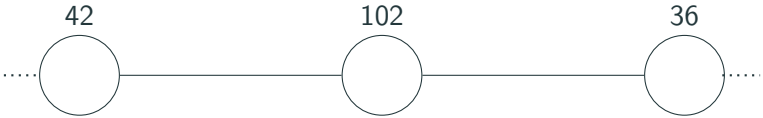
Distributed Computing
6 - Speed Up Simulation

Mikaël Rabie
Université Paris-Cité, IRIF

INSTITUT
DE RECHERCHE
EN INFORMATIQUE
FONDAMENTALE

Speed Up for 3-coloring a Path (LOCAL Model)

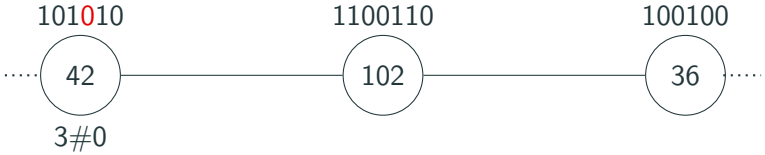
From n colors to $\log n$ colors



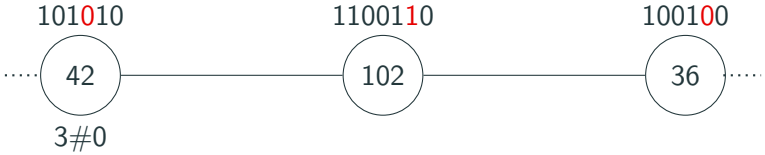
From n colors to $\log n$ colors

From n colors to $\log n$ colors

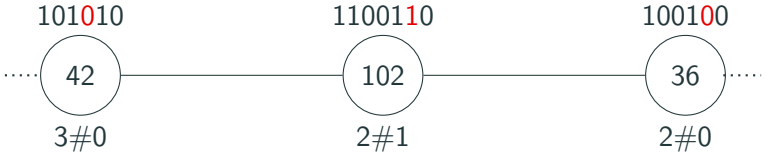
From n colors to $\log n$ colors



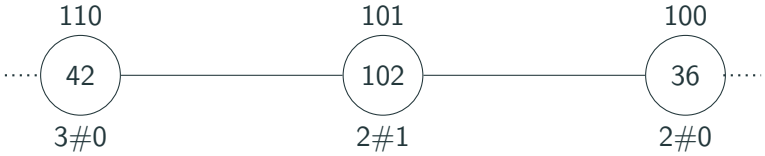
From n colors to $\log n$ colors



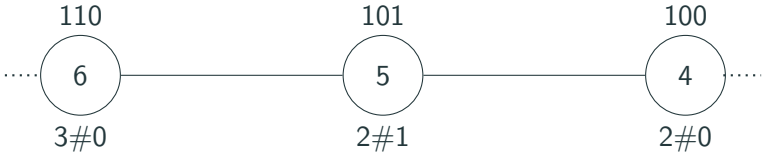
From n colors to $\log n$ colors



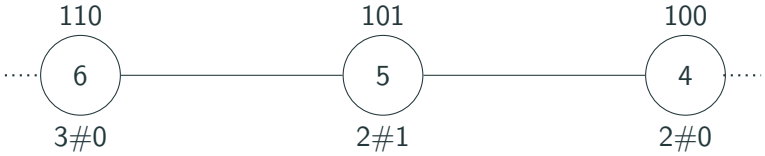
From n colors to $\log n$ colors



From n colors to $\log n$ colors

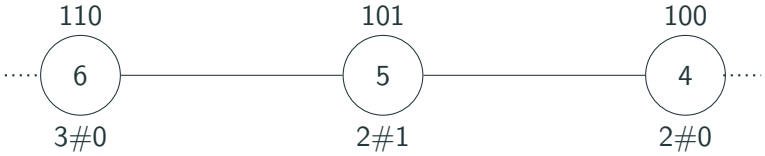


From n colors to $\log n$ colors



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

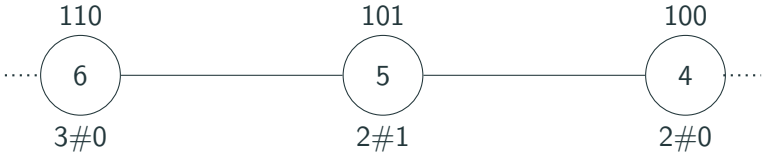
From n colors to $\log n$ colors



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

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

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.

Coloration Lower Bound

Linial (1992)

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.

