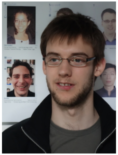
Approximate Single-Source Shortest Paths: Distributed and Dynamic Algorithms

Sebastian Krinninger

Max Planck Institute for Informatics

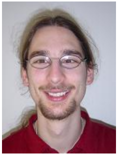
joint works with



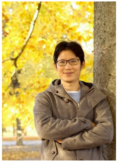
Ruben Becker

Monika Henzinger

Andreas Karrenbauer

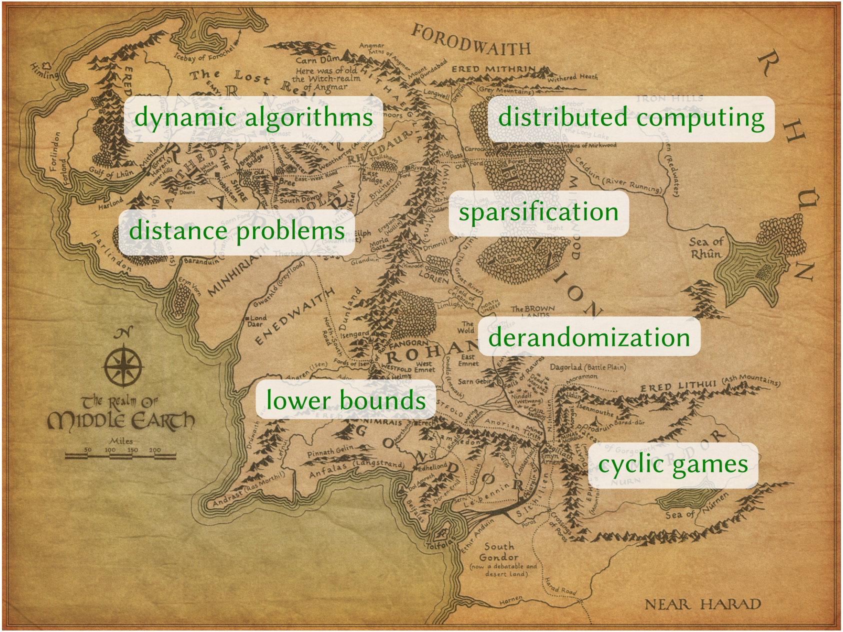


Christoph Lenzen



Danupon Nanongkai


FORODWAITH
RHÛN
Angmar
Carn Dûm
Here was of old the Witch-realm of Angmar
Mount Gundabad
ERED MITHRIN
(Grey Mountains)
Withered Heath
IRON HILLS
Erebor
The Lonely Mountain
Esgaroth
the Long Lake
Wood Elves
The Lost Realm of ARNOR
ERIADOR
ERED LUIN
Himling
Icebay of Forochel
Forlindon
Forlond
Gulf of Lhûn
Mithlond
Harlond
Harlindon
(Blue Mountains)
Fornost
North Downs
Weather Hills
Ettenmoors
Weathertop
RHUDAUR
Trollshaws
Rivendell
High Pass
Carrock
Old Ford
Old Forest Road
East-West Road
Bree
South Downs
THE SHIRE
Hobbiton
Far Downs
White Downs
Tower Hills
Brandywine Bridge
Old Forest
CARDOLAN
Mitheithel
East Bridge
Sarn Ford
MINHIRIATH
Baranduin (Brandywine)
Tharbad
Nîn-in-Eilph (Swanfleet)
Eregion (Hollin)
Moria Gate
Glanduin
Dimrill Dale
LÓRIEN
Nimrodel
Gladden Fields
Celduin (River Running)
Carnen (Redwater)
Mountains of Mirkwood
RHOVANION
MIRKWOOD
East Bight
DOL GULDUR
Sea of Rhûn
Anduin (The Great River)
Field of Celebrant
Limlight
NORTH UNDEEP
SOUTH UNDEEP
The BROWN LANDS
The Wold
Eryn Vorn
Gwathló (Greyflood)
ENEDWAITH
Lond Daer
Dunland
North-South Road
Isengard
FANGORN
ROHAN
West Emnet
East Emnet
WESTFOLD
Helm's Deep
Fords of Isen
Angren (Isen)
Adorn
Entwash
Sarn Gebir
EMYN MUIL
Falls of Rauros
Dead Marshes
Nindalf (Wetwang)
CAIR ANDROS
Dagorlad (Battle Plain)
Morannon
ERED LITHUI (Ash Mountains)
Udûn
Isenmouthe
Barad-dûr
Orodruin
Plateau of Gorgoroth
MORDOR
NURN
Sea of Nûrnen
EPHEL DÚATH
MINAS MORGUL
Snowbourn
Edoras
EASTFOLD
ERED NIMRAIS
Erech
Anórien
Minas Tirith
Osgiliath
Emyn Arnen
Ithilien
GONDOR
Lamedon
Pinnath Gelin
Anfalas (Langstrand)
Andrast (Ras Morthil)
Edhellond
Dol Amroth
Belfalas
Lebennin
Ethir Anduin
Tolfalas
Poros
Crossings of Poros
South Gondor
(now a debatable and desert land)
Harad Road
Harnen
NEAR HARAD
N
The Realm of MIDDLE EARTH
Miles
50
100
150
200


dynamic algorithms
distributed computing
distance problems
sparsification
derandomization
lower bounds
cyclic games

One Problem – Two Results

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

One Problem – Two Results

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

1. Distributed algorithm: Deterministically compute approximate shortest paths in $n^{1/2+o(1)} + Diam^{1+o(1)}$ rounds [HKN '16]

