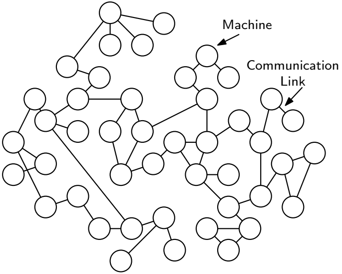
# Almost Global Problems in the LOCAL Model

Alkida Balliu, Sebastian Brandt, **Dennis Olivetti**, and Jukka Suomela

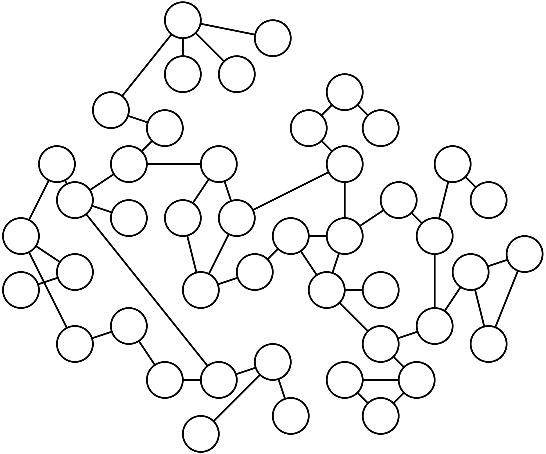
*Aalto University, Finland*

# LOCAL Model

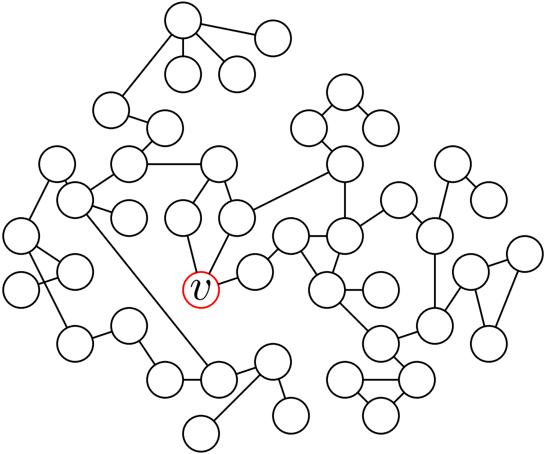



- Distributed network
- Nodes represent machines
- Edges represent communication links
- Synchronous
- Messages of arbitrary size, arbitrary computational power
- Nodes have distinct IDs
- Nodes know the size of the graph
- Complexity measure: number of rounds required to solve a task

