Awake Complexity of Distributed Minimum Spanning Tree

John Augustine[§], **William K. Moses Jr.**[★], Gopal Pandurangan[♦]

[§]Indian Institute of Technology Madras, [★]Durham University, [♦]University of Houston

April 26th, 2024

Outline

Problem Statement

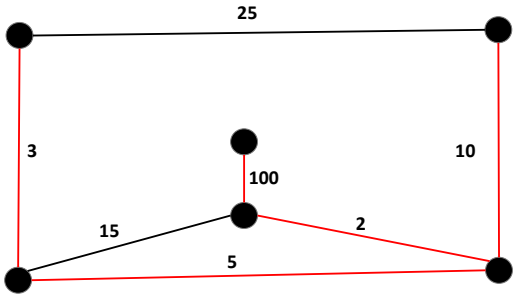
Final Goal

Find the **minimum spanning tree (MST)** of an input graph in the **CONGEST model** in an **awake efficient/energy efficient** manner.

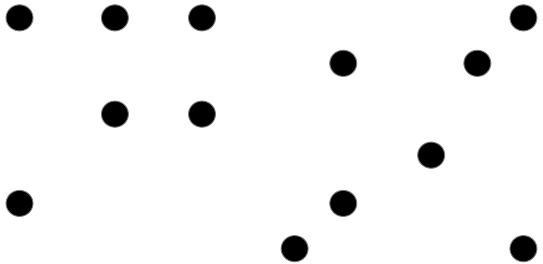
Minimum Spanning Tree (MST)

Minimum Spanning Tree

For a graph $G(V, E)$ where each edge $e \in E$ has associated weight, the minimum spanning tree of G is a connected subgraph of G that connects all nodes of V, has no cycles, and has the minimum sum of edge weights amongst all possible subgraphs satisfying the previous two conditions.

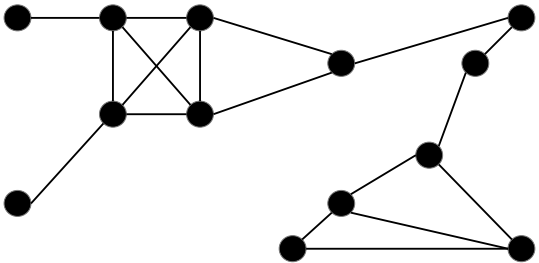


Distributed Computing Model



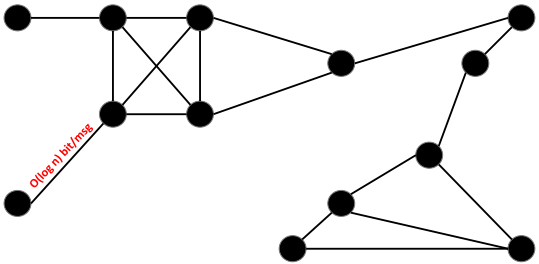
- n nodes with unique IDs, value of n known to all nodes

Distributed Computing Model



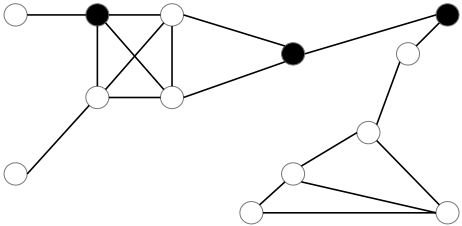
- n nodes with unique IDs, value of n known to all nodes
- Arbitrary graph

Distributed Computing Model



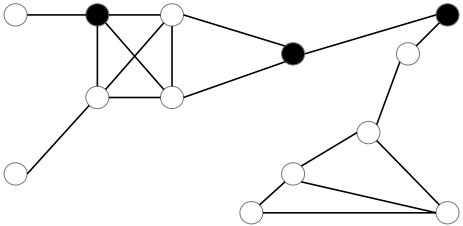
- n nodes with unique IDs, value of n known to all nodes
- Arbitrary graph
- CONGEST model: $O(\log n)$ bits per message over an edge

Awake/Energy Efficiency Defined & Measured



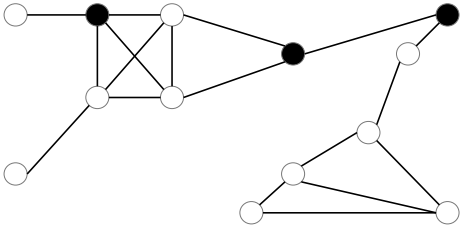
- Synchronous system

Awake/Energy Efficiency Defined & Measured



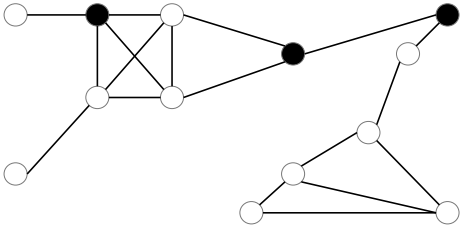
- Synchronous system
- In each round, each node's state $\in$ {awake, asleep}

