Distributed Approximate Single-Source Shortest Paths

Sebastian Krinninger

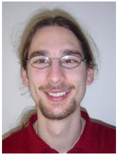
University of Vienna

joint works with

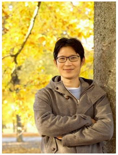
Ruben Becker

Monika Henzinger

Andreas Karrenbauer



Christoph Lenzen



Danupon Nanongkai

One Problem – Two Results

Distributed $(1+\varepsilon)$-approximate single-source shortest paths (SSSP)

One Problem – Two Results

Distributed $(1+\varepsilon)$-approximate single-source shortest paths (SSSP)

1. Deterministically compute approximate shortest paths in $(\sqrt{n} + Diam) \cdot n^{o(1)}$ rounds for $\varepsilon \geq 1/polylog(n)$ [Henzinger/K/Nanongkai 16]

One Problem – Two Results

Distributed $(1+\varepsilon)$**-approximate single-source shortest paths (SSSP)**

1. Deterministically compute approximate shortest paths in $(\sqrt{n} + \mathit{Diam}) \cdot n^{o(1)}$ rounds for $\varepsilon \geq 1/\mathit{polylog}(n)$ [Henzinger/K/Nanongkai 16]
2. Deterministically compute approximate shortest paths in $(\sqrt{n} + \mathit{Diam}) \cdot \mathit{poly}(\log n, \varepsilon)$ rounds [Becker/Lenzen/Karrenbauer/K 16]

One Problem – Two Results

Distributed $(1+\varepsilon)$**-approximate single-source shortest paths (SSSP)**

1. Deterministically compute approximate shortest paths in $(\sqrt{n} + \mathit{Diam}) \cdot n^{o(1)}$ rounds for $\varepsilon \geq 1/\mathit{polylog}(n)$ [Henzinger/K/Nanongkai 16]
2. Deterministically compute approximate shortest paths in $(\sqrt{n} + \mathit{Diam}) \cdot \mathit{poly}(\log n, \varepsilon)$ rounds [Becker/Lenzen/Karrenbauer/K 16]

Comparison:

- Lower bound: $\tilde{\Omega}(\sqrt{n} + \mathit{Diam})$ rounds [Das Sarma et al '11]
- Exact SSSP: $O((n \log n)^{2/3} \mathit{Diam}^{1/3})$ rounds (randomized) [Elkin '17]
- $1+\varepsilon$: $O(n^{1/2} \mathit{Diam}^{1/4} + \mathit{Diam})$ (randomized) [Nanongkai '14]

One Problem – Two Results

Distributed $(1+\varepsilon)$**-approximate single-source shortest paths (SSSP)**

1. Deterministically compute approximate shortest paths in $(\sqrt{n} + \mathit{Diam}) \cdot n^{o(1)}$ rounds for $\varepsilon \geq 1/\mathit{polylog}(n)$ [Henzinger/K/Nanongkai 16]
2. Deterministically compute approximate shortest paths in $(\sqrt{n} + \mathit{Diam}) \cdot \mathit{poly}(\log n, \varepsilon)$ rounds [Becker/Lenzen/Karrenbauer/K 16]

Comparison:

- Lower bound: $\tilde{\Omega}(\sqrt{n} + \mathit{Diam})$ rounds [Das Sarma et al '11]
- Exact SSSP: $O((n \log n)^{2/3} \mathit{Diam}^{1/3})$ rounds (randomized) [Elkin '17]
- $1+\varepsilon$: $O(n^{1/2} \mathit{Diam}^{1/4} + \mathit{Diam})$ (randomized) [Nanongkai '14]

Today: Weighted undirected graphs

Tight and Tighter

Tight and Tighter

Combinatorics & Optimization

Model and Problem Statement

Many distributed models measure amount of communication
Running time = number of rounds

Model and Problem Statement

Many distributed models measure amount of communication
Running time = number of rounds

CONGEST model:

- Synchronous rounds (global clock)
- Message size $O(\log n)$
- In each round, every node sends (at most) one message to each neighbor
- Local computation is free

Model and Problem Statement

Many distributed models measure amount of communication
Running time = number of rounds

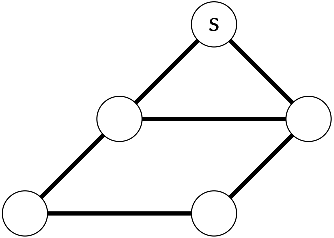
CONGEST model:

- Synchronous rounds (global clock)
- Message size $O(\log n)$
- In each round, every node sends (at most) one message to each neighbor
- Local computation is free

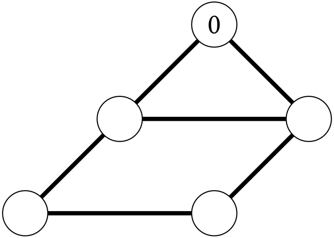
Problem statement:

- Initially, each node knows whether it is the source or not
- Finally: Every node knows its approximate distance to the source
  Often also: Implicit tree; every node knows next edge on approximate shortest path to source

