Distributed Algorithms on a Congested Clique

Christoph Lenzen
The LOCAL Model

1. compute
2. send
3. receive
The LOCAL Model

1. compute
2. send
3. receive
The LOCAL Model

1. compute
2. send
3. receive
The LOCAL Model

1. compute
2. send
3. receive
The LOCAL Model

1. compute
2. send
3. receive
The LOCAL Model

1. compute
2. send
3. receive
LOCAL + restricted bandwidth = CONGEST !

synchr. rounds:
1. compute
2. send
3. receive

message size: $O(\log n)$ bits

round complexity?

...content can differ between neighbors!
What happens here?
Disclaimer

Practical relevance of this model is questionable!

Algorithms for overlay networks?
Subroutines for small cliques in larger networks?

So why should we care?!?
what lower bound graphs look like:

what "real" networks look like:
History: MST Lower Bound

Input: weighted graph
Output: spanning tree
Goal: minimize weight of tree

≈ √n x √n

Peleg and Rubinovich
SIAM J. on Comp.'00
History: MST Lower Bound

Input: weighted graph
Output: spanning tree
Goal: minimize weight of tree

- Alice gets bit string $b$ as input

Peleg and Rubinovich
SIAM J. on Comp.‘00

Diagram of a weighted graph with labeled edges and vertices labeled Alice and Bob.
History: MST Lower Bound

Input: weighted graph
Output: spanning tree
Goal: minimize weight of tree

- Alice gets bit string $b$ as input
- assign weight $2b_i$ to $i^{th}$ edge

Peleg and Rubinovich
SIAM J. on Comp.‘00
History: MST Lower Bound

Input: weighted graph
Output: spanning tree
Goal: minimize weight of tree

- Alice gets bit string $b$ as input
- assign weight $2b_i$ to $i^{th}$ edge
- compute MST

$\Rightarrow$ Bob now knows $b$!

$\Rightarrow$ Alice sent $\geq |b|$ bits to Bob

How long does this take?

Peleg and Rubinovich
SIAM J. on Comp.’00
History: MST Lower Bound

Input: weighted graph
Output: spanning tree
Goal: minimize weight of tree

<table>
<thead>
<tr>
<th>bits sent in time $T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
</tr>
</tbody>
</table>

$T \leq o(\sqrt{n})$

$\Rightarrow$ paths use tree edges to "shortcut" $\Omega(\sqrt{n})$ hops

$\approx \sqrt{n} \times \sqrt{n}$

Peleg and Rubinovich
SIAM J. on Comp.'00
History: MST Lower Bound

Input: weighted graph
Output: spanning tree
Goal: minimize weight of tree

for each path p:
- $p_i$ subpaths in tree
- $h(p_i)$ max. dist. from leaves
- $\sum_i 2^{h(p_i)} \geq \Omega(\sqrt{n})$

but $\sum_p \sum_i 2^{h(p_i)} \leq \sqrt{n}\log n$

$\Rightarrow O(\log n)$ paths, $T \geq \Omega(\sqrt{n}/\log^2 n)$

Peleg and Rubinovich
SIAM J. on Comp.’00
MST Lower Bound: Summary

- general technique
- yields lower bounds of roughly $\Omega(\sqrt{n})$
- helped finding many near-matching algorithms

<table>
<thead>
<tr>
<th>Authors</th>
<th>Conference/Journal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Das Sarma et al.</td>
<td>STOC `11</td>
</tr>
<tr>
<td>Das Sarma et al.</td>
<td>SPAA `12</td>
</tr>
<tr>
<td>L. and Peleg</td>
<td>PODC `12</td>
</tr>
<tr>
<td>Peleg et al</td>
<td>ICALP`12</td>
</tr>
<tr>
<td>Elkin</td>
<td>ACM Trans. on Alg. `05</td>
</tr>
<tr>
<td>Elkin</td>
<td>SIAM J. on Comp. `06</td>
</tr>
<tr>
<td>Elkin and Peleg</td>
<td>SIAM J. on Comp. `04</td>
</tr>
<tr>
<td>Frischknecht et al.</td>
<td>SODA `12</td>
</tr>
<tr>
<td>Khan et al.</td>
<td>Dist. Computing `12</td>
</tr>
<tr>
<td>Khan and Pandurangan</td>
<td>Dist. Computing `08</td>
</tr>
<tr>
<td>Kutten and Peleg</td>
<td>J. Algorithms `98</td>
</tr>
<tr>
<td>L. and Patt-Shamir</td>
<td>STOC `13</td>
</tr>
<tr>
<td>Holzer and Wattenhofer</td>
<td>PODC `12</td>
</tr>
</tbody>
</table>
But How About Well-Connected Graphs?

<table>
<thead>
<tr>
<th>diameter</th>
<th>upper bound</th>
<th>lower bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>O(log n)</td>
<td>O(n^{1/2} log* n)</td>
<td>Ω(n^{1/2}/log^2 n)</td>
</tr>
<tr>
<td>4</td>
<td>?</td>
<td>Ω(n^{1/3}/log n)</td>
</tr>
<tr>
<td>3</td>
<td>?</td>
<td>Ω(n^{1/4}/log n)</td>
</tr>
<tr>
<td>2</td>
<td>O(log n)</td>
<td>?</td>
</tr>
<tr>
<td>1</td>
<td>O(log log n)</td>
<td>?</td>
</tr>
</tbody>
</table>

Lotker et al. Dist. Computing’06
Lotker et al. SIAM J. on Comp.’05
But How About Well-Connected Graphs?