Awake/Energy Efficiency Defined & Measured



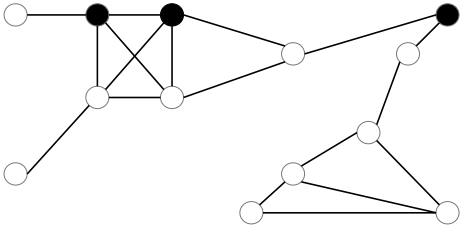
- Synchronous system
- In each round, each node's state $\in$ {awake, asleep}
- Algorithm designer controls when nodes are awake, asleep. Nodes know current round #

Awake/Energy Efficiency Defined & Measured



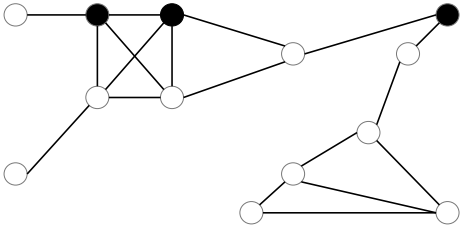
- Synchronous system
- In each round, each node's state $\in$ {awake, asleep}
- Algorithm designer controls when nodes are awake, asleep. Nodes know current round #
- Message exchange on an edge (u, v) only possible when u,v awake in same round

Awake/Energy Efficiency Defined & Measured



- Synchronous system
- In each round, each node's state $\in$ {awake, asleep}
- Algorithm designer controls when nodes are awake, asleep. Nodes know current round #
- Message exchange on an edge (u, v) only possible when u,v awake in same round

Awake/Energy Efficiency Defined & Measured



- Synchronous system
- In each round, each node's state $\in$ {awake, asleep}
- Algorithm designer controls when nodes are awake, asleep. Nodes know current round #
- Message exchange on an edge (u, v) only possible when u,v awake in same round
- **Metric 1:** awake time; **Metric 2:** running time/round complexity

Outline

Why Care About Awake Efficient MST in CONGEST

1. MST long studied in CONGEST. Approx. 40 years now.
   Now established that $\Theta(D + \sqrt{n})$ is tight (up to log factors) bound for running time.
   D - diameter of graph, n - # of nodes of graph

Why Care About Awake Efficient MST in CONGEST

1. MST long studied in CONGEST. Approx. 40 years now.
   Now established that $\Theta(D + \sqrt{n})$ is tight (up to log factors) bound for running time.
   D - diameter of graph, n - # of nodes of graph
2. But do all nodes really need to participate for that much time?

Why Care About Awake Efficient MST in CONGEST

1. MST long studied in CONGEST. Approx. 40 years now.
   Now established that $\Theta(D + \sqrt{n})$ is tight (up to log factors) bound for running time.
   D - diameter of graph, n - # of nodes of graph
2. But do all nodes really need to participate for that much time?
3. Work on other problems like MIS showing awake time asymptotically less than run time.
   [Chatterjee et al., PODC 2020],[Barenboim & Maimon, DISC 2021],
   [Maimon, Manuscript 2021],[Ghaffari & Portmann, SPAA 2022],
   [Dufoulon et al., PODC 2023],

Why Care About Awake Efficient MST in CONGEST

1. MST long studied in CONGEST. Approx. 40 years now.
   Now established that $\Theta(D + \sqrt{n})$ is tight (up to log factors) bound for running time.
   D - diameter of graph, n - # of nodes of graph
2. But do all nodes really need to participate for that much time?
3. Work on other problems like MIS showing awake time asymptotically less than run time.
   [Chatterjee et al., PODC 2020],[Barenboim & Maimon, DISC 2021],
   [Maimon, Manuscript 2021],[Ghaffari & Portmann, SPAA 2022],
   [Dufoulon et al., PODC 2023],
4. But what about MST?

Why Care About Awake Efficient MST in CONGEST

1. MST long studied in CONGEST. Approx. 40 years now.
   Now established that $\Theta(D + \sqrt{n})$ is tight (up to log factors) bound for running time.
   D - diameter of graph, n - # of nodes of graph
2. But do all nodes really need to participate for that much time?
3. Work on other problems like MIS showing awake time asymptotically less than run time.
   [Chatterjee et al., PODC 2020],[Barenboim & Maimon, DISC 2021],
   [Maimon, Manuscript 2021],[Ghaffari & Portmann, SPAA 2022],
   [Dufoulon et al., PODC 2023],
4. But what about MST? Nothing so far.