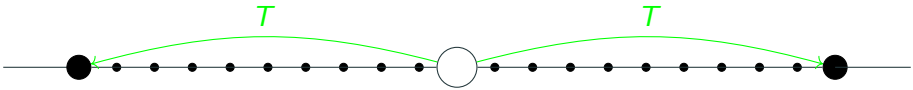
Speed up 3-coloring

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

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

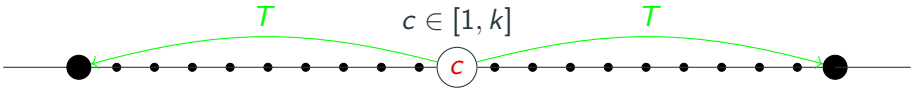
Speed up 3-coloring

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



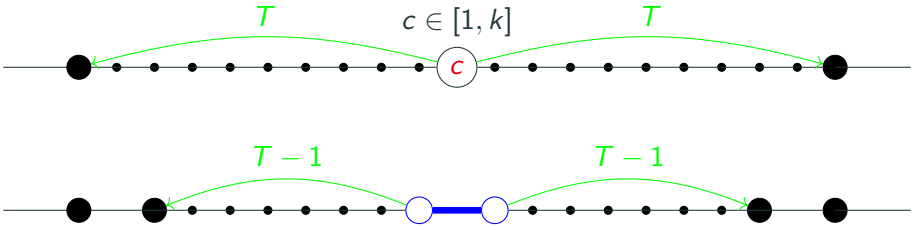
Speed up 3-coloring

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



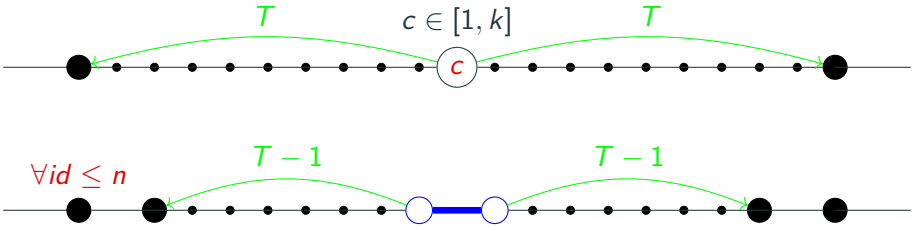
Speed up 3-coloring

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



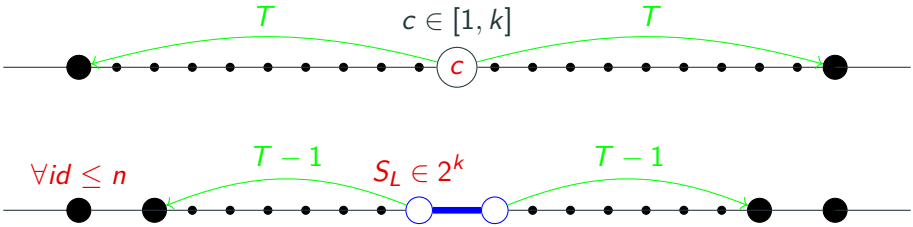
Speed up 3-coloring

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



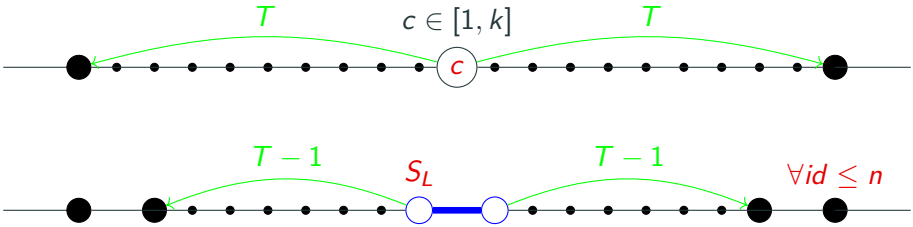
Speed up 3-coloring

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



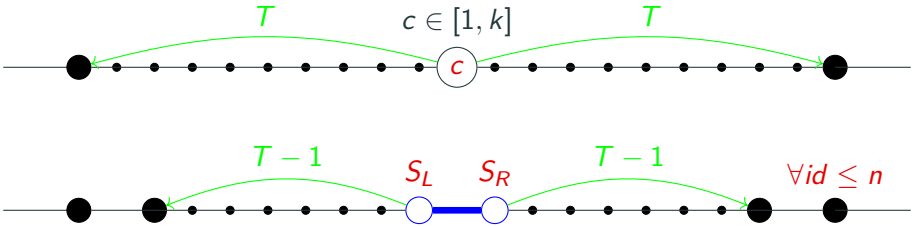
Speed up 3-coloring

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



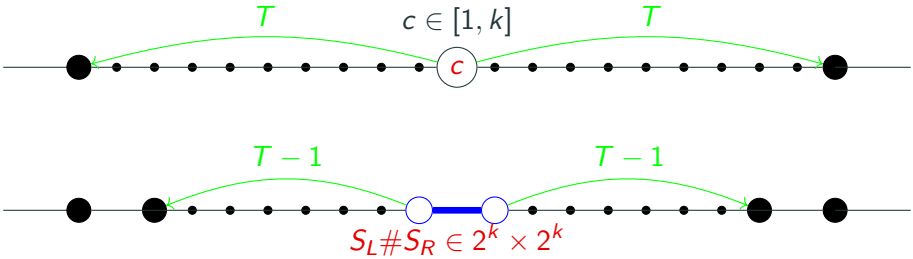
Speed up 3-coloring

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



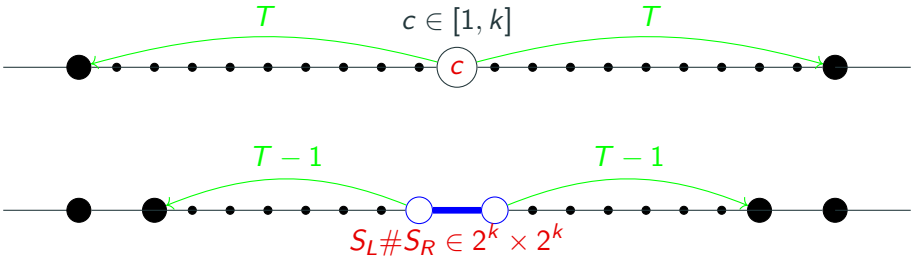
Speed up 3-coloring

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



Speed up 3-coloring

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



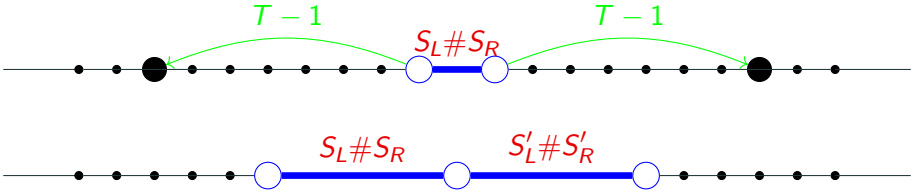
$$S_L \cap S_R = \emptyset$$

Speed up 3-coloring

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

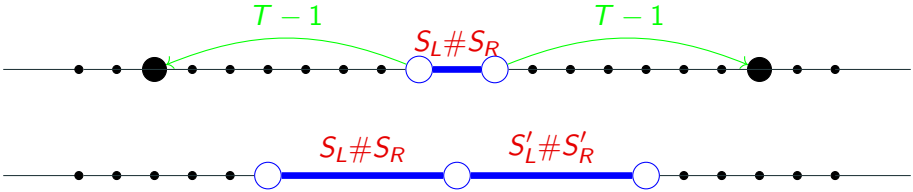
Speed up 3-coloring

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



Speed up 3-coloring

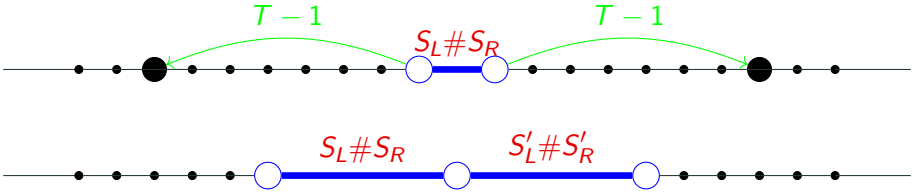
$\mathcal{A}$: algorithm that k-colors nodes in T rounds.



