Distributed Algorithms

- Message Passing
- Shared Memory
Example: Maximal Independent Set (MIS)

- Given a network with $n$ nodes, nodes have unique IDs.
- Find a Maximal Independent Set (MIS)
  - a non-extendable set of pair-wise non-adjacent nodes
Example: Maximal Independent Set (MIS)

- Given a network with \( n \) nodes, nodes have unique IDs.
- Find a Maximal Independent Set (MIS)
  - a non-extendable set of pair-wise non-adjacent nodes
Example: Maximal Independent Set (MIS)

- Given a network with $n$ nodes, nodes have unique IDs.
- Find a Maximal Independent Set (MIS)
  - a non-extendable set of pair-wise non-adjacent nodes
Example: Maximal Independent Set (MIS)

- Given a network with $n$ nodes, nodes have unique IDs.
- Find a Maximal Independent Set (MIS)
  - a nonextendable set of pair-wise non-adjacent nodes
Example: Maximal Independent Set (MIS)

• Given a network with $n$ nodes, nodes have unique IDs.
• Find a Maximal Independent Set (MIS)
  – a non-extendable set of pair-wise non-adjacent nodes
Example: Maximal Independent Set (MIS)

- Given a network with $n$ nodes, nodes have unique IDs.
- Find a Maximal Independent Set (MIS) – a non-extendable set of pair-wise non-adjacent nodes

![Graph with nodes 69, 17, 11, 10, and 7]

- Traditional (sequential) computation: The simple greedy algorithm finds MIS (in linear time)
What about a Distributed Algorithm?

- Nodes are agents with unique ID’s that can communicate with neighbors by sending messages. In each synchronous round, every node can send a (different) message to each neighbor.
What about a Distributed Algorithm?

- Nodes are agents with unique ID’s that can communicate with neighbors by sending messages. In each synchronous round, every node can send a (different) message to each neighbor.

```
each round:
1. send msgs
2. rcv msgs
3. compute
```
A Simple Distributed Algorithm

- Wait until all neighbors with higher ID decided
- If no higher ID neighbor is in MIS \(\rightarrow\) join MIS

![Graph with nodes 69, 17, 11, 10, 7]

**each round:**
1. send msgs
2. rcv msgs
3. compute
A Simple Distributed Algorithm

- Wait until all neighbors with higher ID decided
- If no higher ID neighbor is in MIS $\rightarrow$ join MIS
A Simple Distributed Algorithm

- Wait until all neighbors with higher ID decided
- If no higher ID neighbor is in MIS → join MIS

What’s the problem with this distributed algorithm?

Each round:
1. send msgs
2. rcv msgs
3. compute
Example

- Wait until all neighbors with higher ID decided
- If no higher ID neighbor is in MIS $\rightarrow$ join MIS
Example

- Wait until all neighbors with higher ID decided
- If no higher ID neighbor is in MIS → **join MIS**
Example

• Wait until all neighbors with higher ID decided
• If no higher ID neighbor is in MIS → join MIS

What if we have minor changes?
Example

- Wait until all neighbors with higher ID decided
- If no higher ID neighbor is in MIS $\rightarrow$ join MIS

![Node IDs Diagram]

- What if we have minor changes?
Example

- Wait until all neighbors with higher ID decided
- If no higher ID neighbor is in MIS → join MIS

What if we have minor changes?
Example

- Wait until all neighbors with higher ID decided
- If no higher ID neighbor is in MIS → join MIS

![Graph with nodes 69, 17, 11, 10, 7, 4, 3, 1]

- What if we have minor changes?

![Updated graph with nodes 69, 17, 11, 10, 7, 4, 3, 1]
Example

- Wait until all neighbors with higher ID decided
- If no higher ID neighbor is in MIS → join MIS

What if we have minor changes?
Example

- Wait until all neighbors with higher ID decided
- If no higher ID neighbor is in MIS → join MIS

Proof by animation: In the worst case, the algorithm is slow (linear in the number of nodes). In addition, we have a terrible „butterfly effect“. 
What about a Fast Distributed Algorithm?

• Can you find a distributed algorithm that is polylogarithmic in the number of nodes $n$, for any graph?
What about a **Fast** Distributed Algorithm?

- Surprisingly, for **deterministic** distributed algorithms, this is an open problem!

- However, **randomization** helps! In each synchronous round, nodes should choose a random value. If your value is larger than the value of your neighbors, join MIS!
What about a Fast Distributed Algorithm?

