Efficient Distributed Approximation Algorithms

Gopal Pandurangan

Department of Computer Science
Purdue University
Distributed Network Algorithms

- Emerging networking technologies: peer-to-peer networks, overlay networks, ad hoc wireless and sensor networks --- resource-constrained, dynamic, unreliable..

- Critical to design and analyze efficient, localized, distributed algorithms for solving various fundamental network optimization problems.
Distributed Algorithms

- Algorithms designed to work on an inter-connected network of computers distributed across many sites.

- Typically executed **concurrently**, with separate parts of the algorithm being run simultaneously on independent processors.

- Needed for efficient operation of large-scale communication networks: inherently scalable, robust, usually requires no global information.

- More challenging to design and analyze than traditional (sequential) algorithms.

- Complexity measures: Communication (messages), time (number of rounds), energy (power), ...
Fundamental Network Optimization Problems

- **Minimum Spanning Tree (MST):** leader election, broadcast, convergecast, data aggregation,…
- **Minimum Steiner Tree:** multicast routing, …
- **Shortest Paths:** shortest path routing, data aggregation …
- **Highly Connected Subgraphs:** fault-tolerance,…
- **Spanner:** Sparse backbone for efficient routing..
- **Dominating Set:** communication backbone, resource-efficient routing,…
- ….
Distributed Approximation Algorithms

Trade-off optimality of the solution for the amount of resources consumed by the distributed algorithm.

Motivation:

- Ad hoc wireless sensor networks and peer-to-peer networks operate under inherent resource constraints such as energy, bandwidth etc.
- Topology can also change dynamically.
- Low communication complexity, fast running time, low energy, even at the cost of reduced quality of solution.
Traffic Monitoring with Sensors
Data Aggregation - Low Cost Tree

- Aggregate data on a tree
- Use a low cost tree
Desirable Features

Due to the Presidential visit to Portland, some TripCheck cameras may not be displaying current images.

- Simple and local
Desirable Features

Due to the Presidential visit to Portland, some TripCheck cameras may not be displaying current images.

- Simple and local
Desirable Features

- Simple and local
- Dynamic - handle node failures

Due to the Presidential visit to Portland, some TripCheck cameras may not be displaying current images.
Desirable Features

Due to the Presidential visit to Portland, some TripCheck cameras may not be displaying current images.

- Simple and local
- Dynamic - handle node failures
- Distributed
- Low energy
- Low synchronization
- Small number of messages
- Low degree
Problem

- Find a Minimum Spanning Tree (MST) rooted at a given node.

- MST is a difficult problem to solve distributively.

- Can we construct an approximately good spanning tree efficiently in a distributed manner?
Road Map

- Distributed Approximation Algorithm for MST
  - Nearest Neighbor Tree (NNT) Scheme

- Wireless Networks: Energy-Efficient Distributed MST Algorithms

- A Uniform Approach to Distributed Approximation
  - Shortest Paths, Steiner Forest, Routing Cost Tree …
  - Leader Election
  - Probabilistic tree embedding
Distributed Minimum Spanning Tree (MST) Problem

- Smallest (weighted) set of edges needed to maintain network connectivity.

- Distributed algorithms for (exact) MST are well-known:
  - Optimal with respect to either message or time complexity.
  - Time can be $\sqrt{n}$ and messages can be $\Omega(n^2)$.
  - Relatively complex.

- Motivates simple, efficient, distributed, approximate MST algorithms.
- Tradeoff optimality of MST for low communication and time complexity.
Distributed Computation Model

- Undirected weighted network
- Each node has unique ID
- Each node initially knows only the weights of the adjacent edges
- Communication is synchronous
  - Occurs in discrete time steps (rounds)
  - In each time step, a node can send a message along each adjacent edge
- Size of a message: $O(\log n)$ (CONGEST Model)
- Weight of an edge $O(n^k)$
- Focus: Message and time complexity
Prior Distributed MST Algorithms

- Gallager, Humblet, & Spira ’83 (GHS): $O(n \log n)$ running time
  message: $O(|E| + n \log n)$ (optimal)

- Chin & Ting ’85: $O(n \log \log n)$ time

- Gafni ’85: $O(n \log^* n)$

- Awerbuch ’87: $O(n)$, existentially optimal

- Garay, Kutten, & Peleg ’98: $O(D + n^{0.61})$, Diameter $D$

- Kutten & Peleg ’98: $O(D + \sqrt{n \log^* n})$

- Elkin ’04: $\tilde{O}(\mu + \sqrt{n})$, $\mu$ is called MST radius
  - Cannot detect termination unless $\mu$ is given as input.