Why Care About Awake Efficient MST in CONGEST

1. MST long studied in CONGEST. Approx. 40 years now.
   Now established that $\Theta(D + \sqrt{n})$ is tight (up to log factors) bound for running time.
   D - diameter of graph, n - # of nodes of graph
2. But do all nodes really need to participate for that much time?
3. Work on other problems like MIS showing awake time asymptotically less than run time.
   [Chatterjee et al., PODC 2020],[Barenboim & Maimon, DISC 2021],
   [Maimon, Manuscript 2021],[Ghaffari & Portmann, SPAA 2022],
   [Dufoulon et al., PODC 2023],
4. But what about MST? Nothing so far.
5. **Closest:** [Barenboim & Maimon, DISC 2021] show $O(\log n)$ awake time needed for ST.
   But technique can't be extended to MST.

Why Care About Awake Efficient MST in CONGEST

1. MST long studied in CONGEST. Approx. 40 years now.
   Now established that $\Theta(D + \sqrt{n})$ is tight (up to log factors) bound for running time.
   D - diameter of graph, n - # of nodes of graph
2. But do all nodes really need to participate for that much time?
3. Work on other problems like MIS showing awake time asymptotically less than run time.
   [Chatterjee et al., PODC 2020],[Barenboim & Maimon, DISC 2021],
   [Maimon, Manuscript 2021],[Ghaffari & Portmann, SPAA 2022],
   [Dufoulon et al., PODC 2023],
4. But what about MST? Nothing so far.
5. **Closest:** [Barenboim & Maimon, DISC 2021] show $O(\log n)$ awake time needed for ST.
   But technique can't be extended to MST.
6. Note that lots of work on energy complexity in RADIO model as well.

Our Contributions

Table: Summary of our Results.

Algorithm	Type	Awake Time (AT)	Run Time (RT)	AT Lower Bound	AT $\times$ RT Lower Bound
*Randomized-MST	Randomized	$O(\log n)$	$O(n \log n)$	$\Omega(\log n)$	$\tilde{\Omega}(n)$†
Deterministic-MST	Deterministic	$O(\log n)$	$O(n \log^5 n)$	$\Omega(\log n)$	$\tilde{\Omega}(n)$†
Modified Deterministic-MST	Deterministic	$O(\log n \log^* n)$	$O(n \log n \log^* n)$	$\Omega(\log n)$	$\tilde{\Omega}(n)$†
*♣Trade-Off-MST	Randomized	$\tilde{O}(n/2^k)$	$\tilde{O}(D + 2^k + n/2^k)$	-	-

*The algorithm outputs an MST with high probability.
♣This algorithm takes integer k as an input parameter.
†Holds for (some) graphs with diameter $\tilde{\Omega}(\sqrt{n})$.
Our lower bounds also apply to Monte Carlo randomized algorithms with constant success probability.
n is the number of nodes in the network and N is an upper bound on the largest ID of a node.

Why Care About This Talk

1. Technique for lower bound w.r.t. awake complexity.
2. Techniques for designing algs with low awake complexity.
3. In particular, one concept (LDT) used in to other work [Dufoulon et al., PODC 2023]. Potential applications to other problems.

Outline

Product (of Awake and Run Time Complexities) Lower Bound

Theorem Statement (Simplified)

For graphs with diameter $D = \omega(\sqrt{n}\log^3 n)$, any randomized alg. $\mathcal{A}$ that solves MST that is correct with probability $1 - \epsilon$ for any small fixed $\epsilon > 0$ has the following property:
If the running time of $\mathcal{A}$ is some $o(c)$, for some c between $\tilde{\Omega}(\sqrt{n})$ to $\tilde{O}(n)$, then awake time is at least $\tilde{\Omega}(n/c)$.

Example Implication

For any algorithm $\mathcal{A}$ that solves MST and runs in time $o(n^{3/4})$, its awake time must be at least $\tilde{\Omega}(n^{1/4})$.

Proof Sketch

Problems Considered

SD - Set disjointness (in classical setting)
DSD - Distributed set disjointness (in sleeping model)
CSS - Connected spanning subgraph (in sleeping model)

Proof Sketch

Problems Considered

SD - Set disjointness (in classical setting)
DSD - Distributed set disjointness (in sleeping model)
CSS - Connected spanning subgraph (in sleeping model)

1. SD is reducible to DSD.

Proof Sketch

Problems Considered

SD - Set disjointness (in classical setting)
DSD - Distributed set disjointness (in sleeping model)
CSS - Connected spanning subgraph (in sleeping model)

1. SD is reducible to DSD.
2. If solve DSD with low awake complexity $\implies$ break fundamental SD lower bound. So lower bound on awake complexity.

