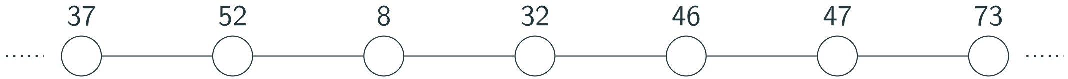
Distributed Computing
2 - Port Numbering Model

Mikaël Rabie (mikael.rabie@irif.fr - he/him)
Université Paris-Cité, IRIF

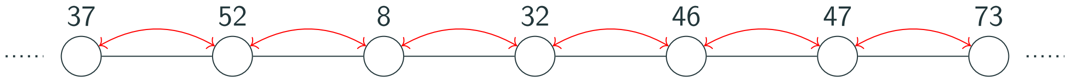
INSTITUT
DE RECHERCHE
EN INFORMATIQUE
FONDAMENTALE

Model Definition

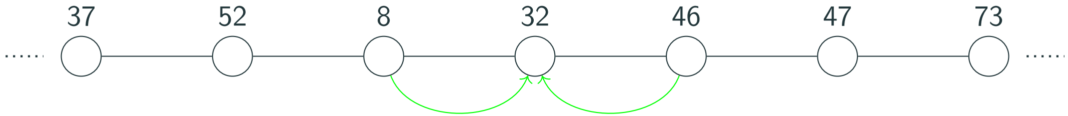
From LOCAL to Port-Numbering



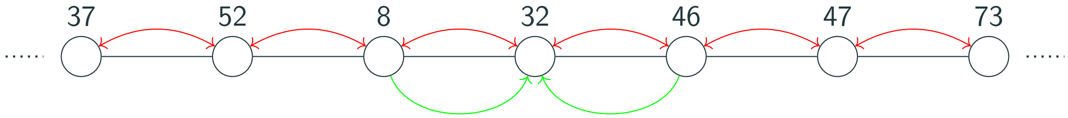
From LOCAL to Port-Numbering



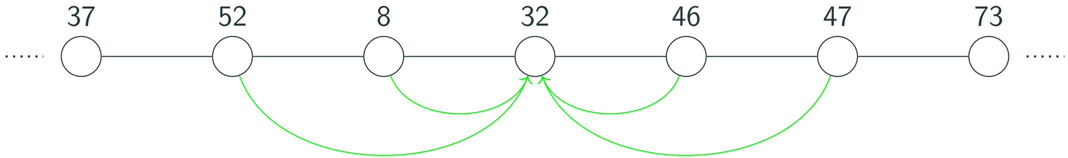
From LOCAL to Port-Numbering



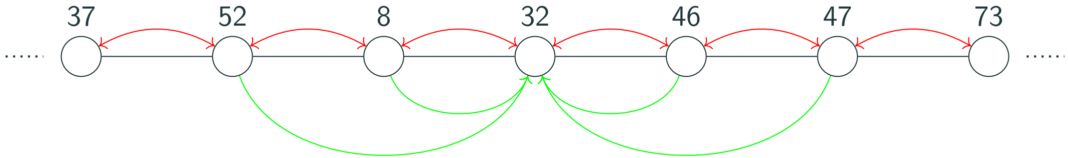
From LOCAL to Port-Numbering



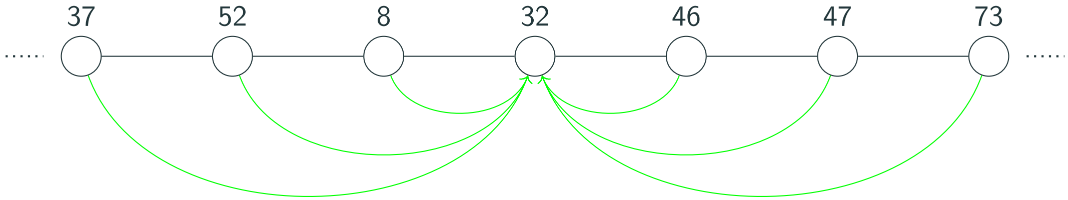
From LOCAL to Port-Numbering



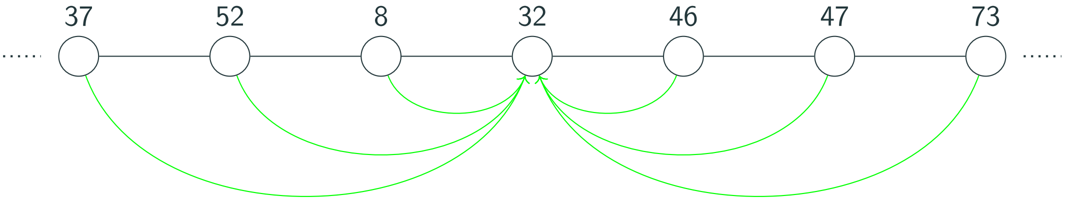
From LOCAL to Port-Numbering

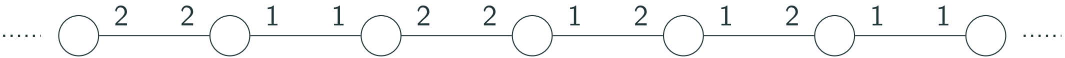


