Algorithms for solving time-dependent routing problems with exponential output size

Martin Skutella, Miriam Schlöter

TU Berlin

DFG-SPP 1736 "Algorithms for Big Data"
Frankfurt a. M.

October 1st, 2014

Overview of Project

This project consists of two main parts:

- i *Time-dependent* routing problems induce dramatic explosion of data.

  Q: To what extend can the curse of big data be broken by using algorithmic and structural insights?

On a more fundamental level:

- ii Some optimization problems seem to require exponential output size (e. g., parametric linear programming, multicriteria optimization, . . .).

  Q: How to classify/substantiate the difficulty of such problems?

  Q: Are there 'good reasons' for related algorithms to have exponential running time?

Adding Temporal Dimension

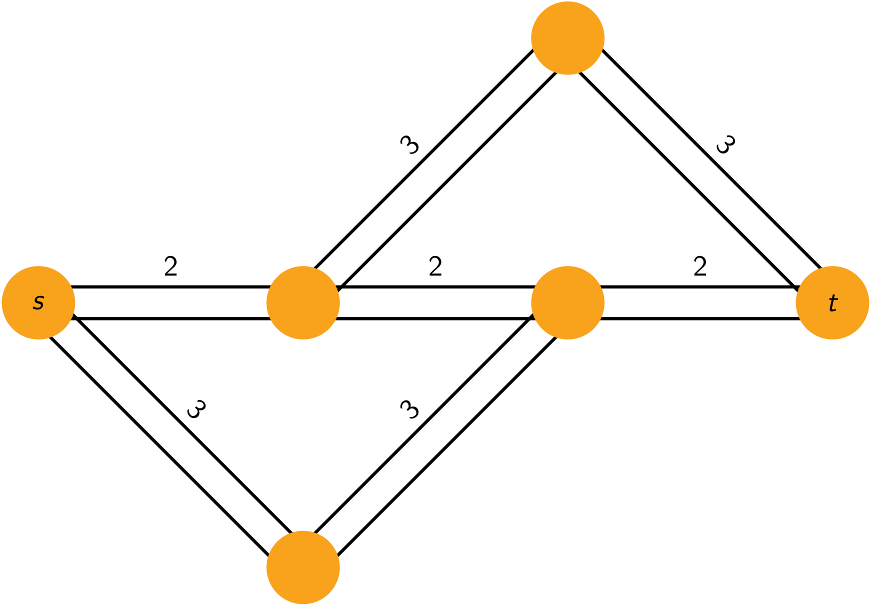
Classical network routing deals with steady state flows, . . .

. . . in many applications, however, time plays a vital role!

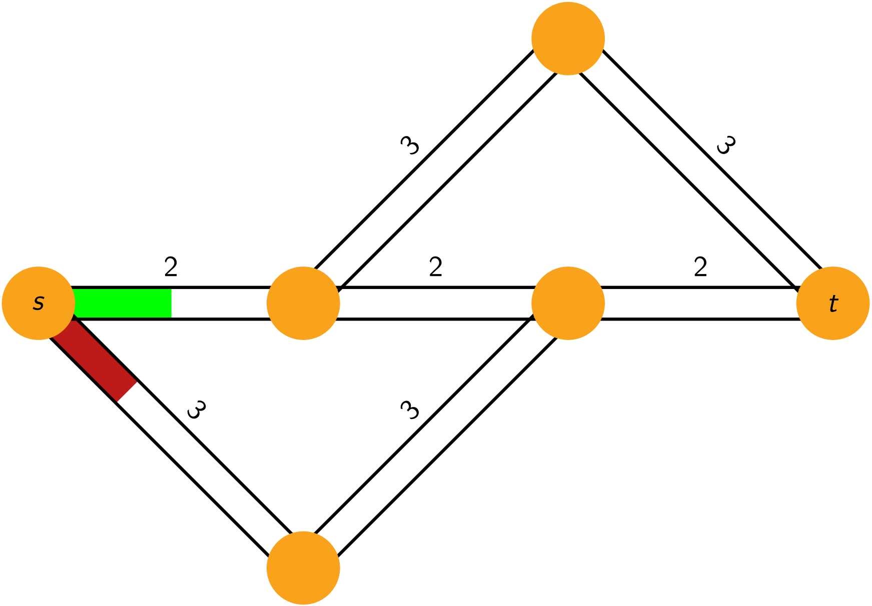
Examples.

- Flow variation over time due to seasonal altering demands, supplies, and/or arc capacities.
- Flow travels only at a certain pace through the network, that is, there are transit times on the arcs.

