Distributed Computing

10 - Speed Up Simulation

Mikaël Rabie
Université de Paris, IRIF
Speed Up for 3-coloring a Path
(LOCAL Model)
From $n$ colors to $\log n$ colors

\[
42 \rightarrow \log n \rightarrow 2\log n \rightarrow \log \log n + 1 \text{ bits}
\]

After $\log^* n$ iterations, $O(1)$ bits.

After $O(1)$ greedy recoloring steps, 3-coloring.
From $n$ colors to $\log n$ colors

After $\log^* n$ iterations, $O(1)$ bits.

After $O(1)$ greedy recoloring steps, 3-coloring.
From $n$ colors to $\log n$ colors

After $\log^* n$ iterations, $O(1)$ bits.

After $O(1)$ greedy recoloring steps, 3-coloring.
From \( n \) colors to \( \log n \) colors

\[
\begin{align*}
&101010 \quad 1100110 \quad 100100 \\
&\quad 42 \quad 102 \quad 36
\end{align*}
\]

\( \#0 \)

After \( \log^* n \) iterations, \( O(1) \) bits.

After \( O(1) \) greedy recoloring steps, 3-coloring.
From $n$ colors to $\log n$ colors

\[ n \text{ colors} \Rightarrow \log n \text{ colors} \Rightarrow 2^{\log n} \text{ new colors} \Rightarrow \log \log n + 1 \text{ bits} \]

After $\log^* n$ iterations, $O(1)$ bits.

After $O(1)$ greedy recoloring steps, 3-coloring.
From $n$ colors to $\log n$ colors

After $\log^* n$ iterations, $O(1)$ bits.

After $O(1)$ greedy recoloring steps, 3-coloring.
From $n$ colors to $\log n$ colors

\[
110 \Rightarrow \log n \text{ bits} \Rightarrow 2^{\log n} \text{ new colors} \Rightarrow \log \log n + 1 \text{ bits}
\]

After $\log \ast n$ iterations, $O(1)$ bits.

After $O(1)$ greedy recoloring steps, 3-coloring.
From $n$ colors to $\log n$ colors

$\ldots \quad 110 \quad \ldots$

$6 \quad \ldots$

3#0

$\ldots \quad 101 \quad \ldots$

5

2#1

$\ldots \quad 100 \quad \ldots$

4

2#0

After $\log* n$ iterations, $O(1)$ bits.

After $O(1)$ greedy recoloring steps, 3-coloring.
From \( n \) colors to \( \log n \) colors

\[ n \text{ colors} \Rightarrow \log n \text{ bits} \Rightarrow 2 \log n \text{ new colors} \Rightarrow \log \log n + 1 \text{ bits} \]
From \( n \) colors to \( \log n \) colors

\[
n \text{colors} \Rightarrow \log n \text{ bits} \Rightarrow 2 \log n \text{ new colors} \Rightarrow \log \log n + 1 \text{ bits}
\]

After \( \log^* n \) iterations, \( O(1) \) bits.

After \( O(1) \) greedy recoloring steps, 3-coloring.
An algorithm which colors the $n$-cycle with three colors requires at least $\frac{1}{2}(\log^* n - 3)$ communication rounds.

The same bound holds also for randomized algorithms.
Speed up 3-coloring

$\mathcal{A}$ : algorithm that $k$-colors nodes in $T$ rounds.

$\mathcal{A}'$ : algorithm that $k'$-colors nodes in $T - 1$ rounds.
$A :$ algorithm that $k$-colors nodes in $T$ rounds.
Speed up 3-coloring

$A$ : algorithm that $k$-colors nodes in $T$ rounds.

$c \in [1, k]$
Speed up 3-coloring

\[ A : \text{algorithm that } k\text{-colors nodes in } T \text{ rounds.} \]
\( \mathcal{A} \): algorithm that \( k \)-colors nodes in \( T \) rounds.

\[ \forall i \leq n \]

\[ T - 1 \quad T - 1 \]
\( \mathcal{A} \) : algorithm that \( k \)-colors nodes in \( T \) rounds.