From LOCAL to Port-Numbering



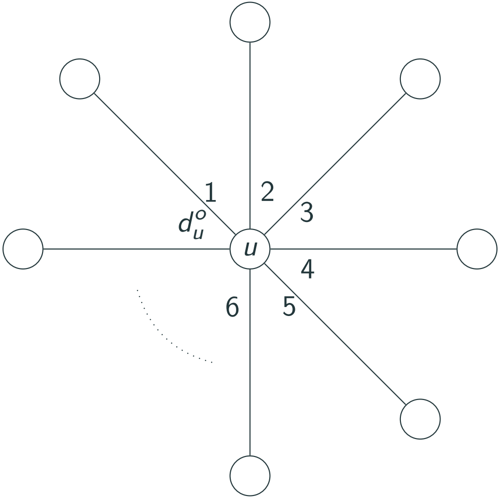
From LOCAL to Port-Numbering





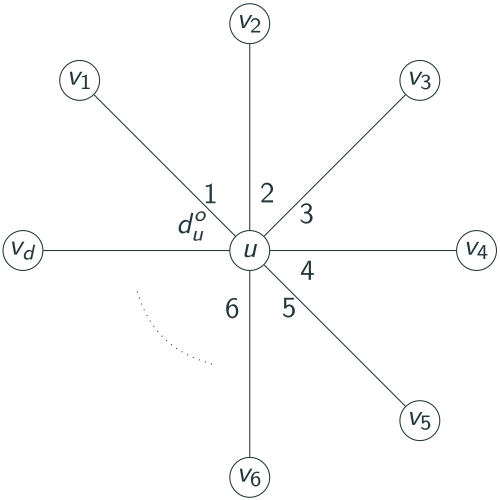
Communication

0 communication round



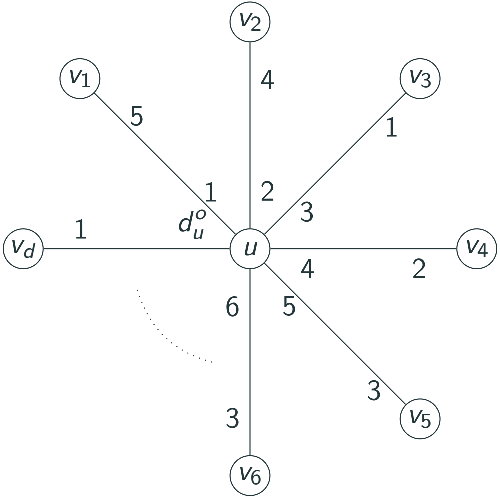
Communication

0 communication round

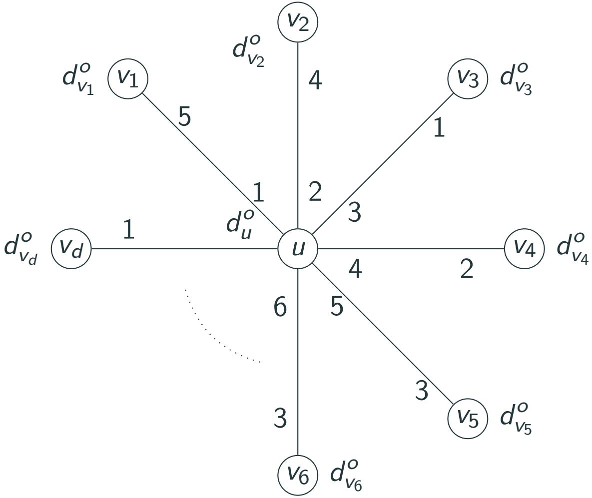


Communication

1 communication round



1 communication round



Formal Definition - PN

A **Port-Numbered Network** is a triple (V, P, p) :

- V is the set of nodes
- P is the set of ports. $P \subseteq V \times \mathbb{N}$
- $p : P \to P$ is the function that connects ports.
  $p(u, i)$ gives the node v to which u is connected, and to which port of v u is connected.
  We have $p(p(u, i)) = (u, i)$.

