Smoothed Analysis of the Successive Shortest Path Algorithm

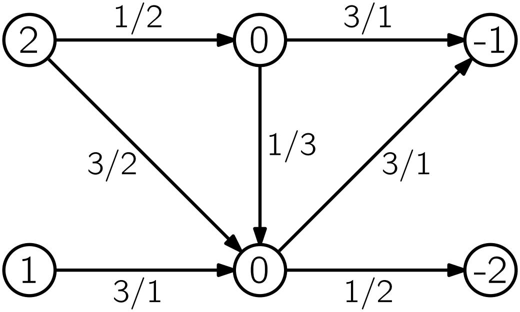
Tobias Brunsch[1] Kamiel Cornelissen[2]
Bodo Manthey[2] Heiko Röglin[1] Clemens Rösner[1]

[1]Department of Computer Science
University of Bonn, Germany

[2]Department of Applied Mathematics
University of Twente, The Netherlands

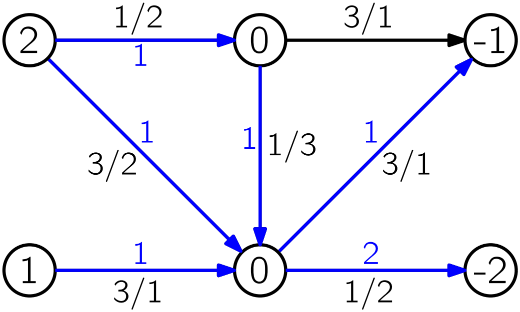
Minimum-Cost Flow Network

flow network: $G = (V, E)$
balance values: $b : V \to \mathbb{Z}$
costs: $c : E \to \mathbb{R}_{\geq 0}$
capacities: $u : E \to \mathbb{N}$



cost/capacity

Minimum-Cost Flow Problem

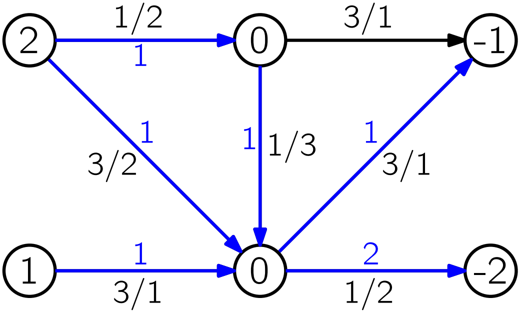


flow: $f : E \to \mathbb{R}_{\geq 0}$

capacity constraints: $\forall e \in E : f(e) \leq u(e)$

Kirchhoff's law: $\forall v \in V : b(v) = \text{out}(v) - \text{in}(v)$

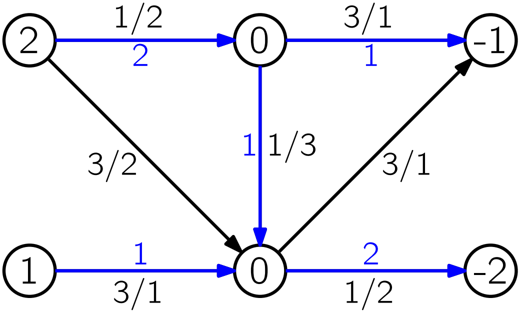
Minimum-Cost Flow Problem



flow:	$f : E \to \mathbb{R}_{\geq 0}$
capacity constraints:	$\forall e \in E : f(e) \leq u(e)$
Kirchhoff's law:	$\forall v \in V : b(v) = \text{out}(v) - \text{in}(v)$

Goal: $\min_{\text{flow } f} \sum_{e \in E} f(e) \cdot c(e)$

Minimum-Cost Flow Problem



flow: $f : E \to \mathbb{R}_{\geq 0}$
capacity constraints: $\forall e \in E : f(e) \leq u(e)$
Kirchhoff's law: $\forall v \in V : b(v) = \text{out}(v) - \text{in}(v)$

Goal: $\min_{\text{flow } f} \sum_{e \in E} f(e) \cdot c(e)$

Short History

Pseudo-Polynomial Algorithms:

Out-of-Kilter algorithm [Minty 60, Fulkerson 61]
Cycle Canceling algorithm
Successive Shortest Path algorithm

Short History

Pseudo-Polynomial Algorithms:

Out-of-Kilter algorithm [Minty 60, Fulkerson 61]
Cycle Canceling algorithm
Successive Shortest Path algorithm

Polynomial Time Algorithms:

Capacity Scaling algorithm [Edmonds and Karp 72]
Cost Scaling algorithm

Short History

Pseudo-Polynomial Algorithms:

Out-of-Kilter algorithm [Minty 60, Fulkerson 61]
Cycle Canceling algorithm
Successive Shortest Path algorithm

Polynomial Time Algorithms:

Capacity Scaling algorithm [Edmonds and Karp 72]
Cost Scaling algorithm

Strongly Polynomial Algorithms:

Tardos' algorithm [Tardos 85]
Minimum-Mean Cycle Canceling algorithm
Network Simplex algorithm
Enhanced Capacity Scaling algorithm [Orlin 93]

Theory vs. Practice

Theory

Fastest algorithm:
Enhanced Capacity Scaling

Practice

Fastest algorithm:
Network Simplex

Theory vs. Practice

Theory

Fastest algorithm:
Enhanced Capacity Scaling

Successive Shortest Path:
exponential in worst case

Minimum-Mean Cycle Canceling:
strongly polynomial

Practice

Fastest algorithm:
Network Simplex

Successive Shortest Path

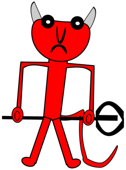
much faster than

Minimum-Mean Cycle Canceling

Reason for Gap between Theory and Practice

- Worst-case complexity is too pessimistic!
- There are artificial worst-case inputs. These inputs, however, do not occur in practice.

Adversary

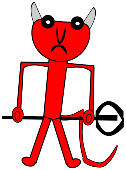


"I will trick your algorithm!"

Reason for Gap between Theory and Practice

- Worst-case complexity is too pessimistic!
- There are artificial worst-case inputs. These inputs, however, do not occur in practice.
- This phenomenon occurs also for many other problems and algorithms.

Adversary

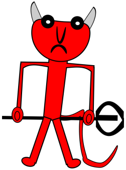


"I will trick your algorithm!"

Reason for Gap between Theory and Practice

- Worst-case complexity is too pessimistic!
- There are artificial worst-case inputs. These inputs, however, do not occur in practice.
- This phenomenon occurs also for many other problems and algorithms.

Adversary



"I will trick your algo-rithm!"

Goal

Find a more realistic performance measure that is not just based on the worst case.

Smoothed Analysis

Observation: In worst-case analysis, the adversary is too powerful.
Idea: Let's weaken him!

Input model:

- Adversarial choice of flow network
- Adversarial real arc capacities u_e and node balance values b_v
- Adversarial densities $f_e\colon [0,1] \to [0,\phi]$
- Arc costs c_e independently drawn according to f_e

Smoothed Analysis

Observation: In worst-case analysis, the adversary is too powerful.
Idea: Let's weaken him!

Input model:

- Adversarial choice of flow network
- Adversarial real arc capacities u_e and node balance values b_v
- Adversarial densities $f_e\colon [0,1] \to [0,\phi]$
- Arc costs c_e independently drawn according to f_e

Randomness models, e.g., measurement errors, numerical imprecision, rounding, ...

Smoothed Analysis

Worst-case Analysis: $\max_{c_e} T$

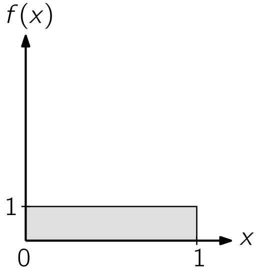
Smoothed Analysis: $\max_{f_e} \mathbf{E}[T]$