\( \forall id \leq n \)

\( T \quad c \in [1, k] \quad T \)

\( T - 1 \quad S_L \in 2^k \quad T - 1 \)
A : algorithm that \( k \)-colors nodes in \( T \) rounds.

\[ c \in [1, k] \]

\[ T - 1 \]

\[ S_L \]

\[ \forall \text{id} \leq n \]
\( \mathcal{A} \): algorithm that \( k \)-colors nodes in \( T \) rounds.
\( \mathcal{A} \) : algorithm that \( k \)-colors nodes in \( T \) rounds.

\[ c \in [1, k] \]

\[ T - 1 \]

\[ S_L \# S_R \in 2^k \times 2^k \]
A : algorithm that $k$-colors nodes in $T$ rounds.

$S_L \# S_R \in 2^k \times 2^k$

$S_L \cap S_R = \emptyset$
\[ A : \text{algorithm that } k\text{-colors nodes in } T\text{ rounds}. \]
\( \mathcal{A} \) : algorithm that \( k \)-colors nodes in \( T \) rounds.
A : algorithm that $k$-colors nodes in $T$ rounds.

\[ SL \cap SR = \emptyset \land S'_L \cap S'_R = \emptyset \]
\(A\) : algorithm that \(k\)-colors nodes in \(T\) rounds.

\[
S_L \cap S_R = \emptyset \land S_L' \cap S_R' = \emptyset
\]

\[
S_L' \cap S_R \neq \emptyset
\]
\[ A : \text{algorithm that}\ k\text{-colors nodes in}\ T\ \text{rounds.} \]

\[ S_L \cap S_R = \emptyset \land S'_L \cap S'_R = \emptyset \]

\[ S'_L \cap S_R \neq \emptyset \]

\[ S_L \neq S'_L \land S_R \neq S'_R \]
\[ A : \text{algorithm that } k\text{-colors nodes in } T \text{ rounds.} \]
\[ A_1 : \text{algorithm that } 4^k\text{-colors edges in } T - 1 \text{ rounds.} \]
\( \mathcal{A} \): algorithm that \( k \)-colors nodes in \( T \) rounds.

\( \mathcal{A}_1 \): algorithm that \( 4^k \)-colors edges in \( T - 1 \) rounds.

c \( \in \) \([1, 4^k]\)
\[ \mathcal{A} : \text{algorithm that } k\text{-colors nodes in } T \text{ rounds.} \]
\[ \mathcal{A}_1 : \text{algorithm that } 4^k\text{-colors edges in } T - 1 \text{ rounds.} \]
\( \mathcal{A} \): algorithm that \( k \)-colors nodes in \( T \) rounds.
\( \mathcal{A}_1 \): algorithm that \( 4^k \)-colors edges in \( T - 1 \) rounds.
\( \mathcal{A} \) : algorithm that \( k \)-colors nodes in \( T \) rounds.

\( \mathcal{A}_1 \) : algorithm that \( 4^k \)-colors edges in \( T - 1 \) rounds.
\( \mathcal{A} \) : algorithm that \( k \)-colors nodes in \( T \) rounds.

\( \mathcal{A}_1 \) : algorithm that \( 4^k \)-colors edges in \( T - 1 \) rounds.
Speed up 3-coloring

\( \mathcal{A} \): algorithm that \( k \)-colors nodes in \( T \) rounds.

\( \mathcal{A}_1 \): algorithm that \( 4^k \)-colors edges in \( T - 1 \) rounds.

\( \mathcal{A}_2 \): algorithm that \( 4^{4^k} \)-colors nodes in \( T - 1 \) rounds.

\( c \in [1, 4^{4^k}] \)
Speed up 3-coloring

\( \mathcal{A} \) : algorithm that \( k \)-colors nodes in \( T \) rounds.
\( \mathcal{A}_1 \) : algorithm that \( 4^k \)-colors edges in \( T - 1 \) rounds.
\( \mathcal{A}_2 \) : algorithm that \( 4^{4^k} \)-colors nodes in \( T - 1 \) rounds.
\[ \ldots \]
\( \mathcal{A}_{2^l} \) : algorithm that \( 4^{4^{4^{\ldots^k}}} \)-colors nodes in \( T - l \) rounds.
Speed up 3-coloring

\( A \) : algorithm that \( k \)-colors nodes in \( T \) rounds.
\( A_1 \) : algorithm that \( 4^k \)-colors edges in \( T - 1 \) rounds.
\( A_2 \) : algorithm that \( 4^4^k \)-colors nodes in \( T - 1 \) rounds.
\ldots
\( A_{2^l} \) : algorithm that \( 4^4^4^\ldots^k \)-colors nodes in \( T - l \) rounds.