- Peleg & Rubinovich (’99) showed a lower bound of $\tilde{\Omega}(\sqrt{n})$ for running time.
Approximation Algorithms for MST

- Peleg & Rubinovich (FOCS 99) first raised the question of approximation algorithm for MST
  "To the best of our knowledge nothing nontrivial is known about this problem."

- An important hardness result by Elkin (STOC ’04): lower bound on running time for any H-approx. algorithm of
  \[ \Omega \left( \sqrt{\frac{n}{H \log n}} \right) \]

- An approx. algorithm by Elkin (STOC ’04) with running time
  \[ O \left( D + \frac{w_{\text{max}}}{H - 1} \log^* n \right), \quad w_{\text{max}} \text{ is max-weight/min-weight} \]
  depends on edge weights

17
Distributed Approximation for MST


- First time-optimal distributed $O(\log n)$-approximation algorithm for MST.

- **Running time:** $O(D + L \log n)$
  - $D$ is the **diameter** of the network
  - $L$ is called the **local shortest path diameter (LSPD)** of the network
  - $1 \leq L \leq n - 1$
  - Typically, $L$ can be much smaller than $\sqrt{n}$
  - Independent of edge weights

- **Message complexity:** $O(|E| \log L \log n)$
Distributed Approximation for MST

- $L$ is not arbitrary – captures the hardness quite precisely.
  - there is a family of graphs, for which any distributed algorithm needs $\Omega(D + L)$ time to compute $H$-approximate MST for any $H \in [1, \log n]$.
  - Our algorithm is existentially optimal (up to polylogarithmic factor).
- For some graphs, our algorithm is exponentially faster than any exact MST algorithm: our algorithm takes $\tilde{O}(1)$ time while any MST algorithm will take $\tilde{\Omega}(\sqrt{n})$ time.
- Our algorithm can be used to find an approximate MST in wireless networks (modeled as unit-disk graphs) and in random weighted networks (which can model power-law networks such as the Internet) in almost optimal $\Tilde{O}(D(G))$ time.
- Distributed approximation algorithm for Minimum Steiner Trees.
Local Shortest Path Diameter, $L$

For node $v$, we first define $L(v)$:

- **Maximum** of the adjacent edge weights, $W(v) = 7$

Consider all nodes within distance $W(v)$ from $v$: 7-neighborhood of $v$.

$L(v)$ is the *maximum number of hops in a shortest path* from $v$ to any node in this neighborhood.

In the above, $L(v) = 2$

$L$ is the maximum of $L(v)$'s over all $v$. 

Nodes with red circle are in $\Gamma_{W(v)}(v)$.
Unit disk graphs (UDG):
- Euclidean graphs where \((u,v) \in E \iff \text{dist}(u, v) \leq R\).
- Popular models for wireless networks.
- Theorem: \(L = 1\).

Graphs with random edge weights:
- Arbitrary topology.
- Edge weights are chosen independently and randomly from any arbitrary distribution in \([0,1]\) (with constant mean).
- Can model real-world networks such as Internet and peer-to-peer networks.
- Theorem: \(L = O(\log n)\) W.H.P.
Given: A (connected) undirected weighted graph G.

- Each node chooses a unique rank.
- Each node connects to its nearest node (via a shortest path) of higher rank.
NNT Construction: Example

Produces a spanning subgraph --- can contain cycles. Output is a spanning tree of this subgraph --- NNT.
Choosing Ranks

Random NNT: each node chooses a rank randomly and independently in [0,1].

Coordinate NNT (geometric setting): use (x,y) coordinates as rank.
Theorem:
On any graph G, NNT scheme (regardless of how ranks are chosen) produces a spanning tree that has a cost of at most $O(\log n)$ times the (optimal) MST.
Proof of NNT Theorem

Without loss of generality, assume that the given graph $G$ is a metric (complete) graph.

(For an arbitrary graph, construct a complete graph with edge weights given by the shortest paths.)