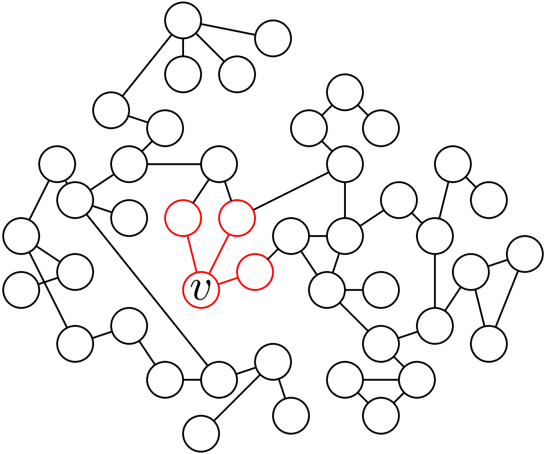
# LOCAL Model: easier description



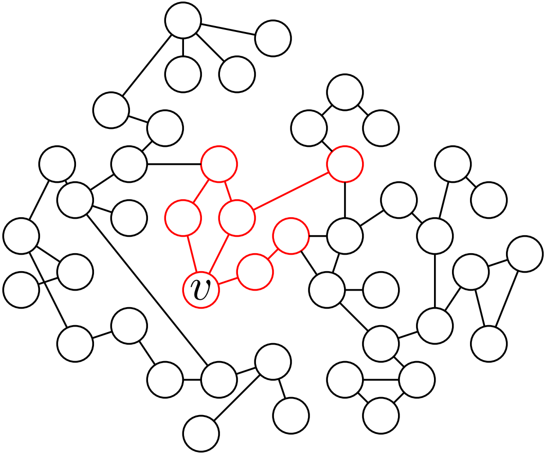
# LOCAL Model: easier description



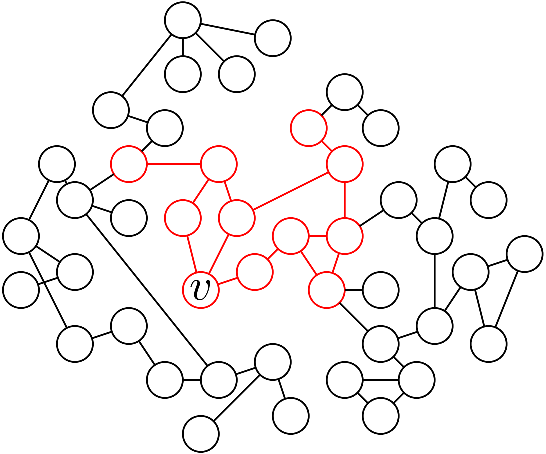
# LOCAL Model: easier description



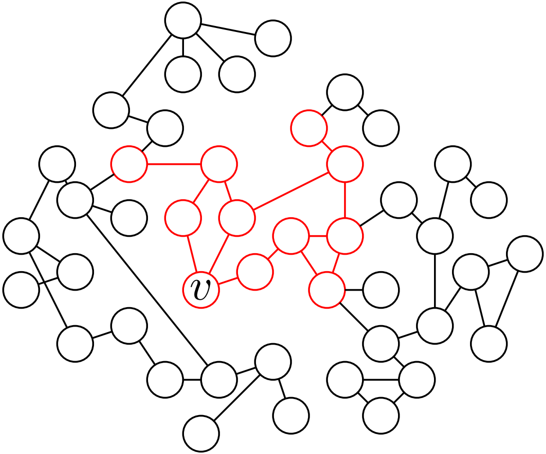
# LOCAL Model: easier description



# LOCAL Model: easier description



# LOCAL Model: easier description



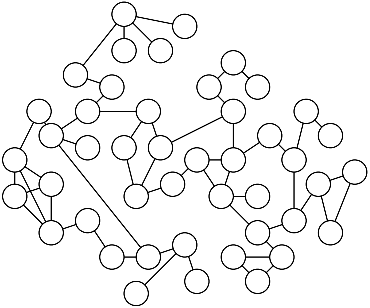

A $t$-round algorithm for the LOCAL model is a mapping from $t$-radius balls to valid outputs.

# Locally Checkable Labellings

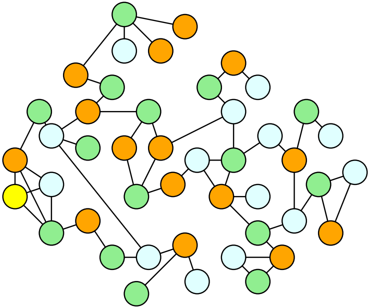
LCL Problems:

- Introduced by Naor and Stockmeyer in 1995
- Constant-size input labels
- Constant-size output labels
- The maximum degree is constant
- Validity of the output is locally checkable

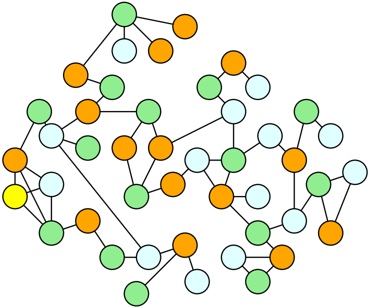
# Locally Checkable Labellings (Example)



# Locally Checkable Labellings (Example)



# Locally Checkable Labellings (Example)



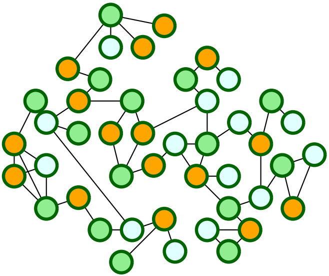
$\Delta + 1$ vertex colouring:

- The input is empty
- The output is in $\{1, \ldots, \Delta + 1\}$
- Nodes can check in 1 round if the colouring is valid

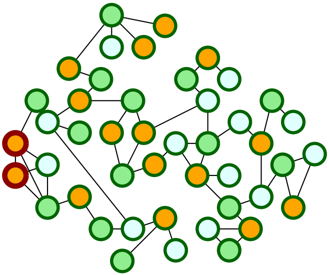
# Local checkability

There must be a constant time distributed algorithm that is able to check the solution, such that:

- If the output is globally correct, all nodes accept.



