Parallel Breadth-First Search and Exact Shortest Paths and Stronger Notions for Approximate Distances

Goran Zuzic

ETH Zürich → Google Research

Václav Rozhoň

Bernhard Haeupler

Anders Martinsson

Christoph Grunau
Today’s Topic: The Shortest Path Problem.

- Undirected graph with non-negative weights $w(e)$.
- Single source.

Definition
We assume some source $A$ is clear from context. We define:

- Exact distances: $\text{dist}(B)$.
- Sometimes $\text{dist}(A, B)$.

- Approximate distances: $\tilde{d}(B)$.
Today’s Topic: The Shortest Path Problem.

- Undirected graph with non-negative weights $w(e)$.
- Single source.

\[ \text{dist}(A) = 0 \quad \text{dist}(B) = 5 \]

\[ \begin{align*}
A & \quad 5 & B \\
C & \quad 20 & D \\
\end{align*} \]

\[ \text{dist}(C) = 15 \quad \text{dist}(D) = 25 \]
Today’s Topic: The Shortest Path Problem.

- Undirected graph with non-negative weights $w(e)$.
- Single source.

\[
\begin{align*}
\text{dist}(A) &= 0 \\
\text{dist}(B) &= 5 \\
\text{dist}(C) &= 15 \\
\text{dist}(D) &= 25
\end{align*}
\]

Definition
We assume some source $A$ is clear from context. We define:

- **Exact distances**: $\text{dist}(B)$. Sometimes $\text{dist}(A, B)$. 

![Graph Image]
Today’s Topic: The Shortest Path Problem.

- Undirected graph with non-negative weights $w(e)$.
- Single source.

\[
\begin{align*}
\text{dist}(A) &= 0 \\
\text{dist}(B) &= 5 \\
\text{dist}(C) &= 15 \\
\text{dist}(D) &= 25
\end{align*}
\]

**Definition**

We assume some source $A$ is clear from context. We define:

- **Exact distances**: $\text{dist}(B)$. Sometimes $\text{dist}(A, B)$.
- **Approximate distances**: $\tilde{d}(B)$. 

The diagram illustrates the graph with labeled edges and distances.
Approximate distances.

Distance estimate = any function $\tilde{d} : V \to \mathbb{R}_{\geq 0}$.

**Definition**

A function $\tilde{d} : V \to \mathbb{R}_{\geq 0}$ is a **weak** $(1 + \varepsilon)$-approximation (with respect to some source $s \in V$) if:

$$\forall v \in V \quad \text{dist}(v) \leq \tilde{d}(v) \leq (1 + \varepsilon)\text{dist}(v).$$
Approximate distances.

Distance estimate = any function $\tilde{d}: V \to \mathbb{R}_{\geq 0}$.

**Definition**

A function $\tilde{d}: V \to \mathbb{R}_{\geq 0}$ is a **weak** $(1 + \varepsilon)$-approximation (with respect to some source $s \in V$) if:

$$\forall v \in V \quad dist(v) \leq \tilde{d}(v) \leq (1 + \varepsilon)\text{dist}(v).$$

**Example:** $G$ = a path graph (from left to right). Source = leftmost node.
Shortest path (i.e., distance computation) is an important building block. For example:

- Maximum flows [Edmonds-Karp’72] [Dinitz’70],
- Embeddings (embedding a graph into L1 [Bourgain’85]),
- Clustering (doing low-diameter decompositions [MPX’13]),
- Etc.
Shortest path (i.e., distance computation) is an important building block. For example:

- Maximum flows [Edmonds-Karp’72] [Dinitz’70],
- Embeddings (embedding a graph into L1 [Bourgain’85]),
- Clustering (doing low-diameter decompositions [MPX’13]),
- Etc.

**Warning!**

But these algorithms typically assume **exact** distance computations. This is easy in the sequential model (e.g., Dijkstra = simple and near linear).
Shortest path (i.e., distance computation) is an important building block. For example:

- Maximum flows [Edmonds-Karp’72] [Dinitz’70],
- Embeddings (embedding a graph into L1 [Bourgain’85]),
- Clustering (doing low-diameter decompositions [MPX’13]),
- Etc.