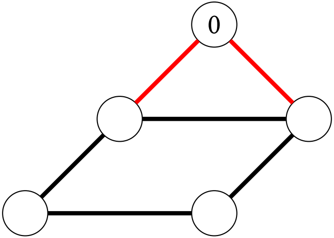
Unweighted Graphs: BFS



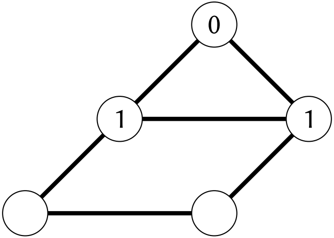
Unweighted Graphs: BFS



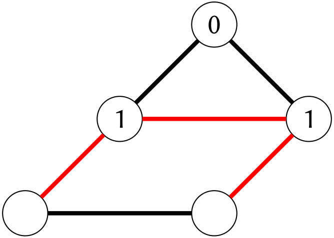
Unweighted Graphs: BFS



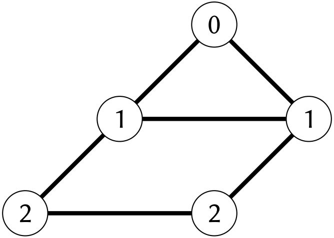
Unweighted Graphs: BFS



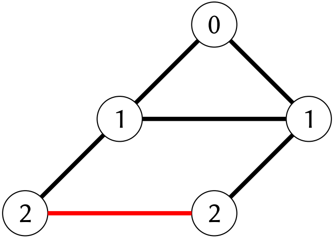
Unweighted Graphs: BFS



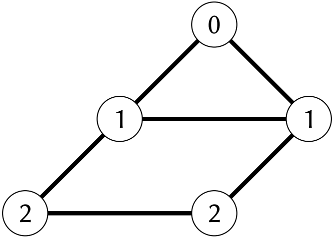
Unweighted Graphs: BFS



Unweighted Graphs: BFS



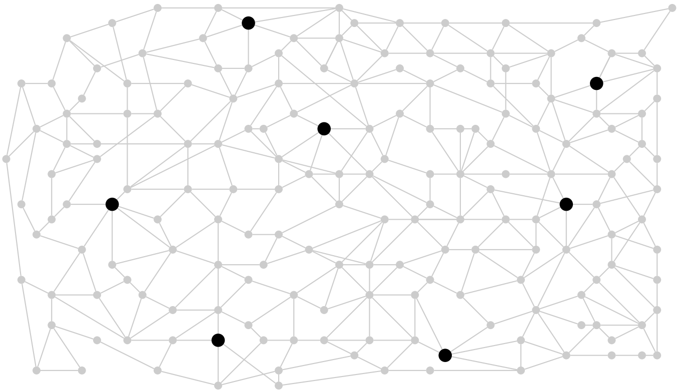
Unweighted Graphs: BFS



BFS tree can be computed in $O(Diam)$ rounds

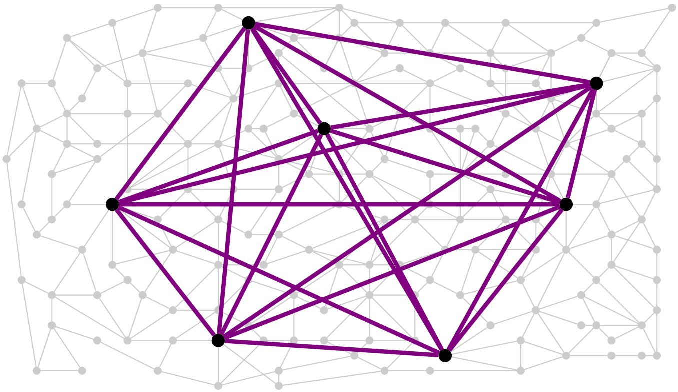
Reduce to SSSP on Overlay Network [Nanongkai '14]

Reduce to SSSP on Overlay Network [Nanongkai '14]



1. Solve SSSP on overlay network and make global knowledge
2. Combine local knowledge of local neighborhoods with global knowledge

Reduce to SSSP on Overlay Network [Nanongkai '14]

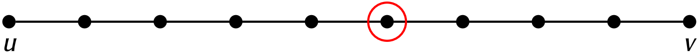


1. Solve SSSP on overlay network and make global knowledge
2. Combine local knowledge of local neighborhoods with global knowledge

Sample $N = \tilde{O}(n^{1/2})$ centers (+ source s)
$\Rightarrow$ Every shortest path with $\geq n^{1/2}$ edges contains center whp

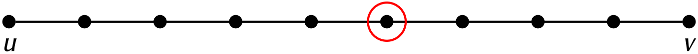
Derandomization of Overlay Network [HKN '16]

Randomization: Hit every shortest path with $\geq \sqrt{n}$ edges

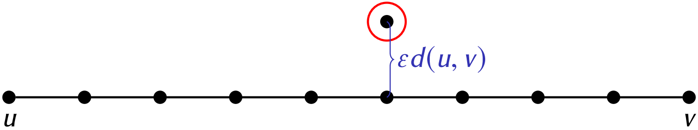


Derandomization of Overlay Network [HKN '16]

Randomization: Hit every shortest path with $\geq \sqrt{n}$ edges

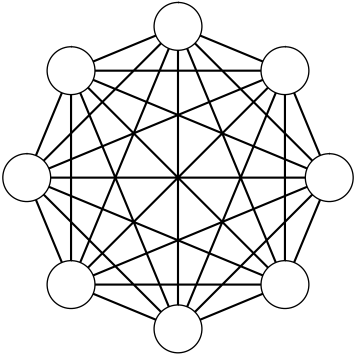


Deterministic relaxation: Almost hit every path $\geq \sqrt{n}$ edges



Congested Clique

Special model: Communication not restricted to neighbors

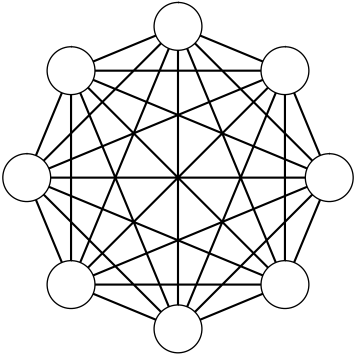


In each round, each node can send one message to each other node

Heavily studied in recent years!

Congested Clique

Special model: Communication not restricted to neighbors



In each round, each node can send one message to each other node

Heavily studied in recent years!

Simulation: Overlay network as congested clique

t rounds in Congested Clique $\rightarrow \tilde{O}(t \cdot (\sqrt{n} + Diam))$ rounds in CONGEST

Hop Reduction

Well Known: Spanners

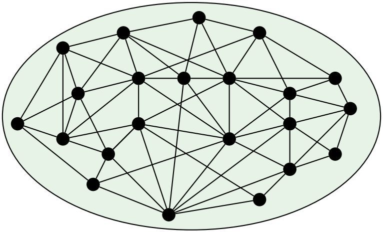
Definition

A k-spanner is a subgraph H of G such that, for all pairs of nodes u and v, $dist_H(u, v) \leq k \cdot dist_G(u, v)$.

Well Known: Spanners

Definition

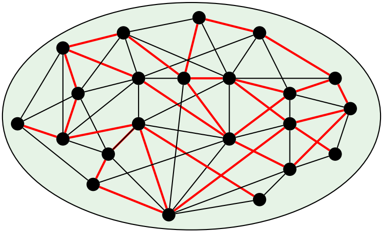
A k-spanner is a subgraph H of G such that, for all pairs of nodes u and v, $dist_H(u, v) \leq k \cdot dist_G(u, v)$.



Well Known: Spanners

Definition

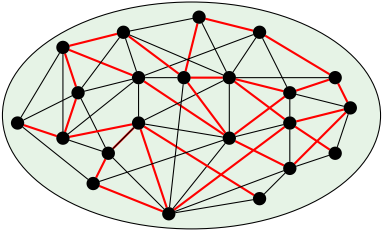
A k-spanner is a subgraph H of G such that, for all pairs of nodes u and v, $dist_H(u, v) \leq k \cdot dist_G(u, v)$.



Well Known: Spanners

Definition

A k-spanner is a subgraph H of G such that, for all pairs of nodes u and v, $dist_H(u, v) \leq k \cdot dist_G(u, v)$.



Fact: Every graph has a k-spanner of size $n^{1+1/k}$ [Folklore]

Application: Running time $T(m, n) \Rightarrow T(n^{1+1/k}, n)$

Less Known: Hop Sets

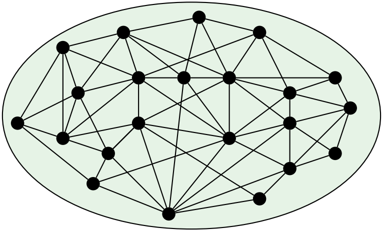
Definition

An (h, ε)-hop set is a set of weighted edges F such that, for all pairs of nodes u and v, in the 'shortcut graph' $G \cup F$ there is a path from u to v with **at most h edges** of weight at most $(1 + \varepsilon)dist(u, v)$.

