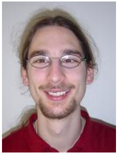
# Near-Optimal Approximate Shortest Paths and Transshipment in Distributed and Streaming Models

**Sebastian Krinninger**

University of Vienna
→ University of Salzburg

joint work with



Ruben Becker
MPI Saarbrücken

Andreas Karrenbauer
MPI Saarbrücken

Christoph Lenzen
MPI Saarbrücken

# Approximate Single-Source Shortest Paths

| | **Our** $(1+\varepsilon)$**-approx** |
|---|---|
| **CONGEST** | $(\sqrt{n} + D) \cdot poly(\log n, \varepsilon)$ rounds |

1

2

3

# Approximate Single-Source Shortest Paths

| | **Our $(1+\varepsilon)$-approx** | **Previous best** |
|---|---|---|
| **CONGEST** | $(\sqrt{n}+D)\cdot poly(\log n, \varepsilon)$ rounds | $(\sqrt{n}+D)\cdot 2^{O(\sqrt{\log n \log(\varepsilon^{-1}\log n)})}$ rounds[1] |

**Comments**:

- Undirected graphs with weights $\in \{1, 2, \ldots, poly(n)\}$
- $D$ = Diameter, $n$ = #nodes
- CONGEST lower bound: $\tilde{\Omega}(\sqrt{n} + Diam)$ rounds [Das Sarma et al '11]

[1] [Henzinger/K/Nanongkai '16]

[2]

[3]

# Approximate Single-Source Shortest Paths

| | **Our $(1+\varepsilon)$-approx** | **Previous best** |
|---|---|---|
| **CONGEST** | $(\sqrt{n} + D) \cdot poly(\log n, \varepsilon)$ rounds | $(\sqrt{n} + D) \cdot 2^{O(\sqrt{\log n \log(\varepsilon^{-1} \log n)})}$ rounds[1] |
| **Cong. Clique** | $poly(\log n, \varepsilon)$ rounds | $2^{O(\sqrt{\log n \log(\varepsilon^{-1} \log n)})}$ rounds[2] |

**Comments**:

- Undirected graphs with weights $\in \{1, 2, \ldots, poly(n)\}$
- $D$ = Diameter, $n$ = #nodes
- CONGEST lower bound: $\tilde{\Omega}(\sqrt{n} + Diam)$ rounds [Das Sarma et al '11]

[1] [Henzinger/K/Nanongkai '16]
[2] [Henzinger/K/Nanongkai '16]
[3]

# Approximate Single-Source Shortest Paths

| | Our $(1+\varepsilon)$-approx | Previous best |
|---|---|---|
| **CONGEST** | $(\sqrt{n}+D)\cdot poly(\log n, \varepsilon)$ rounds | $(\sqrt{n}+D)\cdot 2^{O(\sqrt{\log n \log(\varepsilon^{-1}\log n)})}$ rounds[1] |
| **Cong. Clique** | $poly(\log n, \varepsilon)$ rounds | $2^{O(\sqrt{\log n \log(\varepsilon^{-1}\log n)})}$ rounds[2] |
| **Streaming** | $poly(\log n, \varepsilon)$ passes $O(n \log n)$ space | $(2+1/\varepsilon)^{O(\sqrt{\log n \log\log n})}$ passes $O(n \log^2 n)$ space[3] |

**Comments**:

- Undirected graphs with weights $\in \{1, 2, \ldots, poly(n)\}$
- $D$ = Diameter, $n$ = #nodes
- CONGEST lower bound: $\tilde{\Omega}(\sqrt{n} + Diam)$ rounds [Das Sarma et al '11]

---

[1] [Henzinger/K/Nanongkai '16]
[2] [Henzinger/K/Nanongkai '16]
[3] [Elkin/Neiman '16]

# Approximate Single-Source Shortest Paths

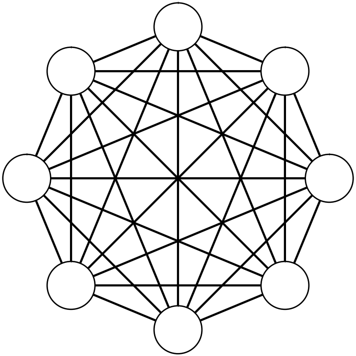
| | Our $(1+\varepsilon)$-approx | Exact computation |
|---|---|---|
| **CONGEST** | $(\sqrt{n}+D)\cdot poly(\log n, \varepsilon)$ rounds | $n^{5/6} + D^{1/3}(n \log n)^{2/3}$ rounds[1] |
| **Cong. Clique** | $poly(\log n, \varepsilon)$ rounds | $O(n^{0.158})$ rounds[2] |
| **Streaming** | $poly(\log n, \varepsilon)$ passes $O(n \log n)$ space | $O(\frac{n}{k})$ passes $O(nk)$ space[3] |

**Comments**:

- Undirected graphs with weights $\in \{1, 2, \ldots, poly(n)\}$
- $D$ = Diameter, $n$ = #nodes
- CONGEST lower bound: $\tilde{\Omega}(\sqrt{n} + Diam)$ rounds [Das Sarma et al '11]

[1] [Elkin '17]
[2] [Censor-Hillel et al. '15]
[3] [Elkin '17]

# Broadcast Congested Clique



**Model:**

- Network topology: clique on $n$ nodes
- Synchronous rounds (global clock)
- In each round, every node sends one message to all other nodes
- Message size $O(\log n)$
- Local computation is free

# Problem Statement

- Initially: Every node knows weight of its incident edges and whether it is the source or not
- Finally: Every node knows its approximate distance to the source

# Problem Statement

- Initially: Every node knows weight of its incident edges and whether it is the source or not
- Finally: Every node knows its approximate distance to the source
- Desirable addon: Implicit tree; every node knows next edge on approximate shortest path to source

# Problem Statement

- Initially: Every node knows weight of its incident edges and whether it is the source or not
- Finally: Every node knows its approximate distance to the source
- Desirable addon: Implicit tree; every node knows next edge on approximate shortest path to source