Warning!

But these algorithms typically assume **exact** distance computations. This is easy in the sequential model (e.g., Dijkstra = simple and near linear).

But in other models (parallel, distributed, streaming, etc.) exact distances are notoriously hard to compute. **Approximate** distances are much easier. But the above applications **typically fail** with (weak) approximations.
Our Contributions

1 Definitions: Introduce new stronger notions of distance approximations.

Main Result: Find efficient algorithms to construct them from weak (usual) distance approximations.

Consequence: Give the first $O\left(\frac{m}{\sqrt{n}}\right)$-work sublinear-depth parallel exact SSSP algorithm.

Not in this talk!

Our result: $O\left(\frac{m}{\sqrt{n}}\right)$-work and $O\left(\sqrt{n}\right)$ depth.

Same result achieved independently by [Cao, Fineman'23].
Our Contributions

1. **Definitions**: Introduce new stronger notions of distance approximations.

2. **Main Result**: Find efficient algorithms to construct them from weak (usual) distance approximations.

...Not in this talk!

Our result: $bO(m)$ work and $bO(\sqrt{n})$ depth.

Same result achieved independently by [Cao, Fineman'23].
Our Contributions

1 Definitions: Introduce new stronger notions of distance approximations.

2 Main Result: Find efficient algorithms to construct them from weak (usual) distance approximations.

3 Consequence: Give the first $\tilde{O}(m)$-work sublinear-depth parallel exact SSSP algorithm.
   - Not in this talk!
   - Our result: $\tilde{O}(m)$ work and $\tilde{O}(\sqrt{n})$ depth.
   - Same result achieved independently by [Cao, Fineman’23].
1 Introduction

2 New Stronger Notions of Distance Approximations.
   - Stronger Notion: Smoothness
   - Stronger Notion: Tree-likeness
   - Smoothness + Tree-likeness = <3

3 Efficient Constructions: Lifting Weak Approximations to Smooth Ones

4 Conclusion
Stronger Notion: Smoothness

Definition
A function (called "distance estimate") \(\tilde{d}: V \to \mathbb{R} \geq 0\) is a smooth \((1+\varepsilon)\)-approximation (with respect to a source \(s \in V\)) if:

\[
\tilde{d}(s) = 0 \quad \text{and} \quad \forall u, v \in V | \tilde{d}(u) - \tilde{d}(v) | \leq (1+\varepsilon) \text{dist}(u, v).
\]

Example: \(G = \) a path graph (from left to right). Source = leftmost node.
Definition

A function (called “distance estimate”) \( \tilde{d} : V \to \mathbb{R}_{\geq 0} \) is a smooth \((1 + \varepsilon)\)-approximation (with respect to a source \( s \in V \)) if:

\[
\tilde{d}(s) = 0 \quad \text{and} 
\]

\[
\forall u, v \in V \quad |\tilde{d}(u) - \tilde{d}(v)| \leq (1 + \varepsilon) \text{dist}(u, v).
\]
**Definition**

A function (called “distance estimate”) \( \tilde{d} : V \rightarrow \mathbb{R}_{\geq 0} \) is a smooth \((1 + \varepsilon)\)-approximation (with respect to a source \( s \in V \)) if:

\[
\tilde{d}(s) = 0 \text{ and } \forall u, v \in V \quad |\tilde{d}(u) - \tilde{d}(v)| \leq (1 + \varepsilon)\text{dist}(u, v).
\]

**Example:** \( G \) = a path graph (from left to right). Source = leftmost node.
Stronger Notion: Tree-likeness

Definition (Informal)

\( \tilde{d} : V \rightarrow \mathbb{R} \geq 0 \) is tree-like if:

1. \( \tilde{d}(s) = 0 \), and
2. every other node \( v \) has a neighbor \( u \) whose estimate \( \tilde{d} \) is smaller by at least \( w(u, v) \).