If we have \( k = 3 \) and \( T < \frac{\log^* n}{2} \),
\[ \Rightarrow \] We create a \( c \)-coloring algorithm in 0 rounds, with \( c = 4^4^3 < n \).
The \((d, \delta)\) Bipartite Algorithm
Biregular Trees

- Black-White Bipartite Tree
- Black nodes of degree \(d\) or 1, White nodes of degree \(\delta\) or 1.
The Black-White Algorithm

- **Bipartite locally verifiable problem** on a \((d, \delta)\) Biregular Trees is a 3-tuple \(\Pi = (\Sigma, A, P)\):
  - \(\Sigma\) : output alphabet
  - \(A \subseteq \Sigma^d\) : outputs on Black nodes of degree \(d\)
  - \(P \subseteq \Sigma^\delta\) : outputs on White nodes of degree \(\delta\)

- Black nodes are **Active** : they produce an output of \(A\) on their edges.
- White nodes are **Passive** : they check that the output on their edges is in \(P\).
Examples

- 5-edge coloring, $\Sigma = \{1, 2, 3, 4, 5\}$
- $A = \{[c_1, c_2, c_3, c_4] \text{ with } c_1 < c_2 < c_3 < c_4\}$, $P = \{[c_1, c_2, c_3] \text{ with } c_1 < c_2 < c_3\}$
Examples

- Maximal Matching, $\Sigma = \{M, U, P\}$

Examples

- Sinkless Orientation, $\Sigma = \{I, O\}$
- $A = \{[O, _, _, _]\}, P = \{[I, _, _]\}$
Examples

- Weak 3-labeling, $\Sigma = \{1, 2, 3\}$
- $A = \{[c_1, c_2, c_3, c_4] \text{ with } c_1 \neq c_4\}, \ P = \{[c_1, c_2, c_3] \text{ with } c_1 \neq c_3\}$
Round Elimination
Principle of Round Elimination

- Problem $\Pi_0$ solved in $T$ rounds on $(d, \delta)$-biregular trees
- $\Rightarrow$ Construct $\Pi_1 = re(\Pi_0)$ solvable in $T - 1$ rounds on $(\delta, d)$-biregular trees
- $\Pi_0 \rightarrow \Pi_1 \rightarrow \ldots \rightarrow \Pi_T = re^T(\Pi_0)$ solvable in 0 round
- $\Pi_T$ not solvable with no communication $\Rightarrow \Pi_0$ not solvable in $T$ rounds

$\Pi_0 = (\Sigma_0, A_0, P_0), \Pi_1 = (\Sigma_1, A_1, P_1)$

- $\Sigma_1 \subseteq (P(\Sigma_0) \setminus \emptyset)$
- $[X_1, \ldots, X_\delta] \in A_1 \iff \forall x_1 \in X_1 \ldots \forall x_\delta \in X_\delta, [x_1, \ldots, x_\delta] \in P_0$
- $[X_1, \ldots, X_d] \in P_1 \iff \exists x_1 \in X_1 \ldots \exists x_d \in X_d, [x_1, \ldots, x_d] \in A_0$
Weak 3-labeling

\(\Pi_0\) on \((3, 2)\)-biregular trees:

- \(\Sigma_0 = \{1, 2, 3\}\)
- \(A_0 = \{[1, 1, 2], [1, 1, 3], [1, 2, 2], [1, 2, 3], [1, 3, 3], [2, 2, 3], [2, 3, 3]\}\)
- \(P_0 = \{[1, 1], [2, 2], [3, 3]\}\)

\(\Pi_1\) on \((2, 3)\)-biregular trees
Weak 3-labeling

\( \Pi_0 \) on \((3, 2)\)-biregular trees:

- \( \Sigma_0 = \{1, 2, 3\} \)
- \( A_0 = \{[1, 1, 2], [1, 1, 3], [1, 2, 2], [1, 2, 3], [1, 3, 3], [2, 2, 3], [2, 3, 3]\} \)
- \( P_0 = \{[1, 1], [2, 2], [3, 3]\} \)

\( \Pi_1 \) on \((2, 3)\)-biregular trees:

- \( \Sigma_1 = \{\{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\} \)
- \( A_1 = \{\{\{1\}, \{1\}\}, \{\{2\}, \{2\}\}, \{\{3\}, \{3\}\}\} \)
- \( P_1 = \{\{\{1\}, \{1\}, \{2\}\}, \{\{1\}, \{1\}, \{3\}\}, \{\{1\}, \{2\}, \{2\}\}, \{\{1\}, \{2\}, \{3\}\}, \{\{1\}, \{3\}, \{3\}\}, \{\{2\}, \{2\}, \{3\}\}, \{\{2\}, \{3\}, \{3\}\}\} \)
Neighbourhood Simulation
Neighbourhood Simulation

$\text{ball}(u, 3)$
Neighbourhood Simulation

ball(u, 3)
Neighbourhood Simulation

ball(u, 3)
Neighbourhood Simulation

$ball(u, 3)$
Neighbourhood Simulation

\[ \text{ball}(u, 3) \]
Algorithm of $\Pi_1$

$A_0$ solves $\Pi_0$ in $T$ rounds. $A_1$ does in $T - 1$ rounds:

- For each White node $u$, $u$ gets its ball of radius $T - 1$.
- For each $v \in N(u)$, $u$ simulates all possible balls of radius $T$ of $v$.
- $X_{u,v} = \{\text{outputs of the possible balls of radius } T \text{ of } v\}$.

Correction:

- $\forall x_{v_1} \in X_{u,v_1} \ldots x_{v_\delta} \in X_{u,\delta}$, their exists $\text{ball}(u, T)$ where $A_0$ produces $[x_{v_1}, \ldots, x_{v_\delta}]$ 
  $\Rightarrow [x_{v_1}, \ldots, x_{v_\delta}] \in P_0 \Rightarrow [X_{u,v_1}, \ldots, X_{u,v_\delta}] \in A_1$.
- For any Black node $v$, for any $\text{ball}(v, T)$ and $u \in N(v)$, $X_{u,v}$ contains $A_0(u, v)$ 
  $\Rightarrow [X_{u_1,v}, \ldots, X_{v_d,v}] \in P_1$. 
Back to weak 3-labeling

- $\Pi_1$ on $(2, 3)$-biregular trees:
  - $\Sigma_1 = \{\{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\}$
  - $A_1 = \{[\{1\}, \{1\}], [\{2\}, \{2\}], [\{3\}, \{3\}]\}$
  - $P_1 = \{[\{1\}, \{1\}, \{2\}], [\{1\}, \{1\}, \{3\}], [\{1\}, \{2\}, \{2\}], [\{1\}, \{2\}, \{3\}], [\{1\}, \{3\}, \{3\}], [\{2\}, \{2\}, \{3\}], [\{2\}, \{3\}, \{3\}]\}$
Back to weak 3-labeling

- $\Pi_1$ on $(2, 3)$-biregular trees:
  - $\Sigma_1 = \{1, 2, 3\}$
  - $A_1 = \{[1, 1], [2, 2], [3, 3]\}$
  - $P_1 = \{[1, 1, 2], [1, 1, 3], [1, 2, 2], [1, 2, 3], [1, 3, 3], [2, 2, 3], [2, 3, 3]\}$
Back to weak 3-labeling

- $\Pi_1$ on $(2, 3)$-biregular trees:
  - $\Sigma_1 = \{1, 2, 3\}$
  - $A_1 = \{[1, 1], [2, 2], [3, 3]\}$
  - $P_1 = \{[1, 1, 2], [1, 1, 3], [1, 2, 2], [1, 2, 3], [1, 3, 3], [2, 2, 3], [2, 3, 3]\}$

- $\Pi_1$ cannot be solved in 0 round $\Rightarrow$ $\Pi_0$ needs at least 2 rounds.

- $\Pi_2$ on $(3, 2)$-biregular trees
Back to weak 3-labeling

- \( \Pi_1 \) on \((2, 3)\)-biregular trees:
  - \( \Sigma_1 = \{1, 2, 3\} \)
  - \( A_1 = \{[1, 1], [2, 2], [3, 3]\} \)
  - \( P_1 = \{[1, 1, 2], [1, 1, 3], [1, 2, 2], [1, 2, 3], [1, 3, 3], [2, 2, 3], [2, 3, 3]\} \)
  - \( \Pi_1 \) cannot be solved in 0 round \(\Rightarrow\) \( \Pi_0 \) needs at least 2 rounds.