• Surprisingly, for deterministic distributed algorithms, this is an open problem!

• However, randomization helps! In each synchronous round, nodes should choose a random value. If your value is larger than the value of your neighbors, join MIS!
What about a Fast Distributed Algorithm?

- Surprisingly, for deterministic distributed algorithms, this is an open problem!

- However, randomization helps! In each synchronous round, nodes should choose a random value. If your value is larger than the value of your neighbors, join MIS!
What about a Fast Distributed Algorithm?

- Surprisingly, for **deterministic** distributed algorithms, this is an **open problem**!

- However, **randomization** helps! In each synchronous round, nodes should choose a random value. If your value is larger than the value of your neighbors, join MIS!

- How many synchronous rounds does this take in expectation (or whp)?
Analysis

- Event \((u \rightarrow v)\): node \(u\) got largest random value in combined neighborhood \(N_u \cup N_v\).
- We only count edges of \(v\) as deleted.

Similarly event \((v \rightarrow u)\) deletes edges of \(u\).
- We only double-counted edges.
- Using linearity of expectation, in expectation at least half of the edges are removed in each round.
- In other words, whp it takes \(O(\log n)\) rounds to compute an MIS.
Results: MIS

General Graphs, Randomized
[Alon, Babai, and Itai, 1986]
[Israeli and Itai, 1986]
[Luby, 1986]
[Métivier et al., 2009]

Decomposition, Deterministic
[Awerbuch et al., 1989]
[Panconesi et al., 1996]

Naïve Algorithm
Local Algorithms

- Each node can exchange a message with all neighbors, for $t$ communication rounds, and must then decide.
- Or: Given a graph, each node must determine its decision as a function of the information available within radius $t$ of the node.
- Or: Change can only affect nodes up to distance $t$.
- Or: ...
Local Algorithms

Sublinear Algorithms
Locality is Everywhere!

- Self-Assembling Robots
- Applications e.g. Multicore
- Self-Stabilization
- Local Algorithms
- Dynamics
- Sublinear Algorithms
Locality is Everywhere!

- Self-Assembling Robots
- Applications e.g. Multicore
- Local Algorithms
- Dynamics
- Sublinear Algorithms

Diagram showing overlapping circles with the mentioned topics and a hammer illustration.
What about an **Even Faster** Distributed Algorithm?

- Since the 1980s, nobody was able to improve this simple algorithm.

- What about lower bounds?

- There is an interesting lower bound, essentially using a Ramsey theory argument, that proves that an MIS needs at least $\Omega(\log^* n)$ time.
  - $\log^*$ is the so-called iterated logarithm – how often you need to take the logarithm until you end up with a value smaller than 1.
  - This lower bound already works on simple networks such as the linked list.
Coloring Lower Bound on Oriented Ring

- Build graph $G_t$, where nodes are possible views of nodes for distributed algorithms of time $t$. Connect views that could be neighbors in ring.
- Here is for instance of $G_1$:

- Chromatic number of $G_t$ is exactly minimum possible colors in time $t$. 
Coloring Lower Bound on Oriented Ring

- Build graph $G_t$, where nodes are possible views of nodes for distributed algorithms of time $t$. Connect views that could be neighbors in ring.
- Here is for instance of $G_1$:

```
1 -- 2 -- 3
```

```
2 -- 3 -- 6
```

```
3 -- 6 -- 7
```

```
3 -- 6 -- 9
```

- Chromatic number of $G_t$ is exactly minimum possible colors in time $t$. 
Results: MIS

1 \quad \log^* n \quad \log n \quad n^\epsilon \quad n

Linked List [Linial, 1992]

General Graphs, Randomized
[Alon, Babai, and Itai, 1986]
[Israeli and Itai, 1986]
[Luby, 1986]
[Métivier et al., 2009]

Decomposition, Determ.
[Awerbuch et al., 1989]
[Panconesi et al., 1996]

Naïve Algo
Results: MIS

Linked List, Deterministic [Cole and Vishkin, 1986]

General Graphs, Randomized
[Alon, Babai, and Itai, 1986]
[Israeli and Itai, 1986]
[Luby, 1986]
[Métivier et al., 2009]

Decomposition, Determ.
[Awerbuch et al., 1989]
[Panconesi et al., 1996]

Naïve Algo

Linked List [Linial, 1992]
Results: MIS

1 log* n log n n^ε n

|IS(N_2)| ∈ O(1)