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

Speed up 3-coloring

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

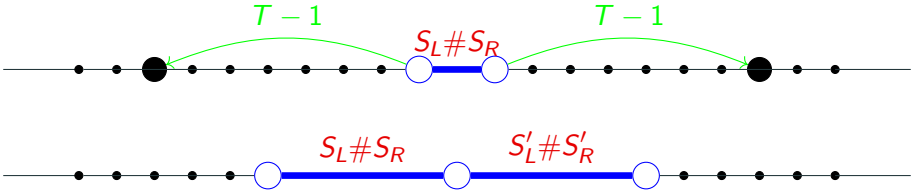


$$S_L \cap S_R = \emptyset \ \& \ S'_L \cap S'_R = \emptyset$$

$$S'_L \cap S_R \neq \emptyset$$

Speed up 3-coloring

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



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

$$S'_L \cap S_R \neq \emptyset$$

$$S_L \# S_R \neq S'_L \# S'_R$$

Speed up 3-coloring

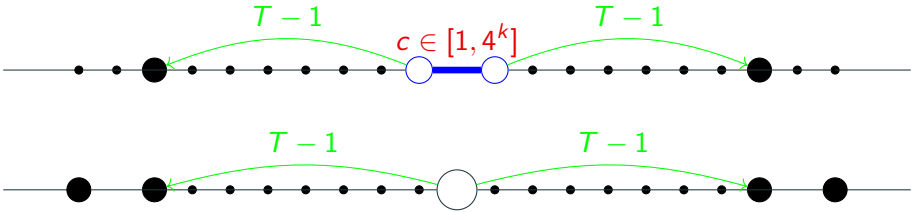
$\mathcal{A}$: algorithm that k-colors nodes in T rounds.

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

Speed up 3-coloring

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

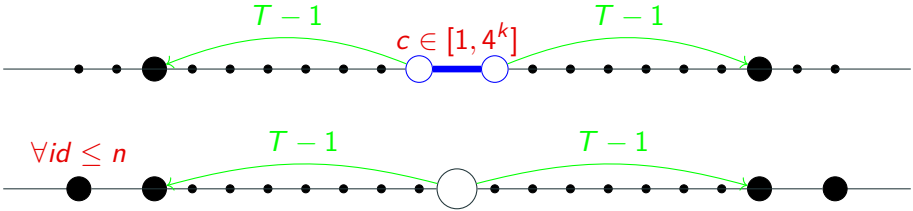
$\mathcal{A}_1$: algorithm that 4^k-colors edges in $T-1$ rounds.



Speed up 3-coloring

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

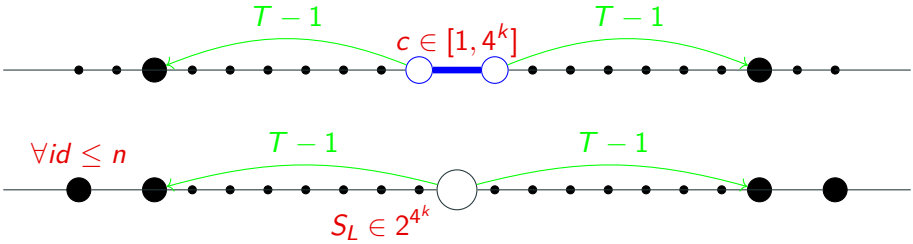
$\mathcal{A}_1$: algorithm that 4^k-colors edges in $T-1$ rounds.



Speed up 3-coloring

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

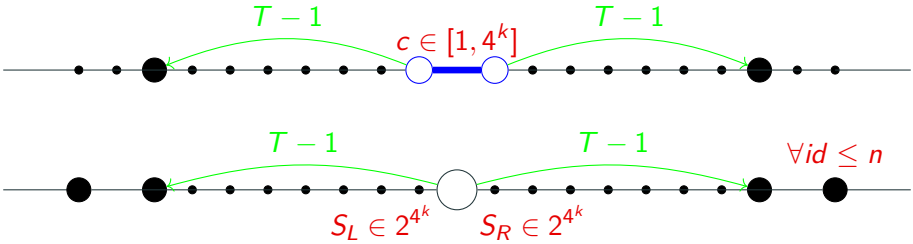
$\mathcal{A}_1$: algorithm that 4^k-colors edges in $T-1$ rounds.



Speed up 3-coloring

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

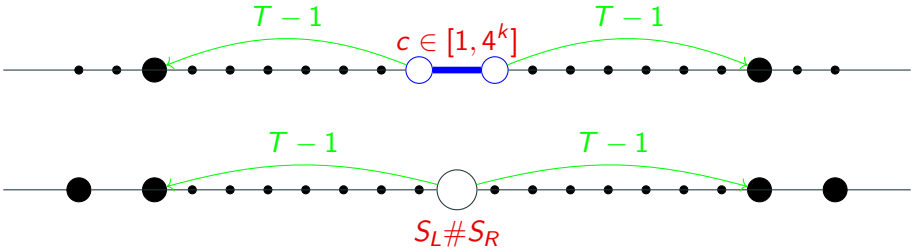
$\mathcal{A}_1$: algorithm that 4^k-colors edges in $T-1$ rounds.



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.

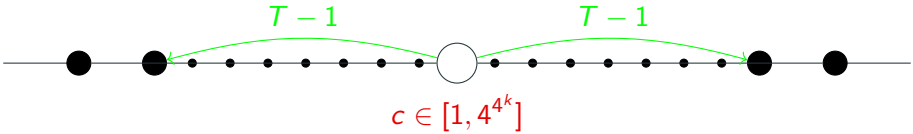


Speed up 3-coloring

$\mathcal{A}$: 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.



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.

...

$\mathcal{A}_{2l}$: algorithm that $4^{4^{\cdot^{\cdot^{\cdot^{k}}}}}$-colors nodes in $T-l$ 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.

...

$\mathcal{A}_{2l}$: algorithm that $4^{4^{\cdot^{\cdot^{\cdot^{k}}}}}$-colors nodes in $T-l$ rounds.

