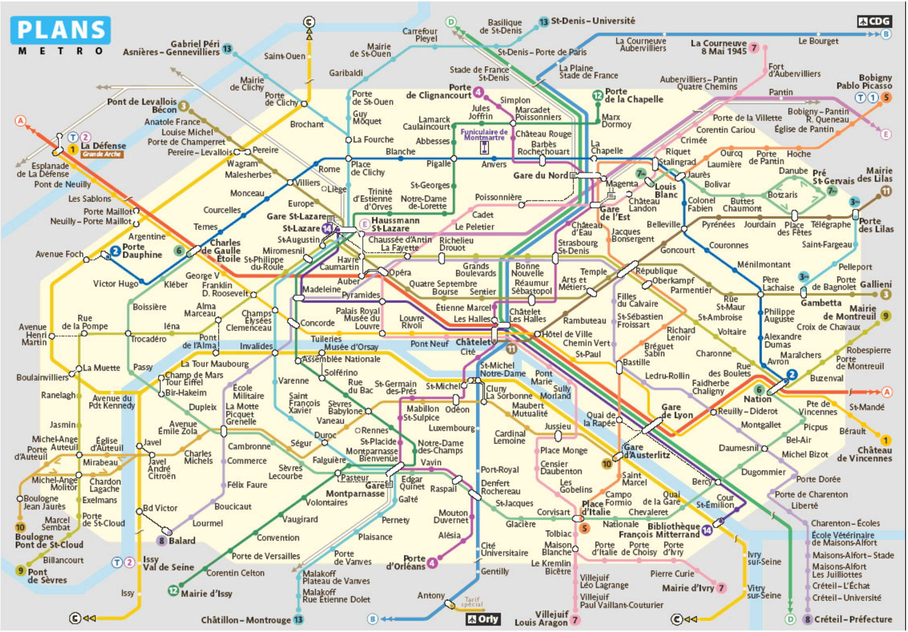
A Faster Distributed Single-Source Shortest Paths Algorithm

Sebastian Forster[1] Danupon Nanongkai[2]

[1]Department of Computer Sciences
University of Salzburg, Austria
Previously known as S. Krinninger

[2]School of Electrical Engineering and Computer Science (EECS)
KTH Royal Institute of Technology, Sweden

FOCS 2018


PLANS
METRO
Gabriel Péri
Asnières – Gennevilliers
Saint-Ouen
Mairie de St-Ouen
Carrefour Pleyel
Basilique de St-Denis
St-Denis – Université
CDG
La Courneuve Aubervilliers
La Courneuve 8 Mai 1945
Le Bourget
St-Denis – Porte de Paris
Garibaldi
Mairie de Clichy
Stade de France St-Denis
La Plaine Stade de France
Fort d'Aubervilliers
Aubervilliers – Pantin Quatre Chemins
Bobigny Pablo Picasso
Pont de Levallois Bécon
Porte de Clichy
Porte de St-Ouen
Porte de Clignancourt
Simplon
Porte de la Chapelle
Pantin
Anatole France
Guy Môquet
Jules Joffrin
Marcadet Poissonniers
Marx Dormoy
Bobigny – Pantin R. Queneau
Louise Michel
Brochant
Lamarck Caulaincourt
Funiculaire de Montmartre
Château Rouge
Porte de la Villette
Église de Pantin
La Défense
Grande Arche
Porte de Champerret
La Fourche
Abbesses
Barbès Rochechouart
La Chapelle
Corentin Cariou
Crimée
Pereire – Levallois
Pereire
Blanche
Riquet
Ourcq
Porte de Pantin
Hoche
Esplanade de La Défense
Wagram
Rome
Place de Clichy
Pigalle
Anvers
Stalingrad
Laumière
Pont de Neuilly
Malesherbes
Villiers
Gare du Nord
Jaurès
Danube
Pré St-Gervais
Mairie des Lilas
Liège
Trinité d'Estienne d'Orves
St-Georges
Notre-Dame de-Lorette
Magenta
Louis Blanc
Bolivar
Les Sablons
Monceau
Poissonnière
Château Landon
Colonel Fabien
Buttes Chaumont
Botzaris
Porte Maillot
Europe
Cadet
Gare de l'Est
Neuilly – Porte Maillot
Courcelles
Gare St-Lazare
Haussmann St-Lazare
Le Peletier
Château d'Eau
Belleville
Pyrénées
Jourdain
Place des Fêtes
Télégraphe
Porte des Lilas
Ternes
St-Lazare
Jacques Bonsergent
Argentine
St-Augustin
Chaussée d'Antin La Fayette
Richelieu Drouot
Strasbourg St-Denis
Goncourt
Couronnes
Saint-Fargeau
Charles de Gaulle Étoile
Miromesnil
Porte Dauphine
Avenue Foch
St-Philippe du-Roule
Havre Caumartin
Ménilmontant
Pelleport
Grands Boulevards
Bonne Nouvelle
Temple
République
George V
Opéra
Père Lachaise
Porte de Bagnolet
Gallieni
Victor Hugo
Kléber
Franklin D. Roosevelt
Auber
Quatre Septembre
Réaumur Sébastopol
Arts et Métiers
Oberkampf
Parmentier
Madeleine
Pyramides
Bourse
Sentier
Filles du Calvaire
Rue St-Maur
Gambetta
Boissière
Alma Marceau
Champs Élysées Clemenceau
Palais Royal Musée du Louvre
Étienne Marcel
Châtelet Les Halles
St-Ambroise
Philippe Auguste
Mairie de Montreuil
Avenue Henri Martin
Rue de la Pompe
Iéna
Concorde
Louvre Rivoli
Les Halles
St-Sébastien Froissart
Rambuteau
Croix de Chavaux
Trocadéro
Pont de l'Alma
Tuileries
Hôtel de Ville
Richard Lenoir
Voltaire
Alexandre Dumas
Invalides
Musée d'Orsay
Pont Neuf
Châtelet
Cité
Chemin Vert
Bréguet Sabin
Charonne
Robespierre
La Muette
Passy
La Tour Maubourg
Assemblée Nationale
St-Paul
Bastille
Maraîchers
Avron
Porte de Montreuil
Boulainvilliers
Champ de Mars Tour Eiffel
St-Michel Notre-Dame
Pont Marie
Rue des Boulets
Solférino
Ledru-Rollin
Faidherbe Chaligny
Buzenval
Bir-Hakeim
École Militaire
Varenne
Rue du Bac
St-Germain des-Prés
St-Michel
Cluny La Sorbonne
Sully Morland
Nation
Ranelagh
Avenue du Pdt Kennedy
Saint François Xavier
Sèvres Babylone
Mabillon
Odéon
Maubert Mutualité
Gare de Lyon
Pte de Vincennes
St-Mandé
Dupleix
La Motte Picquet Grenelle
St-Sulpice
Reuilly – Diderot
Jasmin
Avenue Émile Zola
Vaneau
Luxembourg
Quai de la Rapée
Montgallet
Picpus
Bérault
Rennes
Cardinal Lemoine
Jussieu
Michel-Ange Auteuil
Église d'Auteuil
Javel
Duroc
St-Placide
Notre-Dame des-Champs
Bel-Air
Château de Vincennes
Porte d'Auteuil
Charles Michels
Cambronne
Ségur
Montparnasse Bienvenüe
Place Monge
Gare d'Austerlitz
Daumesnil
Michel Bizot
Mirabeau
Javel André Citroën
Commerce
Falguière
Vavin
Censier Daubenton
Chardon Lagache
Sèvres Lecourbe
Pasteur
Edgar Quinet
Port-Royal
Saint Marcel
Dugommier
Michel-Ange Molitor
Félix Faure
Gare Montparnasse
Raspail
Denfert Rochereau
Les Gobelins
Bercy
Porte Dorée
Boulogne Jean Jaurès
Exelmans
Volontaires
Gaîté
Campo Formio
Quai de la Gare
Cour St-Émilion
Porte de Charenton
Bd Victor
Boucicaut
Mouton Duvernet
St-Jacques
Place d'Italie
Chevaleret
Liberté
Marcel Sembat
Porte de St-Cloud
Lourmel
Vaugirard
Pernety
Corvisart
Glacière
Nationale
Bibliothèque François Mitterrand
Charenton – Écoles
Boulogne Pont de St-Cloud
Balard
Convention
Plaisance
Alésia
Tolbiac
École Vétérinaire de Maisons-Alfort
Billancourt
Cité Universitaire
Maison Blanche
Porte d'Italie
Porte de Choisy
Porte d'Ivry
Maisons-Alfort – Stade
Issy Val de Seine
Porte de Versailles
Porte de Vanves
Porte d'Orléans
Ivry sur-Seine
Maisons-Alfort Les Juilliottes
Pont de Sèvres
Corentin Celton
Malakoff Plateau de Vanves
Gentilly
Le Kremlin Bicêtre
Villejuif Léo Lagrange
Pierre Curie
Créteil – L'Échat
Issy
Mairie d'Issy
Malakoff Rue Étienne Dolet
Antony
Tarif spécial
Villejuif Paul Vaillant-Couturier
Mairie d'Ivry
Vitry sur-Seine
Créteil – Université
Châtillon – Montrouge
Orly
Villejuif Louis Aragon
Créteil – Préfecture