Smoothed Analysis

Worst-case Analysis: $\max_{c_e} T$

Smoothed Analysis: $\max_{f_e} \mathbf{E}[T]$

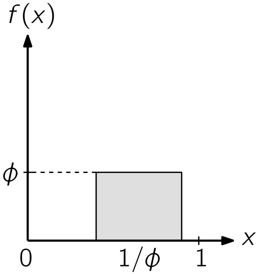
$\phi = 1$: Average-case analysis



Smoothed Analysis

Worst-case Analysis: $\max_{c_e} T$

Smoothed Analysis: $\max_{f_e} \mathbf{E}[T]$

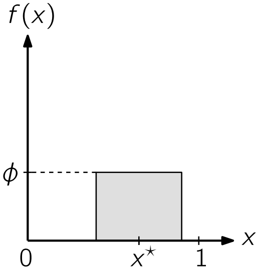


Smoothed Analysis

Worst-case Analysis: $\max_{c_e} T$

Smoothed Analysis: $\max_{f_e} \mathbf{E}[T]$

$\phi \to \infty$: Worst-case analysis

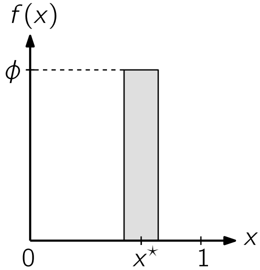


Smoothed Analysis

Worst-case Analysis: $\max_{c_e} T$

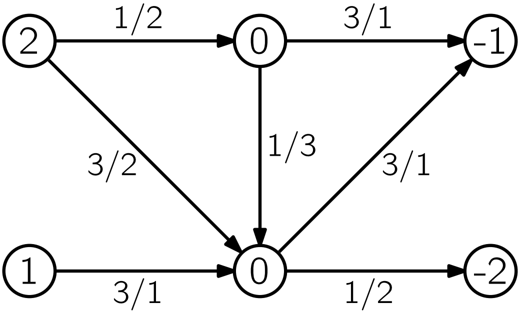
Smoothed Analysis: $\max_{f_e} \mathbf{E}[T]$

