $P$ is a walk in which all vertices are pairwise distinct.

Notice that walks and paths are *strict* since the travel times are assumed strictly positive.
Example of temporal walks

\[ \beta_w = 3 \]

\[ \beta_u = 3 \]

\[ \beta_v = 3 \]
A temporal walk

\[ \beta_w = 3 \]

\[ \beta_v = 3 \]

\[ \beta_u = 3 \]
Not a temporal path

\[ \beta_w = 3 \]
\[ \beta_v = 3 \]
\[ \beta_u = 3 \]

\[ \begin{align*}
  s & \rightarrow u & 1, 1 \\
  w & \rightarrow u & 5, 2 \\
  & \rightarrow v & 10, 2 \\
  v & \rightarrow t & 3, 7 \\
  & \rightarrow w & 2, 3 \\
  t & \rightarrow s & 12, 1
\end{align*} \]
Definition of the reachability problem

A temporal edge \(e\) is \(s\)-reachable if there exists a temporal walk starting from \(s\) ending with \(e\).

**Single-source all-reachable-edges**

Given a temporal graph with waiting constraints \(G = (V, E, \beta)\) and a source node \(s \in V\), compute all the \(s\)-reachable temporal edges.
Definition of the reachability problem

A temporal edge $e$ is $s$-reachable if there exists a temporal walk starting from $s$ ending with $e$.

**Single-source all-reachable-edges**

Given a temporal graph with waiting constraints $G = (V, E, \beta)$ and a source node $s \in V$, compute all the $s$-reachable temporal edges.

This in particular solves the *earliest arrival time* problem.
Definition of the reachability problem

A temporal edge $e$ is $s$-reachable if there exists a temporal walk starting from $s$ ending with $e$.

Single-source all-reachable-edges

Given a temporal graph with waiting constraints $G = (V, E, \beta)$ and a source node $s \in V$, compute all the $s$-reachable temporal edges.

This in particular solves the earliest arrival time problem.

State of the art:
- $O(M \log M)$ [Bentert, Himmel, Nichterlein, Niedermeier, 2020]
- $O(M)$ [Villacis-Llobet, Bui-Xuan, Potop-Butucaru, 2022]
For simplicity we will speak about the case in which all edges have strictly positive travel time ($\lambda > 0$).

**Main result**

A simple algorithm that solves the single-source all-reachable-edge in linear time and space. It receives as input a doubly-sorted representation of the temporal graph.
A *doubly-sorted representation* of a temporal graph \((V, E, \beta)\) consists in two lists \(E^{arr}\) and \(E^{dep}\), where \(E^{arr}\) contains all temporal edges in \(E\) sorted by non-decreasing arrival time, and \(E^{dep}\) by non-decreasing departure time.

**Theorem**

Doubly-sorted and time-expanded are equivalent representations of temporal graphs.
Temporal graph representation

Definition

A *doubly-sorted representation* of a temporal graph \((V, E, \beta)\) consists in two lists \(E^{\text{arr}}\) and \(E^{\text{dep}}\), where \(E^{\text{arr}}\) contains all temporal edges in \(E\) sorted by non-decreasing arrival time, and \(E^{\text{dep}}\) by non-decreasing departure time.

Theorem

Doubly-sorted and time-expanded are equivalent representations of temporal graphs.
Algorithm

Single-source all-reaching-edges

The problem consists in computing all the edges that are reachable (i.e. contained in a walk) from the source $s$. 
Single-source all-reachable-edges

The problem consists in computing all the edges that are reachable (i.e. contained in a walk) from the source $s$.

Extending a walk: $f$ extends $Q$ if $Q.f$ is a walk.

Goal: mark all edges that extend a walk from the source $s$. 
Single-source all-reachable-edges

The problem consists in computing all the edges that are reachable (i.e. contained in a walk) from the source $s$.