Less Known: Hop Sets

Definition

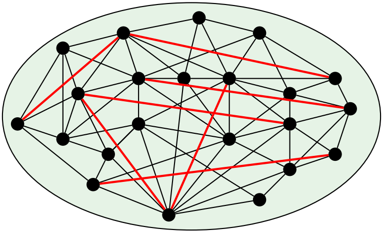
An (h, ε)-hop set is a set of weighted edges F such that, for all pairs of nodes u and v, in the 'shortcut graph' $G \cup F$ there is a path from u to v with **at most h edges** of weight at most $(1 + \varepsilon)dist(u, v)$.



Less Known: Hop Sets

Definition

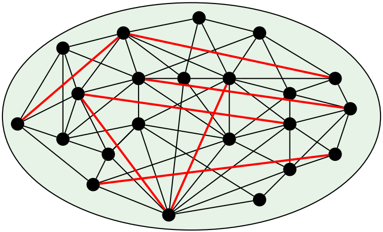
An (h, ε)-hop set is a set of weighted edges F such that, for all pairs of nodes u and v, in the 'shortcut graph' $G \cup F$ there is a path from u to v with **at most h edges** of weight at most $(1 + \varepsilon)dist(u, v)$.



Less Known: Hop Sets

Definition

An (h, ε)-hop set is a set of weighted edges F such that, for all pairs of nodes u and v, in the 'shortcut graph' $G \cup F$ there is a path from u to v with **at most h edges** of weight at most $(1 + \varepsilon)dist(u, v)$.



Fact: Every graph has a $(n^{o(1)}, \varepsilon)$-hop set of size $n^{1+o(1)}$ [Cohen '94] (for $\varepsilon \geq 1/polylogn$)

Less Known: Hop Sets

Definition

An (h, ε)-hop set is a set of weighted edges F such that, for all pairs of nodes u and v, in the 'shortcut graph' $G \cup F$ there is a path from u to v with **at most h edges** of weight at most $(1 + \varepsilon)dist(u, v)$.

Application?

SSSP up to h hops (Bellman-Ford)

- RAM: $O(mh)$ time

Less Known: Hop Sets

Definition

An (h, ε)-hop set is a set of weighted edges F such that, for all pairs of nodes u and v, in the 'shortcut graph' $G \cup F$ there is a path from u to v with **at most h edges** of weight at most $(1 + \varepsilon)dist(u, v)$.

Application?

SSSP up to h hops (Bellman-Ford)

- RAM: $O(mh)$ time
- PRAM: $O(mh)$ with $O(h)$ depth

Less Known: Hop Sets

Definition

An (h, ε)-hop set is a set of weighted edges F such that, for all pairs of nodes u and v, in the 'shortcut graph' $G \cup F$ there is a path from u to v with **at most h edges** of weight at most $(1 + \varepsilon)dist(u, v)$.

Application?

SSSP up to h hops (Bellman-Ford)

- RAM: $O(mh)$ time
- PRAM: $O(mh)$ with $O(h)$ depth
- Congested Clique: $O(h)$ rounds

Less Known: Hop Sets

Definition

An (h, ε)-hop set is a set of weighted edges F such that, for all pairs of nodes u and v, in the 'shortcut graph' $G \cup F$ there is a path from u to v with **at most h edges** of weight at most $(1 + \varepsilon)dist(u, v)$.

Application?

SSSP up to h hops (Bellman-Ford)

- RAM: $O(mh)$ time
- PRAM: $O(mh)$ with $O(h)$ depth
- Congested Clique: $O(h)$ rounds
- Streaming: h passes with $O(n)$ space

Less Known: Hop Sets

Definition

An (h, ε)-hop set is a set of weighted edges F such that, for all pairs of nodes u and v, in the 'shortcut graph' $G \cup F$ there is a path from u to v with **at most h edges** of weight at most $(1 + \varepsilon)dist(u, v)$.

Application?

SSSP up to h hops (Bellman-Ford)

- RAM: $O(mh)$ time
- PRAM: $O(mh)$ with $O(h)$ depth
- Congested Clique: $O(h)$ rounds
- Streaming: h passes with $O(n)$ space
- Incremental/Decremental $O(mh)$ [Even/Shiloach '81, HKN '14]

Less Known: Hop Sets

Definition

An (h, ε)-hop set is a set of weighted edges F such that, for all pairs of nodes u and v, in the 'shortcut graph' $G \cup F$ there is a path from u to v with **at most h edges** of weight at most $(1+\varepsilon)dist(u, v)$.

Application?

SSSP up to h hops (Bellman-Ford)

- RAM: $O(mh)$ time
- PRAM: $O(mh)$ with $O(h)$ depth
- Congested Clique: $O(h)$ rounds
- Streaming: h passes with $O(n)$ space
- Incremental/Decremental $O(mh)$ [Even/Shiloach '81, HKN '14]

Hopset with $h = n^{o(1)}$ and size $n^{1+o(1)}$ gives almost tight algorithms

Less Known: Hop Sets

Definition

An (h, ε)-hop set is a set of weighted edges F such that, for all pairs of nodes u and v, in the 'shortcut graph' $G \cup F$ there is a path from u to v with **at most h edges** of weight at most $(1 + \varepsilon)dist(u, v)$.

Application?

SSSP up to h hops (Bellman-Ford)

- RAM: $O(mh)$ time
- PRAM: $O(mh)$ with $O(h)$ depth
- Congested Clique: $O(h)$ rounds
- Streaming: h passes with $O(n)$ space
- Incremental/Decremental $O(mh)$ [Even/Shiloach '81, HKN '14]

Hopset with $h = n^{o(1)}$ and size $n^{1+o(1)}$ gives almost tight algorithms

Remaining challenge: Compute hop set efficiently

Hop Sets: Approaching Optimality

Authors	Stretch α	Hopbound h	Size
[Baseline]	1	1	$O(n^2)$

Hop Sets: Approaching Optimality

Authors	**Stretch α**	**Hopbound h**	**Size**
[Baseline]	1	1	$O(n^2)$
[Klein/Subramanian '97]	1	$O(\frac{n \log n}{t})$	$O(t^2)$
[Shi/Spencer '99]	1	$O(\frac{n}{t})$	$O(nt)$

Hop Sets: Approaching Optimality

Authors	Stretch α	Hopbound h	Size
[Baseline]	1	1	$O(n^2)$
[Klein/Subramanian '97]	1	$O(\frac{n \log n}{t})$	$O(t^2)$
[Shi/Spencer '99]	1	$O(\frac{n}{t})$	$O(nt)$
[Thorup/Zwick'01]	$2k-1$	2	$O(kn^{1+\frac{1}{k}})$

Hop Sets: Approaching Optimality

Authors	Stretch α	Hopbound h	Size
[Baseline]	1	1	$O(n^2)$
[Klein/Subramanian '97]	1	$O(\frac{n \log n}{t})$	$O(t^2)$
[Shi/Spencer '99]	1	$O(\frac{n}{t})$	$O(nt)$
[Thorup/Zwick'01]	$2k-1$	2	$O(kn^{1+\frac{1}{k}})$
[Cohen'94]	$1+\varepsilon$	$(\frac{\log n}{\varepsilon})^{O(\log k)}$	$O(n^{1+\frac{1}{k}} \log n)$
[Bernstein'09]	$1+\varepsilon$	$O(\frac{3}{\varepsilon})^k \log n$	$O(kn^{1+\frac{1}{k}})$
[Elkin/Neiman'16]	$1+\varepsilon$	$(\frac{\log k}{\varepsilon})^{O(\log k)}$	$O(n^{1+\frac{1}{k}} \log n \log k)$
[Elkin/Neiman'17]	$1+\varepsilon$	$O(\frac{k+1}{\varepsilon})^{k+1}$	$O(n^{1+\frac{1}{2^{k+1}-1}})$
[Huang/Pettie'17]	$1+\varepsilon$	$O(\frac{k}{\varepsilon})^k$	$O(n^{1+\frac{1}{2^{k+1}-1}})$