$\phi \to \infty$: Worst-case analysis



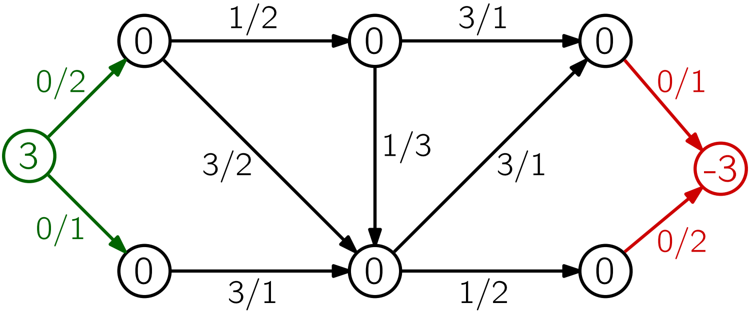
Initial Transformation

Successive Shortest Path algorithm



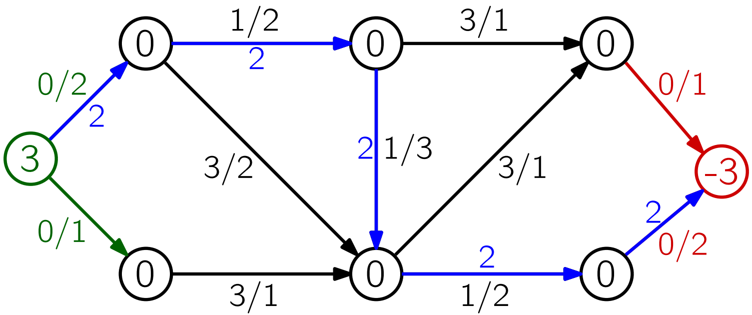
Initial Transformation

Successive Shortest Path algorithm



Augmenting Steps

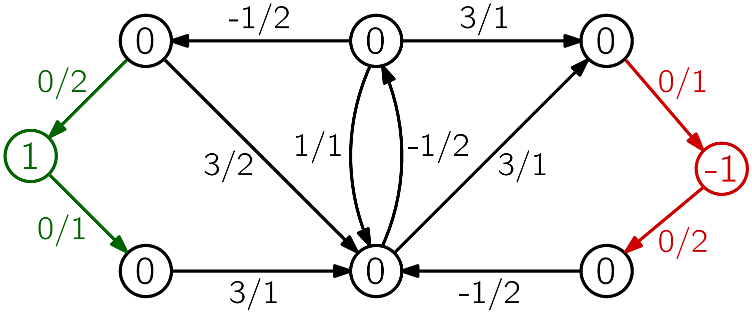
Successive Shortest Path algorithm



path length: 3, augmenting flow value: 2

Augmenting Steps

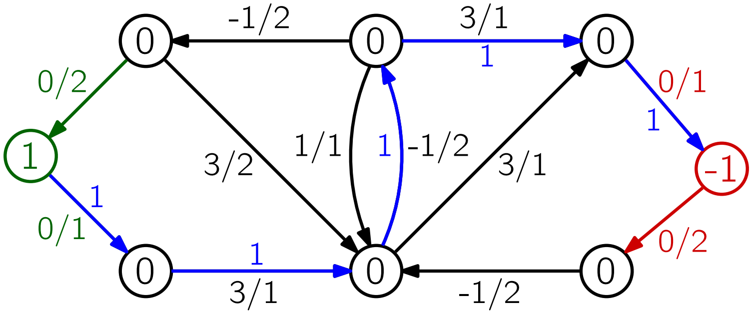
Successive Shortest Path algorithm



update residual network

Augmenting Steps

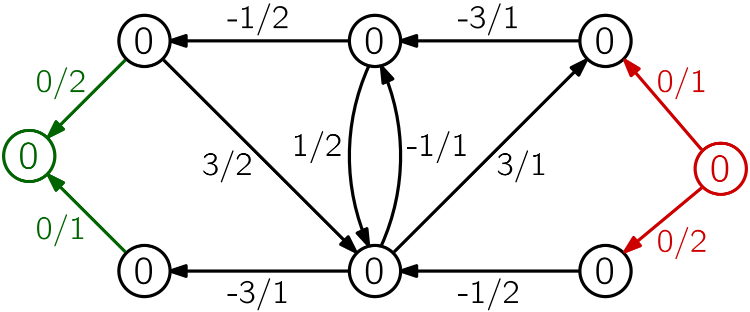
Successive Shortest Path algorithm



path length: 5, augmenting flow value: 1

Augmenting Steps

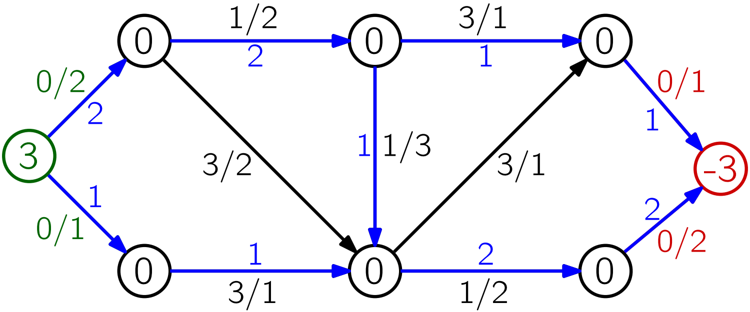
Successive Shortest Path algorithm



update residual network

Resulting Flow

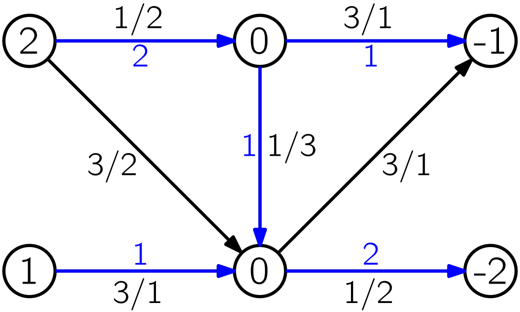
Successive Shortest Path algorithm



flow cost: 11, flow value: 3

Resulting Flow

Successive Shortest Path algorithm



flow cost: 11, flow value: 3

Results

Theorem (Upper Bound)

In expectation, the SSP algorithm requires $O(mn\phi)$ iterations and has a running time of $O(mn\phi(m + n\log n))$.