Proof Sketch

Problems Considered

SD - Set disjointness (in classical setting)
DSD - Distributed set disjointness (in sleeping model)
CSS - Connected spanning subgraph (in sleeping model)

1. SD is reducible to DSD.
2. If solve DSD with low awake complexity $\implies$ break fundamental SD lower bound.
   So lower bound on awake complexity.
3. DSD is reducible to MST via CSS.
   Lower bound on awake compelxity for DSD then applies to MST.

Proof Sketch

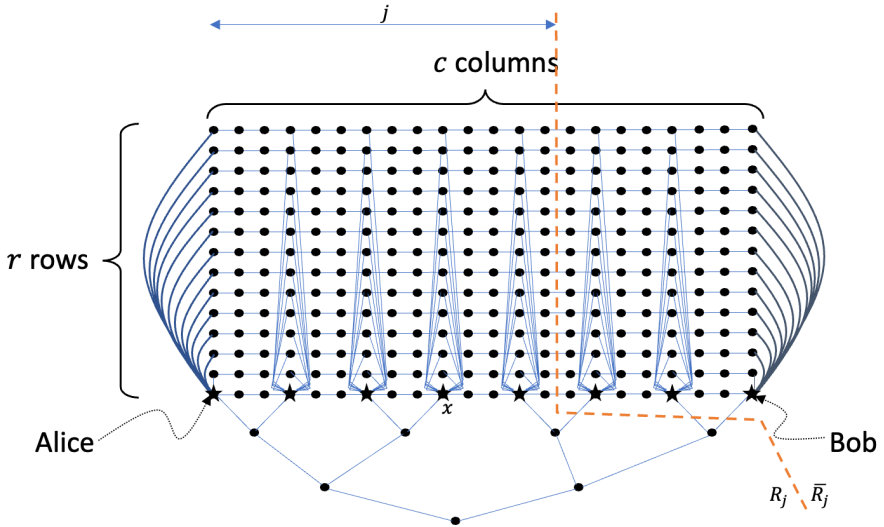
Problems Considered

SD - Set disjointness (in classical setting)
DSD - Distributed set disjointness (in sleeping model)
CSS - Connected spanning subgraph (in sleeping model)

1. SD is reducible to DSD.
2. If solve DSD with low awake complexity $\implies$ break fundamental SD lower bound.
   So lower bound on awake complexity.
3. DSD is reducible to MST via CSS.
   Lower bound on awake compelxity for DSD then applies to MST.

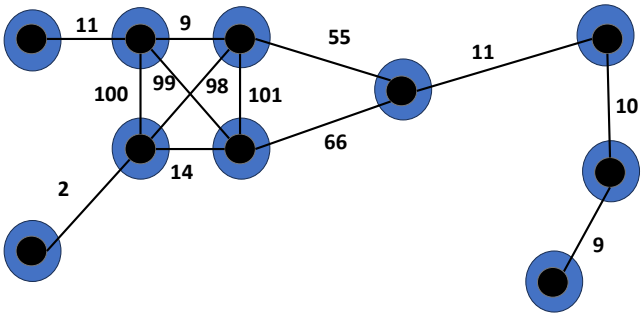
We focus on steps 1 & 2 in this talk.

Steps 1 & 2



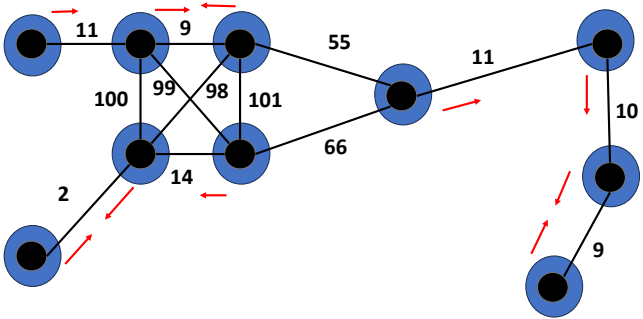
Outline

Intuition Behind Algorithms



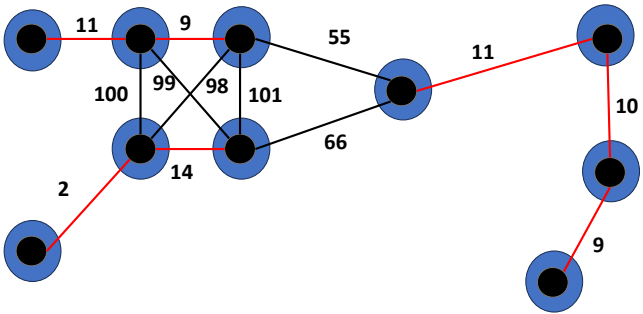
- Algs are variants of classical GHS alg, modified to minimize awake complexity.
- Clusters (initially each node) merge using MOEs into larger clusters.
- Each phase, constant fraction of clusters merge. Totally $O(\log n)$ phases.