**Simulation: Skeleton as congested clique** [Henzinger/K/Nanongkai '16]

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

# Combinatorial Approach

# Sparsification I: Spanners

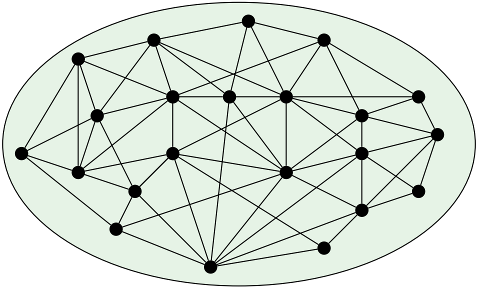
**Definition**

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

# Sparsification I: Spanners

## Definition

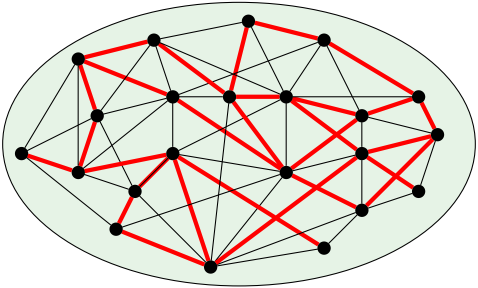
A $k$-spanner is a subgraph $H$ of $G$ such that, for all pairs of nodes $u$ and $v$, $dist_H(u, v) \leq k \cdot dist_G(u, v)$.



# Sparsification I: Spanners

## Definition

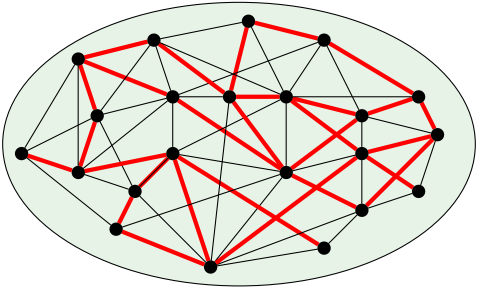
A $k$-spanner is a subgraph $H$ of $G$ such that, for all pairs of nodes $u$ and $v$, $dist_H(u, v) \leq k \cdot dist_G(u, v)$.



# Sparsification I: 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 $(2k-1)$-spanner of size $n^{1+1/k}$

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

# Sparsification II: Hop Sets

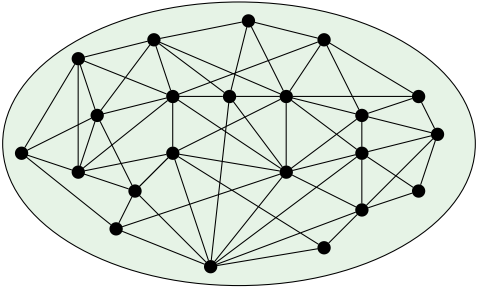
## Definition

An $(h, \varepsilon)$-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 $(1 + \varepsilon)dist(u, v)$.

# Sparsification II: Hop Sets

## Definition

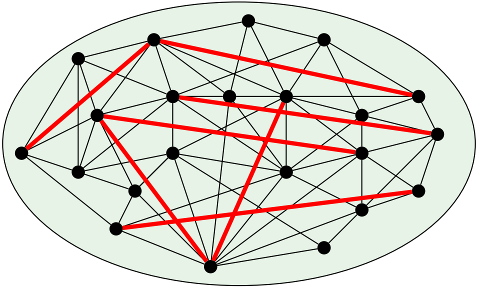
An $(h, \varepsilon)$-hop set is a set of weighted edges $F$ such that, for all pairs of nodes $u$ and $v$, in the 'shortcut graph' $G \cup F$ there is a path from $u$ to $v$ with **at most** $h$ **edges** of weight at most $(1 + \varepsilon)dist(u, v)$.



# Sparsification II: Hop Sets

**Definition**

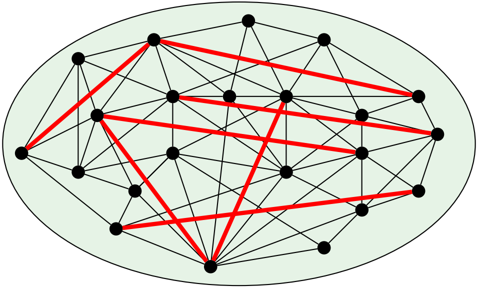
An $(h, \varepsilon)$-hop set is a set of weighted edges $F$ such that, for all pairs of nodes $u$ and $v$, in the 'shortcut graph' $G \cup F$ there is a path from $u$ to $v$ with **at most $h$ edges** of weight at most $(1 + \varepsilon) dist(u, v)$.



# Sparsification II: Hop Sets

**Definition**

An $(h, \varepsilon)$-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 $\varepsilon \geq 1/polylogn$)

# Sparsification II: Hop Sets

**Definition**

An $(h, \varepsilon)$-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 to approximate SSSP**

Almost tight algorithms for Bellman-Ford-like approaches:

- Parallel: $m^{1+o(1)}$ work with $n^{o(1)}$ depth [Cohen '94]
- Congested Clique: $n^{o(1)}$ rounds [Henzinger/K/Nanongkai '16]
- Streaming: $n^{o(1)}$ passes with $n^{1+o(1)}$ space [HKN '16, Elkin/Neiman '16]
- Incremental/Decremental $m^{1+o(1)}$ total time [Henzinger/K/Nanongkai '14]

# Sparsification II: Hop Sets

**Definition**

An $(h, \varepsilon)$-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 to approximate SSSP**

Almost tight algorithms for Bellman-Ford-like approaches:

- Parallel: $m^{1+o(1)}$ work with $n^{o(1)}$ depth [Cohen '94]
- Congested Clique: $n^{o(1)}$ rounds [Henzinger/K/Nanongkai '16]
- Streaming: $n^{o(1)}$ passes with $n^{1+o(1)}$ space [HKN '16, Elkin/Neiman '16]
- Incremental/Decremental $m^{1+o(1)}$ total time [Henzinger/K/Nanongkai '14]