One Problem – Two Results

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

1. Distributed algorithm: Deterministically compute approximate shortest paths in $n^{1/2+o(1)} + Diam^{1+o(1)}$ rounds [HKN '16]

   Similar in spirit:
   Multipass streaming: $n^{1+o(1)}$ space with $n^{o(1)}$ passes [HKN '16]

One Problem – Two Results

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

1. Distributed algorithm: Deterministically compute approximate shortest paths in $n^{1/2+o(1)} + Diam^{1+o(1)}$ rounds [HKN '16]

   Similar in spirit:
   Multipass streaming: $n^{1+o(1)}$ space with $n^{o(1)}$ passes [HKN '16]

2. Dynamic algorithm: Maintain approximate shortest paths under edge deletions with amortized update time $n^{o(1)}$ [HKN '14]

One Problem – Two Results

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

1. Distributed algorithm: Deterministically compute approximate shortest paths in $n^{1/2+o(1)} + \mathit{Diam}^{1+o(1)}$ rounds [HKN '16]

   Similar in spirit:
   Multipass streaming: $n^{1+o(1)}$ space with $n^{o(1)}$ passes [HKN '16]

2. Dynamic algorithm: Maintain approximate shortest paths under edge deletions with amortized update time $n^{o(1)}$ [HKN '14]

Main technique: Iterative computation of hop set

One Problem – Two Results

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

1. Distributed algorithm: Deterministically compute approximate shortest paths in $n^{1/2+o(1)} + Diam^{1+o(1)}$ rounds [HKN '16]

   Similar in spirit:
   Multipass streaming: $n^{1+o(1)}$ space with $n^{o(1)}$ passes [HKN '16]

2. Dynamic algorithm: Maintain approximate shortest paths under edge deletions with amortized update time $n^{o(1)}$ [HKN '14]

Main technique: Iterative computation of hop set

This talk: constant ϵ, positive integer edge weights polynomial in n

Hop Reduction

Well Known: Spanners

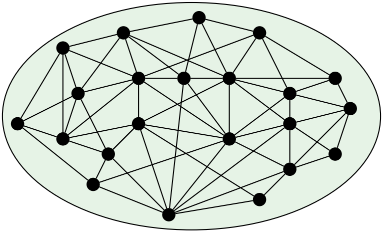
Definition

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

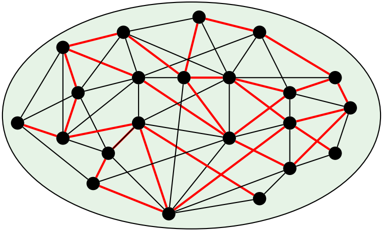
A k-spanner is a subgraph H of G such that, for all pairs of nodes u and v, $dist_H(u,v) \leq k \cdot dist_G(u,v)$.



Well Known: Spanners

Definition

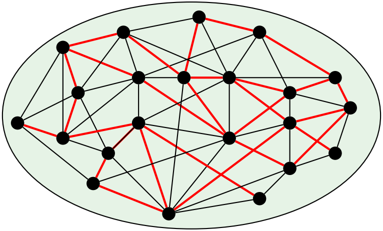
A k-spanner is a subgraph H of G such that, for all pairs of nodes u and v, $dist_H(u, v) \leq k \cdot dist_G(u, v)$.



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

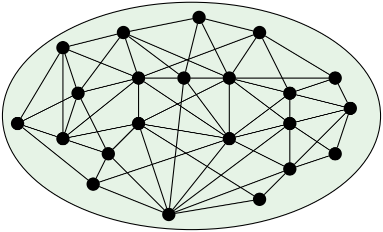
Definition

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 $(1 + \epsilon)dist(u, v)$.

Less Known: Hop Sets

Definition

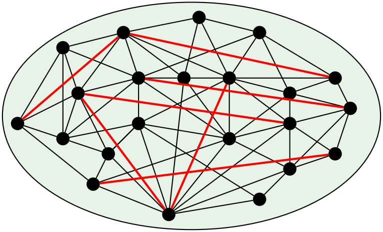
An (h, ϵ)-hop set is a set of weighted edges F such that, for all pairs of nodes u and v, in the 'shortcut graph' $G \cup F$ there is a path from u to v with **at most h edges** of weight at most $(1 + \epsilon) dist(u, v)$.



Less Known: Hop Sets

Definition

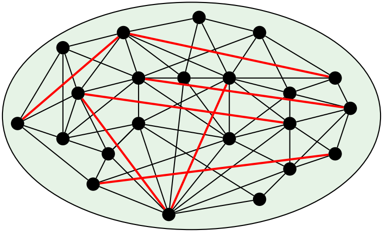
An (h, ϵ)-hop set is a set of weighted edges F such that, for all pairs of nodes u and v, in the 'shortcut graph' $G \cup F$ there is a path from u to v with **at most h edges** of weight at most $(1 + \epsilon)dist(u, v)$.



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 $(1 + \epsilon) dist(u, v)$.



Fact: Every graph has a $(\log^{O(1)} n, \epsilon)$ -hop set of size $m^{1+o(1)}$ [Cohen '94]

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 $(1+\epsilon)dist(u, v)$.

Application?

- Dijkstra: SSSP in time $O(m + n \log n)$
  Not local (global heap), bad for non-centralized models

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 $(1+\epsilon)dist(u,v)$.

Application?