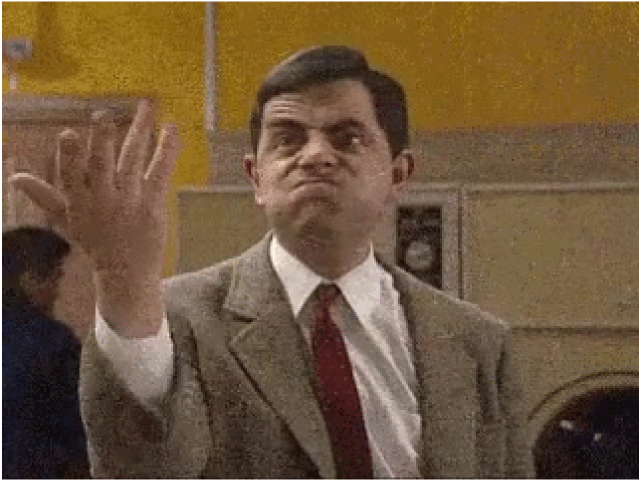
Hop Sets: Approaching Optimality

Authors	Stretch α	Hopbound h	Size
[Baseline]	1	1	$O(n^2)$
[Klein/Subramanian '97]	1	$O(\frac{n \log n}{t})$	$O(t^2)$
[Shi/Spencer '99]	1	$O(\frac{n}{t})$	$O(nt)$
[Thorup/Zwick'01]	$2k-1$	2	$O(kn^{1+\frac{1}{k}})$
[Cohen'94]	$1+\varepsilon$	$(\frac{\log n}{\varepsilon})^{O(\log k)}$	$O(n^{1+\frac{1}{k}} \log n)$
[Bernstein'09]	$1+\varepsilon$	$O(\frac{3}{\varepsilon})^k \log n$	$O(kn^{1+\frac{1}{k}})$
[Elkin/Neiman'16]	$1+\varepsilon$	$(\frac{\log k}{\varepsilon})^{O(\log k)}$	$O(n^{1+\frac{1}{k}} \log n \log k)$
[Elkin/Neiman'17]	$1+\varepsilon$	$O(\frac{k+1}{\varepsilon})^{k+1}$	$O(n^{1+\frac{1}{2^{k+1}-1}})$
[Huang/Pettie'17]	$1+\varepsilon$	$O(\frac{k}{\varepsilon})^k$	$O(n^{1+\frac{1}{2^{k+1}-1}})$
[Abboud/Bodwin/Pettie'16]	$1+\varepsilon$	$\Omega_k(\frac{1}{\varepsilon})^k$	$n^{1+\frac{1}{2^k-1}-\delta}$

Hop Sets: Approaching Optimality

Authors	Stretch α	Hopbound h	Size
[Baseline]	1	1	$O(n^2)$
[Klein/Subramanian '97]	1	$O(\frac{n \log n}{t})$	$O(t^2)$
[Shi/Spencer '99]	1	$O(\frac{n}{t})$	$O(nt)$
[Thorup/Zwick'01]	$2k-1$	2	$O(kn^{1+\frac{1}{k}})$
[Cohen'94]	$1+\varepsilon$	$(\frac{\log n}{\varepsilon})^{O(\log k)}$	$O(n^{1+\frac{1}{k}} \log n)$
[Bernstein'09]	$1+\varepsilon$	$O(\frac{3}{\varepsilon})^k \log n$	$O(kn^{1+\frac{1}{k}})$
[Elkin/Neiman'16]	$1+\varepsilon$	$(\frac{\log k}{\varepsilon})^{O(\log k)}$	$O(n^{1+\frac{1}{k}} \log n \log k)$
[Elkin/Neiman'17]	$1+\varepsilon$	$O(\frac{k+1}{\varepsilon})^{k+1}$	$O(n^{1+\frac{1}{2^{k+1}-1}})$
[Huang/Pettie'17]	$1+\varepsilon$	$O(\frac{k}{\varepsilon})^k$	$O(n^{1+\frac{1}{2^{k+1}-1}})$
[Abboud/Bodwin/Pettie'16]	$1+\varepsilon$	$\Omega_k(\frac{1}{\varepsilon})^k$	$n^{1+\frac{1}{2^k-1}-\delta}$

$\Rightarrow$ Cannot have $\alpha = 1+\varepsilon$, $h = poly(1/\varepsilon)$ and size $n \cdot polylog(n)$.



It was too good to be true…

Hop Set Example

Simple Hop Set Based on Balls (following [Thorup/Zwick '06])

$V = A_0 \supseteq A_1 \supseteq \cdots \supseteq A_k = \emptyset$ where node of A_i goes to A_{i+1} with probability $1/n^{1/k}$

v has **priority** i if $v \in A_i \setminus A_{i+1}$

Simple Hop Set Based on Balls (following [Thorup/Zwick '06])

$V = A_0 \supseteq A_1 \supseteq \cdots \supseteq A_k = \emptyset$ where node of A_i goes to A_{i+1} with probability $1/n^{1/k}$

v has **priority** i if $v \in A_i \setminus A_{i+1}$

For every node u of priority i:

$$Ball(u) = \{v \in V \mid dist(u, v) < dist(u, A_{i+1})\}$$

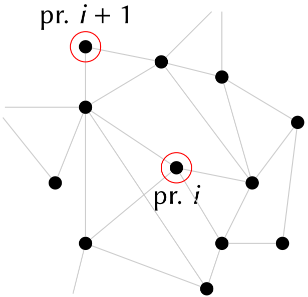
Simple Hop Set Based on Balls (following [Thorup/Zwick ’06])

$V = A_0 \supseteq A_1 \supseteq \cdots \supseteq A_k = \emptyset$ where node of A_i goes to A_{i+1} with probability $1/n^{1/k}$

v has **priority** i if $v \in A_i \setminus A_{i+1}$

For every node u of priority i:

$$Ball(u) = \{v \in V \mid dist(u, v) < dist(u, A_{i+1})\}$$



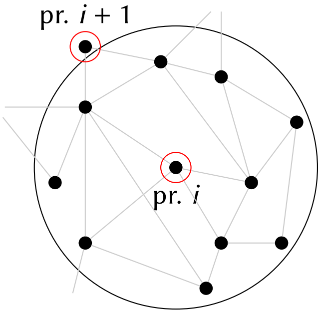
Simple Hop Set Based on Balls (following [Thorup/Zwick '06])