Main steps:

1. Find a MST of $G$.
2. Modify MST into a Hamiltonian path: Euler tour and shortcutting.
3. Induction on segments of path.
Constructing a Hamiltonian Path: Euler Tour and Shortcutting

Cost of path $P(A) \leq 2 \text{cost(MST)}$
Inductive hypothesis: for any set $S$ of $r$ consecutive vertices on this path $\text{cost}(\text{NNT}(S)) \leq \log r \text{ cost}(\text{P}(S))$
Partitioning into Smaller Instances

\[ P(A_1) = \begin{align*}
A_1 & \vdash 7, 2, 6, 5 \\
& \vdash 1, 4, 8, 1
\end{align*} \]

\[ P(A_2) = \begin{align*}
A_2 & \vdash 3, 4, 8 \\
& \vdash 1
\end{align*} \]
Inductive Step

\[ \text{cost}(\text{NNT}(A_1)) \leq (\log n/2) \text{cost}(P(A_1)) \]
Inductive Step

\[
\text{cost}(\text{NNT}(A_2)) \leq (\log n/2) \text{cost}(P(A_2))
\]
Inductive Construction of $\text{NNT}(A_1 \cup A_2)$

Edges incident on non-root nodes can only become shorter

Earlier local root chooses a new edge
Inductive Construction of $\text{NNT}(A_1 \cup A_2)$

$\text{cost}(\text{NNT}(A)) \leq \text{cost}(\text{NNT}(A_1)) + \text{cost}(\text{NNT}(A_2)) + \text{cost}(P(A))$

$\leq (\log n/2) (\text{cost}(P(A_1)) + \text{cost}(P(A_2))) + \text{cost}(P(A))$

$\leq (\log n/2) \text{cost}(P(A)) + \text{cost}(P(A))$

$= \log n \text{ cost}(P(A))$

$\leq 2 \log n \text{ cost}(\text{MST})$. 
Distributed NNT Algorithm


Each node executes the same algorithm simultaneously:

- Rank Selection
  - Every node should have ``close by” node of higher rank.
- Finding the nearest node of higher rank.
  - Controlling congestion.
- Connecting to the nearest node of higher rank.
  - Avoiding cycle formation.
Rank Selection

- Elect a leader \( s \) using a leader election algorithm.
- \( s \) selects an arbitrary number \( p(s) \).
- \( s \) sends \( \text{ID}(s) \) and \( p(s) \) to all of its neighbors in one time step.
- Any other node \( u \) after receiving the first message with \( \text{ID}(v) \) and \( p(v) \) from a neighbor \( v \):
  - Selects a number \( p(u) < p(v) \)
  - Sends \( \text{ID}(u) \) and \( p(u) \) to all of its neighbors
Defining Rank

- For any $u$ and $v$, $r(u) < r(v)$ iff
  - $p(u) < p(v)$
  - or $p(u) = p(v)$ and $\text{ID}(u) < \text{ID}(v)$

- A node with lower number $p()$ has lower rank.
- Ties are broken using $\text{ID}()$
Rank Selection (cont.)

- At the end of the rank selection procedure

  - Each node knows the rank of all of its neighbors
  - The leader $s$ has the highest rank among all nodes in the graph
  - For every node (except $s$), there is a neighbor with higher rank.
Finding a Higher Ranked Node

$W(v) = 7$

Nodes with red circle are in $\Gamma_{W(v)}(v)$

$L(v) = 2$

$v$ needs to explore only the nodes in $\Gamma_{W(v)}(v)$.

In principle, we can do in $O(L)$ time.
Finding a Higher Ranked Node

- $v$ executes the algorithm in one or more phases
  - In the first phase, $v$ sets $\rho = 1$
    - ($\rho$ determines the exploration distance from $v$.)
  - In the subsequent phases, $\rho$ is doubled. In $i^{th}$ phase, $\rho = 2^{i-1}$
  - In each phase, $v$ explores the nodes in $\Gamma_{\rho}(v)$ ($\rho$-neighborhood of $v$)
  - $\rho$ needs to be increased to at most $W(v)$
    - There is a node $u \in \Gamma_{W(v)}(v)$ with $r(u) > r(v)$
Finding a Higher Ranked Node

- Each phase consists of one or more rounds
  - $\lambda$ determines the number of hops from $v$.
  - In the first round, $\lambda = 1$
  - In each subsequent round, $\lambda$ is doubled
  - Phase $\rho$, round $\lambda$: $v$ explores the nodes in $\Gamma_{\rho,\lambda}(v)$ by sending $explore$ messages to all of its neighbors and the neighbors forward the messages
  - If a higher ranked node is found, exploration is finished.
  - If the (total) number of nodes explored in the two successive rounds are the same, move to the next phase.
Controlling Congestion

- Many nodes may have overlapping $\rho$-neighborhood and create congestions by the *explore* messages: can be as much as $\theta(n)$.
- We keep the congestion bounded by $O(1)$.
- When $v$ receives *explore* messages for several originators $u_i$, $v$ forwards only one of these.
Controlling Congestion (cont.)