**Challenge:** Compute/maintain hop set

# Sparsification II: Hop Sets

### Definition

An $(h, \varepsilon)$-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 to approximate SSSP**

Almost tight algorithms for Bellman-Ford-like approaches:

- Parallel: $m^{1+o(1)}$ work with $n^{o(1)}$ depth [Cohen '94]
- Congested Clique: $n^{o(1)}$ rounds [Henzinger/K/Nanongkai '16]
- Streaming: $n^{o(1)}$ passes with $n^{1+o(1)}$ space [HKN '16, Elkin/Neiman '16]
- Incremental/Decremental $m^{1+o(1)}$ total time [Henzinger/K/Nanongkai '14]

**Challenge:** Compute/maintain hop set

# Hop Sets: Approaching Optimality

| **Authors** | **Stretch $\alpha$** | **Hopbound $h$** | **Size** |
|---|---|---|---|
| [Baseline] | 1 | 1 | $O(n^2)$ |

# Hop Sets: Approaching Optimality

| Authors | Stretch $\alpha$ | 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 $\alpha$ | 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 | $O(\frac{n}{t})$ | $O(nt)$ |
| [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 $\alpha$ | 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 | $O(\frac{n}{t})$ | $O(nt)$ |
| [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] | $1+\varepsilon$ | $O(\frac{k}{\varepsilon})^k$ | $O(n^{1+\frac{1}{2^{k+1}-1}})$ |

Hopset analysis of spanner/emulator in [Thorup/Zwick '06]

# Hop Sets: Approaching Optimality

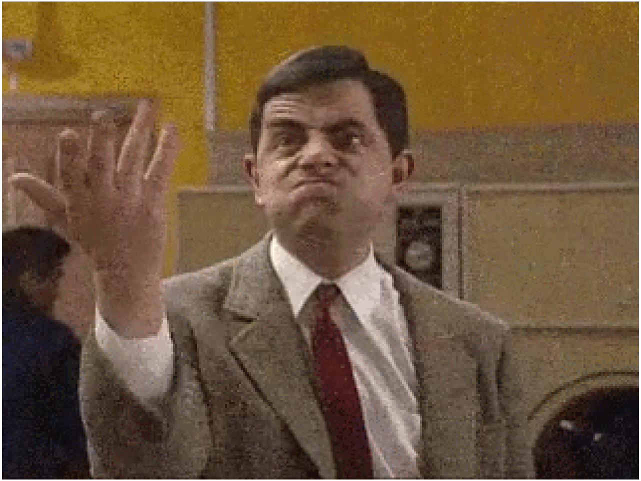
| **Authors** | **Stretch $\alpha$** | **Hopbound $h$** | **Size** |
|---|---|---|---|
| [Baseline] | 1 | 1 | $O(n^2)$ |
| [Klein/Subramanian '97] | 1 | $O(\frac{n \log n}{t})$ | $O(t^2)$ |
| [Shi/Spencer '99] | 1 | $O(\frac{n}{t})$ | $O(nt)$ |
| [Cohen'94] | $1+\varepsilon$ | $(\frac{\log n}{\varepsilon})^{O(\log k)}$ | $O(n^{1+\frac{1}{k}} \log n)$ |
| [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 $\alpha$ | 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 | $O(\frac{n}{t})$ | $O(nt)$ |
| [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 $n \cdot polylog(n)$.

No further (significant) algorithmic improvements by better hop sets :(



It was too good to be true…

# Beyond Hop Sets

# Our Approach

# Our Approach

**Gradient Descent**

# Problem Formulation

## 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.

# Problem Formulation

## 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"

# Problem Formulation

## 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"

## Flow View

Given demand $b(v)$ for each node $v$, find a flow $x(e)$ that:

- meets the demands: $\sum_{e=(u,v)\in E} x(e) = b(v) + \sum_{e=(v,u)\in E} x(e)$ for every node $v$
- and minimizes $\sum_{e\in E} w(e) \cdot x(e)$.

Undirected graphs: arbitrary orientation of edges.

# Problem Formulation

## 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"

## Flow View

Given demand $b(v)$ for each node $v$, find a flow $x(e)$ that:

- meets the demands: $\sum_{e=(u,v)\in E} x(e) = b(v) + \sum_{e=(v,u)\in E} x(e)$ for every node $v$
- and minimizes $\sum_{e\in E} w(e) \cdot x(e)$.

Undirected graphs: arbitrary orientation of edges.

**LP Formulation:** minimize $\|Wx\|_1$ s.t. $Ax = b$

# Problem Formulation

## 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"

## Flow View

Given demand $b(v)$ for each node $v$, find a flow $x(e)$ that:

- meets the demands: $\sum_{e=(u,v)\in E} x(e) = b(v) + \sum_{e=(v,u)\in E} x(e)$ for every node $v$
- and minimizes $\sum_{e\in E} w(e) \cdot x(e)$.

Undirected graphs: arbitrary orientation of edges.

**LP Formulation:** minimize $\|Wx\|_1$ s.t. $Ax = b$

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

# Reformulation

## LP Formulation

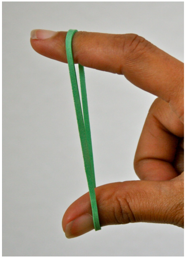
**Primal:** minimize $\|Wx\|_1$ s.t. $Ax = b$

**Dual:** maximize $b^T y$ s.t. $\left\|W^{-1}A^T y\right\|_\infty \leq 1$

# Reformulation

## LP Formulation

**Primal:** minimize $\|Wx\|_1$ s.t. $Ax = b$
**Dual:** maximize $b^T y$ s.t. $\|W^{-1}A^T y\|_\infty \leq 1$

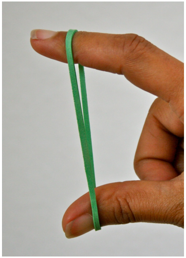


Maximize node potentials restricting stretch:
$|y(u) - y(v)|/w(e) \leq 1$ for every edge $e = (u, v)$
SSSP: potentials = distances to source