$V = A_0 \supseteq A_1 \supseteq \cdots \supseteq A_k = \emptyset$ where node of A_i goes to A_{i+1} with probability $1/n^{1/k}$

v has **priority** i if $v \in A_i \setminus A_{i+1}$

For every node u of priority i:

$$Ball(u) = \{v \in V \mid dist(u, v) < dist(u, A_{i+1})\}$$



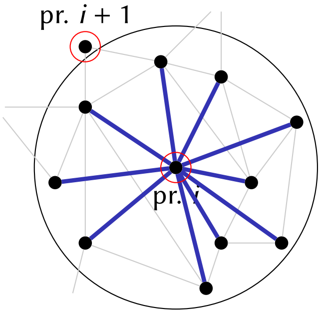
Simple Hop Set Based on Balls (following [Thorup/Zwick '06])

$V = A_0 \supseteq A_1 \supseteq \cdots \supseteq A_k = \emptyset$ where node of A_i goes to A_{i+1} with probability $1/n^{1/k}$

v has **priority** i if $v \in A_i \setminus A_{i+1}$

For every node u of priority i:

$$Ball(u) = \{v \in V \mid dist(u, v) < dist(u, A_{i+1})\}$$



Hop set:

- $(u, v) \in F$ iff $v \in Ball(u)$
- $w(u, v) = dist_G(u, v)$

Simple Hop Set Based on Balls (following [Thorup/Zwick '06])

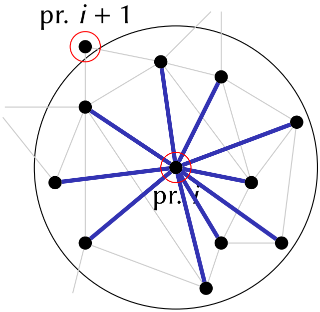
$V = A_0 \supseteq A_1 \supseteq \cdots \supseteq A_k = \emptyset$ where node of A_i goes to A_{i+1} with probability $1/n^{1/k}$

v has **priority** i if $v \in A_i \setminus A_{i+1}$

For every node u of priority i:

$$Ball(u) = \{v \in V \mid dist(u, v) < dist(u, A_{i+1})\}$$

priority	# nodes
0	n
1	$n^{1-1/k}$
$\vdots$	$\vdots$
$k-1$	$n^{1/k}$



Hop set:

- $(u, v) \in F$ iff $v \in Ball(u)$
- $w(u, v) = dist_G(u, v)$

Simple Hop Set Based on Balls (following [Thorup/Zwick '06])

$V = A_0 \supseteq A_1 \supseteq \cdots \supseteq A_k = \emptyset$ where node of A_i goes to A_{i+1} with probability $1/n^{1/k}$

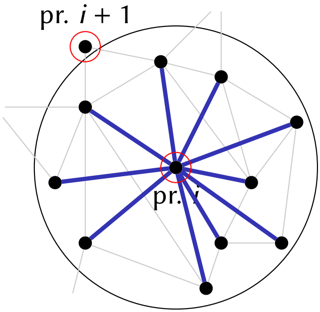
v has **priority** i if $v \in A_i \setminus A_{i+1}$

For every node u of priority i:

$$Ball(u) = \{v \in V \mid dist(u, v) < dist(u, A_{i+1})\}$$

Expected size: $n^{(i+1)/k}$

priority	# nodes	$\lvert Ball(u) \rvert$
0	n	$n^{1/k}$
1	$n^{1-1/k}$	$n^{2/k}$
$\vdots$	$\vdots$	$\vdots$
$k-1$	$n^{1/k}$	n



Hop set:

- $(u, v) \in F$ iff $v \in Ball(u)$
- $w(u, v) = dist_G(u, v)$

Simple Hop Set Based on Balls (following [Thorup/Zwick '06])

$V = A_0 \supseteq A_1 \supseteq \cdots \supseteq A_k = \emptyset$ where node of A_i goes to A_{i+1} with probability $1/n^{1/k}$

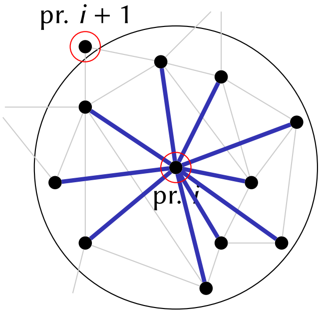
v has **priority** i if $v \in A_i \setminus A_{i+1}$

For every node u of priority i:

$$Ball(u) = \{v \in V \mid dist(u,v) < dist(u, A_{i+1})\}$$

Expected size: $n^{(i+1)/k}$

priority	# nodes	$\lvert Ball(u)\rvert$	# edges
0	n	$n^{1/k}$	$n^{1+1/k}$
1	$n^{1-1/k}$	$n^{2/k}$	$n^{1+1/k}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$k-1$	$n^{1/k}$	n	$n^{1+1/k}$



Hop set:

- $(u,v) \in F$ iff $v \in Ball(u)$
- $w(u,v) = dist_G(u,v)$

Simple Hop Set Based on Balls (following [Thorup/Zwick '06])

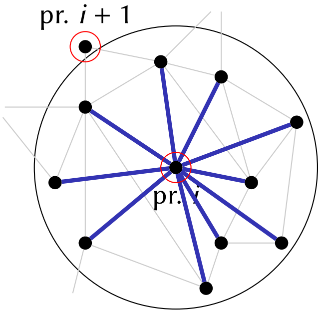
$V = A_0 \supseteq A_1 \supseteq \cdots \supseteq A_k = \emptyset$ where node of A_i goes to A_{i+1} with probability $1/n^{1/k}$

v has **priority** i if $v \in A_i \setminus A_{i+1}$

For every node u of priority i:

$$Ball(u) = \{v \in V \mid dist(u, v) < dist(u, A_{i+1})\}$$

Expected size: $n^{(i+1)/k}$



priority	# nodes	$\lvert Ball(u)\rvert$	# edges
0	n	$n^{1/k}$	$n^{1+1/k}$
1	$n^{1-1/k}$	$n^{2/k}$	$n^{1+1/k}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$k-1$	$n^{1/k}$	n	$n^{1+1/k}$
			$kn^{1+1/k}$

Hop set:

- $(u, v) \in F$ iff $v \in Ball(u)$
- $w(u, v) = dist_G(u, v)$

Parameter Choice

$$k = \frac{\sqrt{\log n}}{\sqrt{\log 4/\varepsilon}}$$

$$\left(\frac{4}{\varepsilon}\right)^k = n^{1/k}$$

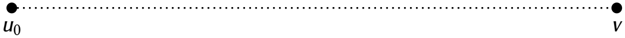
Parameter Choice

$$k = \frac{\sqrt{\log n}}{\sqrt{\log 4/\varepsilon}}$$

$$\left(\frac{4}{\varepsilon}\right)^k = n^{1/k} = n^{o(1)}$$

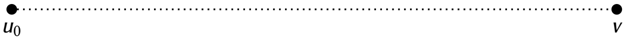
$(n^{1/2+o(1)}, \varepsilon)$-hop set

Case 1: $dist(u_0, v) \leq n^{1/2+1/k}/\varepsilon$



$(n^{1/2+o(1)}, \varepsilon)$-hop set

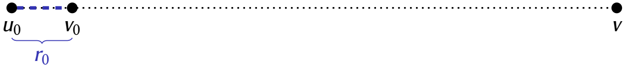
Case 2: $dist(u_0, v) > n^{1/2+1/k}/\varepsilon$