Problem Definition

Goal: Compute shortests paths from a source node s to all other nodes

Problem Definition

Goal: Compute shortests paths from a source node s to all other nodes

How can this be an open problem??

Problem Definition

Goal: Compute shortests paths from a source node s to all other nodes

How can this be an open problem??

- (Nearly) optimal solutions known in RAM model

Problem Definition

Goal: Compute shortests paths from a source node s to all other nodes

How can this be an open problem??

- (Nearly) optimal solutions known in RAM model
- Not fully understood in PRAM model

Problem Definition

Goal: Compute shortests paths from a source node s to all other nodes

How can this be an open problem??

- (Nearly) optimal solutions known in RAM model
- Not fully understood in PRAM model
- Not fully understood in CONGEST model

Problem Definition

Goal: Compute shortests paths from a source node s to all other nodes

How can this be an open problem??

- (Nearly) optimal solutions known in RAM model
- Not fully understood in PRAM model
- Not fully understood in CONGEST model
- To be fair: non-negative weights also not fully understood in RAM model

CONGEST Model

Idea: Measure amount of communication for network to compute result

Running time = #communication rounds

CONGEST Model

Idea: Measure amount of communication for network to compute result
Running time = #communication rounds

Model definition:

- Processors with unique IDs modeled as nodes
- Synchronous rounds (global clock)
- In each round, every node sends (at most) one message to each neighbor
- Message size $O(\log n)$
- Unlimited internal computation between rounds

CONGEST Model

Idea: Measure amount of communication for network to compute result
Running time = #communication rounds

Model definition:

- Processors with unique IDs modeled as nodes
- Synchronous rounds (global clock)
- In each round, every node sends (at most) one message to each neighbor
- Message size $O(\log n)$
- Unlimited internal computation between rounds
- Communication network: unweighted undirected graph of diameter D
- Edges are "annotated" with (non-negative) weights and directions
- Weights represent costs (not time)

CONGEST Model

Idea: *Measure amount of communication for network to compute result*

Running time = #communication rounds

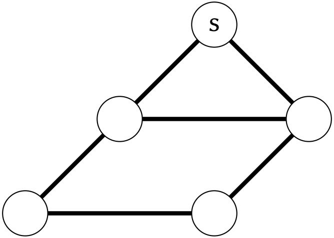
Model definition:

- Processors with unique IDs modeled as nodes
- Synchronous rounds (global clock)
- In each round, every node sends (at most) one message to each neighbor
- Message size $O(\log n)$
- Unlimited internal computation between rounds
- Communication network: unweighted undirected graph of diameter D
- Edges are “annotated” with (non-negative) weights and directions
- Weights represent costs (not time)

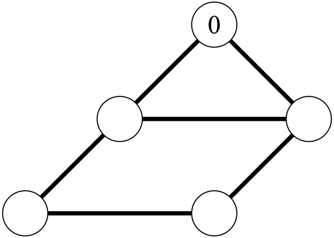
Distributed problem statement:

- Initial knowledge: incident edges, source
- Terminal knowledge: distance to the source, parent on shortest path tree