It matters just the last edge of the walk:

If $e$ is $s$-reachable and $f$ extends $e$, then $f$ is $s$-reachable.
Algorithm overview

Linear scan of $E^{arr}$

$e = (u, v, r, \lambda)$
Algorithm overview

\[ Linear \ scan \ of \ E^{arr} \]

\[ e = (u, v, r, l) \]

1. We consider iteration \( k : e = E^{arr}[k] \)
2. All edges that extend a walk in \( G_{k-1} \) have been marked.
Algorithm overview

Linear scan of $E^{\text{arr}}$

\[ e = (u, v, r, h) \]

\[ \text{dep}(l) < \text{arr}(e) \]

\[ \text{arr}(e) \leq \text{dep}(g) \leq \text{arr}(e) + \beta_r \]

\[ \text{arr}(e) + \beta_r \leq \text{dep}(h) \]
Algorithm overview

Linear scan of $E_{arr}$

$e = (u, v, \tau, h)$

$\text{dep}(\ell) < \text{arr}(e)$

$\text{arr}(e) \leq \text{dep}(g) \leq \text{arr}(e) + \beta_v$

$\text{arr}(e) + \beta_v \leq \text{dep}(h)$

Edges that depart in $[\text{arr}(e), \text{arr}(e) + \beta]$
Algorithm overview

Linear scan of $E^{\text{arr}}$

$e = (u, v, 1, l)$

$\text{dep}(l) < \text{arr}(e)$

$\text{arr}(e) \leq \text{dep}(g) \leq \text{arr}(e) + \beta_v$

$\text{arr}(e) + \beta_v \leq \text{dep}(h)$

$E^{\text{dep}}$

edges that depart in $[\text{arr}(e), \text{arr}(e) + \beta]$

Filippo Brunelli
Computing reachability 5 July 2023
Algorithm overview

Linear scan of $E^{\text{arr}}$

$e = (u, v, \tau, h)$

$\text{dep}(\tau) < \text{arr}(e)$

$\text{arr}(e) \leq \text{dep}(g) \leq \text{arr}(e) + \beta_v$

$\text{arr}(e) + \beta_v < \text{dep}(h)$

$E^{\text{dep}}$

edges that depart in $[\text{arr}(e), \text{arr}(e) + \beta]$
**Input:** A doubly-sorted representation \((E^{\text{arr}}, E^{\text{dep}})\).

**Output:** The sets \((A_v)_{v \in V}\) of \(s\)-reachable edges at each node \(v\) sorted by arrival time.

1. For each node \(v\), generate the list \(E^{\text{dep}}_v\) by bucket sorting \(E^{\text{dep}}\).

2. **For each node \(v\) do**
   3. Set \(A_v := \emptyset\). /* Set of \(s\)-reachable edges (as a sorted list). */
   4. Set \(p_v := 0\). /* Index of the last processed edge in \(E^{\text{dep}}_v\). */

5. Set all the edges in \(E^{\text{arr}}\) as unmarked.

6. Set \(P[e] := \perp\) for each edge \(e \in E^{\text{arr}}\). /* Parent of \(e\), initially null. */

7. **For each edge \(e = (u, v, \tau, \lambda)\) in \(E^{\text{arr}}\) do**
   8. If \(u = s\) or \(e\) is marked then
      /* \(e\) is \(s\)-reachable. */
      9. \(A_v := A_v \cup \{e\}\)
      10. Let \(a = \tau + \lambda\) be the arrival time of \(e\).
          /* Process further edges from \(v\) until \(\text{dep. time} \geq a + \beta_v\): */
      11. Let \(l > p_v\) be the first index of an edge \((v, w, \tau', \lambda') \in E^{\text{dep}}_v\) such that \(\tau' \geq a\) (set \(l := |E^{\text{dep}}_v| + 1\) if no such index exists).
      12. Let \(r \geq l\) be the last index of an edge \((v, w, \tau', \lambda') \in E^{\text{dep}}_v\) such that \(\tau' \leq a + \beta_v\) (set \(r := l - 1\) if no such index exists).
          /* Mark unmarked edges with \(\text{dep. time}\) in \([a, a + \beta_v]\): */
      13. If \(l \leq r\) then mark each edge \(f \in E^{\text{dep}}_v[l : r]\) and set \(P[f] := e\).
      14. Set \(p_v := r\).