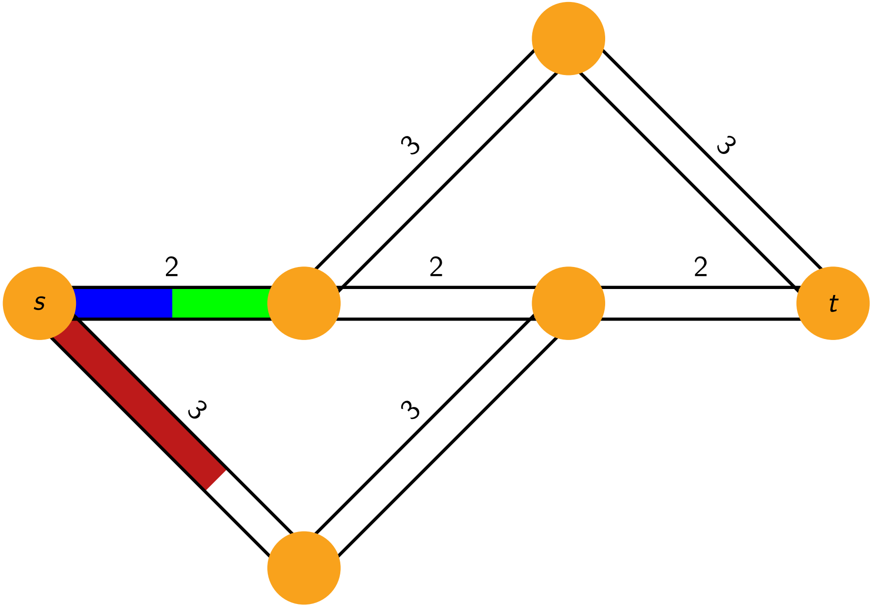
Time-Dependent Routing: Example



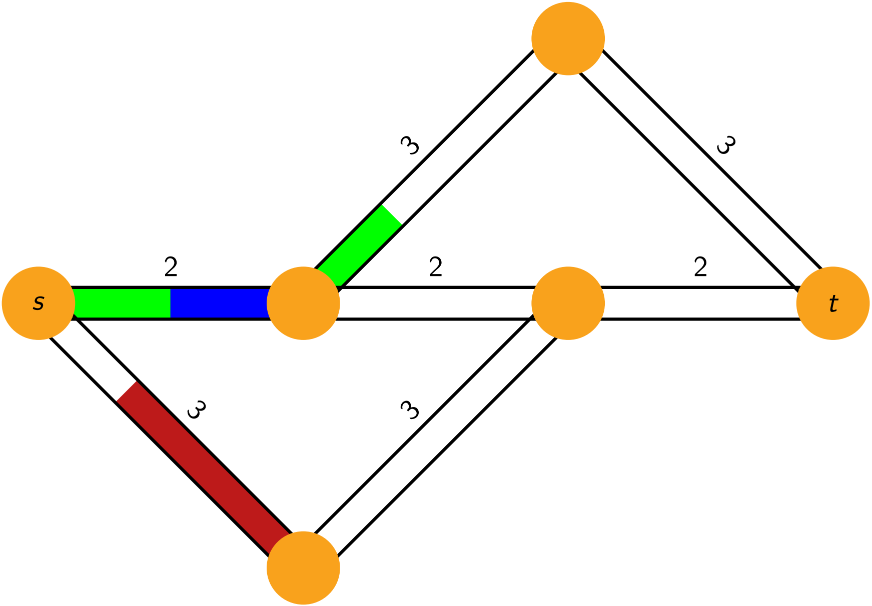
Time-Dependent Routing: Example



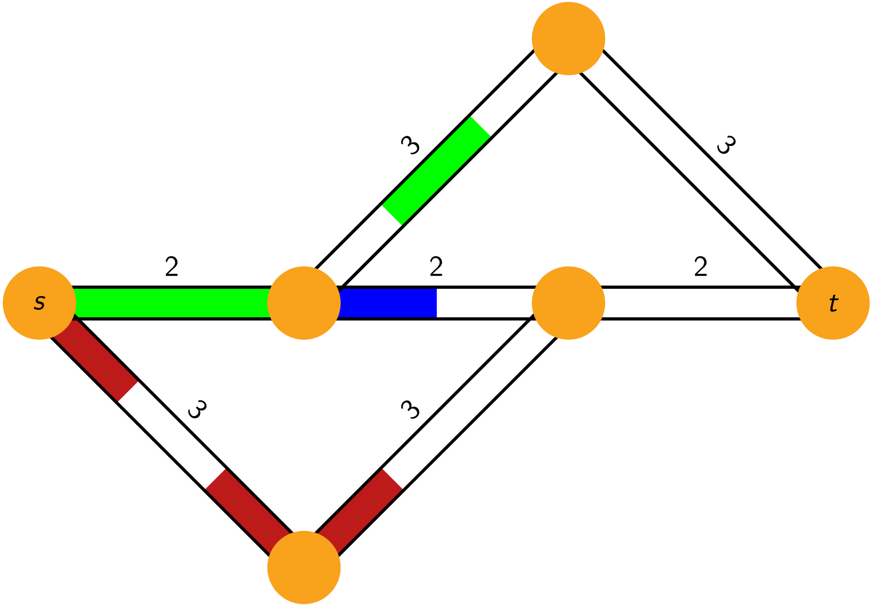
Time-Dependent Routing: Example



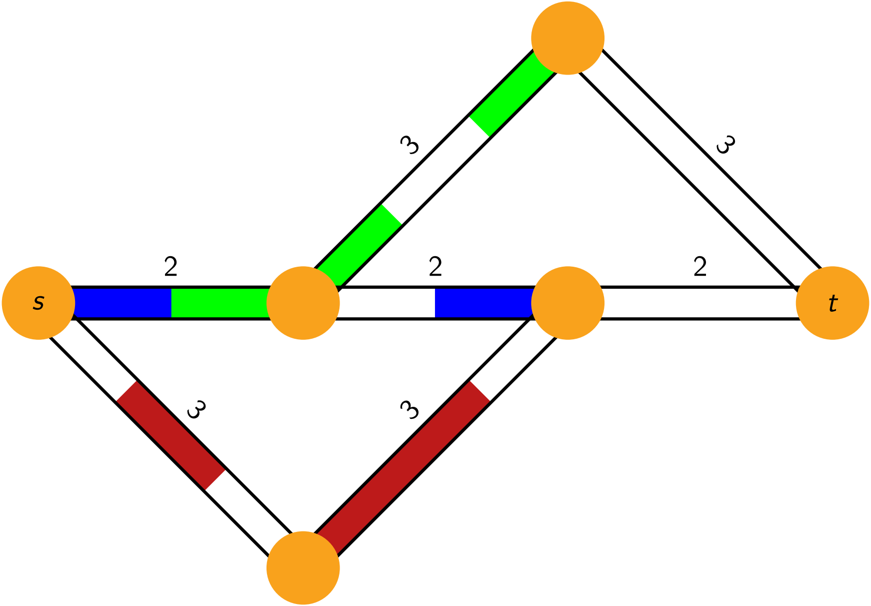
Time-Dependent Routing: Example



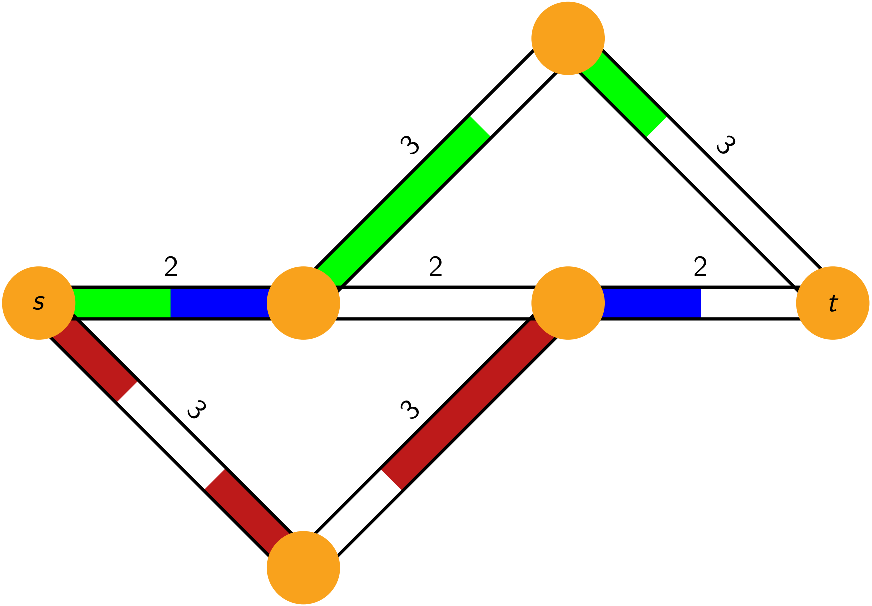
Time-Dependent Routing: Example



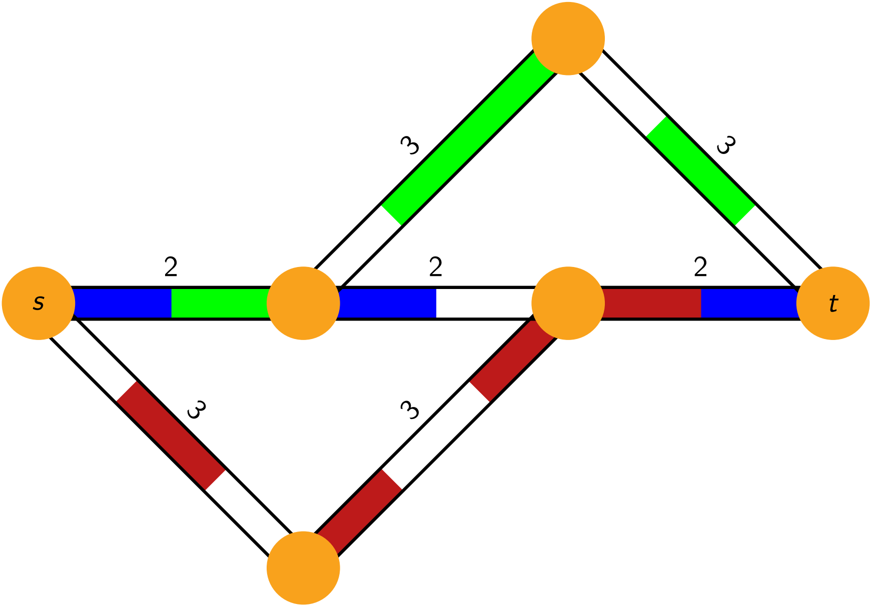
Time-Dependent Routing: Example



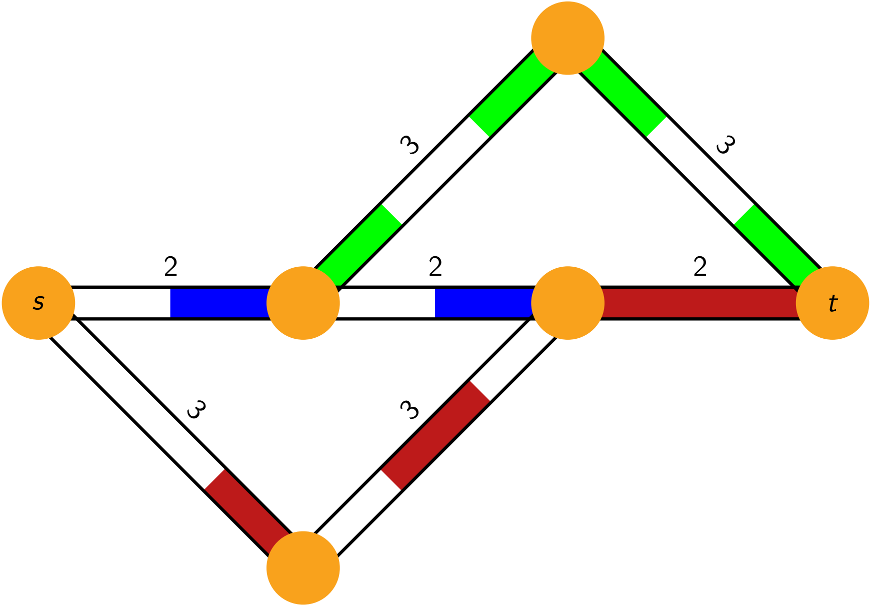
Time-Dependent Routing: Example



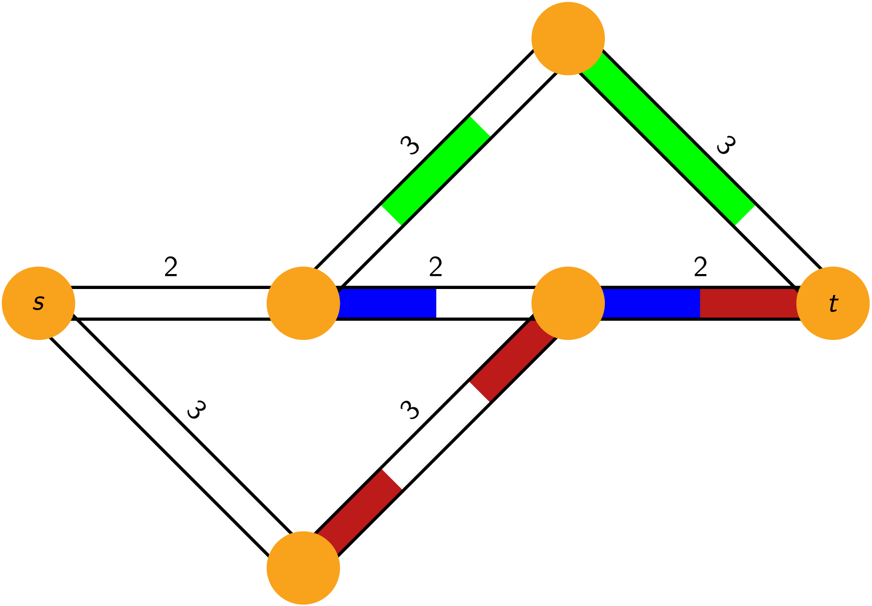
Time-Dependent Routing: Example



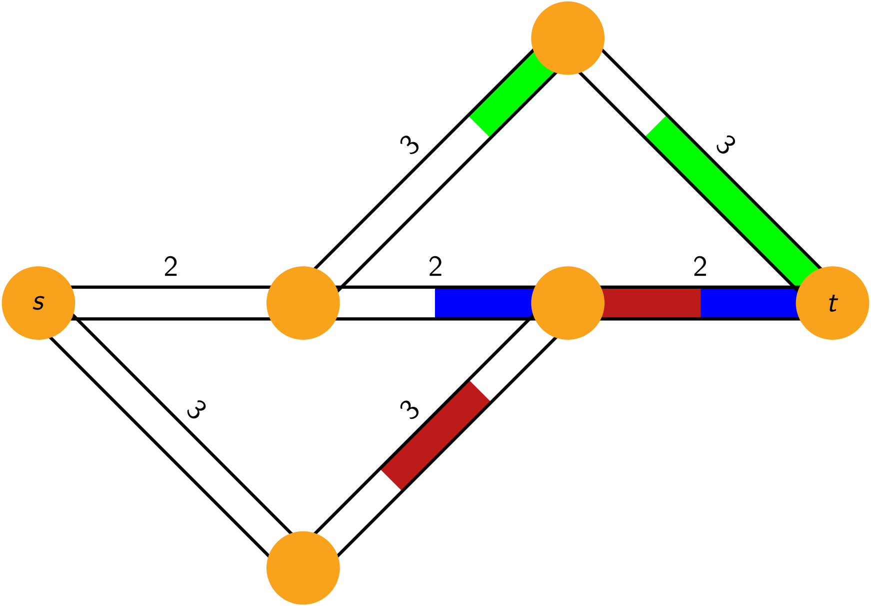
Time-Dependent Routing: Example



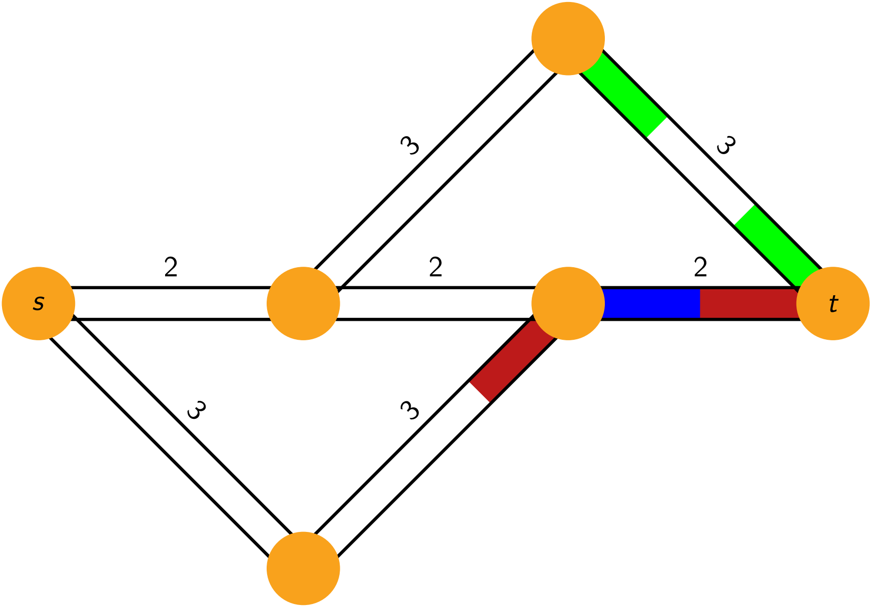
Time-Dependent Routing: Example



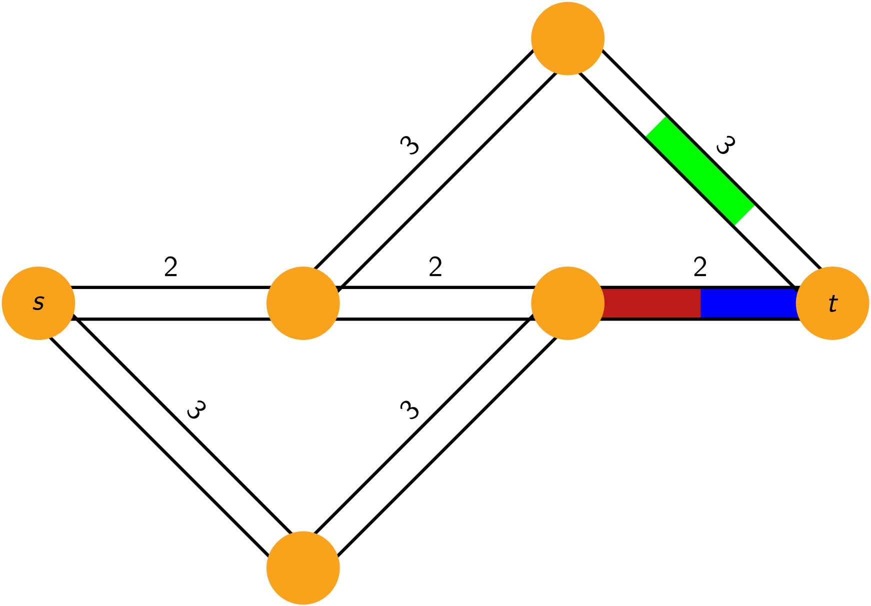
Time-Dependent Routing: Example



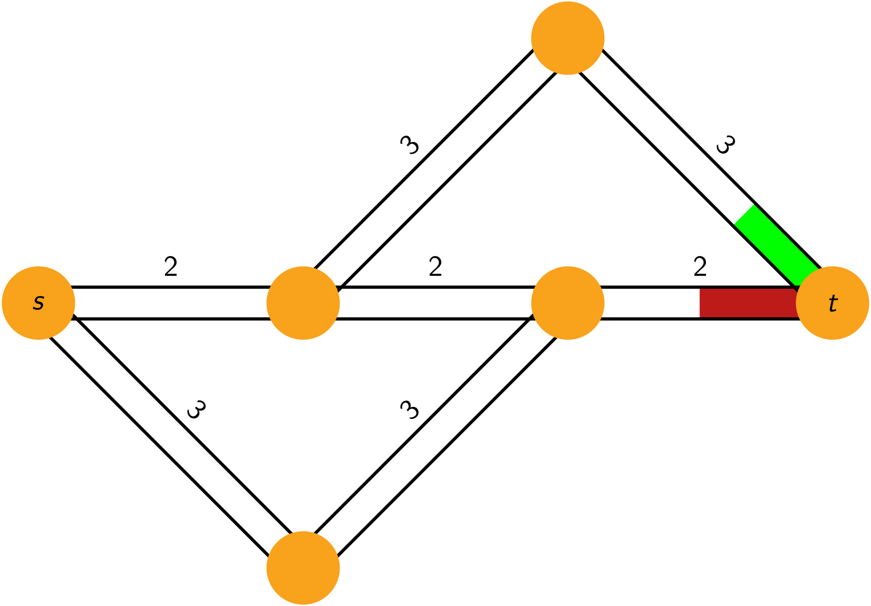
Time-Dependent Routing: Example



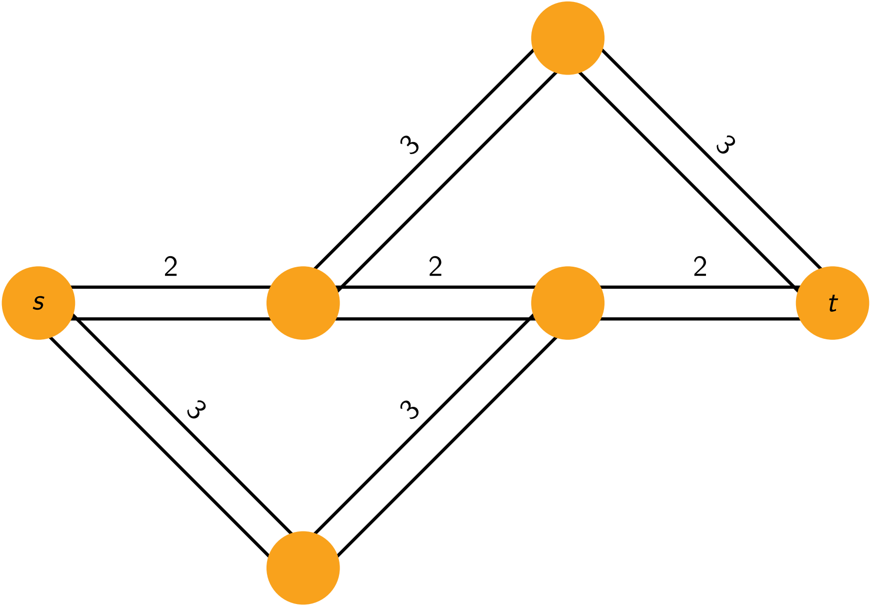