<table>
<thead>
<tr>
<th>diameter</th>
<th>upper bound</th>
<th>lower bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>O(log n)</td>
<td>O(n^{1/2} log* n)</td>
<td>(\Omega(n^{1/2}/\log^2 n))</td>
</tr>
<tr>
<td>4</td>
<td>?</td>
<td>(\Omega(n^{1/3}/\log n))</td>
</tr>
<tr>
<td>3</td>
<td>?</td>
<td>(\Omega(n^{1/4}/\log n))</td>
</tr>
<tr>
<td>2</td>
<td>O(log n)</td>
<td>?</td>
</tr>
<tr>
<td>1</td>
<td>O(log log n)</td>
<td>?</td>
</tr>
</tbody>
</table>

All known lower bounds are based on hardness of spreading information!
What happens here?

What happens if there is no communication bottleneck?

...multi-party communication complexity!
What We Know: MST

input: weight of adjacent edges
output: least-weight spanning tree

- \( O(\log \log n) \) rounds
- no non-trivial lower bound known

Lotker et al., Distr. Comp. ‘06
What We Know: Triangle Detection

input: adjacent edges in input graph
output: whether input contains triangle

- $O(n^{1/3}/\log n)$ rounds
- no non-trivial lower bound known

Dolev et al. DISC’12
What We Know: Metric Facility Location

input: costs for nodes & edges (metric)
output: nodes & edges s.t. selected nodes cover all
goal: minimize cost

- $O(\log \log n \log^* n)$ rounds for $O(1)$-approx.
- no non-trivial lower bound known

Berns et al., ICALP'12
What We Know: Sorting

input: n keys/node
output: indices of keys in global order

- O(1) rounds
- trivially optimal
What We Know: Routing

input: n mess./node, each node dest. of n mess.
goal: deliver all messages

- O(1) rounds
- trivially optimal

PODC‘13
Routing: Known Source/Destination Pairs

input: n messages/node (each to receive n mess.)
source/destination pairs common knowledge

“sources” “destinations”

2 rounds
Routing within Subsets (Known Pairs)

\[ \sqrt{n} \]

\[ \sqrt{n} \]

\[ \sqrt{n} \]

\[ \sqrt{n} \]

send/receive \( n \) messages within subsets
Routing within Subsets (Unknown Pairs)

Within each subset:

1. Broadcast #mess. for each destination  2 rounds
2. Compute communication pattern  local comp.
3. Move messages  2 rounds
Routing: Known Source/Destination Sets

1. Compute pattern on set level  
2. Redistribute messages within sets  
3. Move messages between sets  
4. Redistribute messages within sets  
5. Move messages between sets  
6. Deliver messages within sets

- $n^{1/2}$ supernodes  
- degree $n^{3/2}$  
- $n$ mess. between each pair  
- each pair can handle $n$ mess.
Routing: Unknown Pairs

source/destination pairs only relevant w.r.t. sets
count within sets (one node/dest.) 1 round
broadcast information to all nodes 1 round
Routing: Result

Theorem

Input:
• up to \( n \) messages at each node
• each node destination of up to \( n \) messages

Then:
• all messages can be delivered in 16 rounds
...or in Other Words:

**fully connected CONGEST** $\approx$ **bulk-synchronous (bandwidth $n \log n$)**

- In each round, each node:
  1. Computes
  2. Sends up to $n \log n$ bits
  3. Receives up to $n \log n$ bits
What Do We Want in a Lower Bound?

- caused by “lack of coordination”, not bottleneck
  \[ \text{input per node of size } O(n \log n) \]

ideally, also:

- “natural” problem
- strong bound (e.g. \( \Omega(n^c) \) for constant \( c > 0 \))
- unrestricted algorithms
Triangle Detection: an Algorithm

input: adjacent edges in input graph
output: whether input contains triangle
Triangle Detection: an Algorithm

- partition nodes into subsets of \( n^{2/3} \) nodes
- consider all \( n \) triplets of such subsets
- assign triplets 1:1 to nodes
- responsible node checks for triangle in its triplet
→ needs to learn of \( n^{4/3} \) (pre-determined) edges
→ running time \( O(n^{1/3}/\log n) \)

<table>
<thead>
<tr>
<th>subset</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
</table>

detected by node with triplet \( (3,2,4) \)
Triangle Detection: an Algorithm

“oblivious” algorithm:
- fixed message pattern
- computation only initially and in the end

**Conjecture**
running time $O(n^{1/3}/\log n)$
optimal for oblivious algorithms

...and maybe even in general?
MST and Friends

some doubly logarithmic bounds:

- MST in $O(\log \log n)$ rounds
- Metric Facility Location in $O(\log \log n \log^* n)$ rounds
- no improvement or lower bound on MST for a decade

Open Question

Is running time $O(\log \log n)$ a barrier for some problems?
Connectivity

input: adjacent edges in input graph
output: whether input graph is connected

- natural problem, even simpler than MST
- might be easier to find right approach

Open Question
Can Connectivity be decided within $O(1)$ rounds?
...on a Related Subject

There *is* a lower bound, on Set Disjointness!
(but in a different model)

→ Don‘t miss the next talk!

...*thank you for your attention!*