15. **Return the sets \((A_v)_{v \in V}\).**
\[ E^{arr} = \{(s, u, 1, 1), (u, v, 2, 2), (s, w, 2, 3), (w, u, 5, 2), (v, u, 8, 1), (v, w, 6, 3),
(w, u, 10, 2), (v, t, 3, 7), (u, t, 12, 1)\}\]
\( E^{arr} = \{ (s, u, 1, 1), (u, v, 2, 2), (s, w, 2, 3), (w, u, 5, 2), (v, u, 8, 1), (v, w, 6, 3), (w, u, 10, 2), (v, t, 3, 7), (u, t, 12, 1) \} \)
\[ E^{arr} = \{(s, u, 1, 1), (u, v, 2, 2), (s, w, 2, 3), (w, u, 5, 2), (v, u, 8, 1), (v, w, 6, 3), (w, u, 10, 2), (v, t, 3, 7), (u, t, 12, 1)\} \]
\[ E^{arr} = \{(s, u, 1, 1), (u, v, 2, 2), (s, w, 2, 3), (w, u, 5, 2), (v, u, 8, 1), (v, w, 6, 3), (w, u, 10, 2), (v, t, 3, 7), (u, t, 12, 1)\} \]
\[ E^{\text{arr}} = \{(s, u, 1, 1), (u, v, 2, 2), (s, w, 2, 3), (w, u, 5, 2), (v, u, 8, 1), (v, w, 6, 3),
(w, u, 10, 2), (v, t, 3, 7), (u, t, 12, 1)\} \]

\[ E^{\text{dep}}: (u, v, 2, 2), (u, t, 12, 1), p_u = 0 \]
\( E^{arr} = \{(s, u, 1, 1), (u, v, 2, 2), (s, w, 2, 3), (w, u, 5, 2), (v, u, 8, 1), (v, w, 6, 3), (w, u, 10, 2), (v, t, 3, 7), (u, t, 12, 1)\} \)

\[ \beta_w = 3 \]

\[ \beta_v = 3 \]

\[ \beta_u = 3 \]

\( E^{dep}_u : (u, v, 2, 2), (u, t, 12, 1) \quad \downarrow \quad p_u = 0 \)
\[ E^{arr} = \{(s, u, 1, 1), (u, v, 2, 2), (s, w, 2, 3), (w, u, 5, 2), (v, u, 8, 1), (v, w, 6, 3), \\
(u, t, 3, 7), (u, t, 12, 1)\} \]
$$E^{arr} = \{(s, u, 1, 1), (u, v, 2, 2), (s, w, 2, 3), (w, u, 5, 2), (v, u, 8, 1), (v, w, 6, 3),
(w, u, 10, 2), (v, t, 3, 7), (u, t, 12, 1)\}$$

$\beta_w = 3$

$\beta_v = 3$

$\beta_u = 3$

$E^{dep}_{u} : (u, v, 2, 2), (u, t, 12, 1)$
$E^{arr} = \{(s, u, 1, 1), (u, v, 2, 2), (s, w, 2, 3), (w, u, 5, 2), (v, u, 8, 1), (v, w, 6, 3),
(w, u, 10, 2), (v, t, 3, 7), (u, t, 12, 1)\}$
\[ E^{arr} = \{(s, u, 1, 1), (u, v, 2, 2), (s, w, 2, 3), (w, u, 5, 2), (v, u, 8, 1), (v, w, 6, 3),
(w, u, 10, 2), (v, t, 3, 7), (u, t, 12, 1)\} \]
$$E^{arr} = \{(s, u, 1, 1), (u, v, 2, 2), (s, w, 2, 3), (w, u, 5, 2), (v, u, 8, 1), (v, w, 6, 3), (w, u, 10, 2), (v, t, 3, 7), (u, t, 12, 1)\}$$