Intuition Behind Algorithms



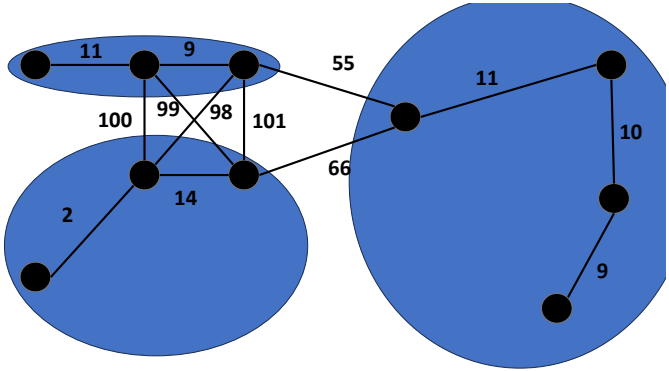
- Algs are variants of classical GHS alg, modified to minimize awake complexity.
- Clusters (initially each node) merge using MOEs into larger clusters.
- Each phase, constant fraction of clusters merge. Totally $O(\log n)$ phases.

Intuition Behind Algorithms



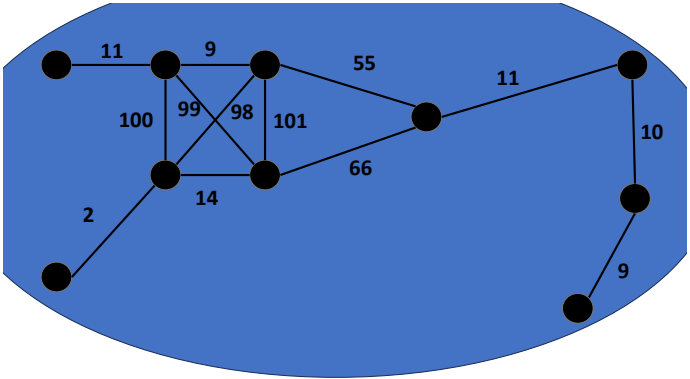
- Algs are variants of classical GHS alg, modified to minimize awake complexity.
- Clusters (initially each node) merge using MOEs into larger clusters.
- Each phase, constant fraction of clusters merge. Totally $O(\log n)$ phases.

Intuition Behind Algorithms



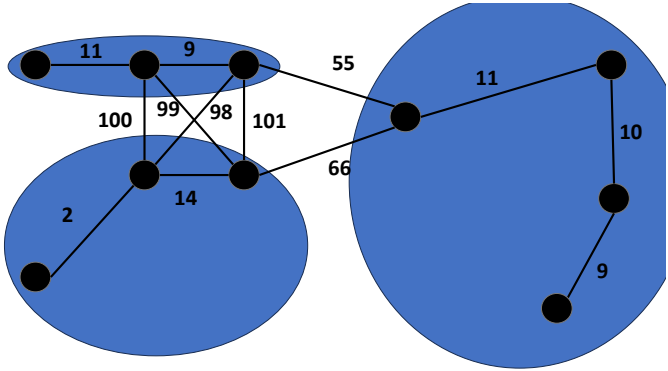
- Algs are variants of classical GHS alg, modified to minimize awake complexity.
- Clusters (initially each node) merge using MOEs into larger clusters.
- Each phase, constant fraction of clusters merge. Totally $O(\log n)$ phases.

Intuition Behind Algorithms



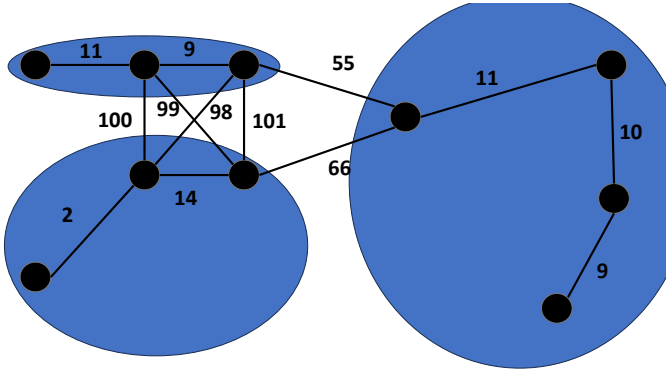
- Algs are variants of classical GHS alg, modified to minimize awake complexity.
- Clusters (initially each node) merge using MOEs into larger clusters.
- Each phase, constant fraction of clusters merge. Totally $O(\log n)$ phases.