- If there is an error, at least a node rejects.



# Locally Checkable Labellings (Motivation)

- Study the complexity of problems where the solution can be checked efficiently (like NP!)
- By restricting to constant degree graphs, we study problems related to distance, while ignoring the influence of other factors.
- It is a simple class that contains many well known problems.
- Lower bounds in this model apply to less powerful models.

# Question

What are the possible time complexities for LCL problems?

# LCL on Cycles and Paths

- There are only three possible time complexities:
  - $\Theta(1)$: trivial problems
  - $\Theta(\log^* n)$: local problems (symmetry breaking)
  - $\Theta(n)$: global problems

# LCL on Cycles and Paths

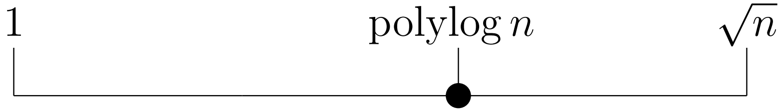
- There are only three possible time complexities:
  - $\Theta(1)$: trivial problems
  - $\Theta(\log^* n)$: local problems (symmetry breaking)
  - $\Theta(n)$: global problems
- Automatic speedups:
  - Any $o(\log^* n)$-rounds algorithm can be converted to a $O(1)$-rounds algorithm [Naor and Stockmeyer, 1995]
  - Any $o(n)$-rounds algorithm can be converted to a $O(\log^* n)$-rounds algorithm [Chang, Kopelowitz and Pettie, 2016]

# LCL on Cycles and Paths

- There are only three possible time complexities:
  - $\Theta(1)$: trivial problems
  - $\Theta(\log^* n)$: local problems (symmetry breaking)
  - $\Theta(n)$: global problems
- Automatic speedups:
  - Any $o(\log^* n)$-rounds algorithm can be converted to a $O(1)$-rounds algorithm [Naor and Stockmeyer, 1995]
  - Any $o(n)$-rounds algorithm can be converted to a $O(\log^* n)$-rounds algorithm [Chang, Kopelowitz and Pettie, 2016]
- On cycles with no input, given an LCL description, we can *decide* its time complexity. [Naor and Stockmeyer, 1995] [Brandt et al, 2017]

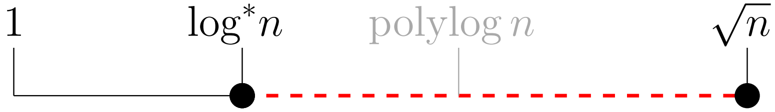
# Motivating Example (Grids)

- $\Delta$–colouring in general graphs can be done in $O(\text{polylog}\, n)$ rounds [Panconesi, Srinivasan 1995]
- 4–colouring in 2–dimensional balanced grids can be done in $O(\text{polylog}\, n)$ rounds



# Motivating Example (Grids)

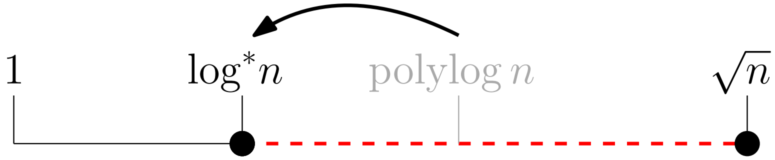
- $\Delta$–colouring in general graphs can be done in $O(\text{polylog } n)$ rounds [Panconesi, Srinivasan 1995]
- 4–colouring in 2–dimensional balanced grids can be done in $O(\text{polylog } n)$ rounds




[Brandt et al. 2017]

# Motivating Example (Grids)

- $\Delta$–colouring in general graphs can be done in $O(\text{polylog}\, n)$ rounds [Panconesi, Srinivasan 1995]
- 4–colouring in 2–dimensional balanced grids can be done in $O(\text{polylog}\, n)$ rounds



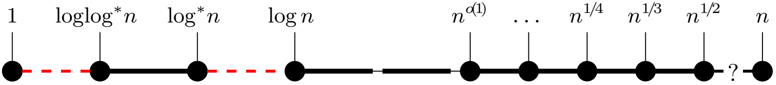
# LCL on Trees

[Chang and Pettie, 2017]:

- Any $n^{o(1)}$-rounds algorithm can be converted to a $O(\log n)$-rounds algorithm
- There are problems of complexity $\Theta(n^{1/k})$

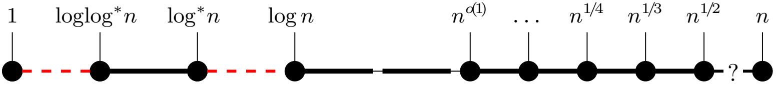
# LCL on General Graphs

- There are problems with complexity $\Theta(\log n)$ [Brandt et al, 2016] [Chang, Kopelowitz and Pettie, 2016] [Ghaffari and Su, 2017]



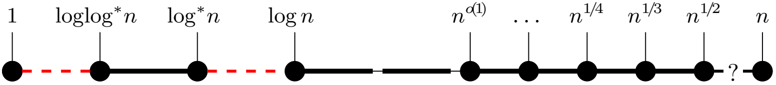
# LCL on General Graphs

- There are problems with complexity $\Theta(\log n)$ [Brandt et al, 2016] [Chang, Kopelowitz and Pettie, 2016] [Ghaffari and Su, 2017]
- Any $o(\log \log^* n)$-rounds algorithm can be converted to a $O(1)$-rounds algorithm using the same techniques of [Naor and Stockmeyer, 1995]



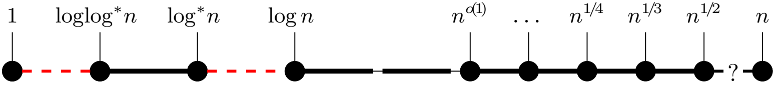
# LCL on General Graphs

- There are problems with complexity $\Theta(\log n)$ [Brandt et al, 2016] [Chang, Kopelowitz and Pettie, 2016] [Ghaffari and Su, 2017]
- Any $o(\log \log^* n)$-rounds algorithm can be converted to a $O(1)$-rounds algorithm using the same techniques of [Naor and Stockmeyer, 1995]
- Any $o(\log n)$-rounds algorithm can be converted to a $O(\log^* n)$-rounds algorithm [Chang, Kopelowitz and Pettie, 2016]



# LCL on General Graphs

- There are problems with complexity $\Theta(\log n)$ [Brandt et al, 2016] [Chang, Kopelowitz and Pettie, 2016] [Ghaffari and Su, 2017]
- Any $o(\log \log^* n)$-rounds algorithm can be converted to a $O(1)$-rounds algorithm using the same techniques of [Naor and Stockmeyer, 1995]
- Any $o(\log n)$-rounds algorithm can be converted to a $O(\log^* n)$-rounds algorithm [Chang, Kopelowitz and Pettie, 2016]
- There are problems with complexities in (almost) all the other regions [Balliu, Hirvonen, Korhonen, Lempiäinen, O., Suomela, 2018]

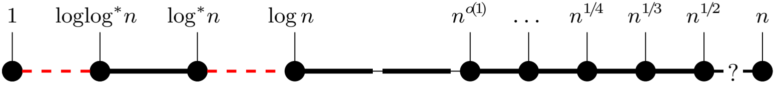


# LCL on General Graphs

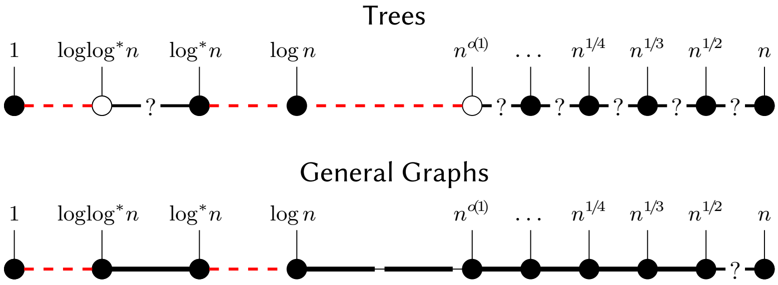
- There are problems with complexity $\Theta(\log n)$ [Brandt et al, 2016] [Chang, Kopelowitz and Pettie, 2016] [Ghaffari and Su, 2017]
- Any $o(\log \log^* n)$-rounds algorithm can be converted to a $O(1)$-rounds algorithm using the same techniques of [Naor and Stockmeyer, 1995]
- Any $o(\log n)$-rounds algorithm can be converted to a $O(\log^* n)$-rounds algorithm [Chang, Kopelowitz and Pettie, 2016]
- There are problems with complexities in (almost) all the other regions [Balliu, Hirvonen, Korhonen, Lempiäinen, O., Suomela, 2018]
- Many problems require $\Omega(\log n)$ and $O(\text{poly} \log n)$