$(n^{1/2+o(1)}, \varepsilon)$-hop set

Case 2: $dist(u_0, v) > n^{1/2+1/k}/\varepsilon$

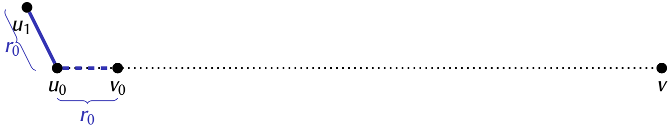
$r_0 = n^{1/2}$



$(n^{1/2+o(1)}, \varepsilon)$-hop set

Case 2: $dist(u_0, v) > n^{1/2+1/k}/\varepsilon$

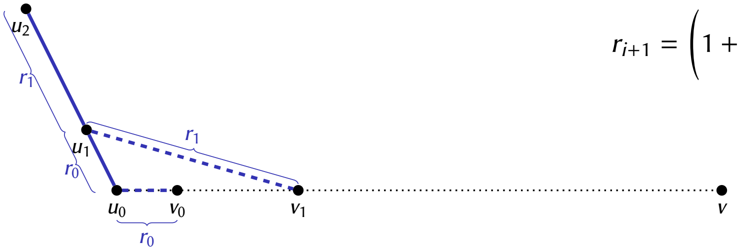
$r_0 = n^{1/2}$



For every node u of priority i and every node v, either $(u, v) \in H$, or $\exists u'$ of priority $i + 1$ s. t. $dist(u, u') \leq dist(u, v)$.

$(n^{1/2+o(1)}, \varepsilon)$-hop set

Case 2: $dist(u_0, v) > n^{1/2+1/k}/\varepsilon$



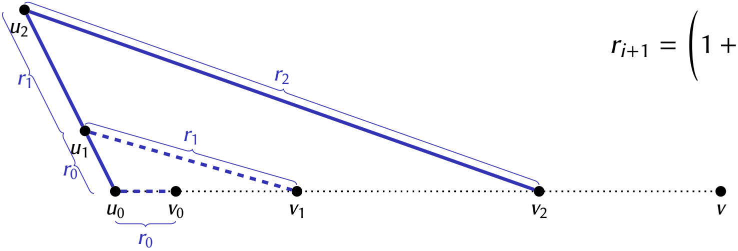
$$r_0 = n^{1/2}$$

$$r_{i+1} = \left(1 + \frac{2}{\varepsilon}\right) \sum_{0 \le j \le i} r_j$$

For every node u of priority i and every node v, either $(u, v) \in H$, or $\exists u'$ of priority $i + 1$ s. t. $dist(u, u') \le dist(u, v)$.

$(n^{1/2+o(1)}, \varepsilon)$-hop set

Case 2: $dist(u_0, v) > n^{1/2+1/k}/\varepsilon$



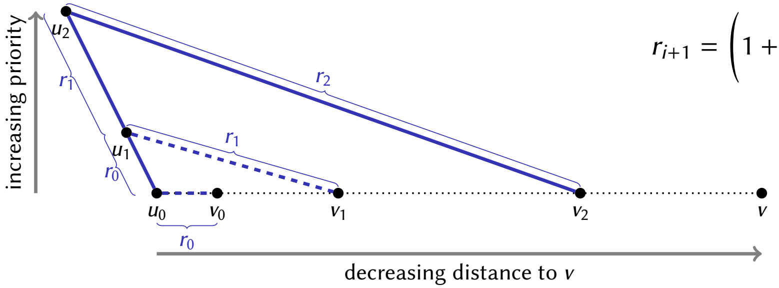
$$r_0 = n^{1/2}$$

$$r_{i+1} = \left(1 + \frac{2}{\varepsilon}\right) \sum_{0 \le j \le i} r_j$$

For every node u of priority i and every node v, either $(u, v) \in H$, or $\exists u'$ of priority $i + 1$ s. t. $dist(u, u') \le dist(u, v)$.

$(n^{1/2+o(1)}, \varepsilon)$-hop set

Case 2: $dist(u_0, v) > n^{1/2+1/k}/\varepsilon$



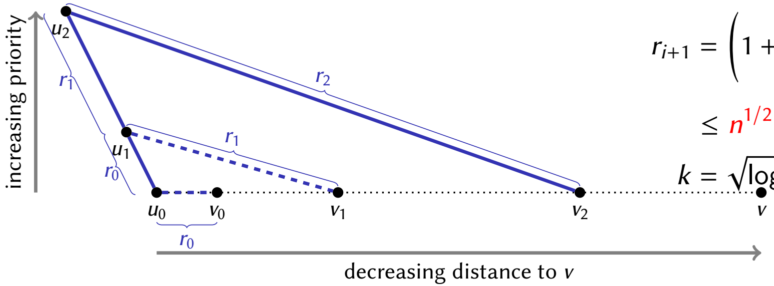
$$r_0 = n^{1/2}$$

$$r_{i+1} = \left(1 + \frac{2}{\varepsilon}\right) \sum_{0 \le j \le i} r_j$$

For every node u of priority i and every node v, either $(u, v) \in H$, or $\exists u'$ of priority $i + 1$ s. t. $dist(u, u') \le dist(u, v)$.

$(n^{1/2+o(1)}, \varepsilon)$-hop set

Case 2: $dist(u_0, v) > n^{1/2+1/k}/\varepsilon$



$$r_0 = n^{1/2}$$

$$r_{i+1} = \left(1 + \frac{2}{\varepsilon}\right) \sum_{0 \le j \le i} r_j$$

$$\le n^{1/2} n^{1/k}$$

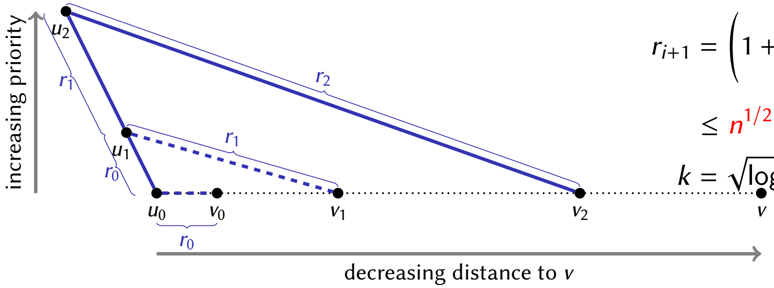
$$k = \sqrt{\log n} / \sqrt{\log 4/\varepsilon}$$

For every node u of priority i and every node v, either $(u, v) \in H$, or $\exists u'$ of priority $i+1$ s. t. $dist(u, u') \le dist(u, v)$.

$$Weight \le (1+\varepsilon) dist(u_0, v)$$

$(n^{1/2+o(1)}, \varepsilon)$-hop set

Case 2: $dist(u_0, v) > n^{1/2+1/k}/\varepsilon$



$$r_0 = n^{1/2}$$

$$r_{i+1} = \left(1 + \frac{2}{\varepsilon}\right) \sum_{0 \le j \le i} r_j$$

$$\le n^{1/2} n^{1/k}$$

$$k = \sqrt{\log n} / \sqrt{\log 4/\varepsilon}$$

For every node u of priority i and every node v, either $(u, v) \in H$, or $\exists u'$ of priority $i + 1$ s. t. $dist(u, u') \le dist(u, v)$.