- Dijkstra: SSSP in time $O(m + n \log n)$
  Not local (global heap), bad for non-centralized models
- Bellman-Ford: SSSP in time $O(mn)$
  Actually: SSSP up to h hops in time $O(mh)$
  With $(n^{o(1)}, \epsilon)$ hop set: $(1+\epsilon)$-approximate SSSP in time $O(m^{1+o(1)})$
  Approach used before in parallel setting [Cohen '94]

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:

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

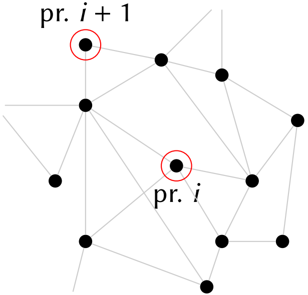
Simple Hop Set Based on Balls (following [Thorup/Zwick '06])

$V = A_0 \supseteq A_1 \supseteq \cdots \supseteq A_k = \emptyset$ 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:

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



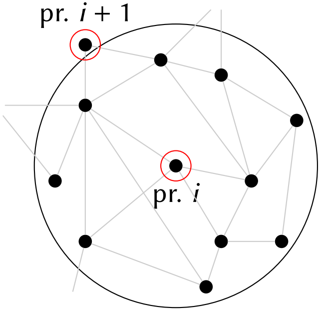
Simple Hop Set Based on Balls (following [Thorup/Zwick '06])

$V = A_0 \supseteq A_1 \supseteq \cdots \supseteq A_k = \emptyset$ 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:

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



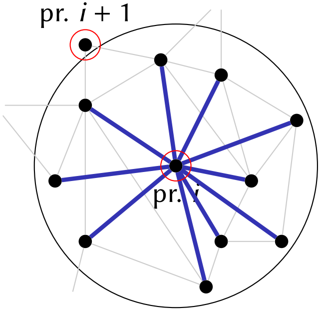
Simple Hop Set Based on Balls (following [Thorup/Zwick '06])

$V = A_0 \supseteq A_1 \supseteq \cdots \supseteq A_k = \emptyset$ 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])

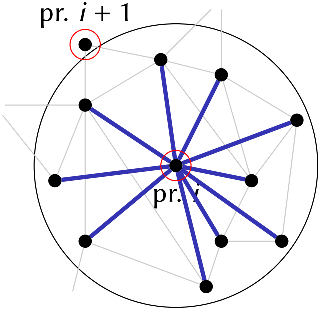
$V = A_0 \supseteq A_1 \supseteq \cdots \supseteq A_k = \emptyset$ where node of A_i goes to A_{i+1} with probability $1/n^{1/k}$

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 $1/n^{1/k}$

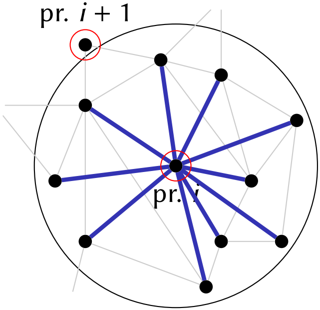
v has **priority** i if $v \in A_i \setminus A_{i+1}$

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 $1/n^{1/k}$

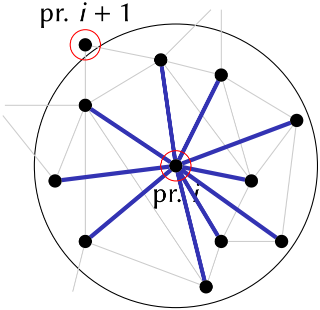
v has **priority** i if $v \in A_i \setminus A_{i+1}$

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])

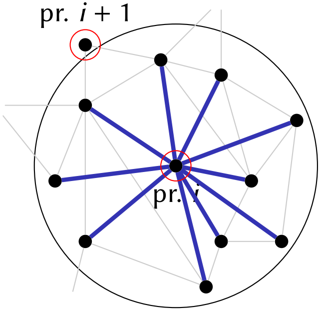
$V = A_0 \supseteq A_1 \supseteq \cdots \supseteq A_k = \emptyset$ where node of A_i goes to A_{i+1} with probability $1/n^{1/k}$

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

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

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

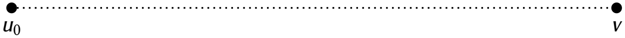
Parameter Choice

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

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

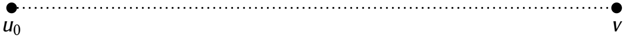
$(n^{1/2+o(1)}, \epsilon)$-hop set

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



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

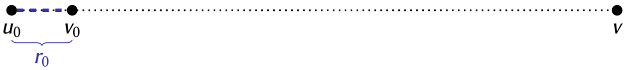
Case 2: $dist(u_0, v) > n^{1/2+1/k}/\epsilon$



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

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

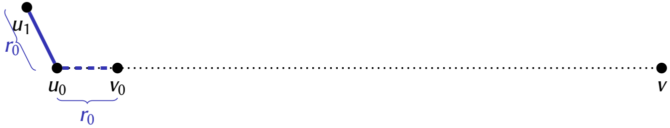
$r_0 = n^{1/2}$



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

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

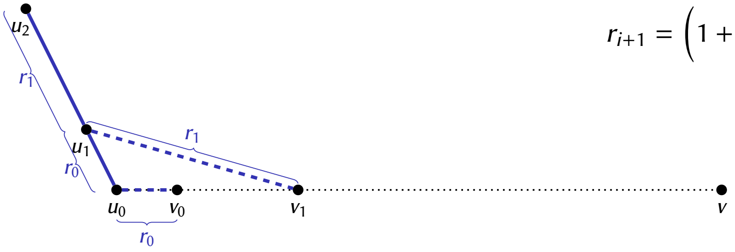
$r_0 = n^{1/2}$



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. $dist(u, u') \leq dist(u, v)$.