Growth-Bounded Graphs [Schneider et al., 2008]

General Graphs, Randomized
[Alon, Babai, and Itai, 1986]
[Israeli and Itai, 1986]
[Luby, 1986]
[Métivier et al., 2009]

Decomposition, Determ.
[Awerbuch et al., 1989]
[Panconesi et al., 1996]

Linked List, Deterministic
[Cole and Vishkin, 1986]

Naïve Algo

Linked List [Linial, 1992]
Results: MIS

1. Linked List, Deterministic [Cole and Vishkin, 1986]
2. Growth-Bounded Graphs [Schneider et al., 2008]
3. General Graphs, Randomized [Alon, Babai, and Itai, 1986], [Israeli and Itai, 1986], [Luby, 1986], [Métivier et al., 2009]
4. Other problems e.g., [Kuhn et al., 2006]
5. e.g., coloring, CDS, matching, max-min LPs, facility location
6. e.g., covering/packing LPs with only local constraints: constant approximation in time $O(\log n)$ or $O(\log^2 \Delta)$
7. Decomposition, Deterministic [Awerbuch et al., 1989], [Panconesi et al., 1996]
8. Naïve Algo

Time Complexity:
- $\log^* n$
- $\log n$
- $n^\epsilon$
- $n$
Results: MIS

1 \log^* n \quad \log n \quad n^\epsilon \quad n

|IS(N_2)| \in O(1)

Growth-Bounded Graphs [Schneider et al., 2008]

Linked List, Deterministic [Cole and Vishkin, 1986]

General Graphs, Randomized [Alon, Babai, and Itai, 1986]
[Israeli and Itai, 1986]
[Luby, 1986]
[Métivier et al., 2009]

Other problems e.g., [Kuhn et al., 2006]

E.g., coloring, CDS, matching, max-min LPs, facility location

E.g., covering/packing LPs with only local constraints: constant approximation in time \(O(\log n)\) or \(O(\log^2 \Delta)\)

Decomposition, Determin. [Awerbuch et al., 1989]
[Panconesi et al., 1996]

Naïve Algo

Linked List [Linial, 1992]

General Graphs [Kuhn et al., 2004, 2006]
Example: Minimum Vertex Cover (MVC)

• Given a network with $n$ nodes, nodes have unique IDs.
• Find a Minimum Vertex Cover (MVC) – a minimum set of nodes such that all edges are adjacent to node in MVC
Example: Minimum Vertex Cover (MVC)

- Given a network with $n$ nodes, nodes have unique IDs.
- Find a Minimum Vertex Cover (MVC)
  - a minimum set of nodes such that all edges are adjacent to node in MVC
Example: Minimum Vertex Cover (MVC)

- Given a network with $n$ nodes, nodes have unique IDs.
- Find a Minimum Vertex Cover (MVC)
  - a minimum set of nodes such that all edges are adjacent to node in MVC
Differences between MIS and MVC

- Central (non-local) algorithms: MIS is trivial, whereas MVC is NP-hard.
- Instead: Find an MVC that is “close” to minimum (approximation).
- Trade-off between time complexity and approximation ratio.

MVC: Various simple (non-distributed) 2-approximations exist!
- What about distributed algorithms?!!
Finding the MVC (by Distributed Algorithm)

- Given the following bipartite graph with $|S_0| = \delta |S_1|$
- The MVC is just all the nodes in $S_1$
- Distributed Algorithm...
Finding the MVC (by Distributed Algorithm)

- Given the following bipartite graph with $|S_0| = \delta |S_1|$
- The MVC is just all the nodes in $S_1$
- Distributed Algorithm...
Finding the MVC (by Distributed Algorithm)

- Given the following bipartite graph with $|S_0| = \delta |S_1|$
- The MVC is just all the nodes in $S_1$
- Distributed Algorithm...
\( N_2(\text{node in } S_0) \)

\( N_2(\text{node in } S_1) \)
Graph is “symmetric”, yet highly non-regular!
Lower Bound: The Argument

- The example graph is for $t = 3$.
- All edges are in fact special bipartite graphs with large enough girth.

- If you use the graph of recursion level $t$, then a distributed algorithm cannot find a good MVC approximation in time $t$. 
Choose degrees $\delta_i$ such that $\delta_{i+1}/\delta_i = 2^i \delta$.