# Reformulation

## LP Formulation

**Primal:** minimize $\|Wx\|_1$ s.t. $Ax = b$
**Dual:** maximize $b^T y$ s.t. $\left\|W^{-1}A^T y\right\|_\infty \leq 1$



Maximize node potentials restricting stretch:
$|y(u) - y(v)|/w(e) \leq 1$ for every edge $e = (u, v)$
SSSP: potentials = distances to source

**Equivalent:**

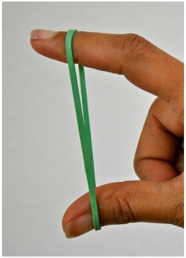
$$\text{minimize } \left\|W^{-1}A^T \pi\right\|_\infty \quad \text{s.t. } b^T \pi = 1$$

# Reformulation

## LP Formulation

**Primal:** minimize $\|Wx\|_1$ s.t. $Ax = b$

**Dual:** maximize $b^T y$ s.t. $\left\|W^{-1}A^T y\right\|_\infty \leq 1$



Maximize node potentials restricting stretch:
$|y(u) - y(v)|/w(e) \leq 1$ for every edge $e = (u, v)$
SSSP: potentials = distances to source

**Equivalent:**

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

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

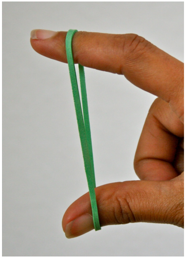
$$\mathsf{lse}_\beta(x) := \frac{1}{\beta} \ln\left(\sum_{1\leq i\leq n}\left(e^{\beta x_i} + e^{-\beta x_i}\right)\right)$$

# Reformulation

## LP Formulation

**Primal:** minimize $\|Wx\|_1$ s.t. $Ax = b$

**Dual:** maximize $b^T y$ s.t. $\left\|W^{-1}A^T y\right\|_\infty \leq 1$



Maximize node potentials restricting stretch:
$|y(u) - y(v)|/w(e) \leq 1$ for every edge $e = (u, v)$
SSSP: potentials = distances to source

**Equivalent:**

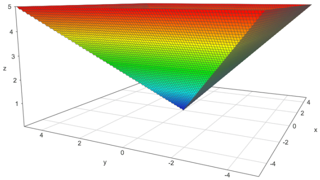
$$\text{minimize } \left\|W^{-1}A^T \pi\right\|_\infty \quad \text{s.t. } b^T \pi = 1$$

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

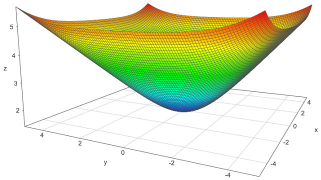
$$\mathsf{lse}_\beta(x) := \frac{1}{\beta} \ln \left( \sum_{1 \leq i \leq n} \left( e^{\beta x_i} + e^{-\beta x_i} \right) \right)$$

**Goal:** minimize $\Phi_\beta(\pi) := \mathsf{lse}_\beta(W^{-1}A^T \pi)$ s.t. $b^T \pi = 1$

# Soft-max approximation



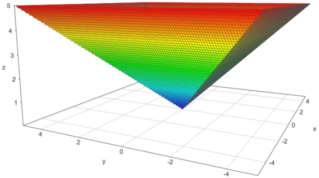

$\Rightarrow$



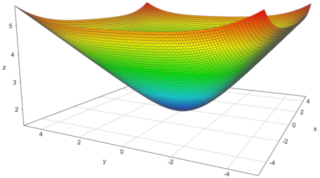

$\|x\|_\infty$ (where $v \in \mathbb{R}^n$)

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

# Soft-max approximation




$\Rightarrow$



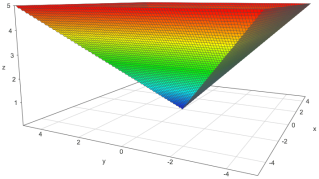

$\|x\|_\infty$ (where $v \in \mathbb{R}^n$)

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

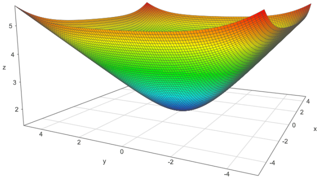
**Additive approximation:**

$$\|x\|_\infty \le \mathsf{lse}_\beta(x) \le \|x\|_\infty + \frac{\ln n}{\beta}$$

# Soft-max approximation




$$\Longrightarrow$$




$\|x\|_\infty$ (where $v \in \mathbb{R}^n$)

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

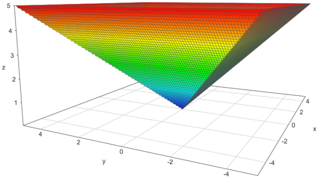
**Additive approximation:**

$$\|x\|_\infty \le \mathrm{lse}_\beta(x) \le \|x\|_\infty + \frac{\ln n}{\beta}$$

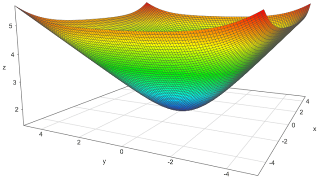
**Lipschitz smoothness:**

$$\left\| \nabla \, \mathrm{lse}_\beta(x) - \nabla \, \mathrm{lse}_\beta(y) \right\|_1 \le \beta \, \|x - y\|_\infty$$

