# Temporal Pathfinding in the Presence of Delays

Malte Renken

Algorithmics and Computational Complexity,
TU Berlin, Germany

July 2022

Joint work with Eugen Füchsle, Hendrik Molter, and Rolf Niedermeier.

# Motivation


(c) CC-BY Frank Kovalchek


Mannheim-Friedrich 11
Gernsheim 17
Köln Hbf 7
Berlin Hbf 9
Passau Hbf 6
Siegen 16
Saarbrücken Hbf 20
Fulda 8
Bruxelles-Midi 19
Hanau Hbf 5
r DB-Zugverkehr beeinträchtigt. Bitte
d informieren Sie sich auch im Internet


Domestic Departures
Domestic Flights
Flights Cancelled
(c) CC-BY-NC-SA flickr/dcmaster

# Main Question

How to plan routes in temporal networks subject to delays?

# Model

We assume that we have:

- a temporal graph $\mathcal{G}$, which is a directed multigraph where every edge $e$ has
  - a departure time $\mathrm{t}(e)$,
  - a duration $\lambda(e)$,
  - an arrival time $\mathrm{t}(e) + \lambda(e)$;
- a start vertex $s$ and destination $d$;
- an upper bound $\delta$ on the delay of any single edge;
- an upper bound $x \in \mathbb{N}$ on the number of delayed edges.

Remarks:

- parallel edges are delayed independently;
- $\lambda(e)$ already includes transfer time
  $\rightsquigarrow$ you can arrive at 3 o'clock and depart again at 3 o'clock.

# Problem variants

## Input:

- Temporal graph $\mathcal{G} = (V, E, \mathrm{t}, \lambda)$
- Start $s \in V$
- Number of delays $x$
- Destination $d \in V$
- Delay time $\delta$

## Question:

Can we get from $s$ to $d$ even if an adversary chooses which edges to delay?

## When do we know which edges are delayed?

- **Delay-Robust Connection:** We are told all the delays before we pick a route.
- **Delayed-Routing Game:** We learn the delays as they occur.
- **Delay-Robust Route:** We have to fix our route before knowing any delays.

# Results overview

$x = \#$ delays

- **Delay-Robust Connection:** We are told all the delays before we pick a route.
  - Solvable in $\mathcal{O}(|V| \cdot |E|)$ time by a flow-based algorithm.
- **Delayed-Routing Game:** We learn the delays as they occur.
  - Solvable in $\mathcal{O}(|V| \cdot |E| \cdot x)$ time by dynamic programming.
- **Delayed-Routing Path Game:** We learn the delays as they occur; we may not revisit earlier vertices.
  - PSPACE-complete.
- **Delay-Robust Route:** We have to fix our route before knowing any delays.
  - Strongly NP-complete;
  - Solvable in $\mathcal{O}(|E|^{x+1}x^2)$.

# Delays

What does it mean for an edge $e$ to be delayed?

Option A: Train stuck in between stations: duration (and arrival) increase.

Option B: Train stuck at the station: departure (and arrival) increase
$\rightsquigarrow$ you might be lucky and still catch it even though you are late.

We care about worst case scenarios $\rightsquigarrow$ may assume A.

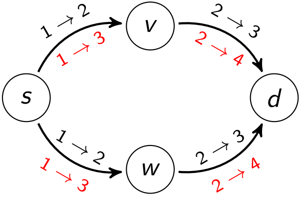
For the same reason: All delays will use the maximum amount $\delta$.

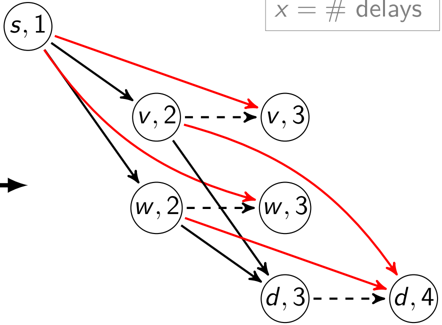
# Delay-Robust Connection

We are told all the delays before we pick a route.

**Idea:** Reduce to a flow problem.

$x = \#$ delays




- Red and dashed edges have capacity $\infty$.
- Solid black edges have capacity 1.

### Lemma

YES iff max flow from $(s, 1)$ to $(d, 4)$ is larger than $x$.

# Delayed-Routing (Path) Game

We learn the delays as they occur.

**Rules for Delayed-Routing Game:**

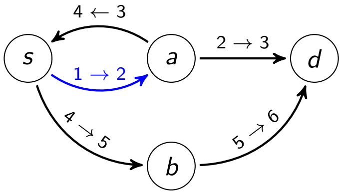
- **Traveler:** choose next edge to traverse.
- **Adversary:** choose whether to delay that edge or not.

Traveler starts at $s$ at time 1 and wins if and only if they reach $d$.

Extra rule for **Delayed-Routing Path Game**: Traveler loses if they revisit a vertex.

**Example:**

- Number of delays $x = 1$.
- Delay time $\delta = 1$.



# Delayed-Routing (Path) Game

## Algorithms

**Delayed-Routing Game:**

Traveler's turn can be described with:

- Current vertex
- Current time step
- Remaining delays

**Delayed-Routing Path Game:**

Traveler's turn can be described with:

- Current vertex
- Current time step
- Remaining delays
- Already visited vertices.

**Strategy for the traveler:**

For each edge the traveler could possibly take next, test if there is a winning strategy in both of these cases:

- edge is delayed, number of delays is reduced by 1,
- edge is not delayed, number of delays remains unchanged.

Dynamic programming $\Rightarrow$ polynomial time

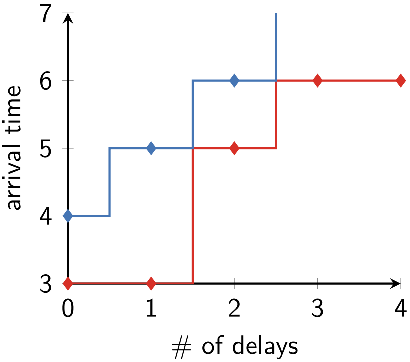
Depth-first search $\Rightarrow$ polynomial space

# Delay-Robust Route

We have to fix our route before knowing any delays.

$x = \#$ delays

Given a route $s = v_0,\ v_1,\ \ldots,\ v_k = d$,
for every prefix $v_0, v_1, \ldots, v_i$ we can draw a delay profile:





Can efficiently compute profile for $v_{i+1}$ from profile of $v_i$.
$\rightsquigarrow$ **Delay-Robust Route** $\in$ NP

Can bound number of possible profiles by $|E|^x$
$\rightsquigarrow$ $\mathcal{O}(|E|^{x+1}x^2)$ algorithm

# Summary

In temporal networks, computing routes that cope with delays can be done

- efficiently, if you know the delays up front;
- pretty efficiently, if you don't know them, but can adjust your route on the go (but only if you can go in cycles);
- efficiently only in special cases if you need to fix you route beforehand.

# Thank you!