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

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



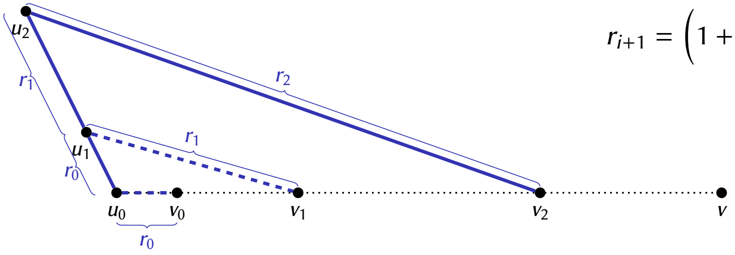
$$r_0 = n^{1/2}$$

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

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. $dist(u, u') \le dist(u, v)$.

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

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



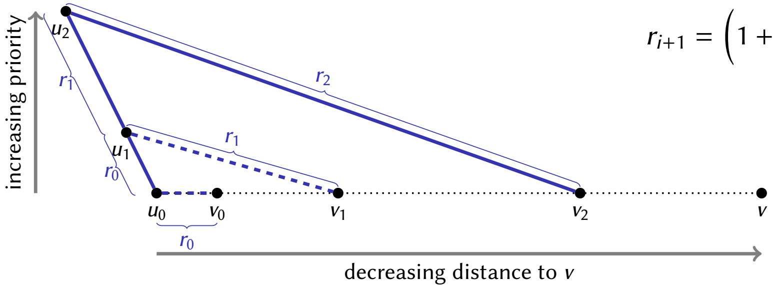
$$r_0 = n^{1/2}$$

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

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. $dist(u, u') \le dist(u, v)$.

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

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



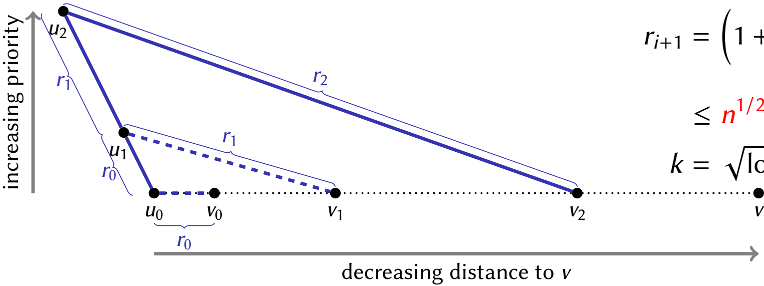
$$r_0 = n^{1/2}$$

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

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. $dist(u, u') \le dist(u, v)$.

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

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



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

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

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

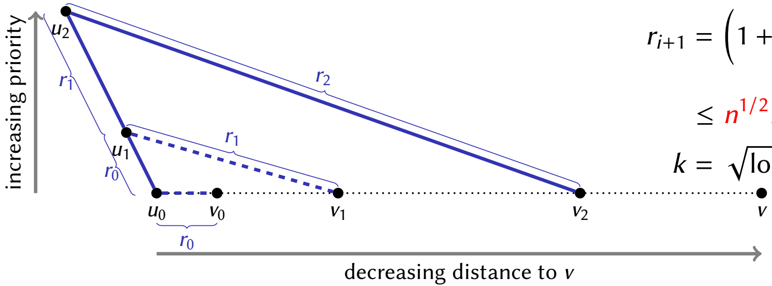
$$k = \sqrt{\log n} / \sqrt{\log 4/\epsilon}$$

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. $dist(u, u') \le dist(u, v)$.

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

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

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



$$r_0 = n^{1/2}$$
$$r_{i+1} = \left(1 + \frac{2}{\epsilon}\right) \sum_{0 \le j \le i} r_j$$
$$\le n^{1/2} n^{1/k}$$
$$k = \sqrt{\log n} / \sqrt{\log 4/\epsilon}$$

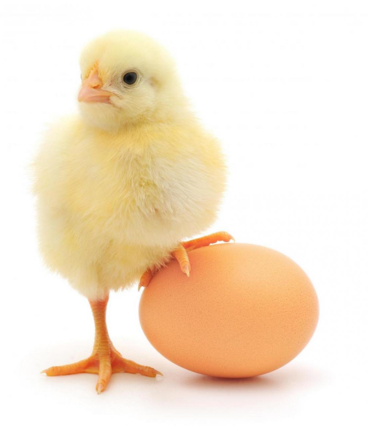
For every node u of priority i and every node v, either $(u, v) \in H$, or $\exists u'$ of priority $i + 1$ s. t. $dist(u, u') \le dist(u, v)$.

$$Weight \le (1 + \epsilon) dist(u_0, v)$$
$$\#Edges \le \frac{k \cdot dist(u, v)}{n^{1/2}} \le \frac{k \cdot n}{n^{1/2}} = kn^{1/2}$$

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

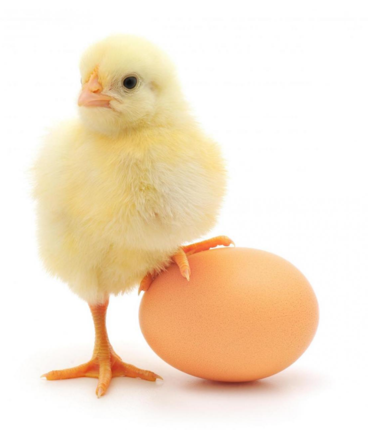
⇒ Back at problem we wanted to solve initially?