$$E^{dep}: (v, t, 3, 7), (v, w, 6, 3), (v, u, 8, 1) \quad \gamma_r = 0$$
\[ E^{arr} = \{(s, u, 1, 1), (u, v, 2, 2), (s, w, 2, 3), (w, u, 5, 2), (v, u, 8, 1), (v, w, 6, 3),\]
\[(w, u, 10, 2), (v, t, 3, 7), (u, t, 12, 1)\}\]
\[ E^{\text{arr}} = \{(s, u, 1, 1), (u, v, 2, 2), (s, w, 2, 3), (w, u, 5, 2), (v, u, 8, 1), (v, w, 6, 3), (w, u, 10, 2), (v, t, 3, 7), (u, t, 12, 1)\} \]

\[ \beta_w = 3 \]

\[ \beta_v = 3 \]

\[ \beta_u = 3 \]

\[ E^{\text{dep}} : (v, t, 3, 7), (v, w, 6, 3), (v, u, 8, 1) , \quad \rho_v = 0 \]
\[ E^{arr} = \{(s, u, 1, 1), (u, v, 2, 2), (s, w, 2, 3), (w, u, 5, 2), (v, u, 8, 1), (v, w, 6, 3), (w, u, 10, 2), (v, t, 3, 7), (u, t, 12, 1)\} \]
$E^{arr} = \{(s, u, 1, 1), (u, v, 2, 2), (s, w, 2, 3), (w, u, 5, 2), (v, u, 8, 1), (v, w, 6, 3),
(w, u, 10, 2), (v, t, 3, 7), (u, t, 12, 1)\}$

$\beta_w = 3$  \hspace{1cm} $\beta_v = 3$

$E^\text{dep} : (v, t, 3, 7), (v, w, 6, 3), (v, u, 8, 1)$, \hspace{5mm} \underbrace{P_v = 0}$
\[ E^{arr} = \{(s, u, 1, 1), (u, v, 2, 2), (s, w, 2, 3), (w, u, 5, 2), (v, u, 8, 1), (v, w, 6, 3), (w, u, 10, 2), (v, t, 3, 7), (u, t, 12, 1)\} \]

\[ \beta_w = 3 \]

\[ \beta_v = 3 \]

\[ \beta_u = 3 \]

\[ E^{dep} : (v, t, 3, 7), (v, w, 6, 3), (v, u, 8, 1) \]
\[ E^{arr} = \{(s, u, 1, 1), (u, v, 2, 2), (s, w, 2, 3), (w, u, 5, 2), (v, u, 8, 1), (v, w, 6, 3), \\
(w, u, 10, 2), (v, t, 3, 7), (u, t, 12, 1)\} \]
\[ E^{arr} = \{(s, u, 1, 1), (u, v, 2, 2), (s, w, 2, 3), (w, u, 5, 2), (v, u, 8, 1), (v, w, 6, 3),
\]
\[ (w, u, 10, 2), (v, t, 3, 7), (u, t, 12, 1)\}\]
\[ E^{arr} = \{(s, u, 1, 1), (u, v, 2, 2), (s, w, 2, 3), (w, u, 5, 2), (v, u, 8, 1), (v, w, 6, 3), (w, u, 10, 2), (v, t, 3, 7), (u, t, 12, 1)\} \]
$$E^{arr} = \{(s, u, 1, 1), (u, v, 2, 2), (s, w, 2, 3), (w, u, 5, 2), (v, u, 8, 1), (v, w, 6, 3),
(w, u, 10, 2), (v, t, 3, 7), (u, t, 12, 1)\}$$

