Solving Evacuation Problems in Polynomial Space

Miriam Schlöter & Martin Skutella
TU Berlin

Flows Over Time & Lex-Max Flows Over Time

Like classical (static) network flows + **time component**:

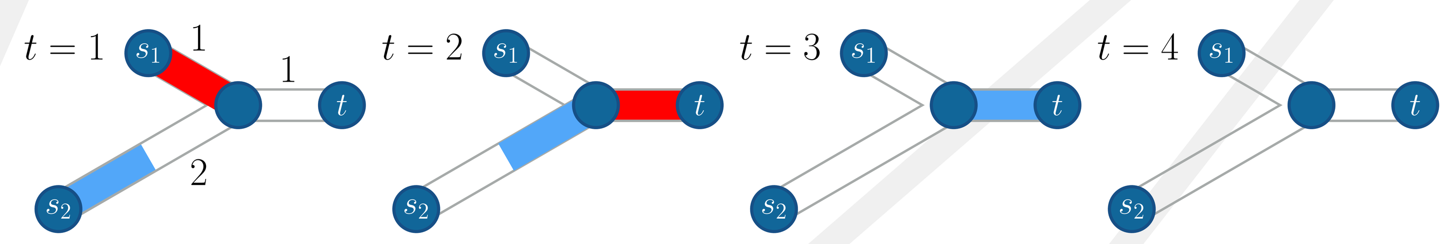
- Each arc a has a **transit time** (length) τ_a
- Each arc a has a **capacity** (width) u_a

dynamic network $\mathcal{N} = (D, u, \tau, S^+, S^-)$: digraph $D = (V, A)$ with capacities u, transit times τ, sources $S^+ \subseteq V$ and sinks $S^- \subseteq V$

A **flow over time f in $\mathcal{N}$** with **time horizon T** specifies the **rate of flow** entering an arc per time, such that no flow is left in $\mathcal{N}$ after time T.

Lex-Max Flows Over Time:
Given: $\mathcal{N} = (D = (V, A), u, \tau, S^+, \{t\})$, total order $\prec$ on S^+, time horizon T
Goal: **lexicographically maximize** the amount of flow leaving each source in the given order within time horizon T

Example: All capacities are 1, lex-max flow over time wrt $s_1 \prec s_2$ and time horizon $T = 4$



A lex-max flow over time f with respect to order $s_1 \prec \ldots \prec s_k$ and time horizon T fulfills

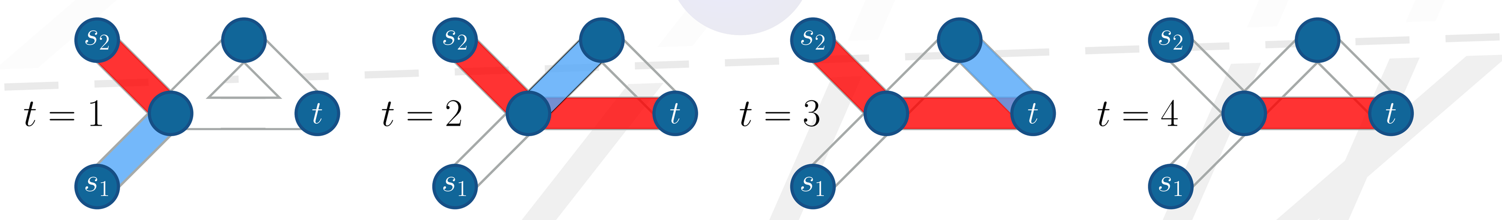
$$|f(\{s_i \ldots, s_k\}|_T = o^T\{s_i \ldots, s_k\} \text{ for all } i \in \{1, \ldots, k\}, \text{ with}$$

$o^\theta(A) :=$ maximum amount of flow that can be send from $A \subseteq S^+$ to t within time θ

Hoppe & Tardos, 1995
Lex-max flow over time problems can be solved in strongly polynomial time.