- Arrange $u_i$'s in increasing order of ranks.
- If $r(u_i) < r(u_j)$ and $\rho_i \geq \rho_j$, $v$ sends a found message to $u_i$.
- From the rest, let $u_k$ be the lowest ranked node. Then $\rho_k$ is also least among the rest.
- Forward the message of $u_k$ and send wait message back to the rest.

**Lemma:** Let, during exploration, $v$ found a higher ranked node $u$ and the path $Q(v, u)$. If $v$'s nearest node of higher rank is $u'$, then $w(Q) \leq 4d(v, u')$. 
Making Connection

- Select the nearest node if more than one node of higher rank is found.
- Let \( u \) found higher ranked node \( v \) through the Path \( Q = <u, ..., x, y, ..., v> \)
- \( u \) sends a connect message though this path to \( v \)
- All the edges in this path are added to NNT
- Any intermediate node, say \( x \),
  - If not already connected, it uses \( (x, y) \) as the connecting edge and stops exploration
  - If it is already connected, removes the previous connecting edge from NNT
- All nodes in this path upgrade their rank to \( r(v) \)
Making Connection

- If in between exploration and connection, any node in path $Q$ gets a higher rank than $r(v)$, connection ends at that node.

- Let $u$ found path $Q$ to $v$

- Then before $u$ sends its connect message, $p$ sends a connect message to $q$

- Let $r(q) > r(p) > r(v)$

- New rank of $x$ is $r(q)$ which is larger than $r(v)$

- $u$’s connection ends at $x$. $x$ does not forward
Other Applications of NNT Scheme

- Message-efficient algorithms for finding low-weight $k$-connected spanning subgraphs in complete networks. (*Theoretical Computer Science, 2007*)


- Efficient dynamic algorithms (*IEEE TPDS 2009*).

- Energy-efficient and low-interference topology control algorithms in unreliable ad hoc wireless networks. (*IEEE INFOCOM 2009*).
Distributed Algorithms For Wireless Networks

- Traditional distributed computing theory assumes point-to-point network communication model.  
  *Complexity measures*: messages, time.

- Wireless uses radio communication.  
  Interference phenomenon, link scheduling.  
  *Other Complexity measures*: power/energy/lifetime.
Radio Broadcast Model

- **Local broadcasting**: A node can broadcast a message that can be (potentially) received by any node within its vicinity (a disk of appropriate radius centered at the node).

- Each node can communicate directly only with nodes within its transmission radius.

- The transmission from a node $u$ to its neighbor $v$ is successful, provided no other neighbor $w$ of $v$ transmits at the same time.
Energy Complexity

- Energy complexity is a measure of the energy needed by the distributed algorithm.

- Various factors affect energy complexity
  - Time needed.
  - Number of messages exchanged.
  - Radiation energy needed to transmit a message through a certain distance --- typically assumed proportional to some power of the distance.
  - Energy overheads of the hardware (startup energy, receiver energy etc.)
  - ....
Energy Complexity

Choi, Khan, Kumar, and Pandurangan. *IEEE Journal on Selected Areas in Communications*, 2009.

Energy complexity is a measure of the energy needed by the distributed algorithm.

\[ E = \sum_{i=1}^{M} r_i^\alpha \]

where \( r_i \) is the transmission distance for message \( i \) and \( M \) is the number of messages exchanged by the nodes to run the algorithm/protocol.

Energy requirements in a wireless communication paradigm:

To transmit a signal over a distance \( r \), the required radiation energy is proportional to \( r^\alpha \). (typically \( \alpha = 2 \).)
Energy-Efficient MST Construction


- Given a random geometric graph: $n$ points uniformly distributed in a unit area. Two nodes are connected if they are within a distance $r$ of each other. Assume that $r = \Theta(\sqrt{\log n/n})$.