How To Minimize Awake Complexity?



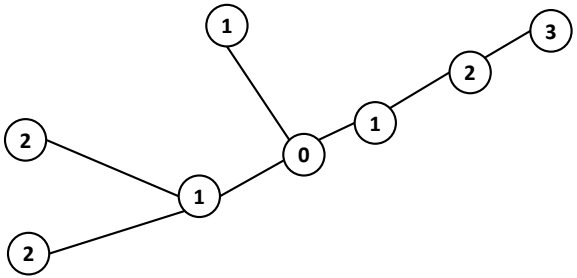
- **Notice:** Not all nodes in a cluster need to be awake at all times in a phase.

How To Minimize Awake Complexity?



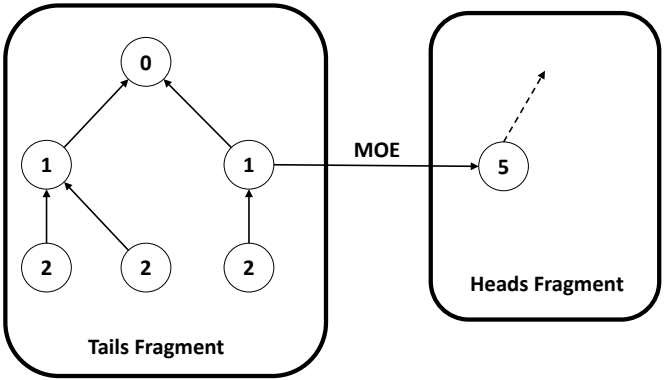
- **Notice:** Not all nodes in a cluster need to be awake at all times in a phase.
- Just need a central computation & communication with adjacent clusters.

How To Minimize Awake Complexity?



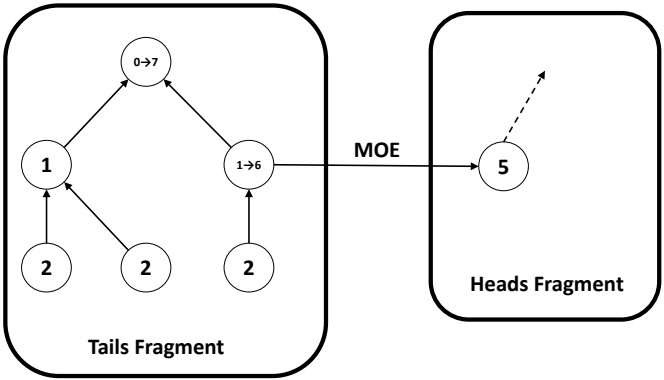
- **Notice:** Not all nodes in a cluster need to be awake at all times in a phase.
- Just need a central computation & communication with adjacent clusters.
- **Idea:** Labeled Distance Tree (LDT) paired with transmission schedules of size $2n + 1$.

Randomized Algorithm



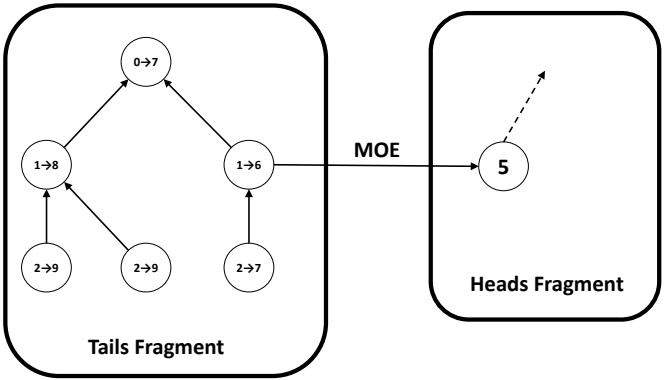
- So in each phase of GHS, merge LDTs (fragments/clusters). **Goal:** only one LDT left.
- In each phase: (i) Each LDT finds MOE (ii) Merge LDTs along MOEs.