Results

Theorem (Upper Bound)

In expectation, the SSP algorithm requires $O(mn\phi)$ iterations and has a running time of $O(mn\phi(m + n\log n))$.

Theorem (Lower Bound)

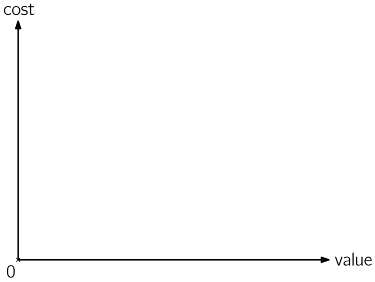
There are smoothed instances on which the SSP algorithm requires $\Omega(m \cdot \min\{n, \phi\} \cdot \phi)$ iterations in expectation.

upper bound tight for $\phi = \Omega(n)$

Useful Properties

Lemma

The distances from the source to any node increase monotonically.

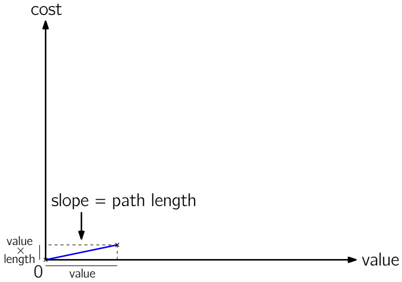


initial solution: empty flow

Useful Properties

Lemma

The distances from the source to any node increase monotonically.

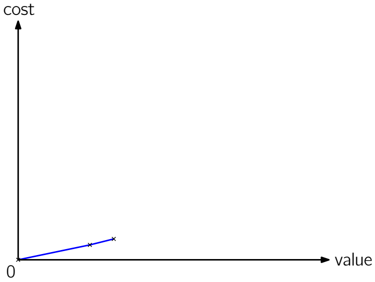


after 1 iteration

Useful Properties

Lemma

The distances from the source to any node increase monotonically.

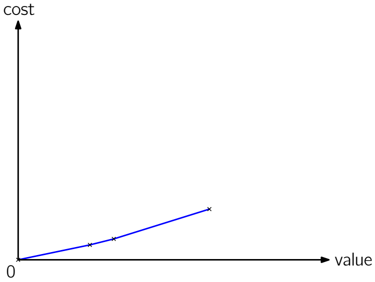


after 2 iterations

Useful Properties

Lemma

The distances from the source to any node increase monotonically.

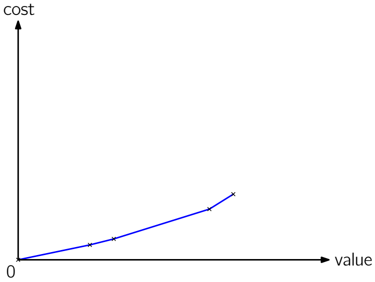


after 3 iterations

Useful Properties

Lemma

The distances from the source to any node increase monotonically.

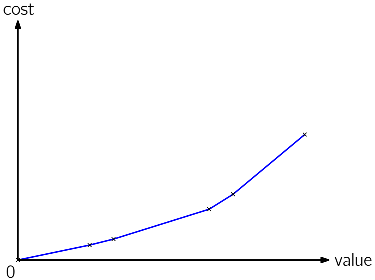


after 4 iterations

Useful Properties

Lemma

The distances from the source to any node increase monotonically.



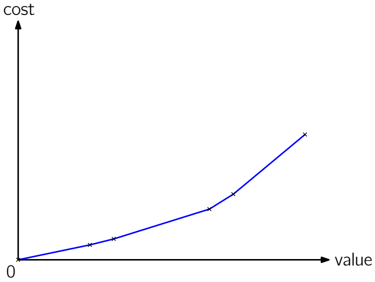
after 5 iterations

#iterations = #distinct slopes

Useful Properties

Lemma

The distances from the source to any node increase monotonically.



after 5 iterations

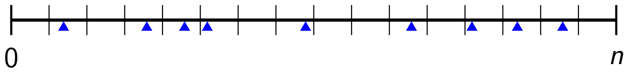
#iterations = #distinct slopes

Lemma

Every intermediate flow is optimal for its flow value.

