Smoothed Analysis of the Successive Shortest Path Algorithm

Tobias Brunsch\textsuperscript{1} \hspace{1em} Kamiel Cornelissen\textsuperscript{2}  
Bodo Manthey\textsuperscript{2} \hspace{1em} Heiko Röglin\textsuperscript{1} \hspace{1em} Clemens Rösner\textsuperscript{1}

\begin{itemize}
\item[\textsuperscript{1}] Department of Computer Science  
University of Bonn, Germany
\item[\textsuperscript{2}] Department of Applied Mathematics  
University of Twente, The Netherlands
\end{itemize}
Minimum-Cost Flow Network

flow network: \( G = (V, E) \)
balance values: \( b : V \rightarrow \mathbb{Z} \)
costs: \( c : E \rightarrow \mathbb{R}_{\geq 0} \)
capacities: \( u : E \rightarrow \mathbb{N} \)
Minimum-Cost Flow Problem

flow:
\[ f : E \rightarrow \mathbb{R}_{\geq 0} \]

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: \( f : E \rightarrow \mathbb{R}_{\geq 0} \)

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: $f : E \rightarrow \mathbb{R}_{\geq 0}$
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: $\min_{f} \sum_{e \in E} f(e) \cdot c(e)$
Short History

**Pseudo-Polynomial Algorithms:**

Out-of-Kilter algorithm
Cycle Canceling algorithm
Successive Shortest Path algorithm

[Minty 60, Fulkerson 61]
Short History

**Pseudo-Polynomial Algorithms:**
Out-of-Kilter algorithm
Cycle Canceling algorithm
Successive Shortest Path algorithm

**Polynomial Time Algorithms:**
Capacity Scaling algorithm
Cost Scaling algorithm

[Minty 60, Fulkerson 61]

[Edmonds and Karp 72]
**Short History**

**Pseudo-Polynomial Algorithms:**
- Out-of-Kilter algorithm
- Cycle Canceling algorithm
- Successive Shortest Path algorithm

[Minty 60, Fulkerson 61]

**Polynomial Time Algorithms:**
- Capacity Scaling algorithm
- Cost Scaling algorithm

[Edmonds and Karp 72]

**Strongly Polynomial Algorithms:**
- Tardos’ algorithm
- Minimum-Mean Cycle Canceling algorithm
- Network Simplex algorithm
- Enhanced Capacity Scaling algorithm

[Tardos 85]

[Orlin 93]
## Theory vs. Practice

<table>
<thead>
<tr>
<th><strong>Theory</strong></th>
<th><strong>Practice</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Fastest algorithm: Enhanced Capacity Scaling</td>
<td>Fastest algorithm: Network Simplex</td>
</tr>
</tbody>
</table>
**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 algorithm!""

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 $f_e : [0,1] \rightarrow [0, \phi]$
- Arc costs $c_e$ independently drawn according to $f_e$
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 $f_e : [0,1] \rightarrow [0,\phi]$
- 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: \( \max_{f_e} E[T] \)
Worst-case Analysis: \( \max_{c_e} T \)

Smoothed Analysis: \( \max_{f_e} \mathbb{E}[T] \)

\( \phi = 1: \) Average-case analysis
**Smooted Analysis**

**Worst-case Analysis:** $\max_{c,e} T$

**Smoothed Analysis:** $\max_{f,e} \mathbb{E}[T]$
Worst-case Analysis: $\max_{c_e} T$

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

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

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

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

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

**Successive Shortest Path algorithm**

```
2  3/2  1/2  3/1  -1
0  3/2  1/3  3/1
1  3/1
0  1/2
-2
```
Initial Transformation

**Successive Shortest Path algorithm**

![Graph with labels and arrows showing the network structure and flows. The labels include costs and capacities, such as 0/2, 3/2, 1/2, 1/3, 3/1, etc.]
Augmenting Steps

Successive Shortest Path algorithm

path length: 3, augmenting flow value: 2
Augmenting Steps

**Successive Shortest Path algorithm**

![Diagram of a network with labels and arrows indicating flow](image)

update residual network
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
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
Lemma

The distances from the source to any node increase monotonically.

slope = path length

after 1 iteration
Useful Properties

Lemma

The distances from the source to any node increase monotonically.
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.

![Graph showing the increase in distance from the source to any node after 4 iterations.](image)
Useful Properties

Lemma

The distances from the source to any node increase monotonically.

After 5 iterations

\#iterations = \#distinct slopes
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 ∈ (0, n]

\[ \text{slope} = \text{augmenting path length} \in (0, n] \]
Counting the Number of Slopes

slope = augmenting path length $\in (0, n] = \bigcup_{\ell=1}^{k} I_\ell$, $|I_\ell| = \frac{n}{k}$
Counting the Number of Slopes

slope = augmenting path length ∈ (0, n] = \bigcup_{\ell=1}^{k} l_{\ell}, \quad |l_{\ell}| = \frac{n}{k}

\[ \Rightarrow \quad \#\text{slopes} = \sum_{\ell=1}^{k} \#\text{slopes} \in l_{\ell} \]
Counting the Number of Slopes

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

\[
\implies \mathbb{E}[\#\text{slopes}] = \sum_{\ell=1}^{k} \mathbb{E}[\#\text{slopes} \in I_{\ell}]
\]
Counting the Number of Slopes

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

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

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

$\approx O\left(mn \phi \epsilon \right)$
Counting the Number of Slopes

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

\[ \Rightarrow \mathbb{E}[\#\text{slopes}] = \sum_{\ell=1}^{k} \mathbb{E}[\#\text{slopes} \in I_{\ell}] \]

\[ \approx \sum_{\ell=1}^{k} \mathbb{P}[\exists \text{slope} \in I_{\ell}] \]

Main Lemma

\[ \forall d \geq 0 : \forall \varepsilon \geq 0 : \mathbb{P}[\exists \text{slope} \in (d, d + \varepsilon)] = O(m\phi \varepsilon) \]
Counting the Number of Slopes

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

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

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

$= O(mn\phi)$

Main Lemma

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

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

Main Lemma

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

\(d\) - slope threshold
Main Lemma

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

- \(d\) - slope threshold
- \(F^*\) - flow at breakpoint
Flow Reconstruction

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

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

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

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

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

d - slope threshold

\( F^* \) - flow at breakpoint

\( P \) - next augmenting path

\( e \) - empty arc of \( P \) in \( G_{f^*} \)

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

Goal: Reconstruct \( F^* \) and \( P \) without knowing \( c_e \)
Main Lemma

∀d ≥ 0 : ∀ε ≥ 0 : \( \Pr[\exists \text{slope} \in (d, d + \varepsilon)] = O(m\phi\varepsilon) \)

Phase 1: Reveal all \( c_{e'} \) for \( e' \neq e \).
Assume this suffices to uniquely identify \( F^* \) and \( P \).
**Main Lemma**

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

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

**Phase 2:**

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

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 \( c' \).
**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 \( c' \).
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 $c'$. 

![Diagram showing cost and value graphs with cost function $c'$ and $c$, Value $F^*$ marked.](image)
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 $c'$.

$F^*$ is the same for $c$ and $c'$