Time-Dependent Routing: Example



Time-Dependent Routing: Example

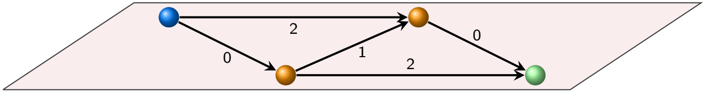


Time-Dependent Routing: Example



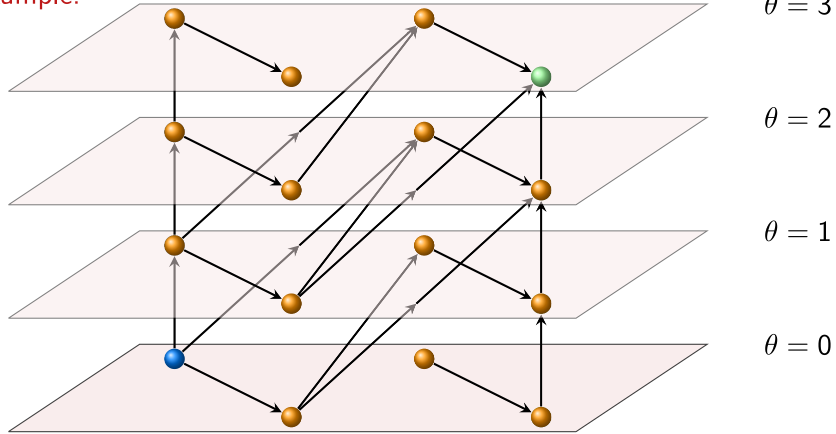
Time Expanded Networks

Example:



Time Expanded Networks

Example:



Big Data and Time-Dependent Routing Problems

- Todays routing networks can consist of billions of nodes and arcs (e. g., `www.openstreetmap.org`).
- Temporal dimension induces further dramatic explosion of data sizes (blow-up by number of time-layers T).
- In particular, time-expanded networks are exponential in input-size.

Working Hypothesis.

An algorithm for time-dependent routing is efficient if its running time is polynomial in the size of the underlying (static) network.

Notice: Such running times are usually sublinear (or even logarithmic) in the size of the time-expanded network.

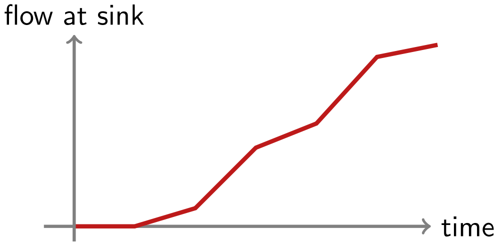
Different Types of Possible Results and Open Problems

- Ford & Fulkerson 1958:
  Problem: max-flow from source s to sink t within T time units
  Result: Static min-cost flow yields opt. solution in poly-time.
- Hall, Hippler & Sk. 2007:
  Problem: multi-commodity flow over time
  Result: NP-hard even if T is polynomial in size of static network
- Fleischer & Sk. 2007:
  Problem: multi-commodity flow over time etc.
  Result: poly-time approximation via condensed time-expansion
- Zadeh 1973:
  Problem: Earliest arrival s-t-flow
  Result: Optimum solution seems to require exponential size???

Earliest Arrival Flows

Earliest arrival flows capture the essence of evacuation planning.

Definition. An earliest arrival s-t-flow maximizes the amount of flow having arrived at sink t for all points in time simultaneously.



- Gale (1959): An earliest arrival flow does always exists.
- Jarvis & Ratliff (1982): Earliest arrival flows minimize average arrival time at sink (and vice versa).