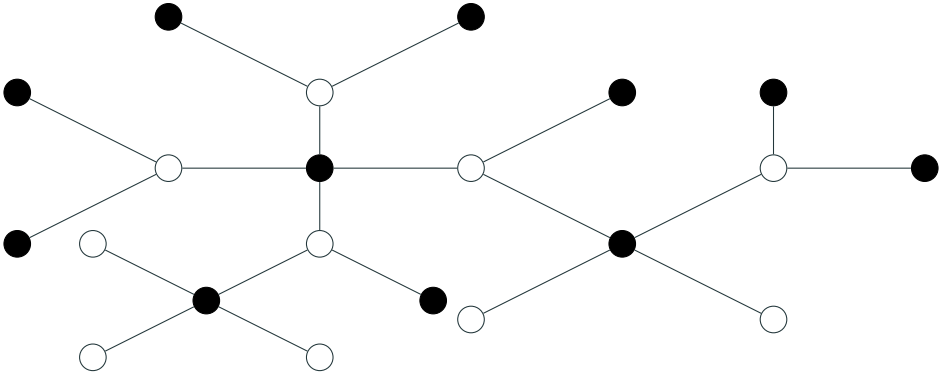
If we have $k=3$ and $T < \dfrac{\log_4^* n}{2}$,

$\Rightarrow$ We create a c-coloring algorithm in 0 rounds, with $c = 4^{4^{\cdot^{\cdot^{\cdot^{3}}}}} < n$.

Contradiction

(d, δ) Biregular Trees

- Black-White Bipartite Tree
- Black nodes of degree d or 1, White nodes of degree δ or 1.

