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 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

![Diagram of a network with nodes 69, 10, 17, 11, 7]
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

• 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:**
  
  **every node:**
  
  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 → join MIS
A Simple Distributed Algorithm

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

**Diagram:**

- 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

What’s the problem with this distributed algorithm?

each round:
every node:
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 → 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 $\rightarrow$ 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

• 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

```
69 17 11 10 7 4 3 1
```

- What if we have minor changes?

```
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

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

- What if we have minor changes?

![Graph with nodes labeled 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

![Diagram](image)

- What if we have minor changes?

![Diagram with butterfly](image)

- 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, Determ.: [Awerbuch et al., 1989], [Panconesi et al., 1996]
- Naïve Algo
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: ...
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.
Results: MIS

1 \log^* n \log n \ n^\varepsilon \ 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]
[Métilvier et al., 2009]

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

Naïve Algo
Results: MIS

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

1 \rightarrow \log^* n \rightarrow \log n \rightarrow n^\epsilon \rightarrow n

\|IS(N_2)\| \in O(1)
Results: MIS

- **|IS(N₂)| ∈ O(1)**
  - Growth-Bounded Graphs [Schneider et al., 2008]
  - Linked List, Deterministic [Cole and Vishkin, 1986]

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

- **log n**
  - e.g., coloring, CDS, matching, max-min LPs, facility location
  - Other problems e.g., [Kuhn et al., 2006]

- **n^ε**
  - e.g., covering/packing LPs with only local constraints: constant approximation in time $O(\log n)$ or $O(\log^2 \Delta)$
  - Decomposition, Determ. [Awerbuch et al., 1989]
  - [Panconesi et al., 1996]

- **n**
  - Naïve Algo

---

Linked List [Linial, 1992]
Results: MIS

- \(|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]
- Decomposition, Deterministic [Awerbuch et al., 1989] [Panconesi et al., 1996]
- Naïve Algo

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

- Linked List [Linial, 1992]
- General Graphs [Kuhn et al., 2004, 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 \(O(\log^2 \Delta)\)
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?!
Results: MIS

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

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]
[Métilier et al., 2009]

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

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

Naïve Algo

\[ 1 \quad \log^* n \quad \sqrt{\log n} \ldots \log n \quad n^\epsilon \quad n \]

Linked List
[Linial, 1992]

General Graphs
[Kuhn et al., 2004, 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 \( O(\log^2 \Delta) \)
Ad Hoc & Sensor Networks?
Unit Disk Graph (UDG)

- Classic computational geometry model, special case of disk graphs

- All nodes are points in the plane, two nodes are connected iff (if and only if) their distance is at most 1, that is \( \{u,v\} \in E \iff |u,v| \leq 1 \)

+ Very simple, allows for strong analysis
- Not realistic: “If you gave me $100 for each paper written with the unit disk assumption, I still could not buy a radio that is unit disk!”
- Particularly bad in obstructed environments (walls, hills, etc.)

- Natural extension: 3D UDG
Unit Ball Graph (UBG)

• ∃ metric $(V,d)$ with constant doubling dimension.

• Metric: Each edge has a distance $d$, with
  1. $d(u,v) \geq 0$ (non-negativity)
  2. $d(u,v) = 0$ iff $u = v$ (identity of indiscernibles)
  3. $d(u,v) = d(v,u)$ (symmetry)
  4. $d(u,w) \leq d(u,v) + d(v,w)$ (triangle inequality)

• Doubling dimension: $\log(#\text{balls of radius } r/2 \text{ to cover ball of radius } r)$
  – Constant: you only need a constant number of balls of half the radius

• Connectivity graph is same as UDG:
  such that: $d(u,v) \leq 1 : (u,v) \in E$
  $d(u,v) > 1 : (u,v) \in E/$
Wireless Interference Models: Protocol Model

• For lower layer protocols, a model needs to be specific about interference. A **simplest interference model** is an extension of the UDG. In the protocol model, a transmission by a node in at most distance 1 is received iff there is no conflicting transmission by a node in distance at most $R$, with $R \geq 1$, sometimes just $R = 2$.

  + Easy to explain
  - Inherits all major drawbacks from the UDG model
  - Does not easily allow for designing distributed algorithms/protocols
  - Lots of interfering transmissions just outside the interference radius $R$ do not sum up

• Can be extended with the same extensions as UDG, e.g. QUDG
Hop Interference (HI)

- An often-used interference model is hop-interference. Here a UDG is given. Two nodes can communicate directly iff they are adjacent, and if there is no concurrent sender in the \( k \)-hop neighborhood of the receiver (in the UDG). Sometimes \( k = 2 \).

- Special case of the protocol model, inheriting all its drawbacks
  + Simple
  + Allows for distributed algorithms
  - A node can be close but not produce any interference (see picture)
- Can be extended with the same extensions as UDG, e.g. QUDG
Physical (SINR) Model

- We look at the signal-to-noise-plus-interference (SINR) ratio.
- Message arrives if SINR is larger than $\beta$ at receiver

$$\frac{P_u}{d(u,v)^\alpha} \geq \beta$$

- Mind that the SINR model is far from perfect as well.

Power level of sender $u$

Path-loss exponent, $\alpha = 2, ..., 6$

Distance between transmitter $w$ and receiver $v$

Minimum signal-to-interference ratio, depending on quality of hardware, etc.
Wireless Media Access?

- **Radio Network Model**
  - Slotted time (unslotted time only costs factor 2)
  - In each slot, each node can either transmit, receive, or sleep
  - Nodes receive transmissions depending on connectivity & interference models
  - With or without collision detection
  - With or without synchronous start
  - With or without ...

- **Beeper Model**
  - Nodes can just beep
  - If at least one neighbor beeps, a node will receive that (no interference)
  - Yes, this can be done in reality, e.g. slotted programming