(Check the full talk for formal details.)
Definition (Informal)

\( \tilde{d} : V \rightarrow \mathbb{R}_{\geq 0} \) is tree-like if:

- \( \tilde{d}(s) = 0 \), and
- every other node \( v \) has a neighbor \( u \) whose estimate \( \tilde{d} \) is smaller by at least \( w(u, v) \).

(Check the full talk for formal details.)
Theorem

The following two are equivalent:

\( \tilde{d} \) is a smooth and tree-like \((1 + \epsilon)\) approximation (from source \(s\)).

Given weights \(w: E \rightarrow \mathbb{R} \geq 0\), there exists a perturbation \(w'\) such that

\[ \tilde{d}(v) = \text{dist}_{w'}(\text{source} = s, v) \]
Theorem

The following two are equivalent:

- \( \tilde{d} \) is a smooth and tree-like \((1 + \varepsilon)\) approximation (from source \( s \)).
Theorem

The following two are equivalent:

- \( \tilde{d} \) is a smooth and tree-like \((1 + \varepsilon)\) approximation (from source \(s\)).

- Given weights \(w : E \rightarrow \mathbb{R}_{\geq 0}\), there exists a perturbation \(w'(e) \in [w(e), (1 + \varepsilon)w(e)]\) such that \(\tilde{d}(v) = \text{dist}_{w'}(\text{source} = s, v)\).
Introduction

New Stronger Notions of Distance Approximations.

Efficient Constructions: Lifting Weak Approximations to Smooth Ones

- Iterative Smoothing
- \((\alpha, \delta) \rightarrow (\alpha \cdot (1 + \frac{\varepsilon}{O(\log n)}), \delta/2)\)

Conclusion
Efficient Constructions: Lifting Weak Approximations to Smooth Ones

Theorem

Suppose we have an oracle that computes weak \((1 + \varepsilon)\)-approximate distances.

There is an efficient algorithm that calls the oracle \(O(\log n)\) times, asks for weak \((1 + \varepsilon/O(\log n))\)-approximations on different graphs, and computes \((1 + \varepsilon)\)-approximate smooth distances.

- Same for tree-likeness.
- I will only the main ideas behind efficiently turning weak \(\rightarrow\) smooth.
Efficient Constructions: Lifting Weak Approximations to Smooth Ones

Goal: \( \forall u, v \in V \quad |\tilde{d}(u) - \tilde{d}(v)| \leq (1 + \varepsilon)\text{dist}(u, v). \)

Definition

A function \( \tilde{d} \) is \((\alpha, \delta)\)-smooth if:

\[ \forall u, v \in V \quad |\tilde{d}(u) - \tilde{d}(v)| \leq (\alpha)\text{dist}(u, v) + \delta. \]
Goal: \( \forall u, v \in V \quad |\tilde{d}(u) - \tilde{d}(v)| \leq (1 + \varepsilon)\text{dist}(u, v). \)

**Definition**

A function \( \tilde{d} \) is \((\alpha, \delta)\)-smooth if:

\[
\forall u, v \in V \quad |\tilde{d}(u) - \tilde{d}(v)| \leq (\alpha)\text{dist}(u, v) + \delta.
\]

I will present an efficient algorithm that will transform \( \tilde{d} \) from

\[
(\alpha, \delta) \rightarrow (\alpha \cdot \left(1 + \frac{\varepsilon}{O(\log n)}\right), \delta/2).
\]

Then we would be done! \((1, n^{100}) \rightarrow \ldots \rightarrow (1 + \varepsilon, \frac{1}{n^{100}})\).
\((\alpha, \delta) \rightarrow (\alpha \cdot (1 + \frac{\epsilon}{O(\log n)}), \delta/2)\)