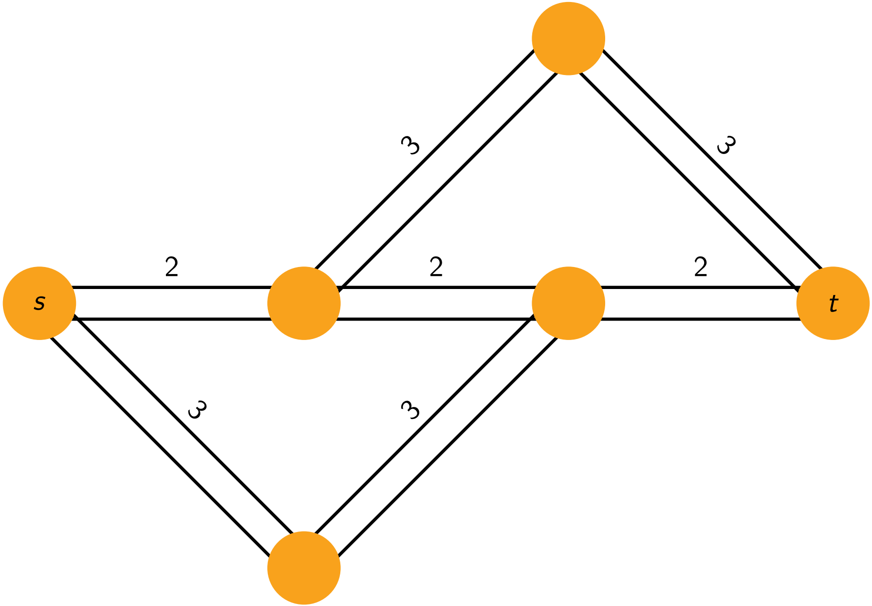
Computing Earliest Arrival Flows

Successive Shortest Path Algorithm (SSPA)

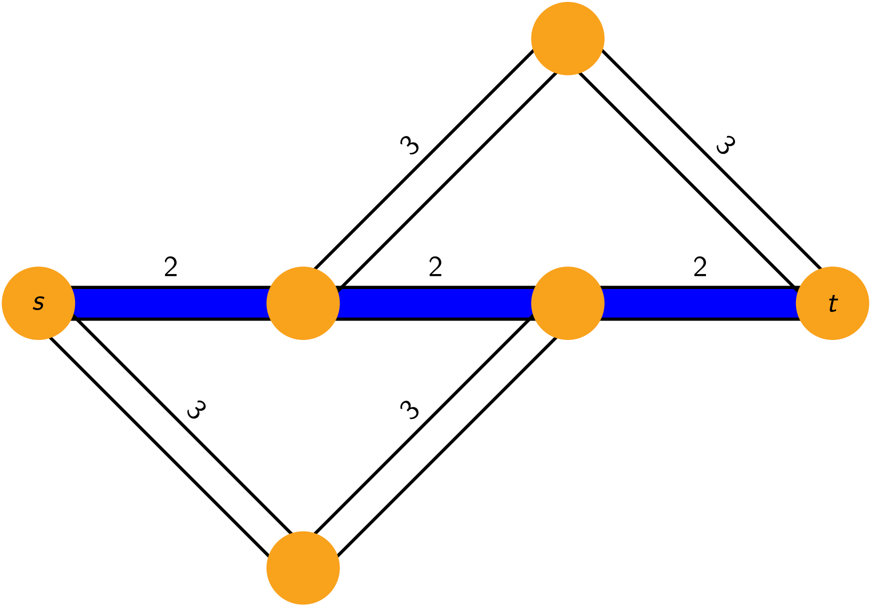
1. start with zero-flow $x \equiv 0$;
2. iteratively augment flow along shortest s-t-path in residual network;

Theorem. (Wilkinson 1971, Minieka 1973)
Sending flow along the paths chosen by SSPA yields an earliest arrival flow.