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

⇒ Back at problem we wanted to solve initially?



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

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

Iterative computation

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

$(n^{1/2+o(1)}, \epsilon)$-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 $F = \bigcup_{1 \le i \le k} F_i$

$(n^{1/2+o(1)}, \epsilon)$-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: $(1 + \epsilon')^k \le (1 + \epsilon)$ for $\epsilon' = 1/(2\epsilon \log n)$

$(n^{1/2+o(1)}, \epsilon)$-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: $(1 + \epsilon')^k \le (1 + \epsilon)$ for $\epsilon' = 1/(2\epsilon \log n)$

Omitted detail: weighted graphs, use rounding technique

Distributed Algorithm

Distributed Algorithm

SSSP in **CONGEST** model: synchronous rounds, message size $O(\log n)$

Running time = number of rounds

- **Exact:** $O(n)$ (Bellman-Ford)
- $(1+\epsilon)$**-approximation:**
  - $\Omega(n^{1/2}/\log n + Diam)$ [Das Sarma et al. '11]
  - $O(\epsilon^{-1}\log\epsilon^{-1})$: $O(n^{1/2+\epsilon} + Diam)$ (randomized) [Lenzen, Patt-Shamir '13]
  - $1+\epsilon$: $O(n^{1/2} Diam^{1/4} + Diam)$ (randomized) [Nanongkai '14]
  - $1+\epsilon$: $O(n^{1/2+o(1)} + Diam^{1+o(1)})$ (deterministic) **(New)**

Distributed Algorithm

SSSP in **CONGEST** model: synchronous rounds, message size $O(\log n)$

Running time = number of rounds

- **Exact:** $O(n)$ (Bellman-Ford)
- $(1+\epsilon)$-**approximation:**
  - $\Omega(n^{1/2}/\log n + Diam)$ [Das Sarma et al. '11]
  - $O(\epsilon^{-1}\log\epsilon^{-1})$: $O(n^{1/2+\epsilon} + Diam)$ (randomized) [Lenzen, Patt-Shamir '13]
  - $1+\epsilon$: $O(n^{1/2}Diam^{1/4} + Diam)$ (randomized) [Nanongkai '14]
  - $1+\epsilon$: $O(n^{1/2+o(1)} + Diam^{1+o(1)})$ (deterministic) **(New)**

Our approach:

1. Compute overlay network
2. Compute hop set and approximate SSSP on overlay network

Distributed Algorithm

SSSP in **CONGEST** model: synchronous rounds, message size $O(\log n)$

Running time = number of rounds

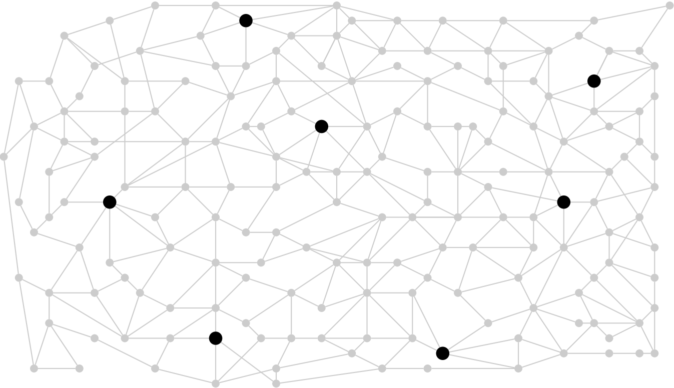
- **Exact:** $O(n)$ (Bellman-Ford)
- $(1+\epsilon)$**-approximation:**
  - $\Omega(n^{1/2}/\log n + Diam)$ [Das Sarma et al. '11]
  - $O(\epsilon^{-1}\log \epsilon^{-1})$: $O(n^{1/2+\epsilon} + Diam)$ (randomized) [Lenzen, Patt-Shamir '13]
  - $1+\epsilon$: $O(n^{1/2} Diam^{1/4} + Diam)$ (randomized) [Nanongkai '14]
  - $1+\epsilon$: $O(n^{1/2+o(1)} + Diam^{1+o(1)})$ (deterministic) **(New)**

Our approach:

1. Compute overlay network
   Derandomization of "hitting paths" argument at cost of approximation
2. Compute hop set and approximate SSSP on overlay network
   Deterministic hop set using greedy hitting set heuristic

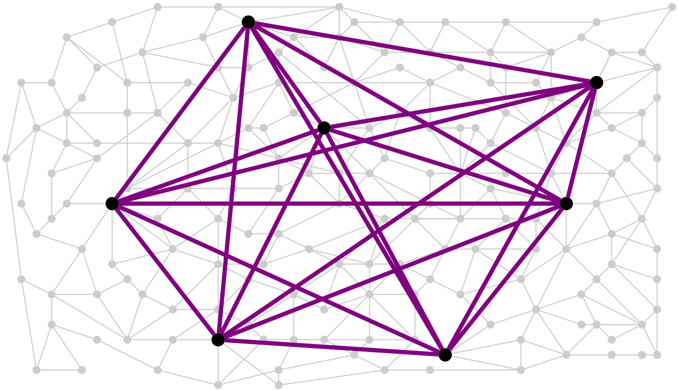
Overlay Network

Overlay Network



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

Overlay Network



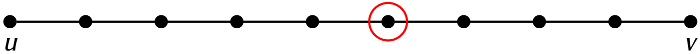
Sample $N = \widetilde{O}(n^{1/2})$ centers (+ source s)