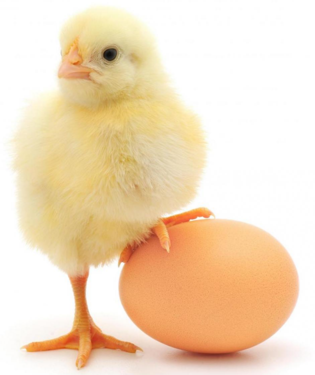
$$\mathit{Weight} \le (1 + \varepsilon) dist(u_0, v)$$

$$\#\mathit{Edges} \le \frac{k \cdot dist(u, v)}{n^{1/2}} \le \frac{k \cdot n}{n^{1/2}} = kn^{1/2}$$

Chicken-Egg Problem?

1. Goal: Faster SSSP via hop set
2. Compute hop set by computing balls
3. Computing balls at least as hard as SSSP

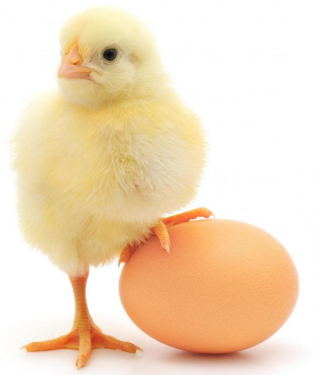
$\Rightarrow$ Back at problem we wanted to solve initially?



Chicken-Egg Problem?

1. Goal: Faster SSSP via hop set
2. Compute hop set by computing balls
3. Computing balls at least as hard as SSSP

$\Rightarrow$ Back at problem we wanted to solve initially?



No! $(n^{1/2+o(1)}, \varepsilon)$-hop set only requires balls up to $n^{1/2+o(1)}$ hops

$(n^{1/2+o(1)}, \varepsilon)$-hop set

Iterative computation

In each iteration number of hops is reduced by a factor of $n^{1/k}$

$(n^{1/2+o(1)}, \varepsilon)$-hop set

Iterative computation

In each iteration number of hops is reduced by a factor of $n^{1/k}$

Algorithm:

for $i = 1$ **to** k **do**

$$H_i = G \cup \bigcup_{1 \leq j \leq i-1} F_j$$

Compute balls with k priorities in H_i up to $n^{2/k}$ hops

$$F_i = \{(u, v) \mid v \in Ball(u)\}$$

end

return $F = \bigcup_{1 \leq i \leq k} F_i$

$(n^{1/2+o(1)}, \varepsilon)$-hop set

Iterative computation
In each iteration number of hops is reduced by a factor of $n^{1/k}$

Algorithm:

for $i = 1$ **to** k **do**

$$H_i = G \cup \bigcup_{1 \le j \le i-1} F_j$$

Compute balls with k priorities in H_i up to $n^{2/k}$ hops

$$F_i = \{(u, v) \mid v \in Ball(u)\}$$

end

return $F = \bigcup_{1 \le i \le k} F_i$

Error amplification: $(1 + \varepsilon')^k \le (1 + \varepsilon)$ for $\varepsilon' = 1/(2\varepsilon \log n)$

$(n^{1/2+o(1)}, \varepsilon)$-hop set

Iterative computation

In each iteration number of hops is reduced by a factor of $n^{1/k}$

Algorithm:

for $i = 1$ **to** k **do**

$$H_i = G \cup \bigcup_{1 \le j \le i-1} F_j$$

Compute balls with k priorities in H_i up to $n^{2/k}$ hops

$$F_i = \{(u, v) \mid v \in Ball(u)\}$$

end

return $F = \bigcup_{1 \le i \le k} F_i$

Error amplification: $(1 + \varepsilon')^k \le (1 + \varepsilon)$ for $\varepsilon' = 1/(2\varepsilon \log n)$

Omitted detail: weighted graphs, use rounding technique

Beyond Hop Sets

New Distributed Algorithm

Theorem ([Becker/Karrenbauer/K/Lenzen arXiv'16])

There is a deterministic algorithm for computing $(1+\varepsilon)$ approximate SSSP in $(\sqrt{n} + Diam) poly(\log n, \varepsilon)$ rounds.

New Distributed Algorithm

Theorem ([Becker/Karrenbauer/K/Lenzen arXiv'16])

There is a deterministic algorithm for computing $(1+\varepsilon)$ *approximate SSSP in* $(\sqrt{n} + \mathit{Diam})\mathit{poly}(\log n, \varepsilon)$ *rounds.*

Key insight: Solve more general problem

Shortest Transshipment Problem

Find the cheapest route for sending units of a single good from sources to sinks along the edges of a graph as specified by demands on nodes.

New Distributed Algorithm

Theorem ([Becker/Karrenbauer/K/Lenzen arXiv'16])

There is a deterministic algorithm for computing $(1+\varepsilon)$ approximate SSSP in $(\sqrt{n} + Diam) poly(\log n, \varepsilon)$ rounds.

Key insight: Solve more general problem

Shortest Transshipment Problem

Find the cheapest route for sending units of a single good from sources to sinks along the edges of a graph as specified by demands on nodes.

"Uncapacitated minimum-cost flow"

New Distributed Algorithm

Theorem ([Becker/Karrenbauer/K/Lenzen arXiv'16])

There is a deterministic algorithm for computing $(1+\varepsilon)$ *approximate SSSP in* $(\sqrt{n} + Diam)poly(\log n, \varepsilon)$ *rounds.*

Key insight: Solve more general problem

Shortest Transshipment Problem

Find the cheapest route for sending units of a single good from sources to sinks along the edges of a graph as specified by demands on nodes.

"Uncapacitated minimum-cost flow"

SSSP: source has demand $-(n-1)$, other nodes have demand 1

Shortest Transshipment Problem

Shortest transshipment as linear program:

$$\text{minimize } \|Wx\|_1 \quad \text{s.t. } Ax = b$$

Shortest Transshipment Problem

Shortest transshipment as linear program:

$$\text{minimize } \|Wx\|_1 \quad \text{s.t. } Ax = b$$

Dual program:

$$\text{maximize } b^T y \quad \text{s.t. } \|W^{-1}A^T y\|_\infty \leq 1$$

Shortest Transshipment Problem

Shortest transshipment as linear program:

$$\text{minimize } \|Wx\|_1 \quad \text{s.t. } Ax = b$$

Dual program:

$$\text{maximize } b^T y \quad \text{s.t. } \|W^{-1}A^T y\|_\infty \leq 1$$

Equivalent:

$$\text{minimize } \|W^{-1}A^T y\|_\infty \quad \text{s.t. } b^T \pi = 1$$

Shortest Transshipment Problem

Shortest transshipment as linear program:

$$\text{minimize } \|Wx\|_1 \quad \text{s.t. } Ax = b$$

Dual program:

$$\text{maximize } b^T y \quad \text{s.t. } \|W^{-1}A^T y\|_\infty \leq 1$$

Equivalent:

$$\text{minimize } \|W^{-1}A^T y\|_\infty \quad \text{s.t. } b^T \pi = 1$$

We approximate $\|\cdot\|_\infty$ by soft-max:

$$\mathsf{lse}_\beta(x) := \frac{1}{\beta} \ln\left(\sum_{i\in[d]} \left(e^{\beta x_i} + e^{-\beta x_i}\right)\right)$$