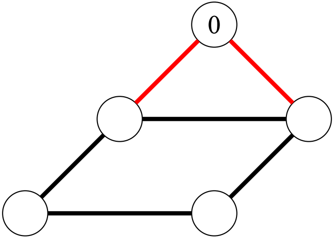
Unweighted Graphs: BFS



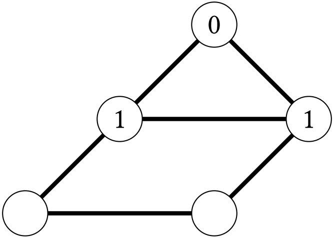
Unweighted Graphs: BFS



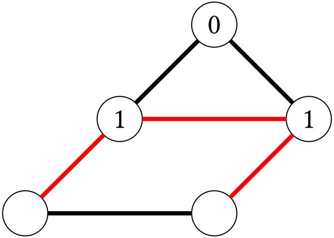
Unweighted Graphs: BFS



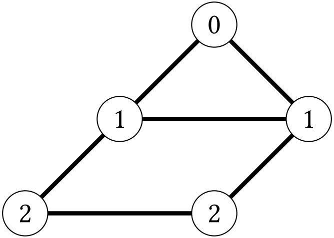
Unweighted Graphs: BFS



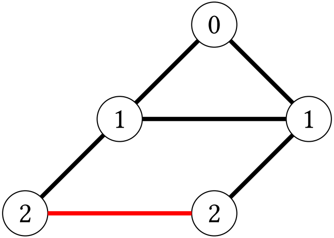
Unweighted Graphs: BFS



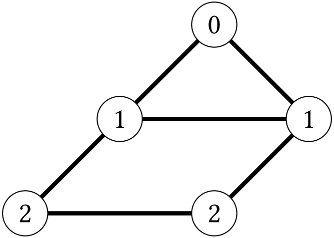
Unweighted Graphs: BFS



Unweighted Graphs: BFS

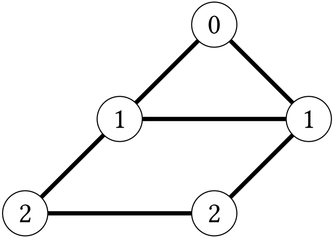


Unweighted Graphs: BFS



Breadth-first search tree can be computed in $O(D)$ rounds.

Unweighted Graphs: BFS



Breadth-first search tree can be computed in $O(D)$ rounds.

Our goal: efficient algorithms for weighted graphs

Comparison Related Work

Lower Bound: $\tilde{\Omega}(\sqrt{n} + D)$ [Peleg/Rubinovich '99] [Das Sarma et al. '11]

Comparison Related Work

Lower Bound: $\tilde{\Omega}(\sqrt{n} + D)$ [Peleg/Rubinovich '99] [Das Sarma et al. '11]

Upper Bounds:

- $O(n)$ (Bellman-Ford)

Comparison Related Work

Lower Bound: $\tilde{\Omega}(\sqrt{n} + D)$ [Peleg/Rubinovich '99] [Das Sarma et al. '11]

Upper Bounds:

- $O(n)$ (Bellman-Ford)
- $\tilde{O}(n^{2/3}D^{1/3} + n^{5/6})$ [Elkin '17]

Comparison Related Work

Lower Bound: $\tilde{\Omega}(\sqrt{n} + D)$ [Peleg/Rubinovich '99] [Das Sarma et al. '11]

Upper Bounds:

- $O(n)$ (Bellman-Ford)
- $\tilde{O}(n^{2/3}D^{1/3} + n^{5/6})$ [Elkin '17]
- $\tilde{O}(n^{3/4}D^{1/4})$ [Ghaffari/Li '18]
- $\tilde{O}(n^{3/4+o(1)} + \min\{n^{3/4}D^{1/6}, n^{6/7}\} + D)$ [Ghaffari/Li '18]

Comparison Related Work

Lower Bound: $\tilde{\Omega}(\sqrt{n} + D)$ [Peleg/Rubinovich '99] [Das Sarma et al. '11]

Upper Bounds:

- $O(n)$ (Bellman-Ford)
- $\tilde{O}(n^{2/3}D^{1/3} + n^{5/6})$ [Elkin '17]
- $\tilde{O}(n^{3/4}D^{1/4})$ [Ghaffari/Li '18]
- $\tilde{O}(n^{3/4+o(1)} + \min\{n^{3/4}D^{1/6}, n^{6/7}\} + D)$ [Ghaffari/Li '18]
- $\tilde{O}(\sqrt{nD})$ **Our result**
- $\tilde{O}(\sqrt{n}D^{1/4} + n^{3/5} + D)$ **Our result**

Comparison Related Work

Lower Bound: $\tilde{\Omega}(\sqrt{n} + D)$ [Peleg/Rubinovich '99] [Das Sarma et al. '11]

Upper Bounds:

- $O(n)$ (Bellman-Ford)
- $\tilde{O}(n^{2/3}D^{1/3} + n^{5/6})$ [Elkin '17]
- $\tilde{O}(n^{3/4}D^{1/4})$ [Ghaffari/Li '18]
- $\tilde{O}(n^{3/4+o(1)} + \min\{n^{3/4}D^{1/6}, n^{6/7}\} + D)$ [Ghaffari/Li '18]
- $\tilde{O}(\sqrt{nD})$ **Our result**
- $\tilde{O}(\sqrt{n}D^{1/4} + n^{3/5} + D)$ **Our result**

All Pairs Shortest Paths: [Holzer/Wattenhofer '12] [Censor-Hillel et al. '15] [Huang/Nanongkai/Saranurak '17] [Agarwal et al. '18] [Agarwal/Ramachandran '18]

Comparison Related Work

Lower Bound: $\tilde{\Omega}(\sqrt{n} + D)$ [Peleg/Rubinovich '99] [Das Sarma et al. '11]

Upper Bounds:

- $O(n)$ (Bellman-Ford)
- $\tilde{O}(n^{2/3}D^{1/3} + n^{5/6})$ [Elkin '17]
- $\tilde{O}(n^{3/4}D^{1/4})$ [Ghaffari/Li '18]
- $\tilde{O}(n^{3/4+o(1)} + \min\{n^{3/4}D^{1/6}, n^{6/7}\} + D)$ [Ghaffari/Li '18]
- $\tilde{O}(\sqrt{nD})$ **Our result**
- $\tilde{O}(\sqrt{n}D^{1/4} + n^{3/5} + D)$ **Our result**

All Pairs Shortest Paths: [Holzer/Wattenhofer '12] [Censor-Hillel et al. '15] [Huang/Nanongkai/Saranurak '17] [Agarwal et al. '18] [Agarwal/Ramachandran '18]

Approximation Algorithms: [Nanongkai '14] [Holzer and Pinsker '15] [Henzinger/K/Nanongkai '16] [Elkin/Neiman '16] [Becker et al. '17]

The Scaling Approach

Two scaling techniques [Gabow '85]:

1. **Bitwise scaling:** In each iteration read next bit of weights
2. **Recursive scaling:** Reduce maximum distance by potential transformation with approximate distances

The Scaling Approach

Two scaling techniques [Gabow '85]:

1. **Bitwise scaling:** In each iteration read next bit of weights
2. **Recursive scaling:** Reduce maximum distance by potential transformation with approximate distances

We follow recursive scaling:

- Similar to [Klein/Subramanian '97] in PRAM model

The Scaling Approach

Two scaling techniques [Gabow '85]:

1. **Bitwise scaling:** In each iteration read next bit of weights
2. **Recursive scaling:** Reduce maximum distance by potential transformation with approximate distances

We follow recursive scaling:

- Similar to [Klein/Subramanian '97] in PRAM model
- Compute approximate distances: $\frac{1}{2} \cdot \mathrm{dist}_G(s, v) \leq \hat{d}(s, v) \leq \mathrm{dist}_G(s, v)$

The Scaling Approach

Two scaling techniques [Gabow '85]:

1. **Bitwise scaling:** In each iteration read next bit of weights
2. **Recursive scaling:** Reduce maximum distance by potential transformation with approximate distances

We follow recursive scaling:

- Similar to [Klein/Subramanian '97] in PRAM model
- Compute approximate distances: $\frac{1}{2} \cdot \mathrm{dist}_G(s, v) \leq \hat{d}(s, v) \leq \mathrm{dist}_G(s, v)$
- Potential transformation: $w'(u, v) = w_G(u, v) + \hat{d}(s, u) - \hat{d}(s, v)$
  Does not change shortest paths

The Scaling Approach

Two scaling techniques [Gabow '85]:

1. **Bitwise scaling:** In each iteration read next bit of weights
2. **Recursive scaling:** Reduce maximum distance by potential transformation with approximate distances

We follow recursive scaling:

- Similar to [Klein/Subramanian '97] in PRAM model
- Compute approximate distances: $\frac{1}{2} \cdot \mathrm{dist}_G(s, v) \leq \hat{d}(s, v) \leq \mathrm{dist}_G(s, v)$
- Potential transformation: $w'(u, v) = w_G(u, v) + \hat{d}(s, u) - \hat{d}(s, v)$
  Does not change shortest paths
- Solve recursively with weights w': Maximum distance has halved!

The Scaling Approach

Two scaling techniques [Gabow '85]:

1. **Bitwise scaling:** In each iteration read next bit of weights
2. **Recursive scaling:** Reduce maximum distance by potential transformation with approximate distances

We follow recursive scaling:

- Similar to [Klein/Subramanian '97] in PRAM model
- Compute approximate distances: $\frac{1}{2} \cdot \mathrm{dist}_G(s, v) \leq \hat{d}(s, v) \leq \mathrm{dist}_G(s, v)$
- Potential transformation: $w'(u, v) = w_G(u, v) + \hat{d}(s, u) - \hat{d}(s, v)$
  Does not change shortest paths
- Solve recursively with weights w': Maximum distance has halved!
- **But:** Want to keep edge weights non-negative

The Scaling Approach

Two scaling techniques [Gabow '85]:

1. **Bitwise scaling:** In each iteration read next bit of weights
2. **Recursive scaling:** Reduce maximum distance by potential transformation with approximate distances

We follow recursive scaling:

- Similar to [Klein/Subramanian '97] in PRAM model
- Compute approximate distances: $\frac{1}{2} \cdot \mathrm{dist}_G(s, v) \leq \hat{d}(s, v) \leq \mathrm{dist}_G(s, v)$
- Potential transformation: $w'(u, v) = w_G(u, v) + \hat{d}(s, u) - \hat{d}(s, v)$
  Does not change shortest paths
- Solve recursively with weights w': Maximum distance has halved!
- **But:** Want to keep edge weights non-negative
- Require: $\hat{d}(s, v) \leq \hat{d}(s, u) + w_G(u, v)$

The Scaling Approach

Two scaling techniques [Gabow '85]:

1. **Bitwise scaling:** In each iteration read next bit of weights
2. **Recursive scaling:** Reduce maximum distance by potential transformation with approximate distances

We follow recursive scaling:

- Similar to [Klein/Subramanian '97] in PRAM model
- Compute approximate distances: $\frac{1}{2} \cdot \mathrm{dist}_G(s, v) \leq \hat{d}(s, v) \leq \mathrm{dist}_G(s, v)$
- Potential transformation: $w'(u, v) = w_G(u, v) + \hat{d}(s, u) - \hat{d}(s, v)$
  Does not change shortest paths
- Solve recursively with weights w': Maximum distance has halved!
- **But:** Want to keep edge weights non-negative
- Require: $\hat{d}(s, v) \leq \hat{d}(s, u) + w_G(u, v)$
- Scaling forces us to solve directed problem

The Scaling Approach

Two scaling techniques [Gabow '85]:

1. **Bitwise scaling:** In each iteration read next bit of weights
2. **Recursive scaling:** Reduce maximum distance by potential transformation with approximate distances

We follow recursive scaling:

- Similar to [Klein/Subramanian '97] in PRAM model
- Compute approximate distances: $\frac{1}{2} \cdot \mathrm{dist}_G(s,v) \leq \hat{d}(s,v) \leq \mathrm{dist}_G(s,v)$
- Potential transformation: $w'(u,v) = w_G(u,v) + \hat{d}(s,u) - \hat{d}(s,v)$
  Does not change shortest paths
- Solve recursively with weights w': Maximum distance has halved!
- **But:** Want to keep edge weights non-negative
- Require: $\hat{d}(s,v) \leq \hat{d}(s,u) + w_G(u,v)$
- Scaling forces us to solve directed problem
- Inherent dependence on $\log(W_{\max})$ due to maximum distance

Reduction

Theorem ([Klein/Subramanian '97])

Suppose auxiliary algorithm computes distance estimate $\hat{d}(s, \cdot)$ *such that*

- *For every node* v*:* $\frac{1}{2} \cdot \mathrm{dist}_G(s, v) \leq \hat{d}(s, v) \leq \mathrm{dist}_G(s, v)$ *(approximation)*
- *For every edge* (u, v)*:* $\hat{d}(s, v) \leq \hat{d}(s, u) + w_G(u, v)$ *(domination)*

Then exact SSSP can be computed by calling auxiliary algorithm $O(\log(nW_{\max}))$ *times (+ bookkeeping work).*

Reduction

Theorem ([Klein/Subramanian '97])

Suppose auxiliary algorithm computes distance estimate $\hat{d}(s, \cdot)$ *such that*