Randomized Algorithm



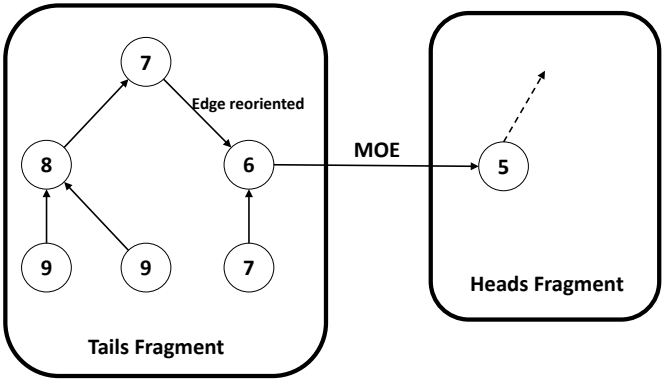
- So in each phase of GHS, merge LDTs (fragments/clusters). **Goal:** only one LDT left.
- In each phase: (i) Each LDT finds MOE (ii) Merge LDTs along MOEs.

Randomized Algorithm



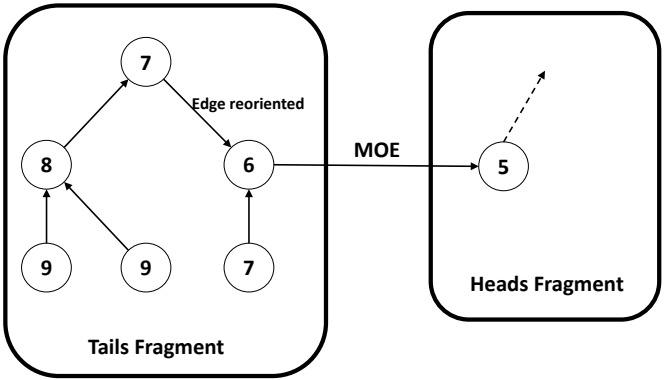
- So in each phase of GHS, merge LDTs (fragments/clusters). **Goal:** only one LDT left.
- In each phase: (i) Each LDT finds MOE (ii) Merge LDTs along MOEs.

Randomized Algorithm



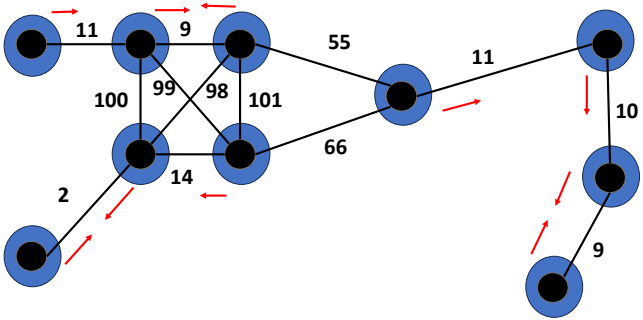
- So in each phase of GHS, merge LDTs (fragments/clusters). **Goal:** only one LDT left.
- In each phase: (i) Each LDT finds MOE (ii) Merge LDTs along MOEs.

Randomized Algorithm



- So in each phase of GHS, merge LDTs (fragments/clusters). **Goal:** only one LDT left.
- In each phase: (i) Each LDT finds MOE (ii) Merge LDTs along MOEs.
- Where could a problem occur?

Randomized Algorithm



- So in each phase of GHS, merge LDTs (fragments/clusters). **Goal:** only one LDT left.
- In each phase: (i) Each LDT finds MOE (ii) Merge LDTs along MOEs.
- Where could a problem occur?
- **Hint:** we want $O(1)$ awake time per node per phase.

Randomized Algorithm

- **Problem:** "Long lines" of LDTs trying to merge.

Randomized Algorithm

- **Problem:** "Long lines" of LDTs trying to merge.
- **Solution:** Each LDT flips a coin. "Valid" MOEs are from Tails to Heads. Ignore others.

Randomized Algorithm

- **Problem:** "Long lines" of LDTs trying to merge.
- **Solution:** Each LDT flips a coin. "Valid" MOEs are from Tails to Heads. Ignore others.
- Through analysis, we show that a constant fraction of LDTs are merged in each phase.

Randomized Algorithm

- **Problem:** "Long lines" of LDTs trying to merge.
- **Solution:** Each LDT flips a coin. "Valid" MOEs are from Tails to Heads. Ignore others.
- Through analysis, we show that a constant fraction of LDTs are merged in each phase.
- So $O(\log n)$ phases.

Randomized Algorithm

- **Problem:** "Long lines" of LDTs trying to merge.
- **Solution:** Each LDT flips a coin. "Valid" MOEs are from Tails to Heads. Ignore others.
- Through analysis, we show that a constant fraction of LDTs are merged in each phase.
- So $O(\log n)$ phases.
- Each phase, $O(1)$ awake complexity per node and $O(n)$ run time.