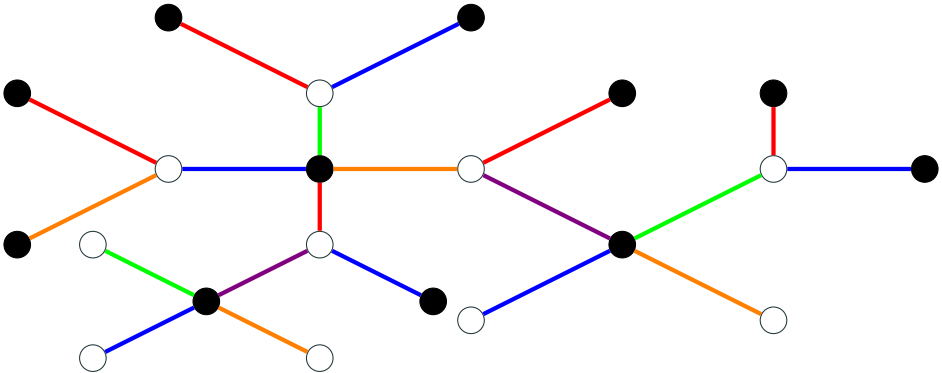


The Black-White Algorithm

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

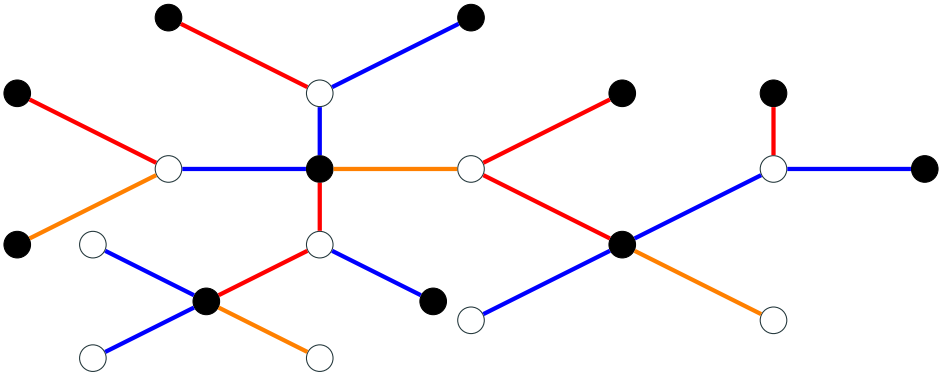
Examples

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



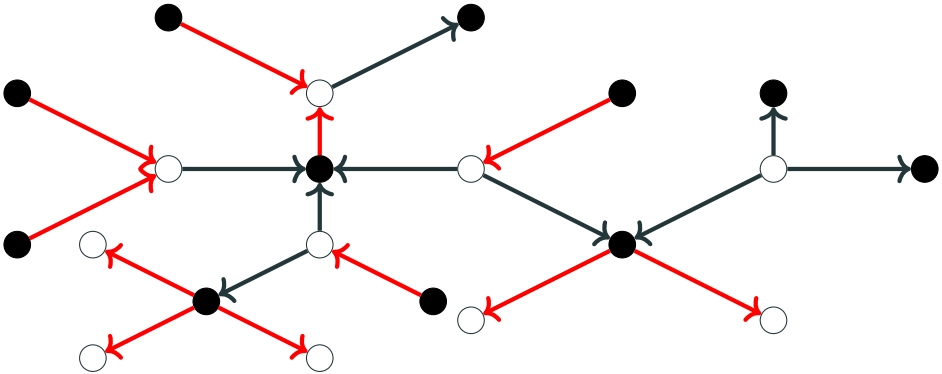
Examples

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



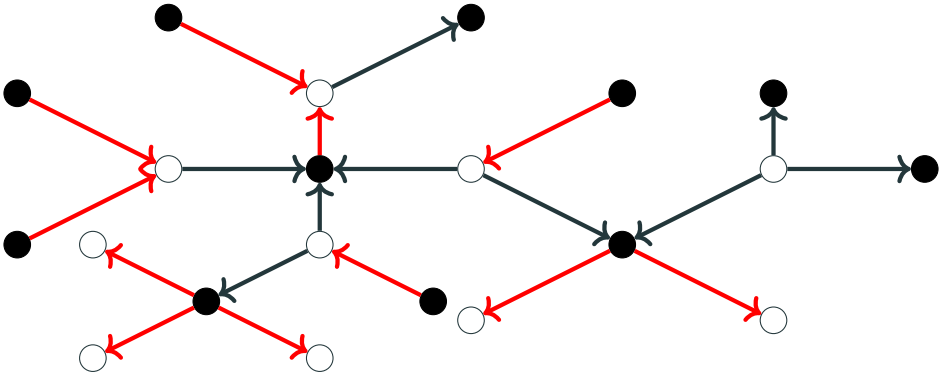
Examples

- Sinkless Orientation



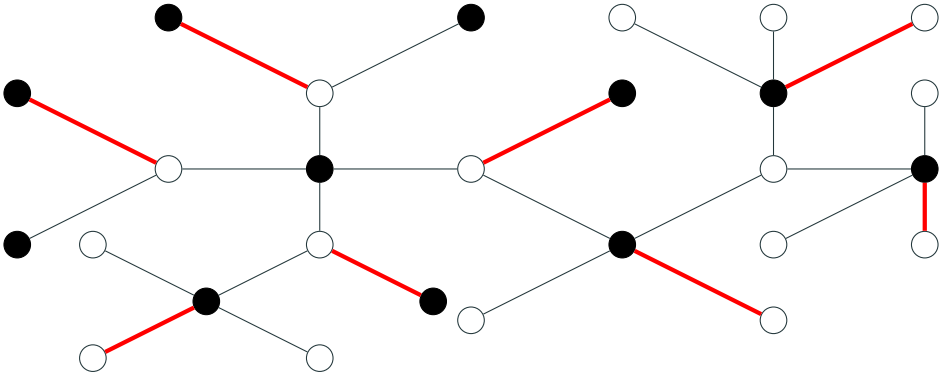
Examples

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



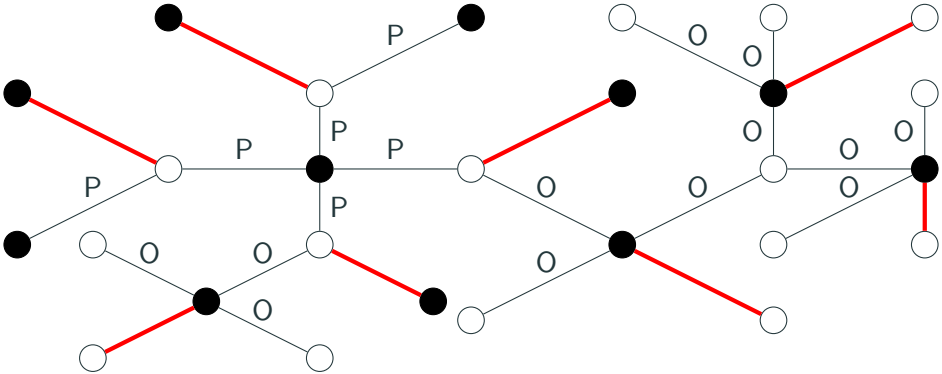
Examples

- Maximal Matching



Examples

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



Round Elimination

Principle of Round Elimination

- Problem Π_0 solved in T rounds on (d, δ)-biregular trees
- $\Rightarrow$ Construct $\Pi_1 = re(\Pi_0)$ solvable in $T - 1$ rounds on (δ, d)-biregular trees
- $\Pi_0 \to \Pi_1 \to \ldots \to \Pi_T = re^T(\Pi_0)$ solvable in 0 round
- Π_T not solvable with no communication $\Rightarrow$ Π_0 not solvable in T rounds

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

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

Weak 3-labeling

Π_0 on $(3,2)$-biregular trees :

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

Π_1 on $(2,3)$-biregular trees

Weak 3-labeling

Π_0 on $(3,2)$-biregular trees :

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

Π_1 on $(2,3)$-biregular trees :

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

Weak 3-labeling

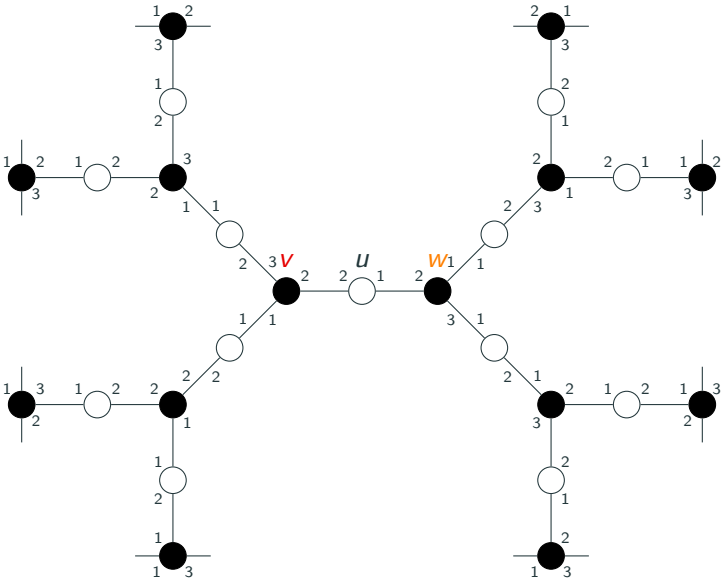
Π_0 on $(3,2)$-biregular trees :

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

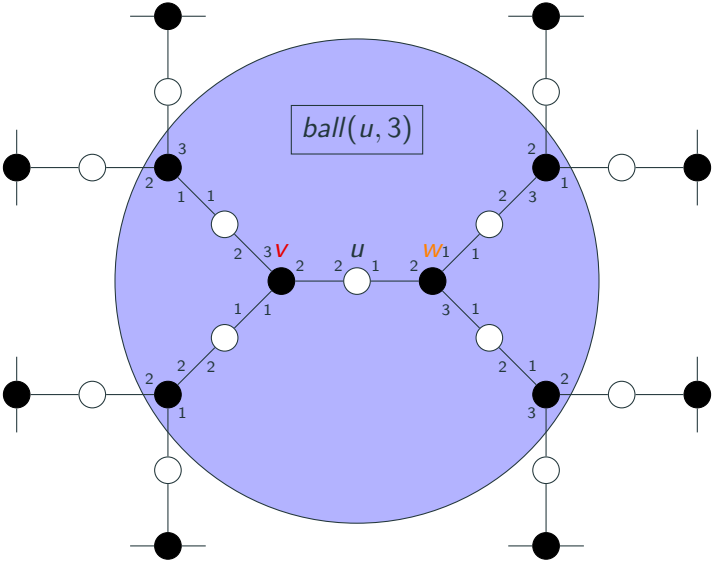
Π_1 on $(2,3)$-biregular trees :