\[\beta_w = 3\]
\[\beta_v = 3\]
\[\beta_u = 3\]
\[ E^{arr} = \{(s, u, 1, 1), (u, v, 2, 2), (s, w, 2, 3), (w, u, 5, 2), (v, u, 8, 1), (v, w, 6, 3),
(w, u, 10, 2), (v, t, 3, 7), (u, t, 12, 1)\} \]

\[ E^{dep} : (u, v, 2, 2), (u, t, 12, 1), P_v = 1 \]
\[ E^{arr} = \{(s, u, 1, 1), (u, v, 2, 2), (s, w, 2, 3), (w, u, 5, 2), (v, u, 8, 1), (v, w, 6, 3),
(w, u, 10, 2), (v, t, 3, 7), (u, t, 12, 1)\} \]
\[ E^{arr} = \{(s, u, 1, 1), (u, v, 2, 2), (s, w, 2, 3), (w, u, 5, 2), (v, u, 8, 1), (v, w, 6, 3),
(w, u, 10, 2), (v, t, 3, 7), (u, t, 12, 1)\} \]
\[ E^{arr} = \{(s, u, 1, 1), (u, v, 2, 2), (s, w, 2, 3), (w, u, 5, 2), (v, u, 8, 1), (v, w, 6, 3),
(w, u, 10, 2), (v, t, 3, 7), (u, t, 12, 1)\} \]
$E^{arr} = \{(s, u, 1, 1), (u, v, 2, 2), (s, w, 2, 3), (w, u, 5, 2), (v, u, 8, 1), (v, w, 6, 3),
(w, u, 10, 2), (v, t, 3, 7), (u, t, 12, 1)\}$

\[\begin{align*}
\beta_w &= 3 \\
\beta_v &= 3 \\
\beta_u &= 3
\end{align*}\]

\text{NOT REACHABLE}
\[ E^{arr} = \{(s, u, 1, 1), (u, v, 2, 2), (s, w, 2, 3), (w, u, 5, 2), (v, u, 8, 1), (v, w, 6, 3),
(w, u, 10, 2), (v, t, 3, 7), (u, t, 12, 1)\} \]
\[ E^{arr} = \{(s, u, 1, 1), (u, v, 2, 2), (s, w, 2, 3), (w, u, 5, 2), (v, u, 8, 1), (v, w, 6, 3), (w, u, 10, 2), (v, t, 3, 7), (u, t, 12, 1)\} \]
\[ E^{\text{arr}} = \{(s, u, 1, 1), (u, v, 2, 2), (s, w, 2, 3), (w, u, 5, 2), (v, u, 8, 1), (v, w, 6, 3),
(w, u, 10, 2), (v, t, 3, 7), (u, t, 12, 1)\} \]

\[ \beta_w = 3 \]

\[ \beta_u = 3 \]

\[ \beta_v = 3 \]

\[ \beta_{w_{\text{dep}}} = 3 \]