# Soft-max approximation



$\Rightarrow$



$\|x\|_\infty \quad (\text{where } v \in \mathbb{R}^n)$

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

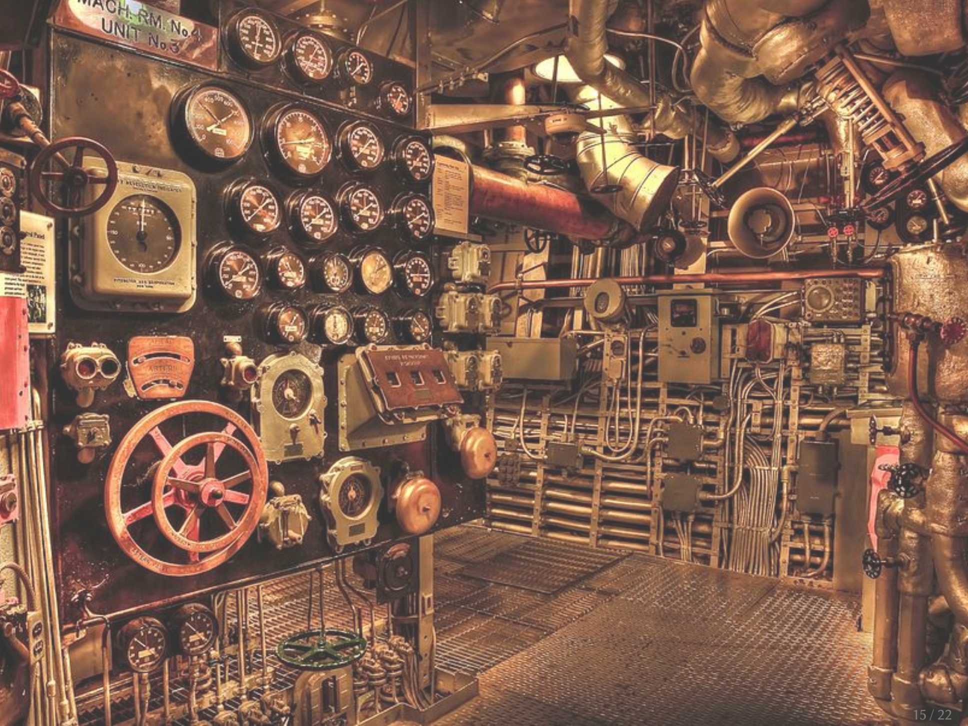
**Additive approximation:**

$$\|x\|_\infty \le \mathsf{lse}_\beta(x) \le \|x\|_\infty + \frac{\ln n}{\beta}$$

**Lipschitz smoothness:**

$$\left\| \nabla \mathsf{lse}_\beta(x) - \nabla \mathsf{lse}_\beta(y) \right\|_1 \le \beta \, \|x - y\|_\infty$$

**Intuition:** Trade off quality of approximation and smoothness


MACH. RM. No.4
UNIT No.3

# Generic Update Step

Bounding improvement in objective for generic update $\pi' = \pi - h$:

# Generic Update Step

Bounding improvement in objective for generic update $\pi' = \pi - h$:

$$\Phi_\beta(\pi) - \Phi_\beta(\pi - h)$$
$$\geq \nabla\Phi_\beta(\pi - h)^T h$$

**Convexity:** $f(y) \geq f(x) + \nabla f(x)^T (y - x)$

# Generic Update Step

Bounding improvement in objective for generic update $\pi' = \pi - h$:

$$
\begin{aligned}
&\Phi_\beta(\pi) - \Phi_\beta(\pi - h) \\
&\quad \geq \nabla\Phi_\beta(\pi - h)^T h \\
&\quad = \nabla\Phi_\beta(\pi)^T h - \left(\nabla\Phi_\beta(\pi)^T h - \nabla\Phi_\beta(\pi - h)^T h\right)
\end{aligned}
$$

# Generic Update Step

Bounding improvement in objective for generic update $\pi' = \pi - h$:

$$\begin{aligned}
&\Phi_\beta(\pi) - \Phi_\beta(\pi - h) \\
&\quad \geq \nabla\Phi_\beta(\pi - h)^T h \\
&\quad = \nabla\Phi_\beta(\pi)^T h - \left(\nabla\Phi_\beta(\pi)^T h - \nabla\Phi_\beta(\pi - h)^T h\right) \\
&\quad = \nabla\Phi_\beta(\pi)^T h - \left(\nabla \operatorname{lse}_\beta(W^{-1}A^T\pi) - \nabla \operatorname{lse}_\beta(W^{-1}A^T(\pi - h))\right)^T W^{-1}A^T h
\end{aligned}$$

**Chain rule:** $\nabla\Phi_\beta(\pi) = AW^{-1}\nabla \operatorname{lse}_\beta(W^{-1}A^T\pi)$

# Generic Update Step

Bounding improvement in objective for generic update $\pi' = \pi - h$:

$$\begin{aligned}
&\Phi_\beta(\pi) - \Phi_\beta(\pi - h) \\
&\quad \geq \nabla\Phi_\beta(\pi - h)^T h \\
&\quad = \nabla\Phi_\beta(\pi)^T h - \left(\nabla\Phi_\beta(\pi)^T h - \nabla\Phi_\beta(\pi - h)^T h\right) \\
&\quad = \nabla\Phi_\beta(\pi)^T h - \left(\nabla \operatorname{lse}_\beta(W^{-1}A^T\pi) - \nabla \operatorname{lse}_\beta(W^{-1}A^T(\pi - h))\right)^T W^{-1}A^T h \\
&\quad \geq \nabla\Phi_\beta(\pi)^T h - \left\|\nabla \operatorname{lse}_\beta(W^{-1}A^T\pi) - \nabla \operatorname{lse}_\beta(W^{-1}A^T(\pi - h))\right\|_1 \left\|W^{-1}A^T h\right\|_\infty
\end{aligned}$$

**Hölder:** $x^T y \leq \|x\|_p \|y\|_q$ for $\frac{1}{p} + \frac{1}{q} = 1$

# Generic Update Step

Bounding improvement in objective for generic update $\pi' = \pi - h$:

**Lipschitz:** $\left\|\nabla \operatorname{lse}_\beta(x) - \nabla \operatorname{lse}_\beta(y)\right\|_1 \leq \beta \|x - y\|_\infty$

$$\begin{aligned}
&\Phi_\beta(\pi) - \Phi_\beta(\pi - h) \\
&\quad \geq \nabla\Phi_\beta(\pi - h)^T h \\
&\quad = \nabla\Phi_\beta(\pi)^T h - \left(\nabla\Phi_\beta(\pi)^T h - \nabla\Phi_\beta(\pi - h)^T h\right) \\
&\quad = \nabla\Phi_\beta(\pi)^T h - \left(\nabla \operatorname{lse}_\beta(W^{-1}A^T\pi) - \nabla \operatorname{lse}_\beta(W^{-1}A^T(\pi - h))\right)^T W^{-1}A^T h \\
&\quad \geq \nabla\Phi_\beta(\pi)^T h - \left\|\nabla \operatorname{lse}_\beta(W^{-1}A^T\pi) - \nabla \operatorname{lse}_\beta(W^{-1}A^T(\pi - h))\right\|_1 \left\|W^{-1}A^T h\right\|_\infty \\
&\quad \geq \nabla\Phi_\beta(\pi)^T h - \beta \left\|W^{-1}A^T h\right\|_\infty^2
\end{aligned}$$

# Generic Update Step

Bounding improvement in objective for generic update $\pi' = \pi - h$:

$$
\begin{aligned}
&\Phi_\beta(\pi) - \Phi_\beta(\pi - h) \\
&\geq \nabla\Phi_\beta(\pi - h)^T h \\
&= \nabla\Phi_\beta(\pi)^T h - \left(\nabla\Phi_\beta(\pi)^T h - \nabla\Phi_\beta(\pi - h)^T h\right) \\
&= \nabla\Phi_\beta(\pi)^T h - \left(\nabla \operatorname{lse}_\beta(W^{-1}A^T\pi) - \nabla \operatorname{lse}_\beta(W^{-1}A^T(\pi - h))\right)^T W^{-1}A^T h \\
&\geq \nabla\Phi_\beta(\pi)^T h - \left\|\nabla \operatorname{lse}_\beta(W^{-1}A^T\pi) - \nabla \operatorname{lse}_\beta(W^{-1}A^T(\pi - h))\right\|_1 \left\|W^{-1}A^T h\right\|_\infty \\
&\geq \nabla\Phi_\beta(\pi)^T h - \beta\left\|W^{-1}A^T h\right\|_\infty^2
\end{aligned}
$$

- Suggests to compute $h$ by solving

$$
\max\{\nabla\Phi_\beta(\pi)^T h : \left\|W^{-1}A^T h\right\|_\infty \leq 1\}
$$

# Generic Update Step

Bounding improvement in objective for generic update $\pi' = \pi - h$:

$$
\begin{aligned}
&\Phi_\beta(\pi) - \Phi_\beta(\pi - h) \\
&\geq \nabla\Phi_\beta(\pi - h)^T h \\
&= \nabla\Phi_\beta(\pi)^T h - \left(\nabla\Phi_\beta(\pi)^T h - \nabla\Phi_\beta(\pi - h)^T h\right) \\
&= \nabla\Phi_\beta(\pi)^T h - \left(\nabla \operatorname{lse}_\beta(W^{-1}A^T\pi) - \nabla \operatorname{lse}_\beta(W^{-1}A^T(\pi - h))\right)^T W^{-1}A^T h \\
&\geq \nabla\Phi_\beta(\pi)^T h - \left\|\nabla \operatorname{lse}_\beta(W^{-1}A^T\pi) - \nabla \operatorname{lse}_\beta(W^{-1}A^T(\pi - h))\right\|_1 \left\|W^{-1}A^T h\right\|_\infty \\
&\geq \nabla\Phi_\beta(\pi)^T h - \beta \left\|W^{-1}A^T h\right\|_\infty^2
\end{aligned}
$$

- Suggests to compute $h$ by solving

$$
\max\{\nabla\Phi_\beta(\pi)^T h : \left\|W^{-1}A^T h\right\|_\infty \leq 1\}
$$

- **Another transshipment problem** with demand vector $\nabla\Phi_\beta(\pi)$.

# Generic Update Step

Bounding improvement in objective for generic update $\pi' = \pi - h$:

$$
\begin{aligned}
&\Phi_\beta(\pi) - \Phi_\beta(\pi - h) \\
&\quad \geq \nabla\Phi_\beta(\pi - h)^T h \\
&\quad = \nabla\Phi_\beta(\pi)^T h - \left(\nabla\Phi_\beta(\pi)^T h - \nabla\Phi_\beta(\pi - h)^T h\right) \\
&\quad = \nabla\Phi_\beta(\pi)^T h - \left(\nabla \operatorname{lse}_\beta(W^{-1}A^T\pi) - \nabla \operatorname{lse}_\beta(W^{-1}A^T(\pi - h))\right)^T W^{-1}A^T h \\
&\quad \geq \nabla\Phi_\beta(\pi)^T h - \left\|\nabla \operatorname{lse}_\beta(W^{-1}A^T\pi) - \nabla \operatorname{lse}_\beta(W^{-1}A^T(\pi - h))\right\|_1 \left\|W^{-1}A^T h\right\|_\infty \\
&\quad \geq \nabla\Phi_\beta(\pi)^T h - \beta \left\|W^{-1}A^T h\right\|_\infty^2
\end{aligned}
$$

- Suggests to compute $h$ by solving

$$\max\{\nabla\Phi_\beta(\pi)^T h : \left\|W^{-1}A^T h\right\|_\infty \leq 1\}$$

- **Another transshipment problem** with demand vector $\nabla\Phi_\beta(\pi)$.
- **Key insight:** $\alpha$-approximation with $\alpha = O(\log n)$ is good enough