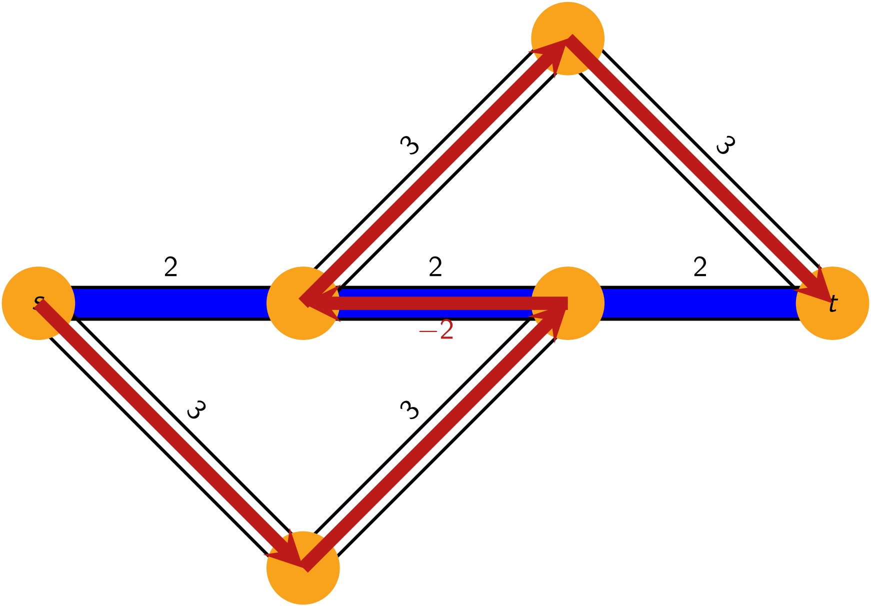
Earliest Arrival Flow: Example



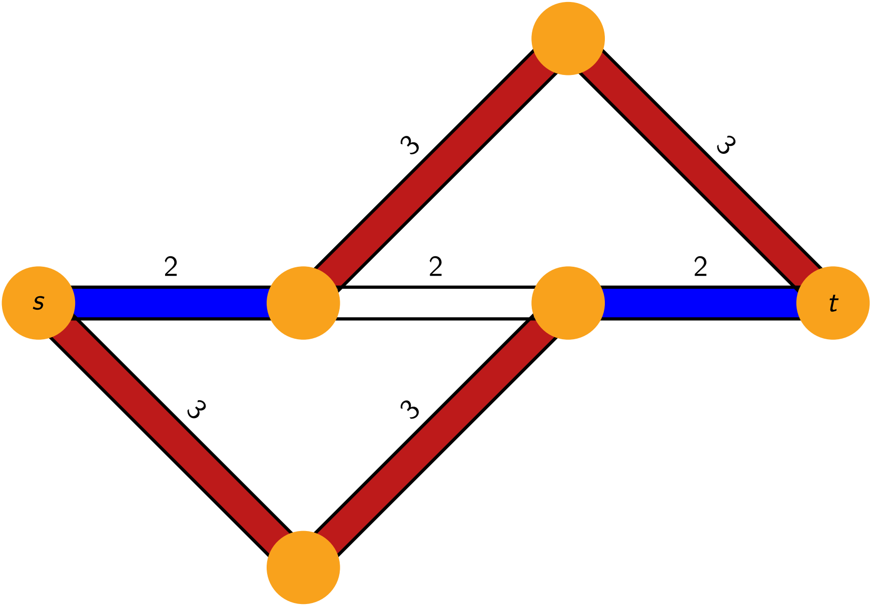
Earliest Arrival Flow: Example



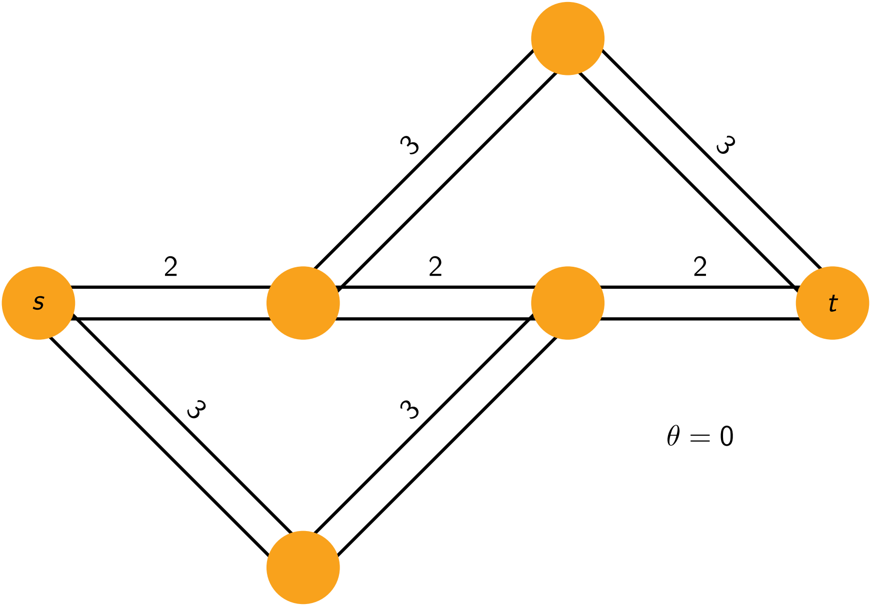
Earliest Arrival Flow: Example



Earliest Arrival Flow: Example

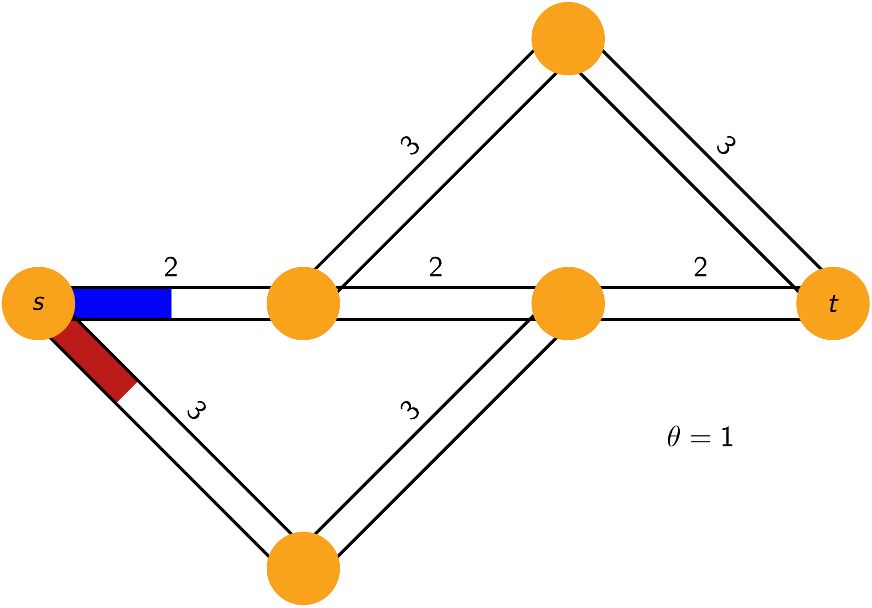


Earliest Arrival Flow: Example

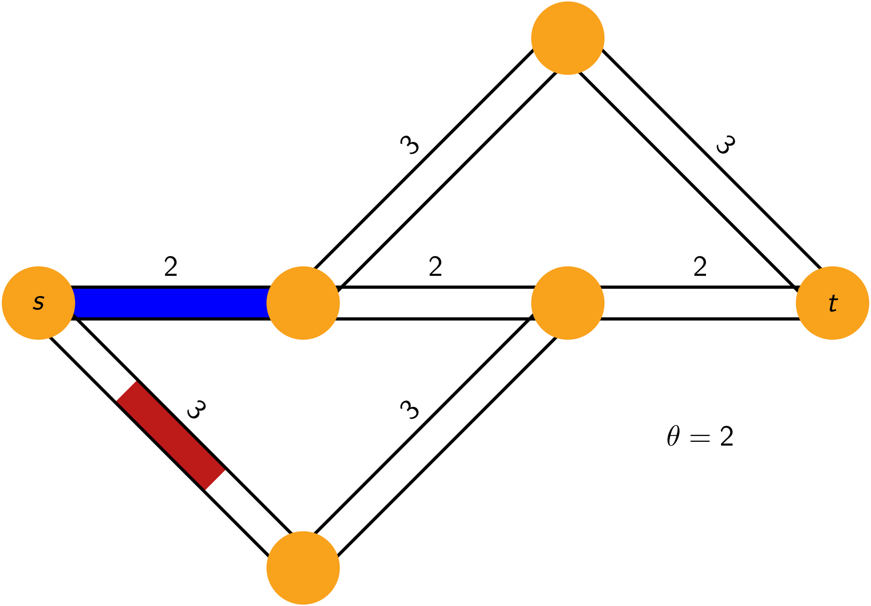


$\theta = 0$

Earliest Arrival Flow: Example

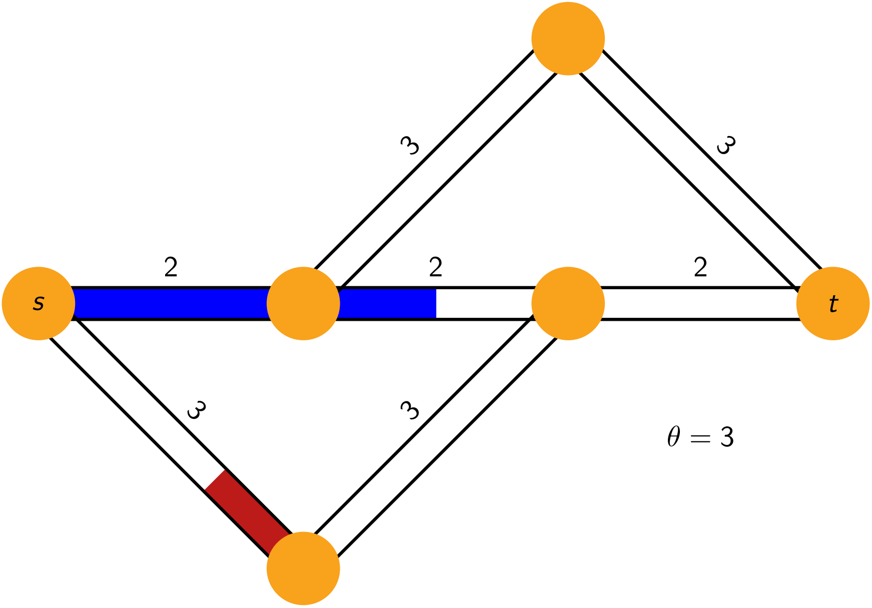


Earliest Arrival Flow: Example



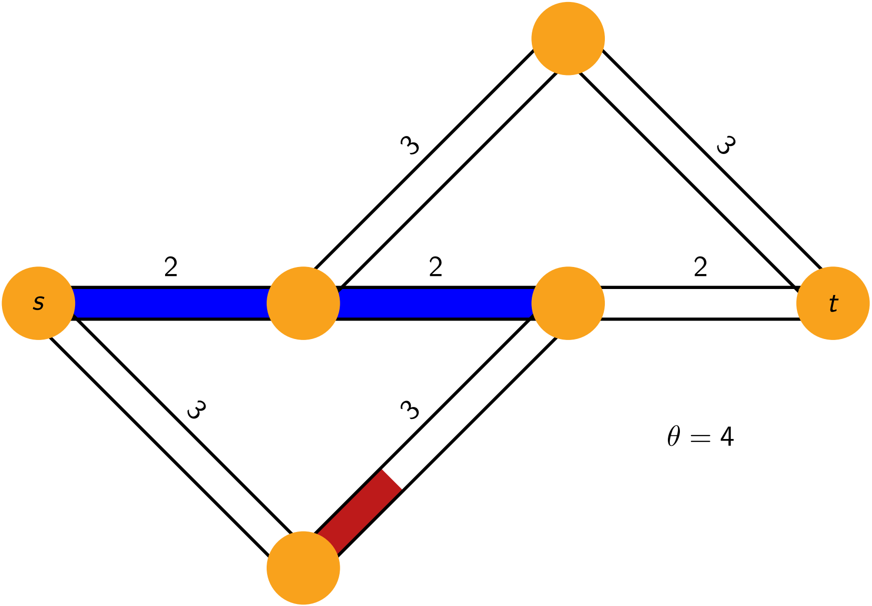
$\theta = 2$

Earliest Arrival Flow: Example

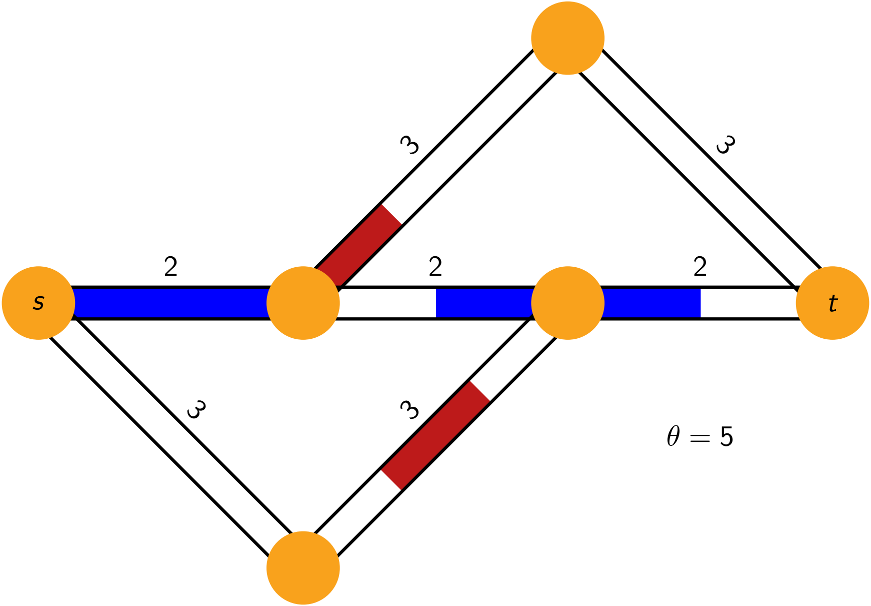


$\theta = 3$

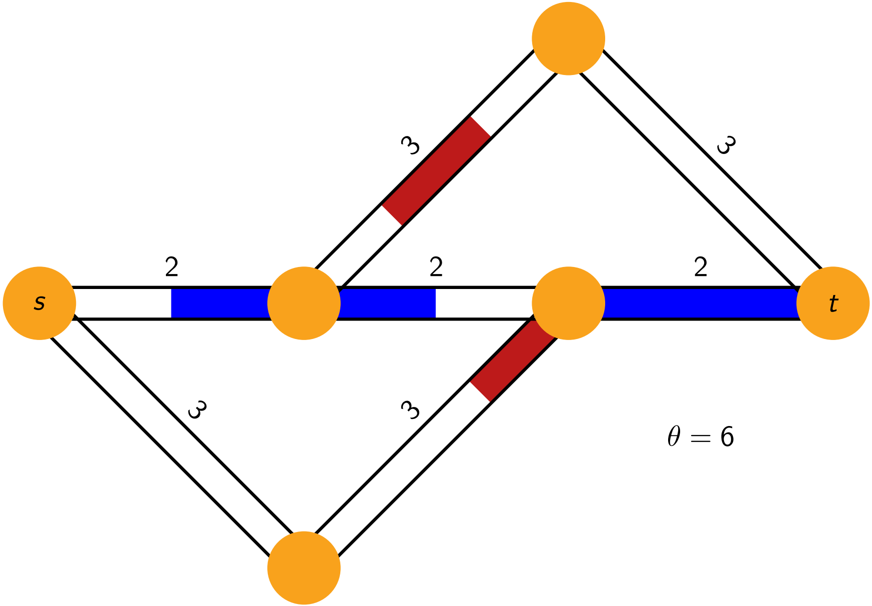
Earliest Arrival Flow: Example



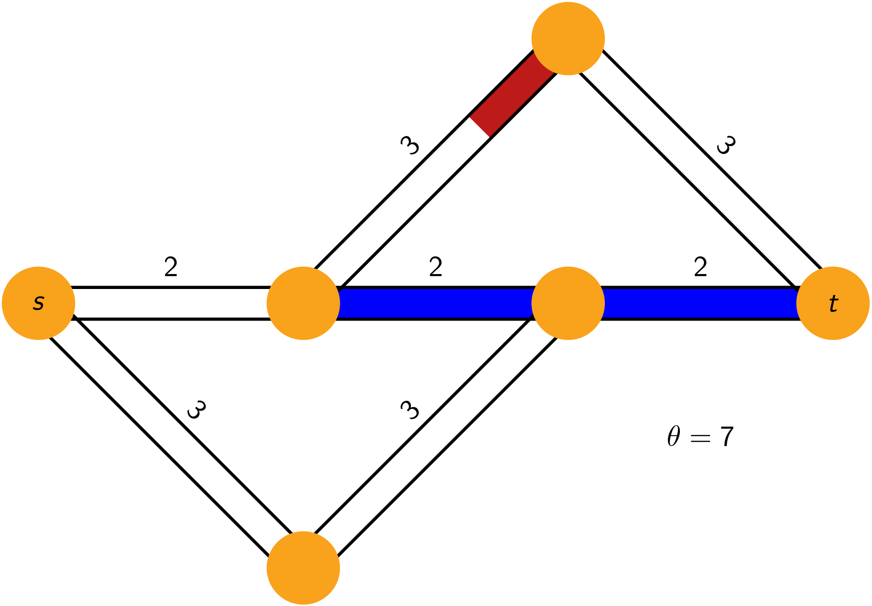
Earliest Arrival Flow: Example



Earliest Arrival Flow: Example

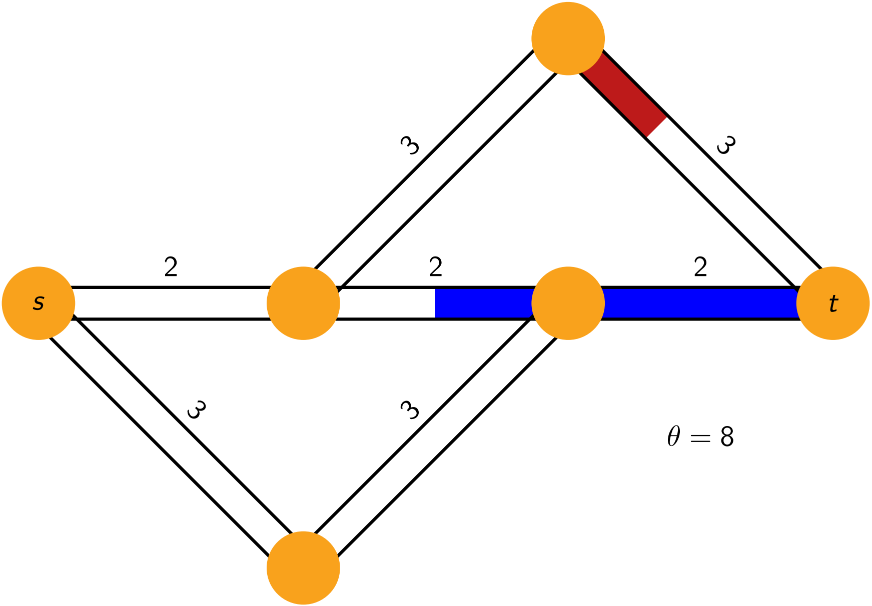


Earliest Arrival Flow: Example



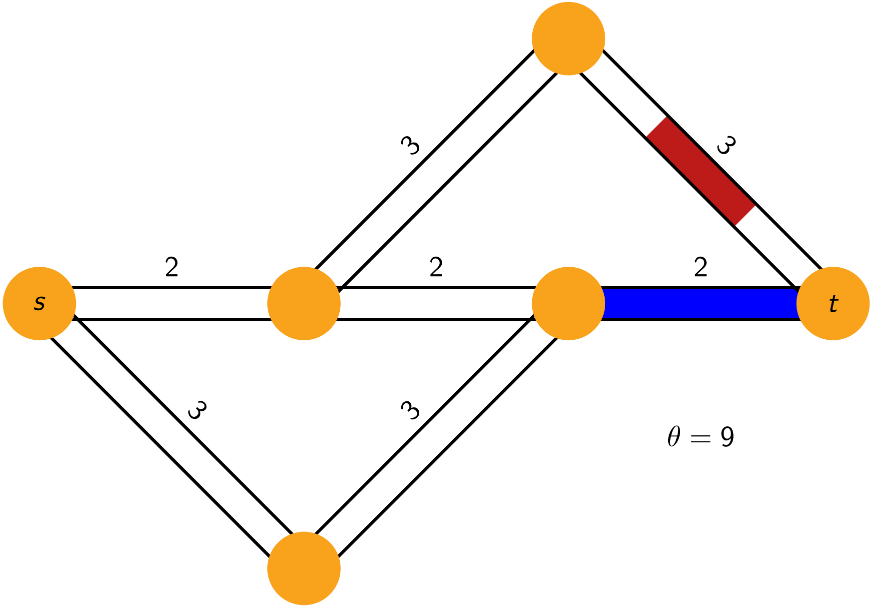
$\theta = 7$

Earliest Arrival Flow: Example



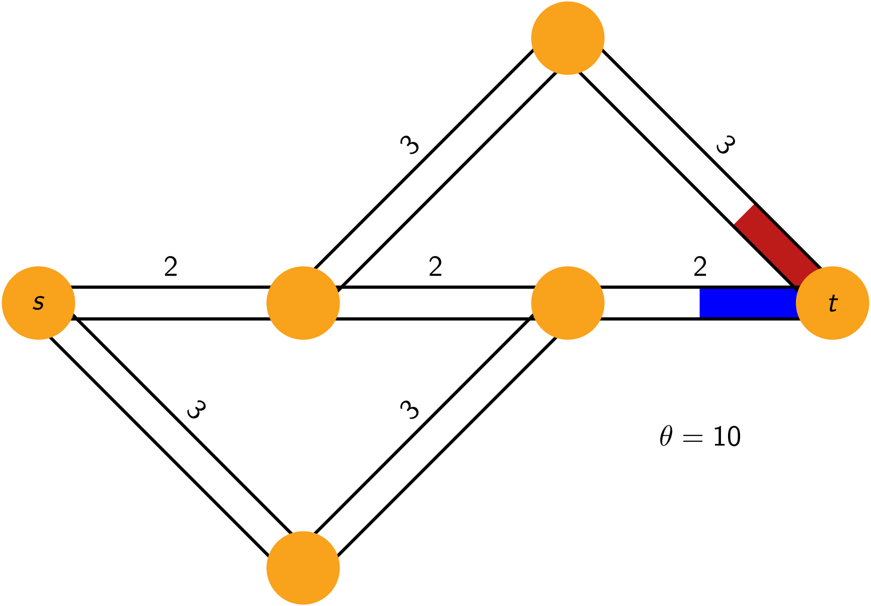
$\theta = 8$

Earliest Arrival Flow: Example



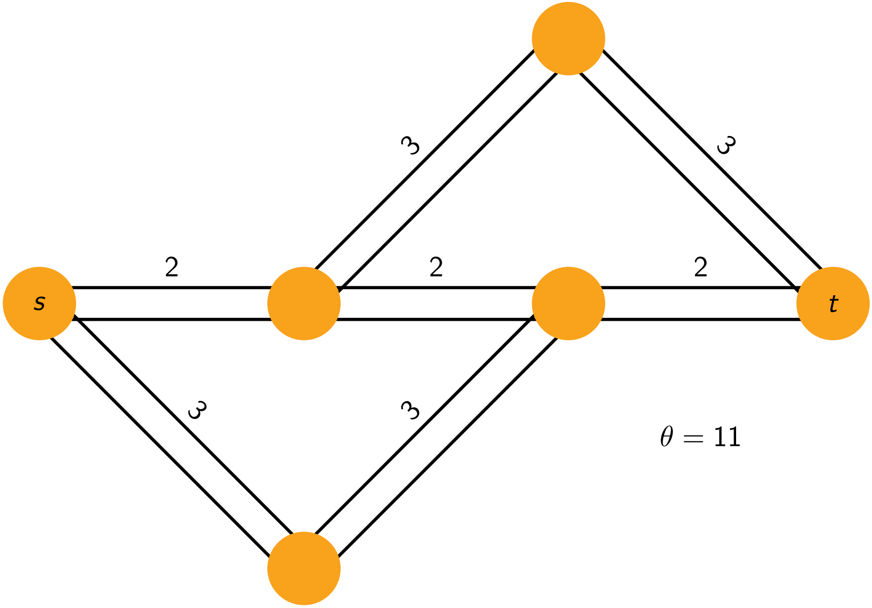
$\theta = 9$

Earliest Arrival Flow: Example



$\theta = 10$

Earliest Arrival Flow: Example



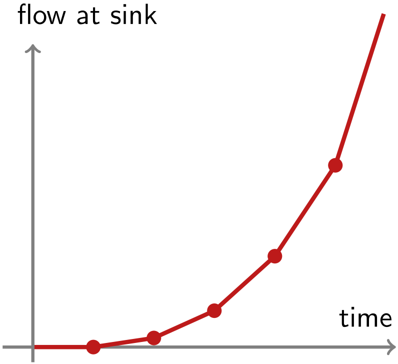
$\theta = 11$

Exponential Worst-Case Running Time and Output Size

Theorem. (Zadeh 1973)

SSPA takes exponentially many iterations in the worst-case.