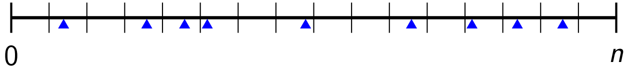
Counting the Number of Slopes

slope = augmenting path length $\in (0, n]$



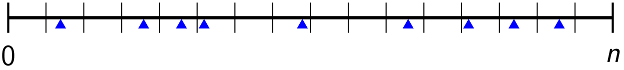
Counting the Number of Slopes

$$\text{slope} = \text{augmenting path length} \in (0, n] = \bigcup_{\ell=1}^{k} I_\ell, \quad |I_\ell| = \frac{n}{k}$$



Counting the Number of Slopes

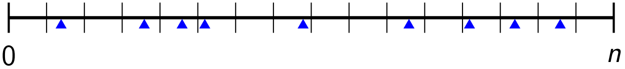
$$\text{slope} = \text{augmenting path length} \in (0, n] = \bigcup_{\ell=1}^{k} I_\ell, \quad |I_\ell| = \frac{n}{k}$$



$$\Longrightarrow \quad \#\text{slopes} = \sum_{\ell=1}^{k} \#\text{slopes} \in I_\ell$$

Counting the Number of Slopes

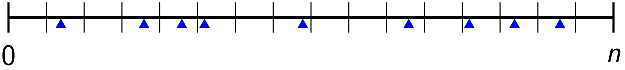
$$\text{slope} = \text{augmenting path length} \in (0, n] = \bigcup_{\ell=1}^{k} I_\ell, \quad |I_\ell| = \frac{n}{k}$$



$$\implies \mathbf{E}[\#\text{slopes}] = \sum_{\ell=1}^{k} \mathbf{E}[\#\text{slopes} \in I_\ell]$$

Counting the Number of Slopes

$$\text{slope} = \text{augmenting path length} \in (0, n] = \bigcup_{\ell=1}^{k} I_\ell, \qquad |I_\ell| = \frac{n}{k}$$

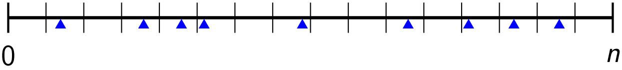


$$\Longrightarrow \mathbf{E}[\#\text{slopes}] = \sum_{\ell=1}^{k} \mathbf{E}[\#\text{slopes} \in I_\ell]$$

$$\approx \sum_{\ell=1}^{k} \mathbf{Pr}[\exists \text{slope} \in I_\ell]$$

Counting the Number of Slopes

$$\text{slope} = \text{augmenting path length} \in (0, n] = \bigcup_{\ell=1}^{k} I_\ell, \quad |I_\ell| = \frac{n}{k}$$



$$\Longrightarrow \mathbf{E}[\#\text{slopes}] = \sum_{\ell=1}^{k} \mathbf{E}[\#\text{slopes} \in I_\ell]$$

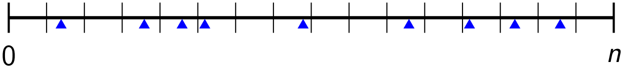
$$\approx \sum_{\ell=1}^{k} \mathbf{Pr}[\exists \text{slope} \in I_\ell]$$

Main Lemma

$\forall d \geq 0 : \forall \varepsilon \geq 0 : \mathbf{Pr}[\exists \text{slope} \in (d, d+\varepsilon]] = O(m\phi\varepsilon)$

Counting the Number of Slopes

$$\text{slope} = \text{augmenting path length} \in (0, n] = \bigcup_{\ell=1}^{k} I_\ell, \quad |I_\ell| = \frac{n}{k}$$



$$\implies \mathbf{E}[\#\text{slopes}] = \sum_{\ell=1}^{k} \mathbf{E}[\#\text{slopes} \in I_\ell]$$

$$\approx \sum_{\ell=1}^{k} \mathbf{Pr}[\exists \text{slope} \in I_\ell]$$

$$= O(mn\phi)$$

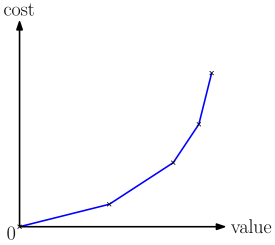
Main Lemma

$\forall d \geq 0 : \forall \varepsilon \geq 0 : \mathbf{Pr}[\exists \text{slope} \in (d, d+\varepsilon]] = O(m\phi\varepsilon)$