$\Rightarrow$ Every shortest path with $\geq n^{1/2}$ edges contains center whp

Solve SSSP on overlay network using hop set

Derandomization of Overlay Network

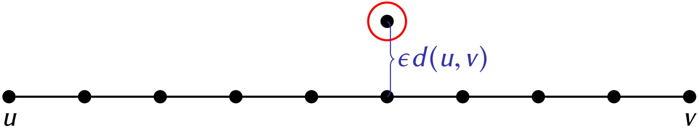
Randomization: Hit every shortest path with $\geq \sqrt{n}$ edges



Derandomization of Overlay Network

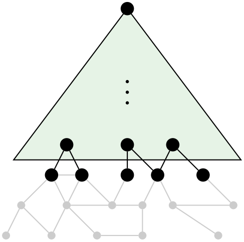
Randomization: Hit every shortest path with $\geq \sqrt{n}$ edges

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



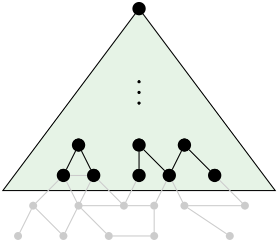
Computing Hop Set on Overlay Network

Shortest paths from source s **up to distance** D**:**



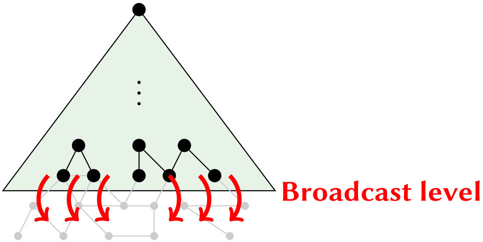
Computing Hop Set on Overlay Network

Shortest paths from source s **up to distance** D**:**



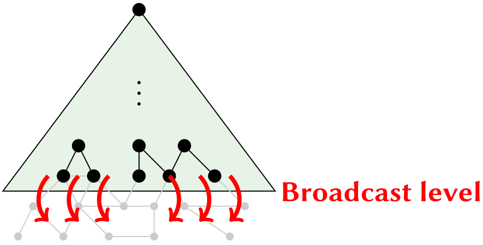
Computing Hop Set on Overlay Network

Shortest paths from source s **up to distance** D**:**



Computing Hop Set on Overlay Network

Shortest paths from source s **up to distance** D:

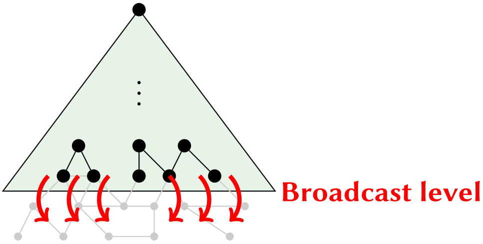


D iterations, each $O(Diam + M_\ell)$ rounds where M_ℓ = #nodes at level ℓ

Running time: $O(D \cdot Diam + \sum_{l \leq D} M_\ell) = O(D \cdot Diam + N)$

Computing Hop Set on Overlay Network

Shortest paths from source s **up to distance** D**:**



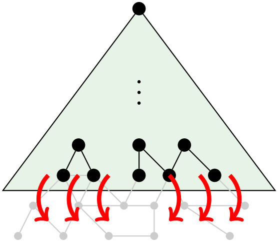
D iterations, each $O(Diam + M_\ell)$ rounds where M_ℓ = #nodes at level ℓ

Running time: $O(D \cdot Diam + \sum_{l \leq D} M_\ell) = O(D \cdot Diam + N)$

Computing balls: $\widetilde{O}(n^{1/k} \cdot Diam + \sum_v |Ball(v)|) = \widetilde{O}(n^{1/k} \cdot Diam + N^{1+1/k})$

Computing Hop Set on Overlay Network

Shortest paths from source s **up to distance** D:



D iterations, each $O(Diam + M_\ell)$ rounds where M_ℓ = #nodes at level ℓ
Running time: $O(D \cdot Diam + \sum_{l \le D} M_\ell) = O(D \cdot Diam + N)$

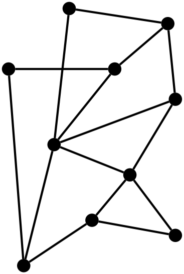
Computing balls: $\widetilde{O}(n^{1/k} \cdot Diam + \sum_v |Ball(v)|) = \widetilde{O}(n^{1/k} \cdot Diam + N^{1+1/k})$

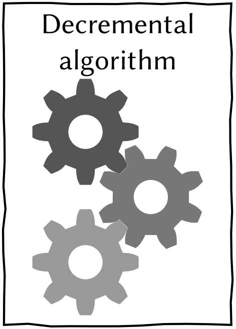
$\Rightarrow$ Hop Set and approximate SSSP: $O(n^{1/2+o(1)} + Diam^{1+o(1)})$

Dynamic Algorithm

Decremental Approximate Shortest Path Problem

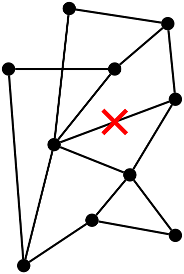
G undergoing deletions:

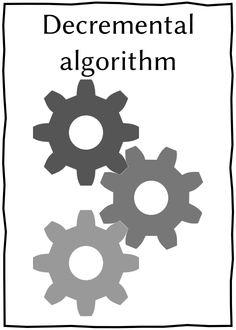




Decremental Approximate Shortest Path Problem

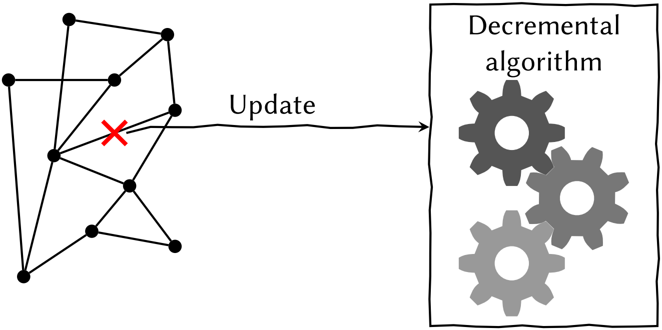
G undergoing deletions:



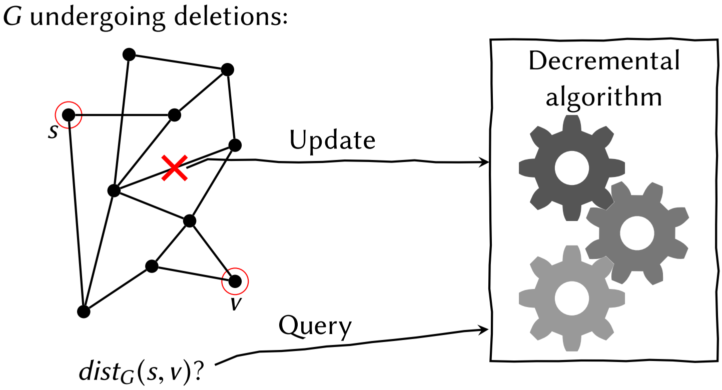


Decremental Approximate Shortest Path Problem

G undergoing deletions:

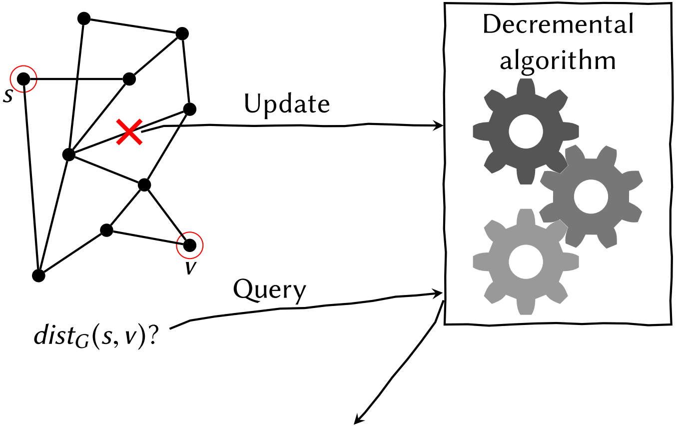


Decremental Approximate Shortest Path Problem



Decremental Approximate Shortest Path Problem

G undergoing deletions:

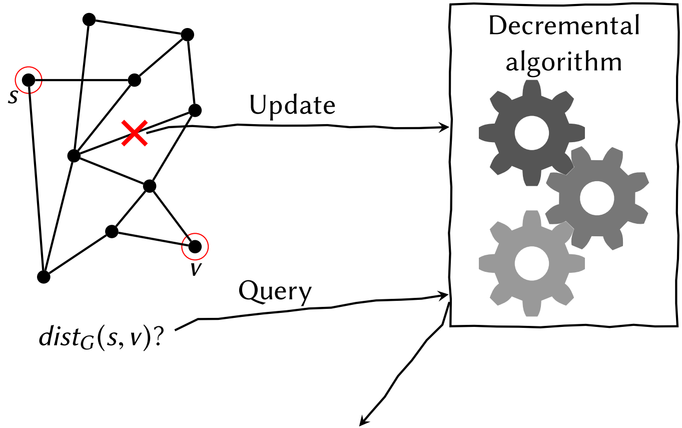


Answer: approximate shortest path of length

$\delta(s, v)$

Decremental Approximate Shortest Path Problem

G undergoing deletions:

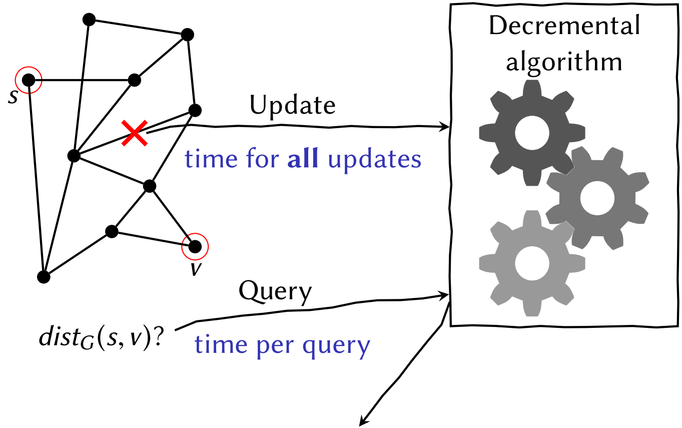


Answer: approximate shortest path of length

$$dist_G(s,v) \leq \delta(s,v) \leq (1+\epsilon)dist_G(s,v)$$

Decremental Approximate Shortest Path Problem

G undergoing deletions:



Answer: approximate shortest path of length

$$dist_G(s, v) \leq \delta(s, v) \leq (1 + \epsilon) dist_G(s, v)$$

Overview of Result

New result:

- Exact: total update time $O(mn)$ (unweighted) [Even/Shiloach '81]
  $\Omega(mn)$ [Roditty/Zwick '04, Henzinger/K/Nanongkai/Saranurak '15]
- $(1+\epsilon)$-approx.: $O(n^{2+o(1)})$ (unweighted) [Bernstein/Roditty '11]
- **New:** $O(m^{1+o(1)})$ (weighted) [Henzinger/K/Nanongkai '14]

Overview of Result

New result:

- Exact: total update time $O(mn)$ (unweighted) [Even/Shiloach '81]
  $\Omega(mn)$ [Roditty/Zwick '04, Henzinger/K/Nanongkai/Saranurak '15]
- $(1+\epsilon)$-approx.: $O(n^{2+o(1)})$ (unweighted) [Bernstein/Roditty '11]
- **New:** $O(m^{1+o(1)})$ (weighted) [Henzinger/K/Nanongkai '14]

Techniques for maintaining balls:

Overview of Result

New result:

- Exact: total update time $O(mn)$ (unweighted) [Even/Shiloach '81]
  $\Omega(mn)$ [Roditty/Zwick '04, Henzinger/K/Nanongkai/Saranurak '15]
- $(1+\epsilon)$-approx.: $O(n^{2+o(1)})$ (unweighted) [Bernstein/Roditty '11]
- **New:** $O(m^{1+o(1)})$ (weighted) [Henzinger/K/Nanongkai '14]

Techniques for maintaining balls:

- Even-Shiloach: $O(mD)$ for SSSP up to depth D

Overview of Result

New result:

- Exact: total update time $O(mn)$ (unweighted) [Even/Shiloach '81]
  $\Omega(mn)$ [Roditty/Zwick '04, Henzinger/K/Nanongkai/Saranurak '15]
- $(1+\epsilon)$-approx.: $O(n^{2+o(1)})$ (unweighted) [Bernstein/Roditty '11]
- **New:** $O(m^{1+o(1)})$ (weighted) [Henzinger/K/Nanongkai '14]

Techniques for maintaining balls:

- Even-Shiloach: $O(mD)$ for SSSP up to depth D
- Restart when distance to next priority changes

Overview of Result

New result:

- Exact: total update time $O(mn)$ (unweighted) [Even/Shiloach '81]
  $\Omega(mn)$ [Roditty/Zwick '04, Henzinger/K/Nanongkai/Saranurak '15]
- $(1+\epsilon)$-approx.: $O(n^{2+o(1)})$ (unweighted) [Bernstein/Roditty '11]
- **New:** $O(m^{1+o(1)})$ (weighted) [Henzinger/K/Nanongkai '14]

Techniques for maintaining balls:

- Even-Shiloach: $O(mD)$ for SSSP up to depth D
- Restart when distance to next priority changes
- Bounding number of nodes in balls not enough
  All edges incident to balls go into running time
  $\Rightarrow$ Sample edges instead of nodes

Overview of Result

New result:

- Exact: total update time $O(mn)$ (unweighted) [Even/Shiloach '81]
  $\Omega(mn)$ [Roditty/Zwick '04, Henzinger/K/Nanongkai/Saranurak '15]
- $(1+\epsilon)$-approx.: $O(n^{2+o(1)})$ (unweighted) [Bernstein/Roditty '11]
- **New:** $O(m^{1+o(1)})$ (weighted) [Henzinger/K/Nanongkai '14]

Techniques for maintaining balls:

- Even-Shiloach: $O(mD)$ for SSSP up to depth D
- Restart when distance to next priority changes
- Bounding number of nodes in balls not enough
  All edges incident to balls go into running time
  $\Rightarrow$ Sample edges instead of nodes
- Deletions-only problem, but edges might be added to hop set
  Monotone ES-tree framework [Henzinger/K/Nanongkai '13]

New Approach

New Distributed Algorithm

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

There is a deterministic algorithm for computing $(1+\epsilon)$ *approximate SSSP in* $\widetilde{O}(\sqrt{n} + Diam)$ *rounds.*

New Distributed Algorithm

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

There is a deterministic algorithm for computing $(1+\epsilon)$ *approximate SSSP in* $\widetilde{O}(\sqrt{n} + Diam)$ *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 $(1+\epsilon)$ *approximate SSSP in* $\widetilde{O}(\sqrt{n} + Diam)$ *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 $(1+\epsilon)$ *approximate SSSP in* $\widetilde{O}(\sqrt{n} + Diam)$ *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:

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

Gradient Descent

Algorithm at a glance:

1. Soft-max is differentiable → apply gradient descent

Gradient Descent

Algorithm at a glance:

1. Soft-max is differentiable → 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* $\widetilde{O}(n)$

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 $\widetilde{O}(n)$
5. Overall: Polylog iterations, each solving $O(\log n)$-approximate transshipment on graph of $\widetilde{O}(n)$ edges

Conclusion

Main contributions:

- Two almost tight algorithms
- Combinatorial and algebraic tools

Conclusion

Main contributions:

- Two almost tight algorithms
- Combinatorial and algebraic tools

Open problems:

- Parallel: improve Cohen's $m^{1+o(1)}$ work with polylog depth?
- Better hop set? $n^{o(1)} \to \log^{O(1)} n$
- Deterministic dynamic SSSP algorithm
  Vision: Dynamic algorithms as data structures inside other algorithms
- Is $O(n)$ rounds for exact distributed SSSP optimal?