Almost Shortest Paths in Streaming and Distributed Models

Michael Elkin and Jian Zhang

Yale University

Supervised by Joan Feigenbaum
Almost Shortest Paths

**Objective:** compute paths and distances in graphs.

Compromise *precision* for *efficiency.*
The Streaming Model

Massive input.

Allowed to store just a small portion of the input and scan the input only a small number of times.

The model is due to
[Alon, Matias, Szegedy, 99],
[Feigenbaum, Kannan, Strauss, Viswanathan, 02].
Results for the Streaming Model

Previous:
[Feigenbaum, Kannan, McGregor, Suri, Zhang, ICALP 2004]

approximation: $\kappa$
space: $O(n^{1+1/\kappa})$
number of passes: 1
processing time/edge: $O(n^{1+1/\kappa})$

Ours:
approximation: $(1 + \epsilon, \beta)$
space: $O(n^{1+1/\kappa})$
number of passes: $O(1)$
processing time/edge: $O(n^{1/\kappa})$
Our Results for the Streaming Model

Our algorithm returns estimates 
\{\delta(u, w) \mid u, w \in V\} \ ((\text{where } G = (V, E)), 
\text{s.t. } \delta(u, w) \leq (1 + \epsilon)\operatorname{dist}_G(u, w) + \beta.

\beta = \beta(\epsilon, \kappa).
\frac{1}{\kappa}, \epsilon > 0 \text{ are arbitrarily small parameters.}
\beta \text{ is constant whenever } \epsilon \text{ and } \kappa \text{ are.}
Our Results: Cont.

The algorithm also returns paths $P(u, w)$ of length $\delta(u, w)$.

*Distances that are greater than constant are approximated arbitrarily well.*

*Lower bound:* Any streaming algorithm for $o(n)$-approximate distance computation requires $\Omega(n)$ bits of space.
Distributed Computing (Internet)

Each vertex hosts a processor.

Processors exchange data over links (graph edges).

They do it in *synchronous rounds*. (Our algorithm applies to *asynchronous* networks as well.)

Efficiency Measures:

1) **Time**: Number of rounds.

2) **Communication**: Number of messages.
Distributed Model: Previous Results

[Elkin PODC’01]

$(1 + \epsilon, \beta)$-approximation.

Time: $O(n^{1+1/\kappa} + s \cdot \Lambda)$, where $\Lambda$ is the diameter of the network, and the paths and estimates are computed for the set of pairs $S \times V$, $|S| = s$.

Communication: $O(|E| \cdot n^{1/\kappa} + s \cdot n^{1+1/\kappa})$. 
Distributed Model: Our Results

$(1 + \epsilon, \beta)$-approximation.

Time: $O(n^{1/\kappa} + s \cdot \Lambda)$.

Communication: $O(|E| \cdot n^{1/\kappa} + s \cdot n^{1+1/\kappa})$.

Also, we devise a more efficient distributed construction of spanners.
Spanners

An \((\alpha, \beta)-\text{spanner}\) \(G'\) of a given graph \(G = (V, E)\) is an edge-subgraph of \(G\) s.t., for every \(u, w \in V\),

\[
\text{dist}_{G'}(u, w) \leq \alpha \cdot \text{dist}_G(u, w) + \beta.
\]

A sparse skeleton:

Red Subgraph

is a \((3,0)\)-spanner.
Applications of Spanners

- Almost shortest paths.
- Routing.
- Network design.
- Synchronizers.
- Approximation algorithms.
Spanners: Our Results

1) The \textit{first streaming algorithm} for constructing \((1 + \epsilon, \beta)\)-spanners.

2) An improved \textit{distributed algorithm} for constructing \((1 + \epsilon, \beta)\)-spanners.
(Its running time is \(O(n^{1/\kappa})\) instead of \(O(n^{1+1/\kappa})\).)

\[ \downarrow \]

Applications to computing almost shortest paths in both \textit{streaming} and \textit{distributed} models.