Flow Reconstruction

Main Lemma

$\forall d \geq 0 : \forall \varepsilon \geq 0 : \mathbf{Pr}[\exists \text{slope} \in (d, d+\varepsilon]] = O(m\phi\varepsilon)$

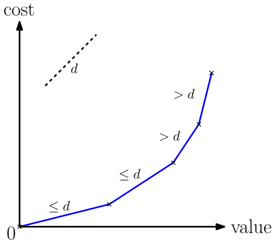


Flow Reconstruction

Main Lemma

$\forall d \geq 0 : \forall \varepsilon \geq 0 : \mathbf{Pr}[\exists \text{slope} \in (d, d+\varepsilon]] = O(m\phi\varepsilon)$

d - slope threshold



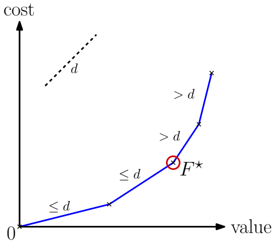
Flow Reconstruction

Main Lemma

$\forall d \geq 0 : \forall \varepsilon \geq 0 : \mathbf{Pr}[\exists \text{slope} \in (d, d+\varepsilon]] = O(m\phi\varepsilon)$

d - slope threshold

$F^\star$- flow at breakpoint

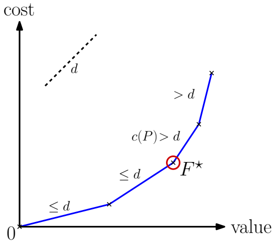


Flow Reconstruction

Main Lemma

$\forall d \geq 0 : \forall \varepsilon \geq 0 : \mathbf{Pr}[\exists \text{slope} \in (d, d+\varepsilon]] = O(m\phi\varepsilon)$

- d - slope threshold
- $F^\star$- flow at breakpoint
- P - next augmenting path
- e - empty arc of P in $G_{f^\star}$

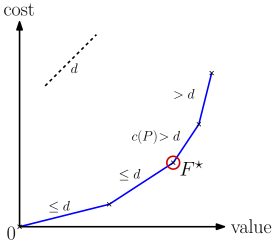


Flow Reconstruction

Main Lemma

$\forall d \geq 0 : \forall \varepsilon \geq 0 : \mathbf{Pr}[\exists \text{slope} \in (d, d+\varepsilon]] = O(m\phi\varepsilon)$

- d - slope threshold
- $F^\star$- flow at breakpoint
- P - next augmenting path
- e - empty arc of P in $G_{f^\star}$



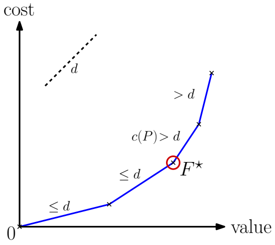
$\exists \text{slope} \in (d, d+\varepsilon] \iff c(P) \in (d, d+\varepsilon]$

Flow Reconstruction

Main Lemma

$\forall d \geq 0 : \forall \varepsilon \geq 0 : \mathbf{Pr}[\exists \text{slope} \in (d, d+\varepsilon]] = O(m\phi\varepsilon)$

- d - slope threshold
- $F^\star$- flow at breakpoint
- P - next augmenting path
- e - empty arc of P in $G_{f^\star}$



$\exists \text{slope} \in (d, d+\varepsilon] \iff c(P) \in (d, d+\varepsilon]$

Goal: Reconstruct $F^\star$ and P without knowing c_e

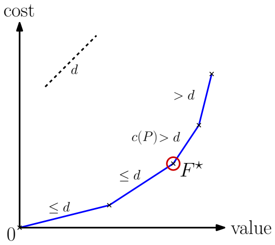
Principle of Deferred Decisions

Main Lemma

$\forall d \geq 0 : \forall \varepsilon \geq 0 : \mathbf{Pr}[\exists \text{slope} \in (d, d+\varepsilon]] = O(m\phi\varepsilon)$

Phase 1: Reveal all $c_{e'}$ for $e' \neq e$.