- $\Sigma_1 = \{\{1\}, \{2\}, \{3\}\}$
- $A_1 = \{[\{1\},\{1\}], [\{2\},\{2\}], [\{3\},\{3\}]\}$
- $P_1 = \{[\{1\},\{1\},\{2\}], [\{1\},\{1\},\{3\}], [\{1\},\{2\},\{2\}], [\{1\},\{2\},\{3\}], [\{1\},\{3\},\{3\}],$
  $[\{2\},\{2\},\{3\}], [\{2\},\{3\},\{3\}]\}$

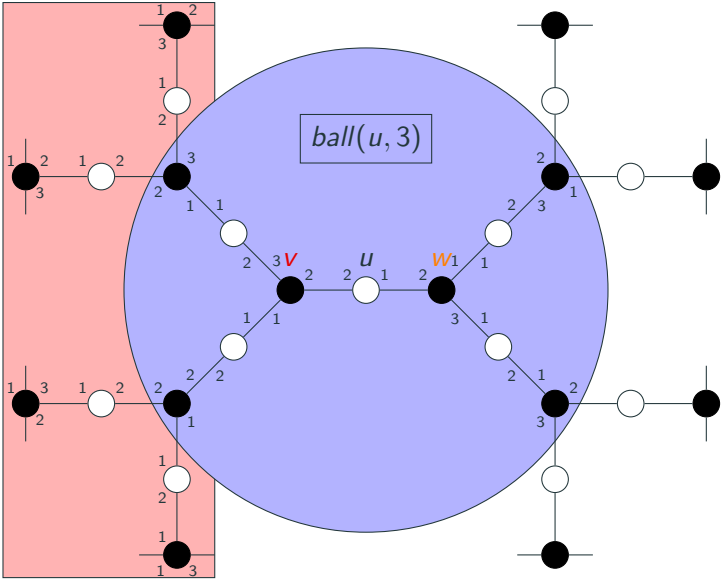
Neighbourhood Simulation



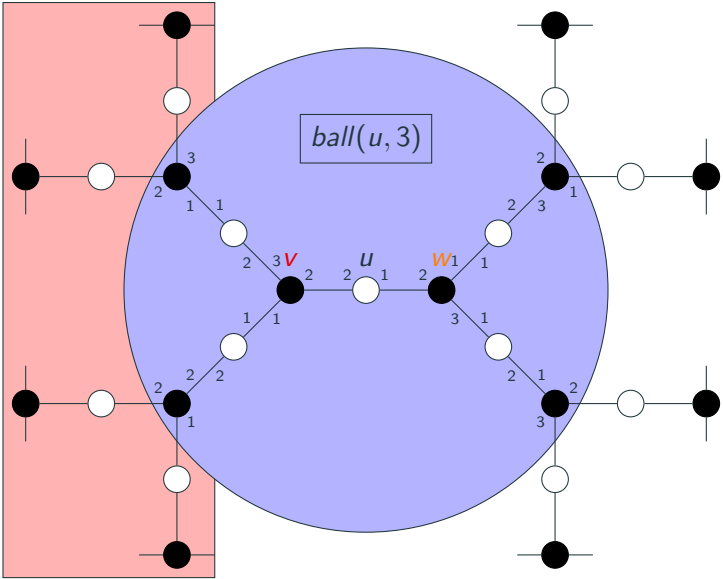
Neighbourhood Simulation



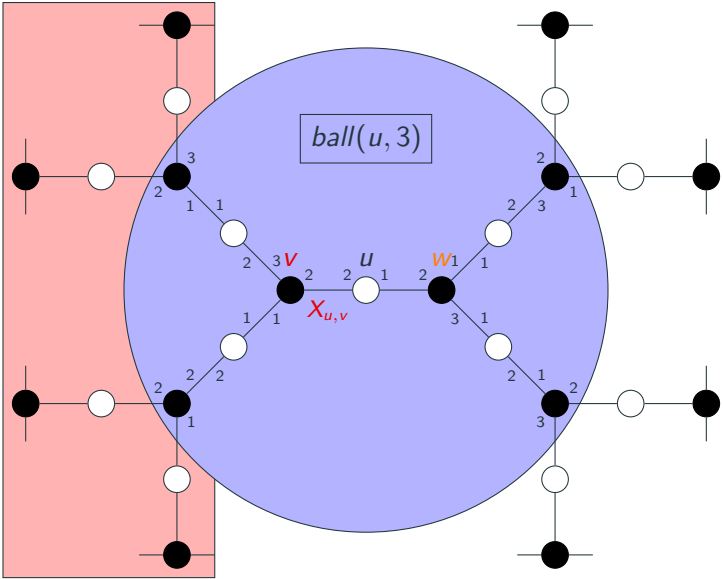
Neighbourhood Simulation



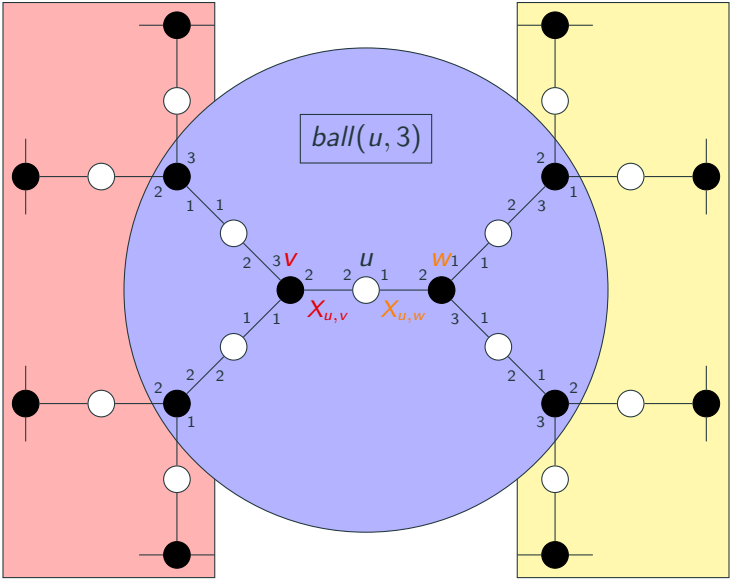
Neighbourhood Simulation



Neighbourhood Simulation



Neighbourhood Simulation



Algorithm of Π_1

$\mathcal{A}_0$ solves Π_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 :