- *For every node* v*:* $\frac{1}{2} \cdot \mathrm{dist}_G(s, v) \leq \hat{d}(s, v) \leq \mathrm{dist}_G(s, v)$ *(approximation)*
- *For every edge* (u, v)*:* $\hat{d}(s, v) \leq \hat{d}(s, u) + w_G(u, v)$ *(domination)*

Then exact SSSP can be computed by calling auxiliary algorithm $O(\log(nW_{\max}))$ *times (+ bookkeeping work).*

Our contribution: Design suitable auxiliary algorithm

Reduction

Theorem ([Klein/Subramanian '97])

Suppose auxiliary algorithm computes distance estimate $\hat{d}(s, \cdot)$ *such that*

- *For every node* v*:* $\frac{1}{2} \cdot \mathrm{dist}_G(s, v) \leq \hat{d}(s, v) \leq \mathrm{dist}_G(s, v)$ *(approximation)*
- *For every edge* (u, v)*:* $\hat{d}(s, v) \leq \hat{d}(s, u) + w_G(u, v)$ *(domination)*

Then exact SSSP can be computed by calling auxiliary algorithm $O(\log(nW_{\max}))$ *times (+ bookkeeping work).*

Our contribution: Design suitable auxiliary algorithm

- Leverage techniques from *approximate* SSSP

Reduction

Theorem ([Klein/Subramanian '97])

Suppose auxiliary algorithm computes distance estimate $\hat{d}(s,\cdot)$ *such that*

- *For every node* v: $\frac{1}{2} \cdot \mathrm{dist}_G(s,v) \leq \hat{d}(s,v) \leq \mathrm{dist}_G(s,v)$ *(approximation)*
- *For every edge* (u,v): $\hat{d}(s,v) \leq \hat{d}(s,u) + w_G(u,v)$ *(domination)*

Then exact SSSP can be computed by calling auxiliary algorithm $O(\log(nW_{\max}))$ *times (+ bookkeeping work).*

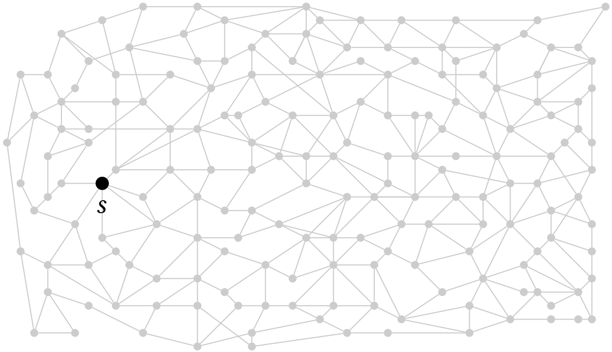
Our contribution: Design suitable auxiliary algorithm

- Leverage techniques from *approximate* SSSP
- Careful design to satisfy domination constraint

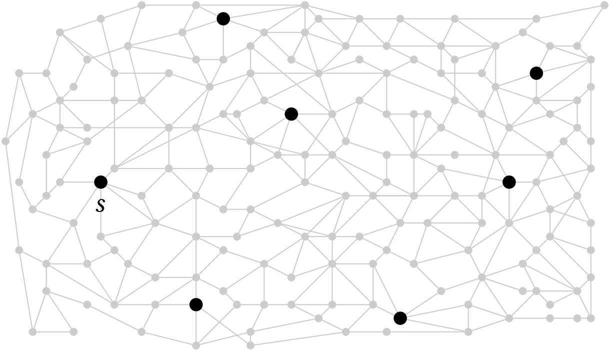
Omitted in this talk:

- Detailed running time analyis
- Dealing with 0-weight edges: Reduce to positive edge weights
- Faster approximation algorithm for directed graphs

Skeleton Graph

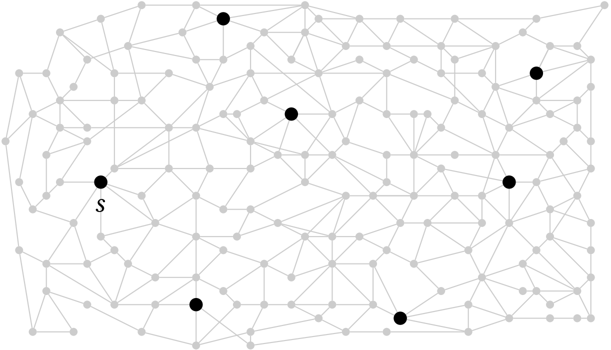


Skeleton Graph



Sample $\tilde{O}(n/h)$ *skeleton nodes* uniformly at random (+ source s)

Skeleton Graph

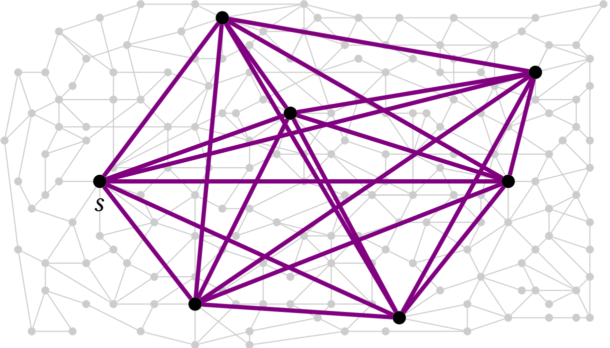


Sample $\tilde{O}(n/h)$ *skeleton nodes* uniformly at random (+ source s)

Lemma (Ullman/Yannakakis '90)

Every shortest path with $h/2$ edges contains skeleton with high probability.

Skeleton Graph



Sample $\tilde{O}(n/h)$ *skeleton nodes* uniformly at random (+ source s)

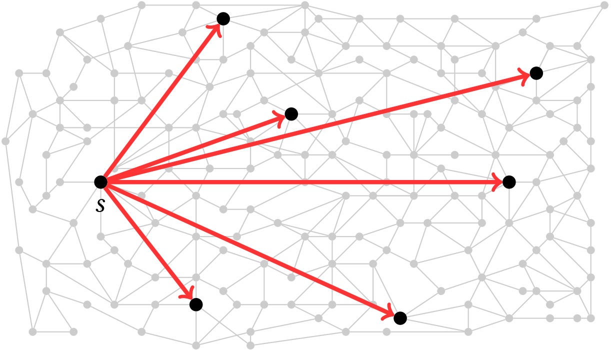
Lemma (Ullman/Yannakakis '90)

Every shortest path with $h/2$ edges contains skeleton with high probability.

