Distributed Computing
8 - Speed Up Simulation

Mikaël Rabie
Université Paris-Cité, IRIF
Speed Up for 3-coloring a Path
(LOCAL Model)
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

\[
\begin{align*}
\text{101010} & \quad \text{1100110} & \quad \text{100100} \\
\text{42} & \quad \text{102} & \quad \text{36}
\end{align*}
\]
From \( n \) colors to \( \log n \) colors

\[ 101010 \quad \rightarrow \quad 1100110 \quad \rightarrow \quad 100100 \]

\[ \ldots \quad 42 \quad \rightarrow \quad 102 \quad \rightarrow \quad 36 \quad \ldots \]

After \( \log \ast n \) 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.

$n$ colors $\Rightarrow \log n$ bits $\Rightarrow 2 \log n$ new colors $\Rightarrow \log \log n + 1$ 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.
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

$n$ colors $\Rightarrow \log n$ bits $\Rightarrow 2 \log n$ new colors $\Rightarrow \log \log n + 1$ bits
From $n$ colors to $\log n$ colors

$n$ colors $\Rightarrow$ log $n$ bits $\Rightarrow$ 2 log $n$ new colors $\Rightarrow$ log log $n + 1$ bits

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

$n$ colors $\Rightarrow \log n$ bits $\Rightarrow 2\log n$ new colors $\Rightarrow \log \log n + 1$ bits

After $\log^* n/2$ 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.
\( \mathcal{A} : \) algorithm that \( k \)-colors nodes in \( T \) rounds.
\( A \) : algorithm that \( k \)-colors nodes in \( T \) rounds.

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

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

\[ \forall \text{id} \leq n \]

\[ T - 1 \]

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

$\forall id \leq n$, $T - 1$, $S_L \in 2^k$, $T - 1$
\(A\) : algorithm that \(k\)-colors nodes in \(T\) rounds.

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

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

\[ T - 1 \quad S_L \quad S_R \quad T - 1 \quad \forall id \leq n \]
\(\mathcal{A}\) : algorithm that \(k\)-colors nodes in \(T\) rounds.

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

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

$c \in [1, k]$

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

$S_L \cap S_R = \emptyset$
Algorithm \( \mathcal{A} \) that \( k \)-colors nodes in \( T \) rounds.

\[ T - 1 \]

\( S_L \# S_R \)

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

\[
S_L \cap S_R = \emptyset \quad \& \quad S'_L \cap S'_R = \emptyset
\]
$\mathcal{A}$ : algorithm that $k$-colors nodes in $T$ rounds.

$S_L \# S_R = \emptyset \land S'_L \# S'_R = \emptyset$

$S_L \cap S_R = \emptyset \land S'_L \cap S'_R = \emptyset$

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

\[ S_L \# S_R = \emptyset \text{ and } S'_L \# S'_R = \emptyset \]

\[ S_L \cap S_R = \emptyset \text{ and } S'_L \cap S'_R = \emptyset \]

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

\[ S_L \# S_R \neq S'_L \# S'_R \]
\(A\) : algorithm that \(k\)-colors nodes in \(T\) rounds.

\(A_1\) : algorithm that \(4^k\)-colors edges in \(T - 1\) rounds.

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

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

$A_1$: 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.

\[ c \in [1, 4^k] \]

\[ \forall id \leq n \]

\[ S_L \in 2^{4^k} \]
\(A\) : algorithm that \(k\)-colors nodes in \(T\) rounds.

\(A_1\) : 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.

\[ c \in [1, 4^k] \]

\( S_L \# S_R \)
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.

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

\[ \mathcal{A} : \text{algorithm that } k\text{-colors nodes in } T \text{ rounds.} \]
\[ \mathcal{A}_1 : \text{algorithm that } 4^k\text{-colors edges in } T - 1 \text{ rounds.} \]
\[ \mathcal{A}_2 : \text{algorithm that } 4^{4^k} \text{-colors nodes in } T - 1 \text{ rounds.} \]
\[ \cdots \]
\[ \mathcal{A}_{2l} : \text{algorithm that } 4^{4^{4^{\cdots^{4^k}}}} \text{-colors nodes in } T - l \text{ rounds.} \]
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.

\[ \vdots \]

$A_{2^l}$: algorithm that $4^{4^{4^{\cdots^k}}}$-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 $c = 4^{4^{4^3}} < n$.  

Contradiction
The $(d, \delta)$ Bipartite Algorithm
Port-Numbering Model

- No identifiers
- Symmetry breaking among neighbours using port numbers
(\(d, \delta\)) Biregular Trees

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

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

- Sinkless Orientation
Examples

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

- Maximal Matching
Examples

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

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

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

- $\Sigma_1 = \{\{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\}$
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:

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

$ball(u, 3)$
Neighbourhood Simulation

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

ball(u, 3)
Neighbourhood Simulation

$ball(u, 3)$
Neighbourhood Simulation

`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 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 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:
  - \( \Sigma_1 = \{\{1\}, \{2\}, \{3\}\} \)
  - \( A_1 = \{[\{1\}, \{1\}], [\{2\}, \{2\}], [\{3\}, \{3\}]\} \)
  - \( P_1 = \{[\{1\}, \{1\}, \{2\}], [\{1\}, \{1\}, \{3\}], [\{1\}, \{2\}, \{2\}], [\{1\}, \{2\}, \{3\}], [\{1\}, \{3\}, \{3\}], [\{2\}, \{2\}, \{3\}], [\{2\}, \{3\}, \{3\}]\} \)
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 rounds
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 rounds $\Rightarrow \Pi_0$ 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 rounds $\Rightarrow \Pi_0$ 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 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 $T(n) \leq (n - 5)/4$:
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 showing sinkless orientation on a path](image)
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 \( T(n) \leq (n - 5)/4 \):

1. 2 1 1 2 2 1 2 1 1 2 2 1 2 1 2 1 2 2 2 1 1
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 $T(n) \leq (n - 5)/4$:
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

\[ \Pi_0 : \]

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

\[ \Pi_1 \]
Sinkless Orientation Speed Up

$\Pi_0$:

- $\Sigma_0 = \{I, O\}$
- $A_0 = \{[O, O, O], [O, O, I], [O, I, I]\}$
- $P_0 = \{[I, I, I], [I, I, O], [I, O, O]\}$

$\Pi_1$:

- $\Sigma_1 = \{\{I\}, \{O, I\}\}$
- $A_1 = \{[\{I\}, \{O, I\}, \{O, I\}]\}$
- $P_1 = \{[\{I\}, \{I\}, \{O, I\}], [\{I\}, \{O, I\}, \{O, I\}], [\{O, I\}, \{O, I\}, \{O, I\}]\}$
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 leafs $\Rightarrow 3 \times 2^T$ 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.
Maximal Matching
Maximal Matching in $\Delta$-regular graphs

- Maximal Matching, $\Sigma = \{M, O, P\}$
- $A_0 = (M^{\Delta-1} \mid P^\Delta)$, $P_0 = (M[PO]^{\Delta-1} \mid O^\Delta)$
Maximal Matching in $\Delta$-regular graphs

- Maximal Matching, $\Sigma = \{M, O, P\}$
- $A_0 = (M O^{\Delta - 1} \mid P^\Delta)$, $P_0 = (M [PO]^{\Delta - 1} \mid O^\Delta)$

Balliu et. al (2019)
Maximal Matching needs $\Omega(\min\{\Delta, \log n / \log \log n\})$ rounds in the LOCAL Model.
Bibliography