- $\forall x_{v_1} \in X_{u,v_1} \dots x_{v_\delta} \in X_{u,\delta}$, their exists $ball(u,T)$ where $\mathcal{A}_0$ produces $[x_{v_1}, \dots, x_{v_\delta}]$
  $\Rightarrow [x_{v_1}, \dots, x_{v_\delta}] \in P_0 \Rightarrow [X_{u,v_1}, \dots, X_{u,v_\delta}] \in A_1$.
- For any Black node v, for any $ball(v,T)$ and $u \in N(v)$, $X_{u,v}$ contains $\mathcal{A}_0(u,v)$
  $\Rightarrow [X_{u_1,v}, \dots, X_{v_d,v}] \in P_1$.

Back to weak 3-labeling

- Π_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

- Π_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

- Π_1 on $(2,3)$-biregular trees :
  - $\Sigma_1 = \{1, 2, 3\}$
  - $A_1 = \{[1,1], [2,2], [3,3]\}$
  - $P_1 = \{[1,1,2], [1,1,3], [1,2,2], [1,2,3], [1,3,3], [2,2,3], [2,3,3]\}$
- Π_1 cannot be solved in 0 rounds

Back to weak 3-labeling

- Π_1 on $(2,3)$-biregular trees :
  - $\Sigma_1 = \{1, 2, 3\}$
  - $A_1 = \{[1,1], [2,2], [3,3]\}$
  - $P_1 = \{[1,1,2], [1,1,3], [1,2,2], [1,2,3], [1,3,3], [2,2,3], [2,3,3]\}$
- Π_1 cannot be solved in 0 rounds $\Rightarrow \Pi_0$ needs at least 2 rounds.
- Π_2 on $(3,2)$-biregular trees

Back to weak 3-labeling

- Π_1 on $(2,3)$-biregular trees :
  - $\Sigma_1 = \{1,2,3\}$
  - $A_1 = \{[1,1],[2,2],[3,3]\}$
  - $P_1 = \{[1,1,2],[1,1,3],[1,2,2],[1,2,3],[1,3,3],\ [2,2,3],[2,3,3]\}$
- Π_1 cannot be solved in 0 rounds $\Rightarrow \Pi_0$ needs at least 2 rounds.
- Π_2 on $(3,2)$-biregular trees :
  - $\Sigma_2 = \{\{1\},\{2\},\{3\},\{1,2\},\{1,3\},\{2,3\},\{1,2,3\}\}$
  - $A_2 = \{[\{1,2,3\},\{1,2\},\{3\}],[\{1,2,3\},\{1,3\},\{2\}],[\{1,2,3\},\{2,3\},\{1\}],$
    $[\{1,2\},\{1,3\},\{2,3\}]\}$
  - $P_2 = \{[X,Y] | X \cap Y \neq \emptyset\}$

Back to weak 3-labeling

- Π_1 on $(2,3)$-biregular trees :
  - $\Sigma_1 = \{1,2,3\}$
  - $A_1 = \{[1,1],[2,2],[3,3]\}$
  - $P_1 = \{[1,1,2],[1,1,3],[1,2,2],[1,2,3],[1,3,3],\ [2,2,3],[2,3,3]\}$
- Π_1 cannot be solved in 0 rounds $\Rightarrow \Pi_0$ needs at least 2 rounds.
- Π_2 on $(3,2)$-biregular trees :
  - $\Sigma_2 = \{\{1\},\{2\},\{3\},\{1,2\},\{1,3\},\{2,3\},\{1,2,3\}\}$
  - $A_2 = \{[\{1,2,3\},\{1,2\},\{3\}],[\{1,2,3\},\{1,3\},\{2\}],[\{1,2,3\},\{2,3\},\{1\}],$
    $[\{1,2\},\{1,3\},\{2,3\}]\}$
  - $P_2 = \{[X,Y] | X \cap Y \neq \emptyset\}$
- Π_2 can be solved in 0 round.

Sinkless Orientation

Sinkless Orientation on Paths

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

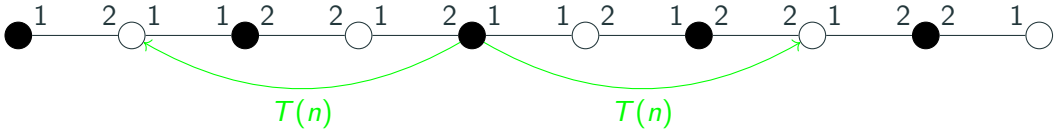
Brandt *et. al* **(2016)**
Sinkless Orientation cannot be solved in $o(n)$ on paths.

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

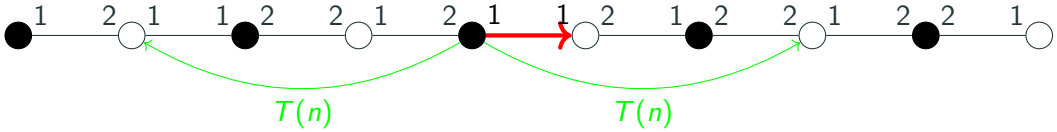


Sinkless Orientation on Paths

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

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

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

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.