- \( \Pi_2 \) on \((3, 2)\)-biregular trees:
  - \( \Sigma_2 = \{\{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\} \)
  - \( A_2 = \{[[1, 2, 3], \{1, 2\}, \{3\}], [[1, 2, 3], \{1, 3\}, \{2\}], [[1, 2, 3], \{2, 3\}, \{1\}], [[1, 2], \{1, 3\}, \{2, 3\}]]\} \)
  - \( P_2 = \{[X, Y]|X \cap Y \neq \emptyset\} \)
  - \( \Pi_2 \) can be solved in 0 round.
Sinkless Orientation
Sinkless Orientation on Paths

- \( \Sigma = \{ I, O \} \)
- \( A = \{ [I, O], [O, O] \} \)
- \( P = \{ [I, O], [I, I] \} \)

**Brandt et. al (2016)**

Sinkless Orientation cannot be solved in \( o(n) \) on paths.
Sinkless Orientation on Paths

- $\Sigma = \{I, O\}$
- $A = \{[I, O], [O, O]\}$
- $P = \{[I, O], [I, I]\}$

Brandt et. al (2016)
Sinkless Orientation cannot be solved in $o(n)$ on paths.

Suppose their exists an algorithm in time $T(n) \leq (n - 5)/4$:
Sinkless Orientation on Paths

- $\Sigma = \{I, O\}$
- $A = \{[I, O], [O, O]\}$
- $P = \{[I, O], [I, I]\}$

**Brandt et. al (2016)**
Sinkless Orientation cannot be solved in $o(n)$ on paths.

Suppose their exists an algorithm in time $T(n) \leq (n - 5)/4$:
Sinkless Orientation on Paths

- \( \Sigma = \{I, O\} \)
- \( A = \{[I, O], [O, O]\} \)
- \( P = \{[I, O], [I, I]\} \)

*Brandt et. al (2016)*

Sinkless Orientation cannot be solved in \( o(n) \) on paths.

Suppose their exists an algorithm in time \( T(n) \leq (n - 5)/4 \) :
Sinkless Orientation on Paths

- $\Sigma = \{l, O\}$
- $A = \{[l, O], [O, O]\}$
- $P = \{[l, O], [l, l]\}$

**Brandt et. al (2016)**
Sinkless Orientation cannot be solved in $o(n)$ on paths.

Suppose their exists an algorithm in time $T(n) \leq (n - 5)/4$:
Sinkless Orientation on Paths

- $\Sigma = \{I, O\}$
- $A = \{[I, O], [O, O]\}$
- $P = \{[I, O], [I, I]\}$

**Brandt et. al (2016)**
Sinkless Orientation cannot be solved in $o(n)$ on paths.

Suppose there exists an algorithm in time $T(n) \leq (n - 5)/4$:
Sinkless Orientation on Paths

- $\Sigma = \{I, O\}$
- $A = \{[I, O], [O, O]\}$
- $P = \{[I, O], [I, I]\}$

**Brandt et. al (2016)**

Sinkless Orientation cannot be solved in $o(n)$ on paths.

Suppose there exists an algorithm in time $T(n) \leq (n - 5)/4$:

![Diagram of Sinkless Orientation on Paths]
Sinkless Orientation on Paths

- $\Sigma = \{I, O\}$
- $A = \{[I, O], [O, O]\}$
- $P = \{[I, O], [I, I]\}$

**Brandt et. al (2016)**
Sinkless Orientation cannot be solved in $o(n)$ on paths.

Suppose there exists an algorithm in time $T(n) \leq (n - 5)/4$:
Sinkless Orientation on Paths

- \( \Sigma = \{I, O\} \)
- \( A = \{[I, O], [O, O]\} \)
- \( P = \{[I, O], [I, I]\} \)

**Brandt et. al (2016)**

Sinkless Orientation cannot be solved in \( o(n) \) on paths.