Even worse, earliest arrival pattern has exponentially many breakpoints!



Questions:

- Does the earliest arrival flow problem have exponential output size?
- How difficult is it to compute an earliest arrival s-t-flow?

Problems Requiring Exponential Output?

Output size

The size of the output depends on the chosen/required encoding!

Examples.

- LPs with exponentially many variables — easy/NP-hard
- flow over time problems — easy/NP-hard
- earliest arrival flows — ???
- parametric linear programming — ???
- parametric min-cost flows — ???
- multi-criteria optimization problems — easy/NP-hard
- enumeration/counting problems — easy/#P-hard

Algorithms with Exponential Worst-Case Running Time

Examples.

- Successive Shortest Path Algorithm (SSPA)
- Simplex Method for solving LPs (standard variants)
- Algorithm that, on input $n \in \mathbb{N}$, counts from 1 to n

Questions.

– Why do these algorithms have poor worst-case behavior?

– Are the 'detours' just a waste of time?

– Or do they solve difficult problems on their way?

Some Partial Answers (Disser & Sk., SODA 2015)

Theorem.

The Simplex Algorithm and SSPA are 'NP-mighty'.

That is, these algorithms can solve NP-complete problems on their way!

More precisely:

- For SSPA, it is NP-complete to decide whether an arc of the given network is ever used.