We have $|S_0| > \delta/2 \cdot |L_1|$, with $|L_1|$ nodes on level 1.
• Choose degrees $\delta_i$ such that $\delta_{i+1}/\delta_i = 2^i \delta$.

• We have $|S_0| > \delta/2 \ |L_1|$, with $|L_1|$ nodes on level 1

• By induction we have a $(1 - \Theta(1/\delta))$ fraction of the nodes is in $S_0$.

• Now $\delta, n, \Delta$ are depending on the recursion level $t$. 
Lower Bound: The Math

- Choose degrees $\delta_i$ such that $\delta_{i+1}/\delta_i = 2^i \delta$.
- We have $|S_0| > \delta/2 |L_1|$, with $|L_1|$ nodes on level 1.

By induction we have a $(1 - \Theta(1/\delta))$ fraction of the nodes is in $S_0$.
- Now $\delta, n, \Delta$ are depending on the recursion level $t$.
Lower Bound: Results

- We can show that for $\varepsilon > 0$, in $t$ time, the approximation ratio is at least

$$\Omega \left( n^{\frac{1/4 - \varepsilon}{t^2}} \right) \text{ and } \Omega \left( \Delta^{\frac{1 - \varepsilon}{t + 1}} \right)$$

- Constant approximation needs at least $\Omega(\log \Delta)$ and $\Omega(\sqrt{\log n})$ time.
- Polylog approximation $\Omega(\log \Delta / \log \log \Delta)$ and $\Omega(\sqrt{\log n / \log \log n})$. 
Lower Bound: Results

- We can show that for $\varepsilon > 0$, in $t$ time, the approximation ratio is at least

  $$\Omega\left(n^{\frac{1}{4}-\varepsilon}\right) \text{ and } \Omega\left(\Delta^{\frac{1-\varepsilon}{t+1}}\right)$$

  [tight for MVC]

- Constant approximation needs at least $\Omega(\log \Delta)$ and $\Omega(\sqrt{\log n})$ time.

- Polylog approximation $\Omega(\log \Delta / \log \log \Delta)$ and $\Omega(\sqrt{\log n / \log \log n})$. 
Many “local looking” problems need non-trivial $t$, in other words, the bounds $\Omega(\log \Delta)$ and $\Omega(\sqrt{\log n})$ hold for a variety of classic problems.
Many “local looking” problems need non-trivial $t$, in other words, the bounds $\Omega(\log \Delta)$ and $\Omega(\sqrt{\log n})$ hold for a variety of classic problems.
Results: MIS

\[ |IS(N_2)| \in O(1) \]

Growth-Bounded Graphs [Schneider et al., 2008]

Linked List, Deterministic [Cole and Vishkin, 1986]

|IS(N_2)| \in O(1)

\[ \log^* n \]

Linked List [Linial, 1992]

General Graphs, Randomized
- [Alon, Babai, and Itai, 1986]
- [Israeli and Itai, 1986]
- [Luby, 1986]
- [Métivier et al., 2009]

Other problems e.g., [Kuhn et al., 2006]

\[ \sqrt{\log n} \ldots \log n \]

General Graphs [Kuhn et al., 2004, 2006]

\[ n^\varepsilon \]

Decomposition, Determ. [Awerbuch et al., 1989]
[Panconesi et al., 1996]

Naïve Algo

\[ n \]

e.g., coloring, CDS, matching, max-min LPs, facility location

e.g., covering/packing LPs with only local constraints: constant approximation in time \( O(\log n) \) or \( O(\log^2 \Delta) \)
Summary

1. \( \log^*n \)

\[ \sqrt{\log n} \ldots \log n \]

Diameter

Growth-Bounded Graphs (various problems)

- E.g., dominating set approximation in planar graphs

Approximations of dominating set, vertex cover, etc.

MIS, maximal matching, etc.

Covering and packing LPs

- E.g., dominating set approximation in planar graphs
Thank You!

Questions & Comments?

Thanks to my co-authors
Fabian Kuhn
Thomas Moscibroda
Johannes Schneider

www.disco.ethz.ch
Open Problems

- Close the gap between $\sqrt{\log n}$ and $\log n$ (for randomized algorithms)!
- Find a fast deterministic MIS algorithm (or strong det. lower bound)!
- Where are the boundaries between constant, log*, log, and diameter?
- What about algorithms that cannot even exchange messages?
- Can the lower bound graph be used in the context of sublinear algorithms?