**Algorithm** Slow Partial Smoothing algorithm (\(n\) oracle calls)

1. Let \(G'\) be the graph \(G\) with distances multiplied by \((1 + \frac{\epsilon}{2\log n})\alpha\).
2. \(\tilde{d} \leftarrow O(G, \text{source} = s, \text{approx} = 1 + \frac{\epsilon}{\log n})\)
3. for each \(u \in V(G)\) do
4. \(\tilde{d}_u \leftarrow O(G', \text{source} = u, \text{approx} = \frac{\epsilon}{10\log n})\)
5. \(\tilde{d}_u(\cdot) \leftarrow \tilde{d}(u) + \tilde{d}_u(\cdot)\)
6. return \(\tilde{d}_\star(\cdot) = \min_{u \in V(G)} \tilde{d}_u(\cdot)\)
$(\alpha, \delta) \rightarrow (\alpha \cdot (1 + \frac{\varepsilon}{O(\log n)}), \delta/2)$

**Algorithm** Slow Partial Smoothing algorithm ($n$ oracle calls)

1. Let $G'$ be the graph $G$ with distances multiplied by $(1 + \frac{\varepsilon}{2 \log n})\alpha$.
2. $\tilde{d} \leftarrow O(G, \text{source} = s, \text{approx} = 1 + \frac{\varepsilon}{\log n})$
3. for each $u \in V(G)$ do
   4. $\tilde{d}_u \leftarrow O(G', \text{source} = u, \text{approx} = \frac{\varepsilon}{10 \log n})$
   5. $\tilde{d}_u(\cdot) \leftarrow \tilde{d}(u) + \tilde{d}_u(\cdot)$
6. return $\tilde{d}_*(\cdot) = \min_{u \in V(G)} \tilde{d}_u(\cdot)$

**Intuition**

*When looking at nodes at most $\frac{\delta \log n}{\varepsilon}$ close to $u$, they are already $(\alpha \cdot (1 + \frac{\varepsilon}{O(\log n)}), \delta/2)$-smooth.*
\[(\alpha, \delta) \rightarrow (\alpha \cdot (1 + \frac{\varepsilon}{O(\log n)}), \delta/2)\]

**Algorithm** Slow Partial Smoothing algorithm (\(n\) oracle calls)

1. Let \(G'\) be the graph \(G\) with distances multiplied by \((1 + \frac{\varepsilon}{2\log n})\alpha\).
2. \(\tilde{d} \leftarrow O(G, \text{source} = s, \text{approx} = 1 + \frac{\varepsilon}{\log n})\)
3. for each \(u \in V(G)\) do
   4. \(\tilde{d}_u \leftarrow O(G', \text{source} = u, \text{approx} = \frac{\varepsilon}{10\log n})\)
   5. \(\tilde{d}_u(\cdot) \leftarrow \tilde{d}(u) + \tilde{d}_u(\cdot)\)
6. return \(\tilde{d}_*(\cdot) = \min_{u \in V(G)} \tilde{d}_u(\cdot)\)

**Intuition**

*When looking at nodes at most \(\frac{\delta\log n}{\varepsilon}\) close to \(u\), they are already \((\alpha \cdot (1 + \frac{\varepsilon}{O(\log n)}), \delta/2)\)-smooth.*

*When \(\text{dist}(u, v) > \frac{\delta\log n}{\varepsilon}\), then \(\tilde{d}_u(v) > \tilde{d}(v)\).*
$(\alpha, \delta) \rightarrow (\alpha \cdot (1 + \frac{\varepsilon}{O(\log n)}), \delta/2)$

**Q:** How to reduce the number of oracle calls from $n$ to $O(1)$?

**A:** (Carefully) carve out the graph into strips of width $\omega := \frac{10\delta \log n}{\varepsilon}$. Connect all nodes to the source. Call the oracle. (And fix certain kind of mistakes.)
1 Introduction

2 New Stronger Notions of Distance Approximations.
   - Stronger Notion: Smoothness
   - Stronger Notion: Tree-likeness
   - Smoothness + Tree-likeness = <3

3 Efficient Constructions: Lifting Weak Approximations to Smooth Ones
   - Iterative Smoothing
     \[(\alpha, \delta) \rightarrow (\alpha \cdot (1 + \frac{\epsilon}{O(\log n)}), \delta/2)\]

4 Conclusion

Thank you!