  Find a tree $T$ spanning $N$, that minimizes the cost $Q_\alpha(T) = \sum_{(u,v) \in T} d^\alpha(u,v)$.

  where $d(u,v)$ is the Euclidean distance between $u$ and $v$.

- The above tree minimizes energy consumption in data aggregation.
# Energy Complexity of Algorithms


<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Energy Complexity</th>
<th>MST Quality</th>
</tr>
</thead>
<tbody>
<tr>
<td>GHS</td>
<td>$\Omega(\log^2 n)$</td>
<td>optimal</td>
</tr>
<tr>
<td>NNT</td>
<td>$O(\log n)$ on average</td>
<td>$O(\log n)$-approximation</td>
</tr>
<tr>
<td>EOPT</td>
<td>$O(\log n)$ on average</td>
<td>optimal</td>
</tr>
<tr>
<td>co-NNT</td>
<td>$O(1)$ on average</td>
<td>$O(1)$-approximation</td>
</tr>
</tbody>
</table>

(nodes know coordinates)
Simulation Results

Energy complexity

MST Quality
Experiments on Real Data

<table>
<thead>
<tr>
<th></th>
<th>Snapshot 1</th>
<th></th>
<th>Snapshot 2</th>
<th></th>
<th>Snapshot 3</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$Q_1$</td>
<td>$Q_2$</td>
<td>Work</td>
<td>Msg</td>
<td>$Q_1$</td>
<td>$Q_2$</td>
</tr>
<tr>
<td>Co-NNT</td>
<td>38.72</td>
<td>6.77</td>
<td>90.54</td>
<td>4832</td>
<td>39.39</td>
<td>8.18</td>
</tr>
<tr>
<td>Rnd-NNT</td>
<td>50.75</td>
<td>14.13</td>
<td>131.42</td>
<td>5241</td>
<td>52.97</td>
<td>20.12</td>
</tr>
<tr>
<td>GHS-Yao</td>
<td>33.16</td>
<td>3.73</td>
<td>1271.11</td>
<td>20592</td>
<td>33.52</td>
<td>3.82</td>
</tr>
</tbody>
</table>
Road Map

- Distributed Approximation Algorithm for MST
  - Nearest Neighbor Tree (NNT) Scheme

- Wireless Networks: Energy-Efficient Distributed MST Algorithms

- A Uniform Approach to Distributed Approximation
  - Shortest Paths, Steiner Forest, Routing Cost Tree …
  - Leader Election
  - Probabilistic tree embedding
A Uniform Approach to Distributed Approximation


- A uniform approach to design efficient distributed approximation algorithms
  - randomized approach.
  - based on a probabilistic tree embedding.
- Expected $O(\log n)$-approximate distributed algorithms for
  - the shortest paths problem.
  - the generalized Steiner forest problem.
  - the minimum cost routing tree problem.
  - …
- The time complexities are within a polylogarithmic factor of the optimum.
Our Approach

- One approach to approximation is:
  - embed the given metric space into a tree metric
  - Solve the problem in the tree (significantly easier).

- Tree embedding was never used before in a distributed setting.

- We use a randomized tree embedding due to Fakcharoenphol, Rao, and Talwar (FRT embedding).

- Other approaches are based on Linear Programming:
  - Primal dual method (Grandoni et al, PODC 05)
  - Randomized rounding (Kuhn et al, SODA 06)

- In FRT embedding, the expected stretch of any edge is $O(\log n)$, leading to $O(\log n)$-approximation algorithms.
Probabilistic Tree Embedding

Let \((V,d)\) be an arbitrary metric, where \(V\) is a set of \(n\) vertices and \(d(u,v)\) gives the distance between any two points \(u\) and \(v\).

A metric \((V,f)\) is said to dominate \((V,d)\) if for all \(u,v\) in \(V\): \(f(u,v) \geq d(u,v)\).

Interested in tree metrics that dominate a given metric. (A tree metric is a metric arising from shortest path distance on a tree containing the given vertices).