Gradient Descent

Algorithm at a glance:

1. Soft-max is differentiable $\rightarrow$ apply gradient descent

Gradient Descent

Algorithm at a glance:

1. Soft-max is differentiable $\rightarrow$ apply gradient descent
2. Each iteration: solve transshipment problem with different demand vector b' depending on current gradient

Gradient Descent

Algorithm at a glance:

1. Soft-max is differentiable $\rightarrow$ apply gradient descent
2. Each iteration: solve transshipment problem with different demand vector b' depending on current gradient
3. Key observation: For b', bad approximation is sufficient

Gradient Descent

Algorithm at a glance:

1. Soft-max is differentiable $\rightarrow$ apply gradient descent
2. Each iteration: solve transshipment problem with different demand vector b' depending on current gradient
3. Key observation: For b', bad approximation is sufficient
4. Compute spanner on overlay network and solving transshipment on overlay spanner
   Spanner has stretch $O(\log n)$ and size $\tilde{O}(n)$

Gradient Descent

Algorithm at a glance:

1. Soft-max is differentiable $\rightarrow$ apply gradient descent
2. Each iteration: solve transshipment problem with different demand vector b' depending on current gradient
3. Key observation: For b', bad approximation is sufficient
4. Compute spanner on overlay network and solving transshipment on overlay spanner
   Spanner has stretch $O(\log n)$ and size $\tilde{O}(n)$
5. Overall: Polylog iterations, each solving $O(\log n)$-approximate transshipment on graph of $\tilde{O}(n)$ edges

Gradient Descent

Algorithm at a glance:

1. Soft-max is differentiable $\rightarrow$ apply gradient descent
2. Each iteration: solve transshipment problem with different demand vector b' depending on current gradient
   Congested Clique: Compute gradient in $O(1)$ rounds
3. Key observation: For b', bad approximation is sufficient
4. Compute spanner on overlay network and solving transshipment on overlay spanner
   Spanner has stretch $O(\log n)$ and size $\tilde{O}(n)$
5. Overall: Polylog iterations, each solving $O(\log n)$-approximate transshipment on graph of $\tilde{O}(n)$ edges

Gradient Descent

Algorithm at a glance:

1. Soft-max is differentiable $\rightarrow$ apply gradient descent
2. Each iteration: solve transshipment problem with different demand vector b' depending on current gradient
   Congested Clique: Compute gradient in $O(1)$ rounds
3. Key observation: For b', bad approximation is sufficient
4. Compute spanner on overlay network and solving transshipment on overlay spanner
   Spanner has stretch $O(\log n)$ and size $\tilde{O}(n)$
   Congested Clique: Spanner can be computed in $O(\log n)$ rounds [Baswana/Sen '03]
5. Overall: Polylog iterations, each solving $O(\log n)$-approximate transshipment on graph of $\tilde{O}(n)$ edges

Technical Details

1. Black-box reduction from SSSP to shortest transshipment only for **exact** solutions

Technical Details

1. Black-box reduction from SSSP to shortest transshipment only for **exact** solutions
2. Transshipment will only guarantee $(1 + \varepsilon)$-approximation on average

Technical Details

1. Black-box reduction from SSSP to shortest transshipment only for **exact** solutions
2. Transshipment will only guarantee $(1+\varepsilon)$-approximation on average
3. How to obtain per-node guarantee:
    - Solve with increased precision
    - Inspect gradient to identify “good nodes”
    - Repeat transshipment for “bad” nodes only
    - Analysis: Total “mass” reduced by constant fraction in each run

Technical Details

1. Black-box reduction from SSSP to shortest transshipment only for **exact** solutions
2. Transshipment will only guarantee $(1 + \varepsilon)$-approximation on average
3. How to obtain per-node guarantee:
    - Solve with increased precision
    - Inspect gradient to identify "good nodes"
    - Repeat transshipment for "bad" nodes only
    - Analysis: Total "mass" reduced by constant fraction in each run

Independent work: Approximate transshipment [Sherman '16]

Technical Details

1. Black-box reduction from SSSP to shortest transshipment only for **exact** solutions
2. Transshipment will only guarantee $(1+\varepsilon)$-approximation on average
3. How to obtain per-node guarantee:
    - Solve with increased precision
    - Inspect gradient to identify "good nodes"
    - Repeat transshipment for "bad" nodes only
    - Analysis: Total "mass" reduced by constant fraction in each run

Independent work: Approximate transshipment [Sherman '16]

- More general solvers based on generalized preconditioning

Technical Details

1. Black-box reduction from SSSP to shortest transshipment only for **exact** solutions
2. Transshipment will only guarantee $(1+\varepsilon)$-approximation on average
3. How to obtain per-node guarantee:
   - Solve with increased precision
   - Inspect gradient to identify “good nodes”
   - Repeat transshipment for “bad” nodes only
   - Analysis: Total “mass” reduced by constant fraction in each run

Independent work: Approximate transshipment [Sherman ’16]

- More general solvers based on generalized preconditioning
- Linear preconditioner based on metric embeddings

Technical Details

1. Black-box reduction from SSSP to shortest transshipment only for **exact** solutions
2. Transshipment will only guarantee $(1+\varepsilon)$-approximation on average
3. How to obtain per-node guarantee:
   - Solve with increased precision
   - Inspect gradient to identify "good nodes"
   - Repeat transshipment for "bad" nodes only
   - Analysis: Total "mass" reduced by constant fraction in each run

Independent work: Approximate transshipment [Sherman '16]

- More general solvers based on generalized preconditioning
- Linear preconditioner based on metric embeddings
- With additional analysis: spanner-based oracle as non-linear preconditioner

Technical Details

1. Black-box reduction from SSSP to shortest transshipment only for **exact** solutions
2. Transshipment will only guarantee $(1 + \varepsilon)$-approximation on average
3. How to obtain per-node guarantee:
    - Solve with increased precision
    - Inspect gradient to identify "good nodes"
    - Repeat transshipment for "bad" nodes only
    - Analysis: Total "mass" reduced by constant fraction in each run

Independent work: Approximate transshipment [Sherman '16]

- More general solvers based on generalized preconditioning
- Linear preconditioner based on metric embeddings
- With additional analysis: spanner-based oracle as non-linear preconditioner
- No straightforward way of obtaining per-node guarantee

Conclusion

Main contributions:

- Two almost tight algorithms in distributed and streaming models
- Combinatorial and continuous tools

Conclusion

Main contributions:

- Two almost tight algorithms in distributed and streaming models
- Combinatorial and continuous tools

Open problems:

- PRAM: improve Cohen's $m^{1+o(1)}$ work with polylog depth?
- Deterministic decremental SSSP algorithm

Tight and Tighter