Idea:

1. Reduce to computing SSSP on skeleton graph

Skeleton Graph



Sample $\tilde{O}(n/h)$ *skeleton nodes* uniformly at random (+ source s)

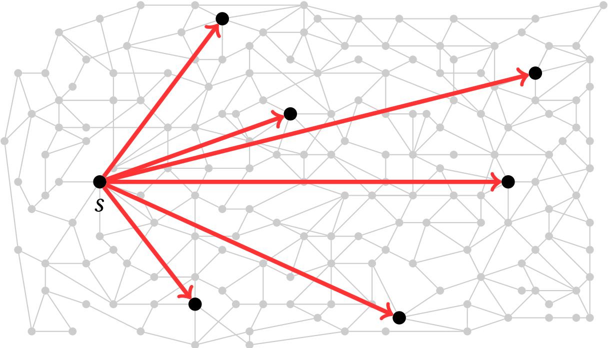
Lemma (Ullman/Yannakakis '90)

Every shortest path with $h/2$ edges contains skeleton with high probability.

Idea:

1. Reduce to computing SSSP on skeleton graph
2. Add skeleton shortcuts

Skeleton Graph



Sample $\tilde{O}(n/h)$ *skeleton nodes* uniformly at random (+ source s)

Lemma (Ullman/Yannakakis '90)

Every shortest path with $h/2$ edges contains skeleton with high probability.

Idea:

1. Reduce to computing SSSP on skeleton graph
2. Add skeleton shortcuts
3. Left to deal with shortest paths with $\leq h$ edges

Auxiliary Algorithm

1. Sample $\tilde{O}(n/h)$ skeleton nodes

Auxiliary Algorithm

1. Sample $\tilde{O}(n/h)$ skeleton nodes
2. Compute approximate skeleton graph H:

Auxiliary Algorithm

1. Sample $\tilde{O}(n/h)$ skeleton nodes
2. Compute approximate skeleton graph H:
   - Compute $\tilde{d}(x, y)$ for every pair of skeleton nodes x, y s.t. $\mathrm{dist}_G(x, y) \leq \tilde{d}(x, y) \leq 2\,\mathrm{dist}_G(x, y)$ [Nanongkai '14]
   - Set $w_H(x, y) := \frac{1}{2}\tilde{d}(x, y)$

Auxiliary Algorithm

1. Sample $\tilde{O}(n/h)$ skeleton nodes
2. Compute approximate skeleton graph H:
   - Compute $\tilde{d}(x, y)$ for every pair of skeleton nodes x, y s.t. $\mathrm{dist}_G(x, y) \leq \tilde{d}(x, y) \leq 2\,\mathrm{dist}_G(x, y)$ [Nanongkai '14]
   - Set $w_H(x, y) := \frac{1}{2}\tilde{d}(x, y)$
3. Solve *exact* SSSP on skeleton

Auxiliary Algorithm

1. Sample $\tilde{O}(n/h)$ skeleton nodes
2. Compute approximate skeleton graph H:
   - Compute $\tilde{d}(x,y)$ for every pair of skeleton nodes x,y s.t. $\mathrm{dist}_G(x,y) \le \tilde{d}(x,y) \le 2\,\mathrm{dist}_G(x,y)$ [Nanongkai '14]
   - Set $w_H(x,y) := \frac{1}{2}\tilde{d}(x,y)$
3. Solve *exact* SSSP on skeleton
   Compute $\mathrm{dist}_H(s,x)$ for every skeleton node x

Auxiliary Algorithm

1. Sample $\tilde{O}(n/h)$ skeleton nodes
2. Compute approximate skeleton graph H:
   - Compute $\tilde{d}(x, y)$ for every pair of skeleton nodes x, y s.t. $\mathrm{dist}_G(x, y) \leq \tilde{d}(x, y) \leq 2\,\mathrm{dist}_G(x, y)$ [Nanongkai '14]
   - Set $w_H(x, y) := \frac{1}{2}\tilde{d}(x, y)$
3. Solve *exact* SSSP on skeleton
   Compute $\mathrm{dist}_H(s, x)$ for every skeleton node x
4. Augment original graph G to G' by adding skeleton shortcuts

Auxiliary Algorithm

1. Sample $\tilde{O}(n/h)$ skeleton nodes
2. Compute approximate skeleton graph H:
   - Compute $\tilde{d}(x, y)$ for every pair of skeleton nodes x, y s.t. $\text{dist}_G(x, y) \leq \tilde{d}(x, y) \leq 2\,\text{dist}_G(x, y)$ [Nanongkai '14]
   - Set $w_H(x, y) := \frac{1}{2}\tilde{d}(x, y)$
3. Solve *exact* SSSP on skeleton
   Compute $\text{dist}_H(s, x)$ for every skeleton node x
4. Augment original graph G to G' by adding skeleton shortcuts
   Set $w_{G'}(s, x) = dist_H(s, x)$ for every skeleton node x

Auxiliary Algorithm

1. Sample $\tilde{O}(n/h)$ skeleton nodes
2. Compute approximate skeleton graph H:
   - Compute $\tilde{d}(x, y)$ for every pair of skeleton nodes x, y s.t. $\mathrm{dist}_G(x, y) \le \tilde{d}(x, y) \le 2\,\mathrm{dist}_G(x, y)$ [Nanongkai '14]
   - Set $w_H(x, y) := \frac{1}{2}\tilde{d}(x, y)$
3. Solve *exact* SSSP on skeleton
   Compute $\mathrm{dist}_H(s, x)$ for every skeleton node x
4. Augment original graph G to G' by adding skeleton shortcuts
   Set $w_{G'}(s, x) = dist_H(s, x)$ for every skeleton node x
5. Compute h-hop distances in G': $\hat{d}(s, v) := \mathrm{dist}^h_{G'}(s, v)$ for every node v

Auxiliary Algorithm

1. Sample $\tilde{O}(n/h)$ skeleton nodes
2. Compute approximate skeleton graph H:
   - Compute $\tilde{d}(x, y)$ for every pair of skeleton nodes x, y s.t. $\mathrm{dist}_G(x, y) \leq \tilde{d}(x, y) \leq 2\,\mathrm{dist}_G(x, y)$ [Nanongkai '14]
   - Set $w_H(x, y) := \frac{1}{2}\tilde{d}(x, y)$
3. Solve *exact* SSSP on skeleton
   Compute $\mathrm{dist}_H(s, x)$ for every skeleton node x
4. Augment original graph G to G' by adding skeleton shortcuts
   Set $w_{G'}(s, x) = dist_H(s, x)$ for every skeleton node x
5. Compute h-hop distances in G': $\hat{d}(s, v) := \mathrm{dist}^h_{G'}(s, v)$ for every node v
   Shortest path using at most h edges: h iterations of Bellman-Ford