Let \(S\) be a family of metrics over \(V\); let \(D\) be a distribution over \(S\):

\((S,D)\ \alpha\text{-probabilistically approximates} \ \) a metric \((V,d)\), if every metric in \(S\) dominates \(d\) and for every pair of vertices \(u, v\) in \(V\):

\[E_f \in (S, D)[f(u, v)] \leq \alpha d(u, v).\]

**Theorem** [FRT]: There is an polynomial-time algorithm that outputs a distribution of tree metrics that \(O(\log n)\)-probabilistically approximates any given metric.
LE-List of Node $v$

Nodes sorted based on their distance from $v_0$

<table>
<thead>
<tr>
<th>Node</th>
<th>Rank</th>
<th>Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_0$</td>
<td>(4)</td>
<td>0.0</td>
</tr>
<tr>
<td>$v_1$</td>
<td>(7)</td>
<td>1.0</td>
</tr>
<tr>
<td>$v_2$</td>
<td>(6)</td>
<td>2.0</td>
</tr>
<tr>
<td>$v_3$</td>
<td>(10)</td>
<td>4.5</td>
</tr>
<tr>
<td>$v_4$</td>
<td>(2)</td>
<td>4.5</td>
</tr>
<tr>
<td>$v_5$</td>
<td>(8)</td>
<td>5.0</td>
</tr>
<tr>
<td>$v_6$</td>
<td>(5)</td>
<td>5.0</td>
</tr>
<tr>
<td>$v_7$</td>
<td>(1)</td>
<td>6.0</td>
</tr>
<tr>
<td>$v_8$</td>
<td>(9)</td>
<td>7.0</td>
</tr>
<tr>
<td>$v_9$</td>
<td>(3)</td>
<td>11.0</td>
</tr>
</tbody>
</table>

LE-list of $v_0$, $L(v_0) = \{ \langle v_0, 0 \rangle, \langle v_4, 4.5 \rangle, \langle v_7, 6.0 \rangle \}$
Least Element (LE)-Lists

- Given distinct ranks of the nodes, every node $v$ maintains a LE-list which stores the smallest ranked node within every distance $d$.
- Distributed algorithm for computing LE-lists is at the heart of our approach.
- When ranks are chosen randomly, LE-lists give an efficient way to compute an FRT embedding.
- Using the distributed representation of the FRT embedding, solve a distributed problem, giving an $O((\log n))$-approximation algorithm.
FRT Tree Embedding

**β-list**

- The leader chooses a random $\beta \in [0.5, 1]$
- Broadcast $\beta$ using the BFS tree
- From LE-list, each node constructs
  
  $\beta$-list: $(u_0, u_1, u_2, \ldots)$

  where $u_i$ is the least element in $2^i \beta$-neighborhood
FRT Tree Embedding

- The $\beta$-lists defines a hierarchical clustering and the FRT tree
- $u_i$ is called level-$i$ cluster center for $\nu$
FRT Tree Construction via LE-lists

Root: cluster containing all nodes

Level-i cluster is decomposed into level-\((i-1)\) clusters

(based on level-\((i-1)\) cluster center)

leaf nodes (level-0) are singleton clusters

corresponds to the nodes in the original graph \(G\)

\[
E[d_{\text{FRT}}(v_i, v_j)] \leq O(\log n) \cdot d_G(v_i, v_j) \quad \text{and} \quad d_{\text{FRT}}(v_i, v_j) \geq d_G(v_i, v_j)
\]
Improved Leader Election Algorithm

Leader election: Elect a unique node as a leader; all other nodes should know this leader.
One of the fundamental problems in distributed computing.

**Theorem:** There is a leader election algorithm that is both time optimal and (almost) message optimal:

- $O(D)$ time (deterministically)
- $O(|E| \cdot \min\{D, \log n\})$ messages (with high probability).

The best known time-optimal algorithm (Peleg, 1990):

- $O(D)$ time
- $O(|E| \cdot D)$ messages

Lower bounds:

- $\Omega(D)$ is a lower bound on the time needed.
- $\Omega(|E| + n \log n)$ is a lower bound on the number of messages required.
Shortest Paths

**Problem:** Find shortest paths (i.e., construct routing tables) between all-pairs of nodes.

Classical (exact) algorithm: Distributed Bellman-Ford (used in the Internet).

Our algorithm: $O(\log n)$ approximation:

$O(n^2 D \log n)$ time, $O(n | E | \log n + n^2 \log n)$ messages.

**Exact Algorithm:**

$O(n^2 D)$ time, $O(n^2 | E |)$ messages.

Our algorithm has a **significantly better message complexity** than Bellman-Ford, while the time complexities are almost equal.
Relevant Publications