# Generic Update Step

Bounding improvement in objective for generic update $\pi' = \pi - h$:

$$\begin{aligned}
&\Phi_\beta(\pi) - \Phi_\beta(\pi - h) \\
&\quad \geq \nabla\Phi_\beta(\pi - h)^T h \\
&\quad = \nabla\Phi_\beta(\pi)^T h - \left(\nabla\Phi_\beta(\pi)^T h - \nabla\Phi_\beta(\pi - h)^T h\right) \\
&\quad = \nabla\Phi_\beta(\pi)^T h - \left(\nabla \operatorname{lse}_\beta(W^{-1}A^T\pi) - \nabla \operatorname{lse}_\beta(W^{-1}A^T(\pi - h))\right)^T W^{-1}A^T h \\
&\quad \geq \nabla\Phi_\beta(\pi)^T h - \left\|\nabla \operatorname{lse}_\beta(W^{-1}A^T\pi) - \nabla \operatorname{lse}_\beta(W^{-1}A^T(\pi - h))\right\|_1 \left\|W^{-1}A^T h\right\|_\infty \\
&\quad \geq \nabla\Phi_\beta(\pi)^T h - \beta \left\|W^{-1}A^T h\right\|_\infty^2
\end{aligned}$$

- Suggests to compute $h$ by solving

$$\max\{\nabla\Phi_\beta(\pi)^T h : \left\|W^{-1}A^T h\right\|_\infty \leq 1\}$$

- **Another transshipment problem** with demand vector $\nabla\Phi_\beta(\pi)$.
- **Key insight:** $\alpha$-approximation with $\alpha = O(\log n)$ is good enough

$\Rightarrow$ Solve on spanner with stretch $\alpha = \log n$ of size $O(n \log n)$ ("oracle")

# Gradient Descent Algorithm

**repeat**

  **while** $\frac{4\ln(4m)}{\varepsilon\beta} \geq \Phi_\beta(\pi)$ **do** $\beta \leftarrow \frac{5}{4}\beta$.

  $\tilde{b} \leftarrow P^T \nabla\Phi_\beta(\pi)$, where $P \leftarrow I - \pi b^T$

  $\tilde{h} \leftarrow \alpha$-approximation of $\max\{\tilde{b}^T h : \left\|W^{-1}A^T h\right\|_\infty \leq 1\}$

  $\delta \leftarrow \frac{\tilde{b}^T\tilde{h}}{\left\|W^{-1}A^T P\tilde{h}\right\|_\infty}$

  **if** $\delta > \frac{\varepsilon}{8\alpha}$ **then** $\pi \leftarrow \pi - \frac{\delta}{2\beta\left\|W^{-1}A^T P\tilde{h}\right\|_\infty} P\tilde{h}$.

**until** $\delta \leq \frac{\varepsilon}{8\alpha}$

# Gradient Descent Algorithm

**repeat**

  **while** $\frac{4\ln(4m)}{\varepsilon\beta} \geq \Phi_\beta(\pi)$ **do** $\beta \leftarrow \frac{5}{4}\beta$.

  $\tilde{b} \leftarrow P^T \nabla \Phi_\beta(\pi)$, where $P \leftarrow I - \pi b^T$

  $\tilde{h} \leftarrow \alpha$-approximation of $\max\{\tilde{b}^T h : \left\|W^{-1}A^T h\right\|_\infty \leq 1\}$

  $\delta \leftarrow \frac{\tilde{b}^T \tilde{h}}{\left\|W^{-1}A^T P\tilde{h}\right\|_\infty}$

  **if** $\delta > \frac{\varepsilon}{8\alpha}$ **then** $\pi \leftarrow \pi - \frac{\delta}{2\beta\left\|W^{-1}A^T P\tilde{h}\right\|_\infty} P\tilde{h}$.

**until** $\delta \leq \frac{\varepsilon}{8\alpha}$

Details:

- $\pi$ must stay feasible (projection onto $b^T\pi = 1$)
- $\beta$ needs to be in the right range

# Gradient Descent Algorithm

**repeat**

**while** $\frac{4\ln(4m)}{\varepsilon\beta} \geq \Phi_\beta(\pi)$ **do** $\beta \leftarrow \frac{5}{4}\beta$.

$\tilde{b} \leftarrow P^T \nabla\Phi_\beta(\pi)$, where $P \leftarrow I - \pi b^T$

$\tilde{h} \leftarrow \alpha$-approximation of $\max\{\tilde{b}^T h : \|W^{-1}A^T h\|_\infty \leq 1\}$

$\delta \leftarrow \frac{\tilde{b}^T\tilde{h}}{\|W^{-1}A^T P\tilde{h}\|_\infty}$

**if** $\delta > \frac{\varepsilon}{8\alpha}$ **then** $\pi \leftarrow \pi - \frac{\delta}{2\beta\|W^{-1}A^T P\tilde{h}\|_\infty} P\tilde{h}$.

**until** $\delta \leq \frac{\varepsilon}{8\alpha}$

Details:

- $\pi$ must stay feasible (projection onto $b^T\pi = 1$)
- $\beta$ needs to be in the right range

## Theorem

*Given an $\alpha$-approximate shortest transshipment oracle, one can compute primal solution $x$ and dual solution $y$ such that $\|Wx\|_1 \leq (1+\varepsilon)b^T y$ with $(\varepsilon^{-3}\alpha^2 \log n \log \alpha)$ oracle calls.*

# Implementation in Brodcast Congested Clique

1. **Evaluate Gradient:**
   - Evaluate $(\nabla\Phi_\beta(\pi))_v$ locally at each node $v$
   - $(\nabla\Phi_\beta(\pi))_v$ is a function of $\pi$ and weight of edges incident to $v$ ("edge stretches under current node potentials")
   - Constant #rounds: Make $\pi$ and $(\nabla\Phi_\beta(\pi))$ global knowledge

# Implementation in Brodcast Congested Clique

1. **Evaluate Gradient:**
   - Evaluate $(\nabla\Phi_\beta(\pi))_v$ locally at each node $v$
   - $(\nabla\Phi_\beta(\pi))_v$ is a function of $\pi$ and weight of edges incident to $v$ ("edge stretches under current node potentials")
   - Constant #rounds: Make $\pi$ and $(\nabla\Phi_\beta(\pi))$ global knowledge