Auxiliary Algorithm

1. Sample $\tilde{O}(n/h)$ skeleton nodes
2. Compute approximate skeleton graph H:
   - Compute $\tilde{d}(x, y)$ for every pair of skeleton nodes x, y s.t. $\mathrm{dist}_G(x, y) \leq \tilde{d}(x, y) \leq 2\,\mathrm{dist}_G(x, y)$ [Nanongkai '14]
   - Set $w_H(x, y) := \frac{1}{2}\tilde{d}(x, y)$
3. Solve *exact* SSSP on skeleton
   Compute $\mathrm{dist}_H(s, x)$ for every skeleton node x
4. Augment original graph G to G' by adding skeleton shortcuts
   Set $w_{G'}(s, x) = dist_H(s, x)$ for every skeleton node x
5. Compute h-hop distances in G': $\hat{d}(s, v) := \mathrm{dist}^h_{G'}(s, v)$ for every node v
   Shortest path using at most h edges: h iterations of Bellman-Ford

Theorem

- *For every node v: $\frac{1}{2} \cdot \mathrm{dist}_G(s, v) \leq \hat{d}(s, v) \leq \mathrm{dist}_G(s, v)$ (approximation)*
- *For every edge (u, v): $\hat{d}(s, v) \leq \hat{d}(s, u) + w_G(u, v)$ (domination)*

Proof of Domination

- Need to show: $\mathrm{dist}^h_{G'}(s, v) \leq \mathrm{dist}^h_{G'}(s, u) + w_G(u, v)$

Proof of Domination

- Need to show: $\mathrm{dist}_{G'}^{h}(s, v) \leq \mathrm{dist}_{G'}^{h}(s, u) + w_G(u, v)$
- We show that $\mathrm{dist}_{G'}^{h}(s, v) = \mathrm{dist}_{G'}(s, v)$

Proof of Domination

- Need to show: $\mathrm{dist}^h_{G'}(s, v) \le \mathrm{dist}^h_{G'}(s, u) + w_G(u, v)$
- We show that $\mathrm{dist}^h_{G'}(s, v) = \mathrm{dist}_{G'}(s, v)$
- Then domination follows from triangle inequality

Proof of Domination

- Need to show: $\mathrm{dist}^h_{G'}(s, v) \leq \mathrm{dist}^h_{G'}(s, u) + w_G(u, v)$
- We show that $\mathrm{dist}^h_{G'}(s, v) = \mathrm{dist}_{G'}(s, v)$
- Then domination follows from triangle inequality

Proof idea:

- Shortest path in G' has the following structure: at most one shortcut edge to skeleton node followed by a shortest path π in G

Proof of Domination

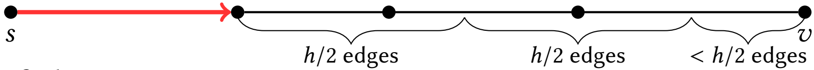
- Need to show: $\mathrm{dist}^h_{G'}(s,v) \leq \mathrm{dist}^h_{G'}(s,u) + w_G(u,v)$
- We show that $\mathrm{dist}^h_{G'}(s,v) = \mathrm{dist}_{G'}(s,v)$
- Then domination follows from triangle inequality

Proof idea:

- Shortest path in G' has the following structure: at most one shortcut edge to skeleton node followed by a shortest path π in G
- Subdivide π into subsequent chunks of $h/2$ edges

Proof of Domination

- Need to show: $\mathrm{dist}^h_{G'}(s,v) \le \mathrm{dist}^h_{G'}(s,u) + w_G(u,v)$
- We show that $\mathrm{dist}^h_{G'}(s,v) = \mathrm{dist}_{G'}(s,v)$
- Then domination follows from triangle inequality

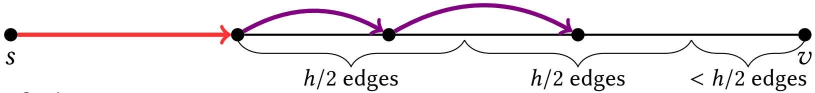


Proof idea:

- Shortest path in G' has the following structure: at most one shortcut edge to skeleton node followed by a shortest path π in G
- Subdivide π into subsequent chunks of $h/2$ edges
- With high probability, each chunk contains a skeleton node

Proof of Domination

- Need to show: $\text{dist}^h_{G'}(s, v) \leq \text{dist}^h_{G'}(s, u) + w_G(u, v)$
- We show that $\text{dist}^h_{G'}(s, v) = \text{dist}_{G'}(s, v)$
- Then domination follows from triangle inequality

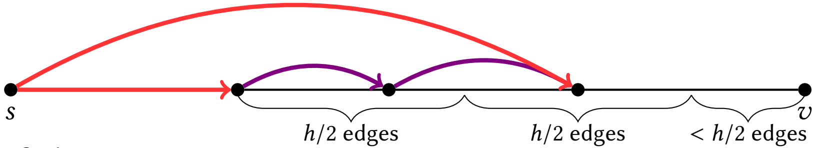


Proof idea:

- Shortest path in G' has the following structure: at most one shortcut edge to skeleton node followed by a shortest path π in G
- Subdivide π into subsequent chunks of $h/2$ edges
- With high probability, each chunk contains a skeleton node
- Following skeleton nodes with skeleton edges would be at least as cheap as following π (underestimated approximation!)

Proof of Domination

- Need to show: $\mathrm{dist}^h_{G'}(s,v) \leq \mathrm{dist}^h_{G'}(s,u) + w_G(u,v)$
- We show that $\mathrm{dist}^h_{G'}(s,v) = \mathrm{dist}_{G'}(s,v)$
- Then domination follows from triangle inequality

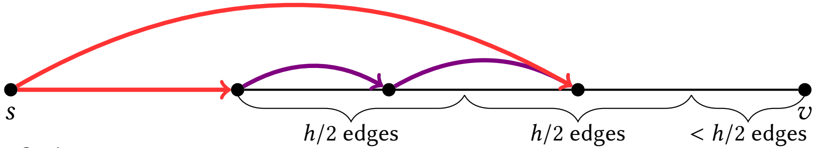


Proof idea:

- Shortest path in G' has the following structure: at most one shortcut edge to skeleton node followed by a shortest path π in G
- Subdivide π into subsequent chunks of $h/2$ edges
- With high probability, each chunk contains a skeleton node
- Following skeleton nodes with skeleton edges would be at least as cheap as following π (underestimated approximation!)
- Shortcut edge in G' to last skeleton node is as least as cheap