Earliest Arrival Flow & Pattern

Given: $\mathcal{N} = (D = (V, A), u, \tau, \{S^+\}, \{t\})$, supplies v on the sources
Goal: fulfill the supplies such that **at each point in time** as much flow as possible has reached the sink t, **earliest arrival flow (EAF)**
Example: $u \equiv 1$, $\tau \equiv 1$, $v(s_1) = 1$, $v(s_2) = 3$, minimal feasible time horizon $T = 5$



Known Results

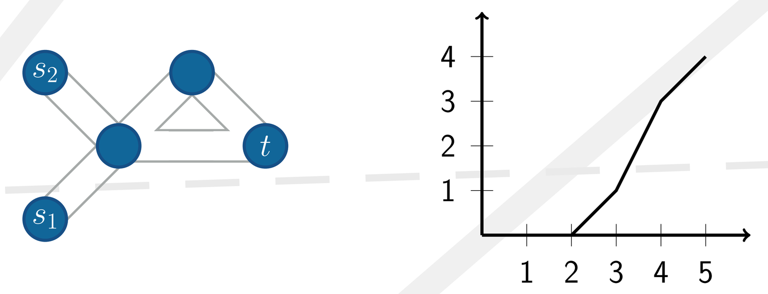
- $|S^+| = 1$: EAFs can be computed by sending flow along paths occuring in the successive shortest path algorithm (SSPA) from the sources to sink – polynomial space algorithm!
- $|S^+| > 1$: All algorithms known so far need **exponential space!**

Earliest Arrival Pattern
$p : [0, \infty) \to \mathbb{R}^+$, $p(\theta) =$ value of an earliest arrival flow at time θ

Structure of The EAF-Pattern (Baumann & Skutella, 2006)
Let $\theta_1 = \max\{\theta | o^\theta(S^+) = p(\theta)\}$. It is

$$p(\theta) = \begin{cases} o^\theta(S^+) \text{ for } \theta < \theta_1 \\ o^\theta(S^+ \setminus S_1) + v(S_1) \text{ for } \theta \leq \theta_1. \end{cases}$$

Intuitive interpretation: In an earliest arrival flow the sources in S_1 have to run empty until time θ_1 even if they send as little flow as possible!



$S_1 = \{s_1\}$, $\theta_1 = 4$
$S_2 = \{s_2\}$, $\theta_2 = 5$

Sets $S_1, \ldots, S_r \subseteq S^+$ and times $\theta_1 < \ldots < \theta_r$ such that S_i has to run empty at time θ_i in an earliest arrival flow can be computed in strongly polynomial time (Baumann & Skutella).

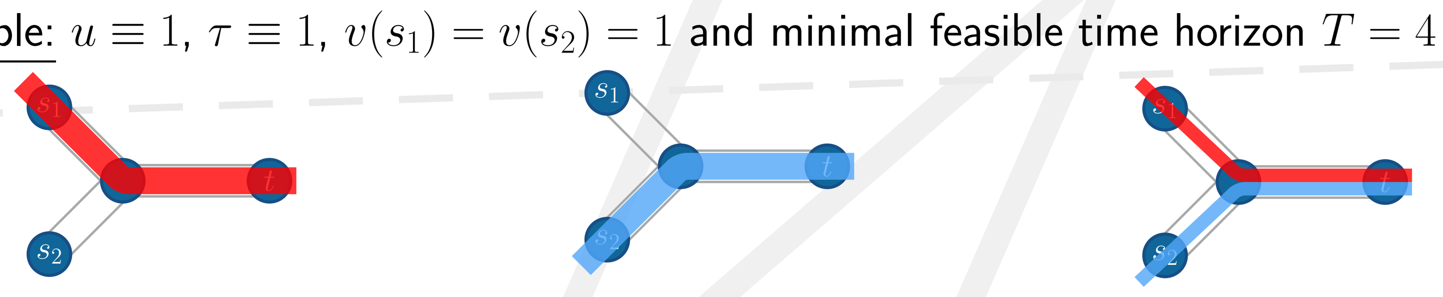
Our Main Result (SODA 2017): A polynomial space algorithm for solving earliest arrival flow problems in networks with multiple sources

Special Cases

Given: a dynamic network $\mathcal{N} = (D = (V, A), u, \tau, \{S^+\}, \{t\})$, supplies v on the sources, minimal feasible time horizon T
Case 1: $o^T(S^+) = v(S^+)$ (tight case) - it is $p(\theta) = o^\theta(S^+)$ for all $\theta \leq T$

Result 1
A flow over time f with time horizon T fulfilling all supplies can be obtained as **convex combination of lex-max flows over time** that can be found by one submodular function minimization - using the SSPA ensures that the lex-max flows respect the EAF-pattern.