Randomized Algorithm

- **Problem:** "Long lines" of LDTs trying to merge.
- **Solution:** Each LDT flips a coin. "Valid" MOEs are from Tails to Heads. Ignore others.
- Through analysis, we show that a constant fraction of LDTs are merged in each phase.
- So $O(\log n)$ phases.
- Each phase, $O(1)$ awake complexity per node and $O(n)$ run time.
- **Result:** MST constructed w.h.p. in $O(\log n)$ awake complexity and $O(n \log n)$ round complexity.

Deterministic Algorithm

- **Problem:** How to ensure no “long lines” without randomness?

Deterministic Algorithm

- **Problem:** How to ensure no "long lines" without randomness?
- **Solution:** Use maximal matching (coloring with a small palette + matching).

Deterministic Algorithm

- **Problem:** How to ensure no "long lines" without randomness?
- **Solution:** Use maximal matching (coloring with a small palette + matching).
- **Issue 1:** How to guarantee small palette enough?

Deterministic Algorithm

- **Problem:** How to ensure no "long lines" without randomness?
- **Solution:** Use maximal matching (coloring with a small palette + matching).
- **Issue 1:** How to guarantee small palette enough?
- **Solution:** Restrict "valid" MOEs to be only at most 4 (3 incoming + 1 outgoing). Color palette 5 enough now.

Deterministic Algorithm

- **Problem:** How to ensure no "long lines" without randomness?
- **Solution:** Use maximal matching (coloring with a small palette + matching).
- **Issue 1:** How to guarantee small palette enough?
- **Solution:** Restrict "valid" MOEs to be only at most 4 (3 incoming + 1 outgoing). Color palette 5 enough now.
- **Issue 2:** How to argue constant fraction of fragments merged in each phase?

Deterministic Algorithm

- **Problem:** How to ensure no "long lines" without randomness?
- **Solution:** Use maximal matching (coloring with a small palette + matching).
- **Issue 1:** How to guarantee small palette enough?
- **Solution:** Restrict "valid" MOEs to be only at most 4 (3 incoming + 1 outgoing). Color palette 5 enough now.
- **Issue 2:** How to argue constant fraction of fragments merged in each phase?
- **Solution:** Combinatorial argument:
  Take highest priority color.
  In any 5 coloring of subgraph, there are a constant fraction of nodes colored with it.
  Those nodes merge in that phase.

Deterministic Algorithm

- **Problem:** How to ensure no "long lines" without randomness?
- **Solution:** Use maximal matching (coloring with a small palette + matching).
- **Issue 1:** How to guarantee small palette enough?
- **Solution:** Restrict "valid" MOEs to be only at most 4 (3 incoming + 1 outgoing). Color palette 5 enough now.
- **Issue 2:** How to argue constant fraction of fragments merged in each phase?
- **Solution:** Combinatorial argument:
  Take highest priority color.
  In any 5 coloring of subgraph, there are a constant fraction of nodes colored with it.
  Those nodes merge in that phase.
- **Result 1:** MST in $O(\log n)$ awake time and $O(n \log^5 n)$ run time.
- **Result 2:** MST in $O(\log n \log^* n)$ awake time and $O(n \log n \log^* n)$ run time.

Outline

Conclusions & Future Work

Conclusions

1. In this talk, we looked at product lower bound and algorithms to find MST.
2. In paper, you can find the trade-off family of algorithms & unconditional awake time lower bound.

Future Work

1. Awake complexity of other problems.
2. Looking at awake complexity in presence of faults and/or dynamism of graph.

References


Leonid Barenboim and Tzalik Maimon.
Deterministic logarithmic completeness in the distributed sleeping model.
In Seth Gilbert, editor, *35th International Symposium on Distributed Computing, DISC 2021, October 4-8, 2021, Freiburg, Germany (Virtual Conference)*, volume 209 of *LIPIcs*, pages 10:1–10:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.
doi:10.4230/LIPIcs.DISC.2021.10.


Soumyottam Chatterjee, Robert Gmyr, and Gopal Pandurangan.
Sleeping is efficient: MIS in $O(1)$-rounds node-averaged awake complexity.
In Yuval Emek and Christian Cachin, editors, *PODC '20: ACM Symposium on Principles of Distributed Computing*, pages 99–108, 2020.


Fabien Dufoulon, William K. Moses Jr., and Gopal Pandurangan.
Distributed MIS in o(log log n) awake complexity.
In *Proceedings of the Symposium on Principles of Distributed Computing (PODC)*, 2023.


Mohsen Ghaffari and Julian Portmann.
Average awake complexity of MIS and matching.
In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 45–55, 2022.


Tzalik Maimon.
Sleeping model: Local and dynamic algorithms, 2021.

The End