Proof of Domination

- Need to show: $\mathrm{dist}^h_{G'}(s, v) \leq \mathrm{dist}^h_{G'}(s, u) + w_G(u, v)$
- We show that $\mathrm{dist}^h_{G'}(s, v) = \mathrm{dist}_{G'}(s, v)$
- Then domination follows from triangle inequality

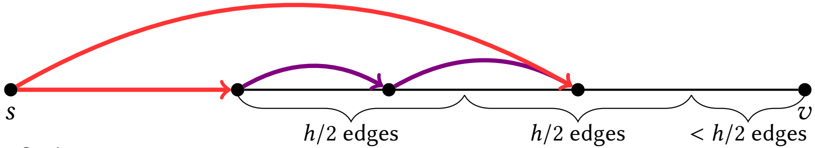


Proof idea:

- Shortest path in G' has the following structure: at most one shortcut edge to skeleton node followed by a shortest path π in G
- Subdivide π into subsequent chunks of $h/2$ edges
- With high probability, each chunk contains a skeleton node
- Following skeleton nodes with skeleton edges would be at least as cheap as following π (underestimated approximation!)
- Shortcut edge in G' to last skeleton node is as least as cheap
- Reason: Triangle inequality for exact distances!

Proof of Domination

- Need to show: $\mathrm{dist}^h_{G'}(s, v) \leq \mathrm{dist}^h_{G'}(s, u) + w_G(u, v)$
- We show that $\mathrm{dist}^h_{G'}(s, v) = \mathrm{dist}_{G'}(s, v)$
- Then domination follows from triangle inequality



Proof idea:

- Shortest path in G' has the following structure: at most one shortcut edge to skeleton node followed by a shortest path π in G
- Subdivide π into subsequent chunks of $h/2$ edges
- With high probability, each chunk contains a skeleton node
- Following skeleton nodes with skeleton edges would be at least as cheap as following π (underestimated approximation!)
- Shortcut edge in G' to last skeleton node is as least as cheap
- Reason: Triangle inequality for exact distances!
- Now: remainder of π has $< h$ edges

How to Solve on Skeleton

Recall: We need *exact* SSSP on skeleton

How to Solve on Skeleton

Recall: We need *exact* SSSP on skeleton

Two Variants:

1. Dijkstra's algorithm

2. Recurse

How to Solve on Skeleton

Recall: We need *exact* SSSP on skeleton

Two Variants:

1. Dijkstra's algorithm
   Running time: $\tilde{O}(\sqrt{nD})$
2. Recurse
   Running time: $\tilde{O}(\sqrt{n}D^{1/4} + n^{3/5} + D)$

How to Solve on Skeleton

Recall: We need *exact* SSSP on skeleton

Two Variants:

1. Dijkstra's algorithm
   Running time: $\tilde{O}(\sqrt{nD})$
2. Recurse
   Running time: $\tilde{O}(\sqrt{n}D^{1/4} + n^{3/5} + D)$

Why is SSSP instance different?

- Small size

How to Solve on Skeleton

Recall: We need *exact* SSSP on skeleton

Two Variants:

1. Dijkstra's algorithm
   Running time: $\tilde{O}(\sqrt{nD})$
2. Recurse
   Running time: $\tilde{O}(\sqrt{n}D^{1/4} + n^{3/5} + D)$

Why is SSSP instance different?

- Small size
- Computation on skeleton via broadcasting in original network

Discussion: Implementation of Klein/Subramanian?

We borrow many ideas from PRAM algorithm of Klein and Subramanian

Discussion: Implementation of Klein/Subramanian?

We borrow many ideas from PRAM algorithm of Klein and Subramanian

Main difference:

- Klein and Subramanian treat skeleton edges as a hop set
- We solve SSSP on skeleton explicitly

Discussion: Implementation of Klein/Subramanian?

We borrow many ideas from PRAM algorithm of Klein and Subramanian

Main difference:

- Klein and Subramanian treat skeleton edges as a hop set
- We solve SSSP on skeleton explicitly

New trade-off for directed graphs in PRAM model:

- Klein and Subramanian: work $\tilde{O}(m\sqrt{n})$ and depth $\tilde{O}(\sqrt{n})$
- Our approach: work $\tilde{O}((n^3/h^3 + mh + mn/h))$ and depth $\tilde{O}(h)$

Open Problems

1. Match the single-source reachability barrier for SSSP!

Open Problems

1. Match the single-source reachability barrier for SSSP!
   - CONGEST model: $\tilde{O}(\sqrt{n}D^{1/4} + D)$ rounds [Ghaffari/Udwani '15]

Open Problems

1. Match the single-source reachability barrier for SSSP!
   - CONGEST model: $\tilde{O}(\sqrt{n}D^{1/4} + D)$ rounds [Ghaffari/Udwani '15] (Is this tight??)

Open Problems

1. Match the single-source reachability barrier for SSSP!
    - CONGEST model: $\tilde{O}(\sqrt{n}D^{1/4} + D)$ rounds [Ghaffari/Udwani '15] (Is this tight??)
    - PRAM model:
        - $\tilde{O}(mh + h^4/n)$ work and $\tilde{O}(n/h)$ depth [Ullman/Yannakakis '90]
        - $\tilde{O}(m)$ work and $\tilde{O}(n^{2/3})$ depth [Fineman '18]

Open Problems

1. Match the single-source reachability barrier for SSSP!
   - CONGEST model: $\tilde{O}(\sqrt{n}D^{1/4} + D)$ rounds [Ghaffari/Udwani '15] (Is this tight??)
   - PRAM model:
     - $\tilde{O}(mh + h^4/n)$ work and $\tilde{O}(n/h)$ depth [Ullman/Yannakakis '90]
     - $\tilde{O}(m)$ work and $\tilde{O}(n^{2/3})$ depth [Fineman '18]
2. Deterministic algorithms?

Open Problems

1. Match the single-source reachability barrier for SSSP!
   - CONGEST model: $\tilde{O}(\sqrt{n}D^{1/4} + D)$ rounds [Ghaffari/Udwani '15] (Is this tight??)
   - PRAM model:
     - $\tilde{O}(mh + h^4/n)$ work and $\tilde{O}(n/h)$ depth [Ullman/Yannakakis '90]
     - $\tilde{O}(m)$ work and $\tilde{O}(n^{2/3})$ depth [Fineman '18]
2. Deterministic algorithms?

Thank you for your attention!

slides: https://www.cosy.sbg.ac.at/~forster/