Example: $u \equiv 1$, $\tau \equiv 1$, $v(s_1) = v(s_2) = 1$ and minimal feasible time horizon $T = 4$



lex-max flow wrt. $s_2 \prec s_1$ — lex-max flow wrt. $s_1 \prec s_2$ — convex combination of both lex-max flows (both weighted with $1/2$)

Case 2: Let $S^+ = \{s_1, \ldots, s_k\}$, s_i runs empty at time θ_i for all $i \in \{1, \ldots, k\}$ with $\theta_1 < \theta_2 < \ldots < \theta_k$.
Observation: For an earliest arrival flow f we have for all $i \in \{1, \ldots, k\}$:

$$|f(\{s_i \ldots, s_k\}|_{\theta_i} = o^{\theta_i}(\{s_i \ldots, s_k\}) \text{ and } |f(\{s_{i+1} \ldots, s_k\}|_{\theta_i} = o^{\theta_i}(\{s_{i+1} \ldots, s_k\})$$

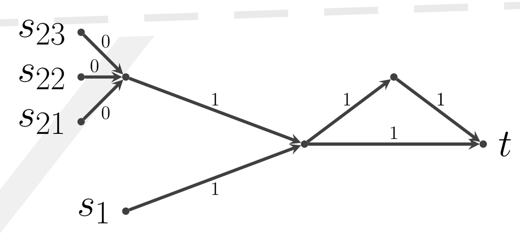
An earliest arrival flow f behaves like a lex-max flow over time with respect to order $s_1 \prec s_2 \prec \ldots \prec s_k$ and growing time horizons $\theta_1 < \ldots < \theta_k$, **generalized lex-max flow over time**: flow respects order $\prec$ and

- flow out of s_1 has time horizon θ_1
- flow out of s_2 has time horizon θ_2
- ...
- flow out of s_k has time horizon θ_k

Result 2
A **polynomial space algorithm** that computes generalized lex-max flows over time that respect the EAF-pattern.

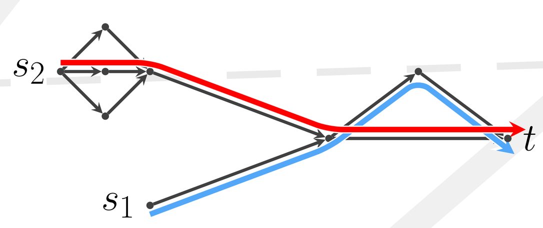
General Case

Example: $u \equiv 1$, $\tau \equiv$ indicated in figure, $v(s_1) = 1$, $v(s_{21}) = v(s_{22}) = v(s_{23}) = 1$



$S_1 = \{s_1\}$, $\theta_1 = 4$
$S_2 = \{s_{21}, s_{22}, s_{23}\}$, $\theta_2 = 5$

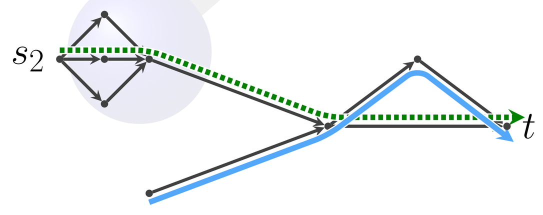
Attaching a supersource s_2 to s_{21}, s_{22} and s_{23} with $v(s_2) = 3$ and $v(s_1) = 1$ results in an EAF-problem in which s_1 runs empty a time 4 and s_2 at time 5!



gen. lex-max flow f' wrt. $s_1 \prec s_2$ and $\theta_1 < \theta_2$

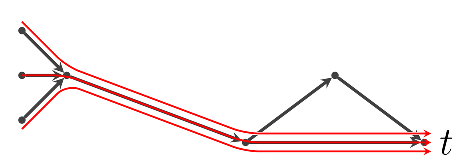
Result 2: flow over time f' with:
$|f'(s_1)|_{\theta_1} = v(s_1) = 1$, $|f'(s_2)|_{\theta_2} = v(s_2) = 3$
and $|f'|_\theta = p(\theta) = o^\theta(\{s_1, s_2\})$

Problem with gen. lex-max flow f': Not each original source might send exactly its supply!
Solution: Use Result 1 to ensure that all sources send exactly their supply!



Tight for $\theta = \theta_1 = 3$ ($v(s_2) = o^{\theta_1}(s_2)$), solved by lex-max flow wrt. $s_1 \prec s_2$

Result 1: A convex combination of lex-max flows solving this tight problem (flow out of s_2 is not of interest)



Tight for $\theta = \theta_2 = 5$, all flows are weighted with $1/3$

Result 1: A convex combination of lex-max flows solving this tight problem

Combine both convex combination → convex combination of generalized lex-max flows solving the EAF problem that can be computed in poly space using **Result 2**.