Suppose their exists an algorithm in time \( T(n) \leq (n - 5)/4 \):

![Diagram of sinkless orientation on paths](image)
Sinkless Orientation on Trees

\[ \Pi : \]

- Nodes of degree \( d \geq 3 \) need an outgoing edge
- Nodes of degree \( \leq 2 \) have no restriction

There exists a \( O(\log n) \) algorithm:

- Nodes of degree \( \leq 2 \) orient edges toward them (in case of conflict, toward the Black node)
- If at the previous round, one of your edge got oriented, it must be outgoing
  Orient the remaining of your edges toward you
- In a tree, there is always at distance \( \log n \) from you a node of degree \( \leq 2 \)
Sinkless Orientation Speed Up

\[ \Pi_0 : \]
- \( \Sigma_0 = \{ I, O \} \)
- \( A_0 = \{ [O, O, O], [O, O, I], [O, I, I] \} \)
- \( P_0 = \{ [I, I, I], [I, I, O], [I, O, O] \} \)

\[ \Pi_1 \]
\( \Pi_0 : \)
- \( \Sigma_0 = \{I, O\} \)
- \( A_0 = \{[O, O, O], [O, O, I], [O, I, I]\} \)
- \( P_0 = \{[I, I, I], [I, I, O], [I, O, O]\} \)

\( \Pi_1 : \)
- \( \Sigma_1 = \{\{I\}, \{O, I\}\} \)
- \( A_1 = \{[\{I\}, \{O, I\}, \{O, I\}]\} \)
- \( P_1 = \{[\{I\}, \{I\}, \{O, I\}], [\{I\}, \{O, I\}, \{O, I\}]\} \)
Sinkless Orientation Speed Up

\[\Pi_0:\]
- \(\Sigma_0 = \{I, O\}\)
- \(A_0 = \{[O, O, O], [O, O, I], [O, I, I]\}\)
- \(P_0 = \{[I, I, I], [I, I, O], [I, O, O]\}\)

\[\Pi_1:\]
- \(\Sigma_1 = \{\{I\}, \{O, I\}\} = \{A, B\}\)
- \(A_1 = \{\{\{I\}, \{O, I\}, \{O, I\}\}\} = \{[A, B, B]\}\)
- \(P_1 = \{\{\{I\}, \{I\}, \{O, I\}\}, \{\{I\}, \{O, I\}, \{O, I\}\}, \{\{O, I\}, \{O, I\}, \{O, I\}\}\}\)
  = \{[A, A, B], [A, B, B], [B, B, B]\}\)
Sinkless Orientation Speed Up

\[ \Pi_1 : \]

- \[ \Sigma_1 = \{A, B\} \]
- \[ A_1 = \{[A, B, B]\} \]
- \[ P_1 = \{[A, A, B], [A, B, B], [B, B, B]\} \]

\[ \Pi_2 \]
Sinkless Orientation Speed Up

\( \Pi_1 : \)
- \( \Sigma_1 = \{A, B\} \)
- \( A_1 = \{[A, B, B]\} \)
- \( P_1 = \{[A, A, B], [A, B, B], [B, B, B]\} \)

\( \Pi_2 : \)
- \( \Sigma_2 = \{\{B\}, \{A, B\}\} \)
- \( A_2 = \{\{\{B\}, \{A, B\}, \{A, B\}\}\} \)
- \( P_2 = \{\{\{B\}, \{B\}, \{A, B\}\}, \{\{B\}, \{A, B\}, \{A, B\}\}, \{\{A, B\}, \{A, B\}, \{A, B\}\}\} \)
Sinkless Orientation Lower Bound

- $\forall i \geq 1, \Pi_i = \Pi_1$
- $A$ solves $\Pi_0$ in $T$ rounds $\Rightarrow \Pi_1$ can be solved in $0$ round

Sinkless Orientation cannot be solved?!
Sinkless Orientation Lower Bound

- $\forall i \geq 1, \Pi_i = \Pi_1$
- $A$ solves $\Pi_0$ in $T$ rounds $\Rightarrow \Pi_1$ can be solved in 0 round

Sinkless Orientation cannot be solved ?!

- A $T$ round algorithm with no leaves $\Rightarrow 3 \times 2^T$ nodes
- With $T(n) = \Omega(\log n)$, we cannot do the speed up

Chang et. al (2016)
Sinkless Orientation cannot be solved in $o(\log n)$ rounds.
Bibliography


- Sebastian Brandt. **An automatic speedup theorem for distributed problems.** In PODC 2019.

- Sebastian Brandt, Orr Fischer, Juho Hirvonen, Barbara Keller, Tuomo Lempiäinen, Joel Rybicki, Jukka Suomela, and Jara Uitto. **A lower bound for the distributed Lovász local lemma.** In STOC 2016.

- Yi-Jun Chang, Tsvi Kopelowitz, and Seth Pettie. **An exponential separation between randomized and deterministic complexity in the LOCAL model.** In FOCS 2016.