# LCL on General Graphs

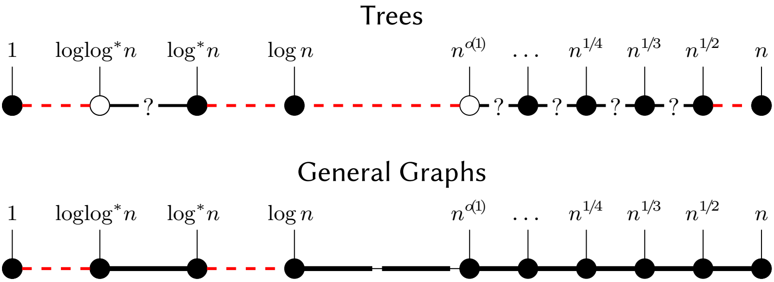
- There are problems with complexity $\Theta(\log n)$ [Brandt et al, 2016] [Chang, Kopelowitz and Pettie, 2016] [Ghaffari and Su, 2017]
- Any $o(\log \log^* n)$-rounds algorithm can be converted to a $O(1)$-rounds algorithm using the same techniques of [Naor and Stockmeyer, 1995]
- Any $o(\log n)$-rounds algorithm can be converted to a $O(\log^* n)$-rounds algorithm [Chang, Kopelowitz and Pettie, 2016]
- There are problems with complexities in (almost) all the other regions [Balliu, Hirvonen, Korhonen, Lempiäinen, O., Suomela, 2018]
- Many problems require $\Omega(\log n)$ and $O(\text{poly} \log n)$
- Different scenario with randomized algorithms



# Trees vs General Graphs



# Our Results




In general graphs, we can construct LCL problems with infinitely many complexities between $\omega(\sqrt{n})$ and $o(n)$.

In trees, problems with complexities between $\omega(\sqrt{n})$ and $o(n)$ do not exist.

# Proof idea

In general graphs, we can construct LCL problems with infinitely many complexities between $\omega(\sqrt{n})$ and $o(n)$

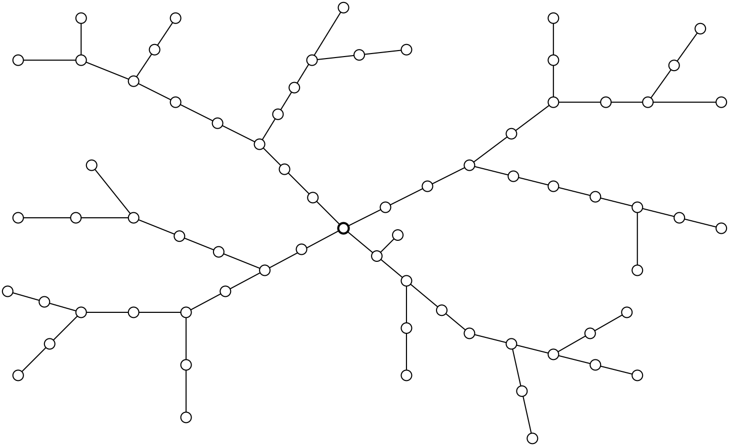
Idea:

- Encode linear bounded automata as LCLs on grids
- Obtain complexities that depend on the execution time of the LBA