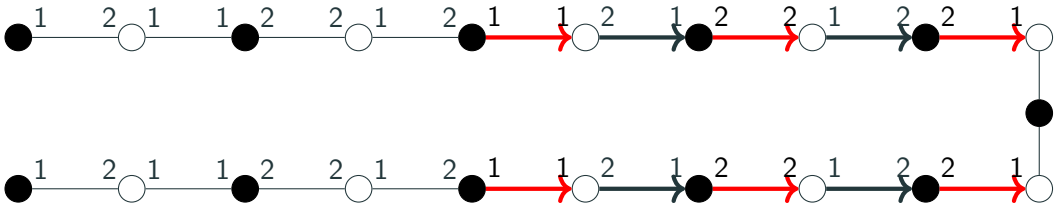
Suppose their exists an algorithm in time $T(n) \leq (n-5)/4$:

Sinkless Orientation on Paths

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

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

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

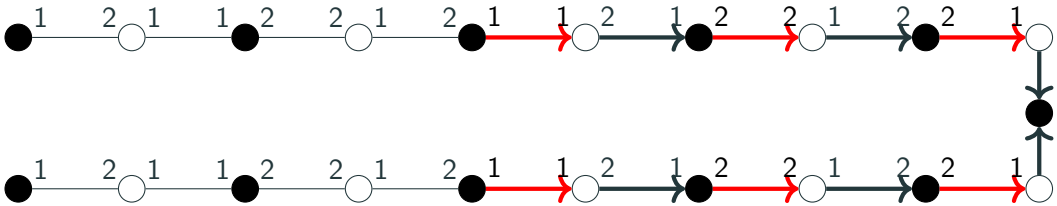


Sinkless Orientation on Paths

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

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 Trees

Π :

- Nodes of degree $d \geq 3$ need an outgoing edge
- Nodes of degree ≤ 2 have no restriction

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

- Nodes of degree ≤ 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 ≤ 2

Sinkless Orientation Speed Up

Π_0 :

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

Π_1

Sinkless Orientation Speed Up

Π_0 :

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

Π_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

Π_0 :

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

Π_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

Π_1 :

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

Π_2

Sinkless Orientation Speed Up

Π_1 :

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

Π_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

- $\forall i \geq 1$, $\Pi_i = \Pi_1$
- $\mathcal{A}$ solves Π_0 in T rounds $\Rightarrow$ Π_1 can be solved in 0 round

Sinkless Orientation cannot be solved ? !

Sinkless Orientation Lower Bound

- $\forall i \geq 1$, $\Pi_i = \Pi_1$
- $\mathcal{A}$ solves Π_0 in T rounds $\Rightarrow$ Π_1 can be solved in 0 round

Sinkless Orientation cannot be solved ? !

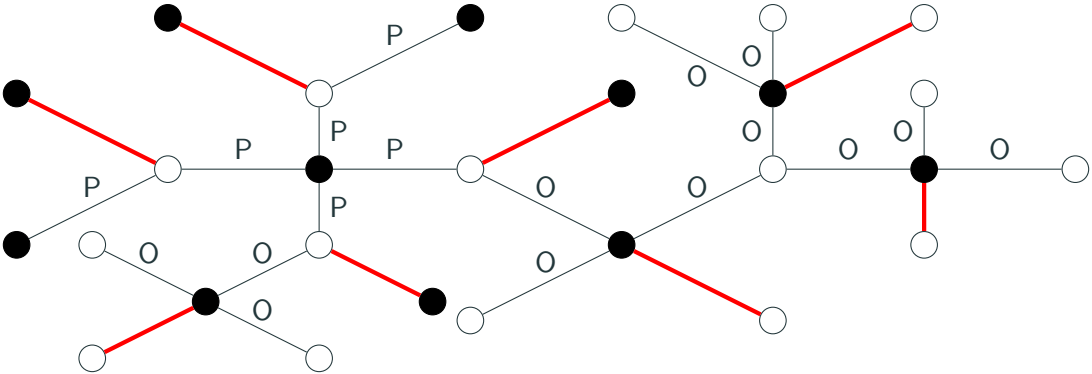
- A T round algorithm with no leafs $\Rightarrow 3 \times 2^T$ nodes
- With $T(n) = \Omega(\log n)$, we cannot do the speed up

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

Maximal Matching

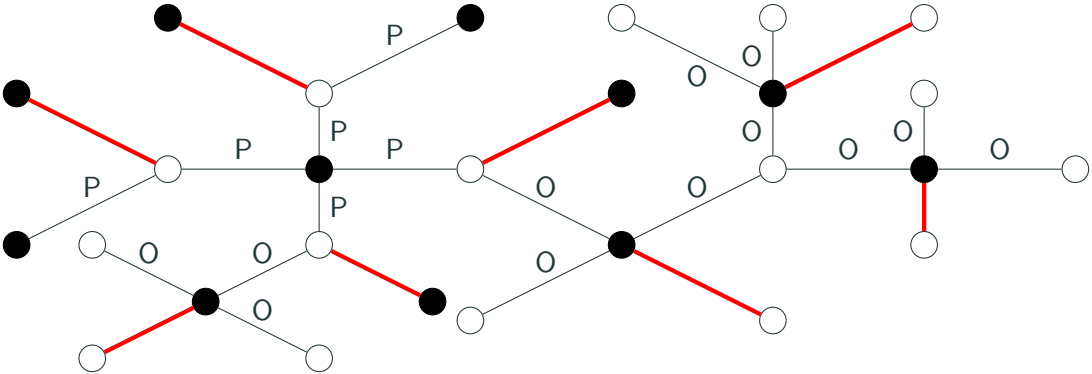
Maximal Matching in Δ-regular graphs

- Maximal Matching, $\Sigma = \{\mathsf{M}, \mathsf{O}, \mathsf{P}\}$
- $A_0 = \left(\mathsf{M}\mathsf{O}^{\Delta-1} \,\middle|\, \mathsf{P}^{\Delta}\right)$, $P_0 = \left(\mathsf{M}[\mathsf{PO}]^{\Delta-1} \,\middle|\, \mathsf{O}^{\Delta}\right)$



Maximal Matching in Δ-regular graphs

- Maximal Matching, $\Sigma = \{\mathsf{M}, \mathsf{O}, \mathsf{P}\}$
- $A_0 = \left(\mathsf{M}\mathsf{O}^{\Delta-1} \,\middle|\, \mathsf{P}^{\Delta}\right)$, $P_0 = \left(\mathsf{M}[\mathsf{PO}]^{\Delta-1} \,\middle|\, \mathsf{O}^{\Delta}\right)$



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

Bibliography

- Jukka Suomela's courses (https ://jukkasuomela.fi/da2020/da2020-09.pdf and https ://jukkasuomela.fi/da2020/da2020-10.pdf)
- Alkida Balliu, Sebastian Brandt, Juho Hirvonen, Dennis Olivetti, Mikaël Rabie, Jukka Suomela. **Lower Bounds for Maximal Matchings and Maximal Independent Sets**. In Proc. FOCS 2019.
- 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.
- Nathan Linial. **Locality in distributed graph algorithms**. SIAM Journal on Computing, 1992.