Assume this suffices to uniquely identify $F^\star$ and P.

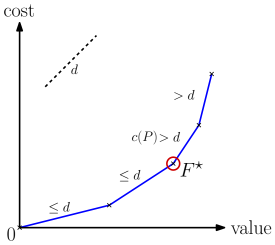


Principle of Deferred Decisions

Main Lemma

$\forall d \geq 0 : \forall \varepsilon \geq 0 : \mathbf{Pr}[\exists \text{slope} \in (d, d+\varepsilon]] = O(m\phi\varepsilon)$

Phase 1: Reveal all $c_{e'}$ for $e' \neq e$.
Assume this suffices to uniquely identify $F^\star$ and P.



Phase 2:

$$\mathbf{Pr}[c(P) \in (d, d+\varepsilon]] = \mathbf{Pr}[c(e) \in (z, z+\varepsilon]] \leq \phi\varepsilon,$$

where z is fixed if $c_{e'}$ for $e' \neq e$ is fixed.

Flow Reconstruction

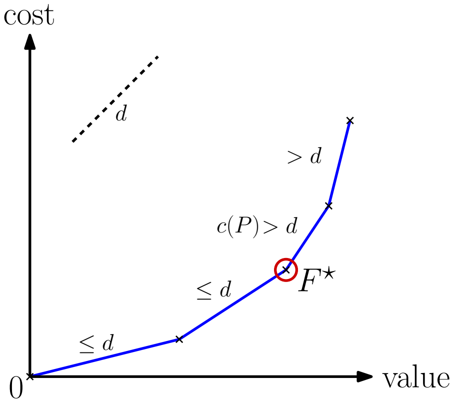
Case 1: e forward arc

Set $c'(e) = 1$ and for all $e' \neq e$ set $c'(e') = c(e')$.
Run SSP with modified costs c'.

Flow Reconstruction

Case 1: e forward arc

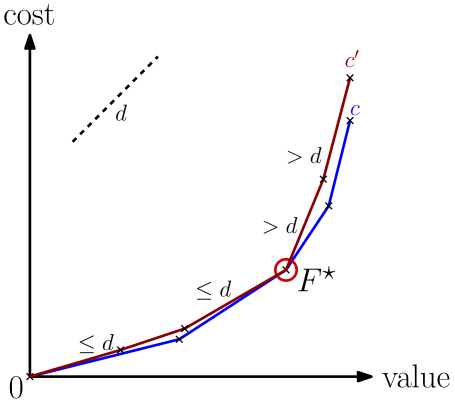
Set $c'(e) = 1$ and for all $e' \neq e$ set $c'(e') = c(e')$.
Run SSP with modified costs c'.



Flow Reconstruction

Case 1: e forward arc

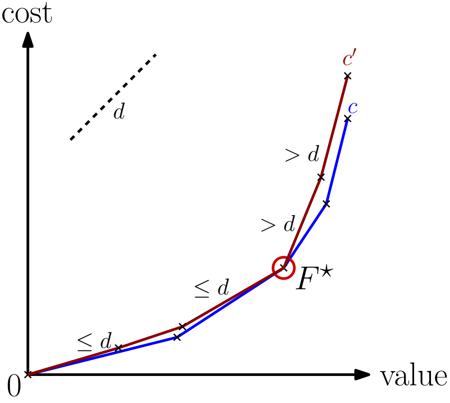
Set $c'(e) = 1$ and for all $e' \neq e$ set $c'(e') = c(e')$.
Run SSP with modified costs c'.



Flow Reconstruction

Case 1: e forward arc

Set $c'(e) = 1$ and for all $e' \neq e$ set $c'(e') = c(e')$.
Run SSP with modified costs c'.



$F^\star$ is the same for c and c'