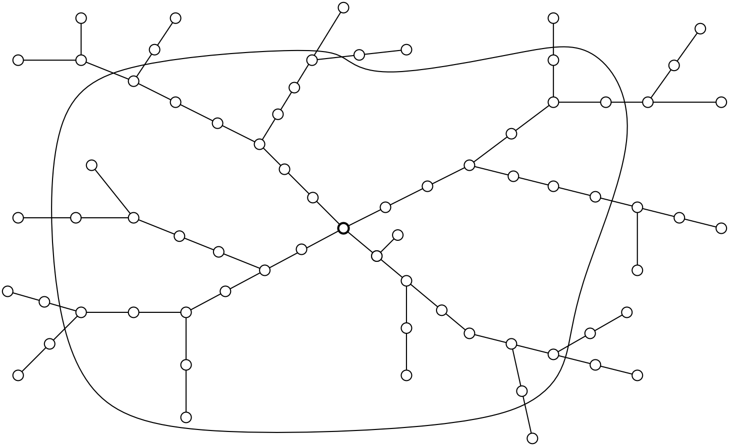
# Proof idea

In trees, problems with complexities between $\omega(\sqrt{n})$ and $o(n)$ do not exist.

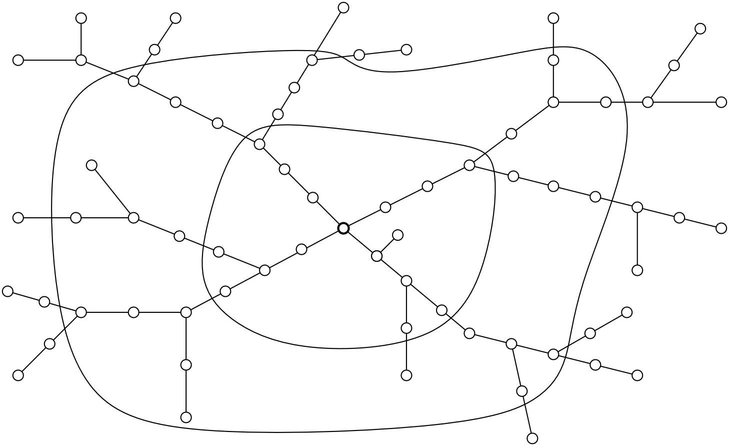
# Proof idea



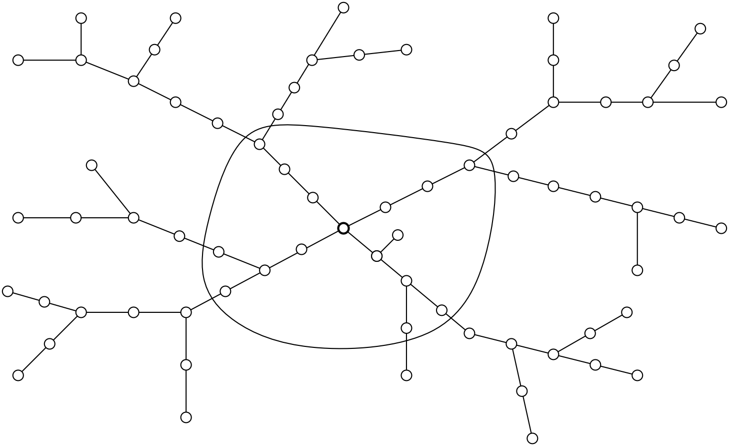
# Proof idea



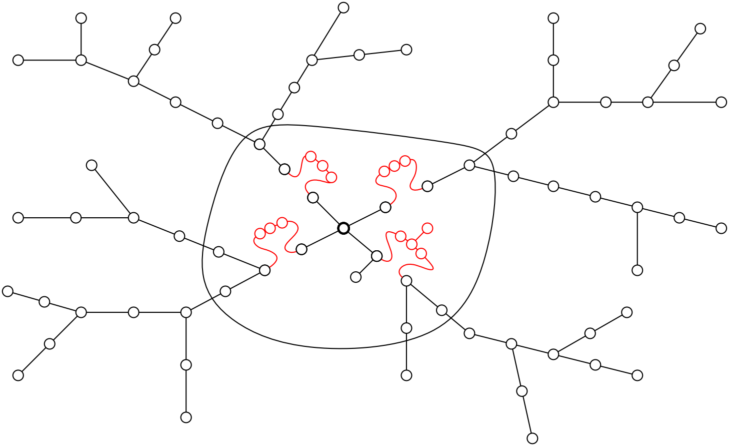
# Proof idea



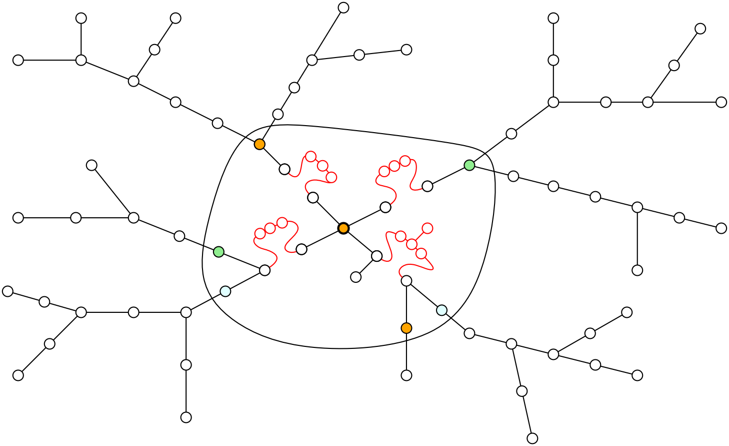
# Proof idea



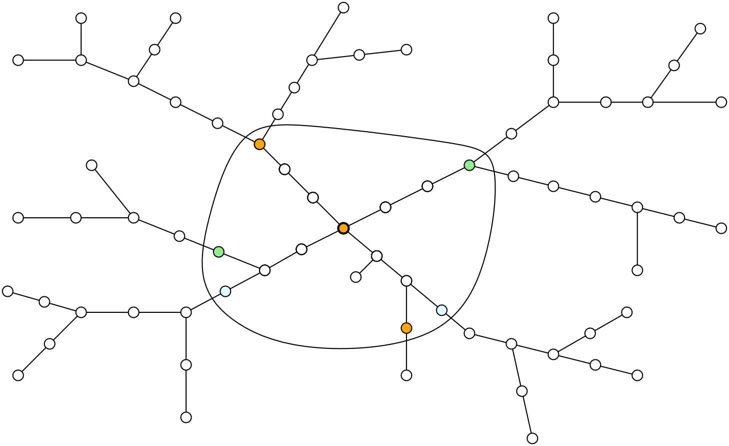