\[ P_{w_{\text{dep}}} = 2 \]
$E^{arr} = \{(s, u, 1, 1), (u, v, 2, 2), (s, w, 2, 3), (w, u, 5, 2), (v, u, 8, 1), (v, w, 6, 3), (w, u, 10, 2), (v, t, 3, 7), (u, t, 12, 1)\}$
\[ E^{arr} = \{(s, u, 1, 1), (u, v, 2, 2), (s, w, 2, 3), (w, u, 5, 2), (v, u, 8, 1), (v, w, 6, 3), \\
(w, u, 10, 2), (v, t, 3, 7), (u, t, 12, 1)\} \]
\[ E^{arr} = \{(s, u, 1, 1), (u, v, 2, 2), (s, w, 2, 3), (w, u, 5, 2), (v, u, 8, 1), (v, w, 6, 3),
(w, u, 10, 2), (v, t, 3, 7), (u, t, 12, 1)\} \]
\[ E^{arr} = \{(s, u, 1, 1), (u, v, 2, 2), (s, w, 2, 3), (w, u, 5, 2), (v, u, 8, 1), (v, w, 6, 3),
(w, u, 10, 2), (v, t, 3, 7), (u, t, 12, 1)\} \]
\[ E^{\text{arr}} = \{(s, u, 1, 1), (u, v, 2, 2), (s, w, 2, 3), (w, u, 5, 2), (v, u, 8, 1), (v, w, 6, 3),
(w, u, 10, 2), (v, t, 3, 7), (u, t, 12, 1)\} \]
\[ E^{\text{arr}} = \{ (s, u, 1, 1), (u, v, 2, 2), (s, w, 2, 3), (w, u, 5, 2), (v, u, 8, 1), (v, w, 6, 3), (w, u, 10, 2), (v, t, 3, 7), (u, t, 12, 1) \} \]
$E^{arr} = \{(s, u, 1, 1), (u, v, 2, 2), (s, w, 2, 3), (w, u, 5, 2), (v, u, 8, 1), (v, w, 6, 3), (w, u, 10, 2), (v, t, 3, 7), (u, t, 12, 1)\}$
$E^{arr} = \{(s, u, 1, 1), (u, v, 2, 2), (s, w, 2, 3), (w, u, 5, 2), (v, u, 8, 1), (v, w, 6, 3), (w, u, 10, 2), (v, t, 3, 7), (u, t, 12, 1)\}$

Extract the temporal walk following parent pointers
\[ E^{arr} = \{(s, u, 1, 1), (u, v, 2, 2), (s, w, 2, 3), (w, u, 5, 2), (v, u, 8, 1), (v, w, 6, 3),
(w, u, 10, 2), (v, t, 3, 7), (u, t, 12, 1)\} \]
$E^{arr} = \{(s, u, 1, 1), (u, v, 2, 2), (s, w, 2, 3), (w, u, 5, 2), (v, u, 8, 1), (v, w, 6, 3),
(w, u, 10, 2), (v, t, 3, 7), (u, t, 12, 1)\}$

Extract the temporal walk following parent pointers
\[ E^{arr} = \{(s, u, 1, 1), (u, v, 2, 2), (s, w, 2, 3), (w, u, 5, 2), (v, u, 8, 1), \ldots\} \]

\[ E^{dep} = \{(s, u, 1, 1), (s, w, 2, 3)(u, v, 2, 2), (v, t, 3, 7), (w, u, 5, 2), \ldots\} \]
Time-expanded representation
Doubly-sorted representation

For simplicity we will speak about the case in which all edges have strictly positive travel time ($\lambda > 0$).

**Definition**

A *doubly-sorted representation* of a temporal graph $(V, E, \beta)$ consists in two lists $E^{arr}$ and $E^{dep}$ where $E^{arr}$ contains all temporal edges in $E$ sorted by non-decreasing arrival time, and $E^{dep}$ by non-decreasing departure time.
Doubly-sorted representation

For simplicity we will speak about the case in which all edges have strictly positive travel time ($\lambda > 0$).

**Definition**

A *doubly-sorted representation* of a temporal graph $(V, E, \beta)$ consists in two lists $E^{arr}$ and $E^{dep}$ where $E^{arr}$ contains all temporal edges in $E$ sorted by non-decreasing arrival time, and $E^{dep}$ by non-decreasing departure time.

**Weaker assumptions:**

- $E^{arr}$ is node-arrival sorted and walk-respecting
- $E^{dep}$ is node-departure sorted

They can be computed in linear time from a space-time representation.
The simple core idea of the algorithm we presented can be exploited to design more involved and efficient algorithms.
The simple core idea of the algorithm we presented can be exploited to design more involved and efficient algorithms. We used it to compute in temporal graphs subject to waiting constraints, in linear time and space:

- shortest duration temporal walks,
- fewest hops temporal walks,
- minimum waiting-time temporal walks,
- many other criteria.
Thank you for your attention