2. **Oracle call:**
   - Initially compute spanner in $O(\log n)$ rounds [Baswana/Sen '03]
   - Spanner then is global knowledge (size $O(n \log n)$)
   - At oracle call, make gradient global knowledge (size $O(n)$)
   - Each node can internally compute solution on spanner

# Are we done?

# Approximate SSSP

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

# Approximate SSSP

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

# Approximate SSSP

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 close-to-optimal nodes
    - Repeat transshipment for “bad” nodes only
    - Analysis: Total “mass” reduced by constant fraction in each run

# Approximate SSSP

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 close-to-optimal nodes
   - Repeat transshipment for "bad" nodes only
   - Analysis: Total "mass" reduced by constant fraction in each run

**Theorem**

*We can compute a $(1+\varepsilon)$-approximate distance estimate for each node in the SSSP problem with polylog$(n, \|w\|_\infty)$ calls to our gradient descent algorithm with precision $\varepsilon' = \Omega(\varepsilon^3/(\alpha^2 \log n))$.*

# Comparison to [Sherman SODA'17]

Both papers solve $(1 + \varepsilon)$-approximate shortest transshipment

# Comparison to [Sherman SODA'17]

Both papers solve $(1+\varepsilon)$-approximate shortest transshipment

**Our approach**

- specialized to shortest transshipment

**Sherman '17**

- general norm-minimization framework

# Comparison to [Sherman SODA'17]

Both papers solve $(1+\varepsilon)$-approximate shortest transshipment

**Our approach**

- specialized to shortest transshipment
- oracle calls

**Sherman '17**

- general norm-minimization framework
- generalized preconditioning

# Comparison to [Sherman SODA'17]

Both papers solve $(1+\varepsilon)$-approximate shortest transshipment

**Our approach**

- specialized to shortest transshipment
- oracle calls
- oracle of stretch $O(\log n)$ based on spanner

**Sherman '17**

- general norm-minimization framework
- generalized preconditioning
- preconditioner of stretch $n^{o(1)}$ based on metric embedding

# Comparison to [Sherman SODA'17]

Both papers solve $(1+\varepsilon)$-approximate shortest transshipment

**Our approach**

- specialized to shortest transshipment
- oracle calls
- oracle of stretch $O(\log n)$ based on spanner
- Sequential RAM model: time $O(n^2 \varepsilon^{-3} polylog(n))$ time

**Sherman '17**

- general norm-minimization framework
- generalized preconditioning
- preconditioner of stretch $n^{o(1)}$ based on metric embedding
- Sequential RAM model: time $m^{1+o(1)} \varepsilon^{-2}$

# Comparison to [Sherman SODA'17]

Both papers solve $(1+\varepsilon)$-approximate shortest transshipment

**Our approach**

- specialized to shortest transshipment
- oracle calls
- oracle of stretch $O(\log n)$ based on spanner
- Sequential RAM model: time $O(n^2\varepsilon^{-3} polylog(n))$ time
- (deterministic) extension to approximate SSSP

**Sherman '17**

- general norm-minimization framework
- generalized preconditioning
- preconditioner of stretch $n^{o(1)}$ based on metric embedding
- Sequential RAM model: time $m^{1+o(1)}\varepsilon^{-2}$
- ??

# Comparison to [Sherman SODA'17]

Both papers solve $(1+\varepsilon)$-approximate shortest transshipment

**Our approach**

- specialized to shortest transshipment
- oracle calls
- oracle of stretch $O(\log n)$ based on spanner
- Sequential RAM model: time $O(n^2 \varepsilon^{-3} polylog(n))$ time
- (deterministic) extension to approximate SSSP
- randomized tree solution

**Sherman '17**

- general norm-minimization framework
- generalized preconditioning
- preconditioner of stretch $n^{o(1)}$ based on metric embedding
- Sequential RAM model: time $m^{1+o(1)} \varepsilon^{-2}$
- ??
- ??

# Comparison to [Sherman SODA'17]

Both papers solve $(1+\varepsilon)$-approximate shortest transshipment

**Our approach**

- specialized to shortest transshipment
- oracle calls
- oracle of stretch $O(\log n)$ based on spanner
- Sequential RAM model: time $O(n^2 \varepsilon^{-3} polylog(n))$ time
- (deterministic) extension to approximate SSSP
- randomized tree solution
  $\Rightarrow$ nearly tight approximate SSSP in distributed and streaming models

**Sherman '17**

- general norm-minimization framework
- generalized preconditioning
- preconditioner of stretch $n^{o(1)}$ based on metric embedding
- Sequential RAM model: time $m^{1+o(1)}\varepsilon^{-2}$
- ??
- ??

# Conclusion

## Contributions

1. New approach tailored to efficient implementation in distributed models
2. Gradient descent for shortest transshipment based on oracle calls
3. Additional refinement gives per-node guarantee for approximate SSSP

# Conclusion

## Contributions

1. New approach tailored to efficient implementation in distributed models
2. Gradient descent for shortest transshipment based on oracle calls
3. Additional refinement gives per-node guarantee for approximate SSSP

## Open Problems

1. **Distributed Model:** Faster exact SSSP?
2. **Parallel Model:** Approximate SSSP with $m \cdot poly(\log n, \varepsilon)$ work and $poly(\log n, \varepsilon)$ depth?
3. **RAM Model:** Approximate shortest transshipment in time $m \cdot poly(\log n, \varepsilon)$?

# Conclusion

## Contributions

1. New approach tailored to efficient implementation in distributed models
2. Gradient descent for shortest transshipment based on oracle calls
3. Additional refinement gives per-node guarantee for approximate SSSP

## Open Problems

1. **Distributed Model:** Faster exact SSSP?
2. **Parallel Model:** Approximate SSSP with $m \cdot poly(\log n, \varepsilon)$ work and $poly(\log n, \varepsilon)$ depth?
3. **RAM Model:** Approximate shortest transshipment in time $m \cdot poly(\log n, \varepsilon)$?

**Thank you!**