# Proof idea



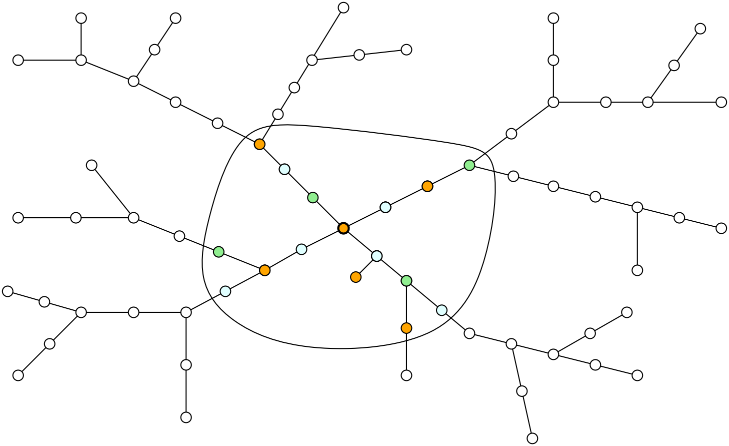
# Proof idea



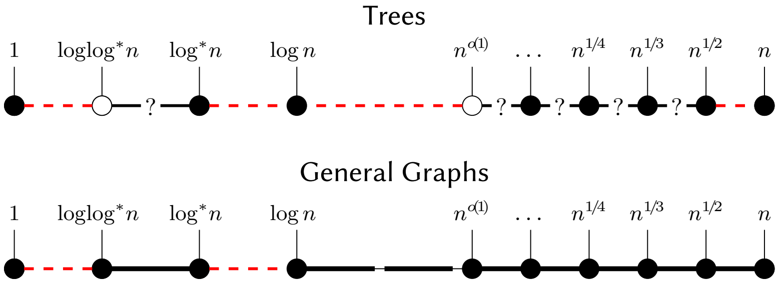
# Proof idea



# Proof idea



# Conclusions and Open Problems




- What changes in the case of randomization?
- What happens if nodes do *not* know the size of the graph?
- Can we prove automatic speedups for some subclass of LCL problems?

# Thank you!

# Questions?