- For the Simplex Algorithm (Dantzig's pivot rule), it is NP-complete to decide whether a variable of a given LP ever occurs in the basis.
- Result even holds for *Network* Simplex Algorithm.

Some Partial Answers (Disser & Sk., SODA 2015)

Further results and consequences:

Corollary. The earliest arrival flow problem is NP-hard to the extent that determining the minimum average arrival time is NP-hard.

Corollary. It is NP-complete to decide whether in the solution to a parametric LP a given variable will ever take a positive value.

Corollary. Determining # iterations of the (Network) Simplex Algorithm and the Successive Shortest Path Algorithm for a given input is NP-hard.

Corollary. Given a d-dimensional polytope P, determining the number of vertices of Ps projection onto a 2-dimensional subspace is NP-hard.

Related Recent/Subsequent Results

Adler, Papadimitriou & Rubinstein (2014).
For the Simplex Algorithm with an artificial pivot rule it is even PSPACE-complete to decide whether a given basis ever appears.

Fearnley & Savani (2014).
The Simplex Algorithm (with Dantzig's pivot rule) is PSPACE-mighty.

Conclusion

- Aim at better structural/algorithmic understanding of problems with exponential solution sizes.
- Primary example: Time-dependent routing problems
- How to classify/substantiate the difficulty of such problems?
- Are there 'good reasons' for related algorithms to have exponential running time?