  Edges can be deduced from p.

Formal Definition - Algorithm

- **States** : S (not necessarily finite)
- **Input** : I (I might be a singleton if we do not provide inputs)
- **Output** : $O \subseteq S$
  - Can be directly vertex output.
  - Can be edge output (gives a mapping of outputs to each port of the vertex).
- **Messages** : M
- **Execution Functions** (depending on degree d of the node) :
  - $init_d : I \to S$
  - $send_d : S \to M^d$ (sends message i through port i)
  - $receive_d : S \times M^d \to S$ (receives the d messages from the ports and updates the state)
    $\forall o \in O$, we have $receive_d(o, m) = o$.
- **Algorithm** : $(I, S, O, M, (init_d)_{d\in\mathbb{N}}, (send_d)_{d\in\mathbb{N}}, (receive_d)_{d\in\mathbb{N}})$

Formal Definition - Distributed Graph Problems

From our network triple (V, P, p) :

- **Initialization** : $f : V \to I$ (pre-labelling of the nodes)
- **Configuration** : $x : V \to S$
- **Transition** : $x_k \to x_{k+1}$
  For each $v \in V$ of degree d, if $p(v, i) = (u, j)$,
  m_i corresponds to the jth element of $send_{d'}(x_k(u))$,
  we have $x_{k+1}(v) = receive_d(x_k(v), m_1, \ldots, m_d)$.
- **Execution** : $x_0, x_1, \ldots$ such that $x_0(v) = init_d(f(v))$ and for each k, $x_k \to x_{k+1}$
- **End** of execution : first k such that $\forall v \in V$, $x_k(v) \in O$

Formal Definition - Problem Solving

- **Distributed Graph Problems** : Π such as, for a PN Network $N = (V, P, p)$,
  $\Pi(N)$ is the set of accepted labellings.
- **Solution** : $f : V \to O$ such as $f \in \Pi(N)$
- Algorithm $\mathcal{A}$ **Solves** Π from input Π' on N in time k if
  For any $f_{init} \in \Pi'$, we have a sequence $x_0, \ldots, x'_k$
  starting from f_{init} with $k' \leq k$ and $x'_k \in \Pi$.
- Algorithm $\mathcal{A}$ **Solves** Π from input Π' on family $\mathfrak{F}$ of graphs in time $T : \mathbb{N} \to \mathbb{N}$ if
  For any PN Network $N = (V, P, p)$ representing a graph $G \in \mathfrak{F}$,
  $\mathcal{A}$ solves Π from input Π' on N in time $T(|V|)$

3 Algorithms

3-coloring a path

- $\mathfrak{F}$: Paths.
- Π : 3-colored paths.
- Π' : proper coloring of the path.

3-coloring a path

- $\mathfrak{F}$: Paths.
- Π : 3-colored paths.
- Π' : proper coloring of the path.

3-coloring a path

- $\mathfrak{F}$: Paths.
- Π : 3-colored paths.
- Π' : proper coloring of the path.

Idea : Local maximums choose the smallest possible color.

3-coloring a path

- $\mathfrak{F}$: Paths.
- Π : 3-colored paths.
- Π' : proper coloring of the path.

Idea : Local maximums choose the smallest possible color.

3-coloring a path

- $\mathfrak{F}$: Paths.
- Π : 3-colored paths.
- Π' : proper coloring of the path.

Idea : Local maximums choose the smallest possible color.

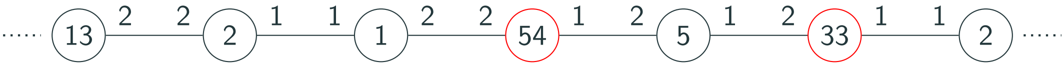
3-coloring a path

- $\mathfrak{F}$: Paths.
- Π : 3-colored paths.
- Π' : proper coloring of the path.

Idea : Local maximums choose the smallest possible color.

3-coloring a path

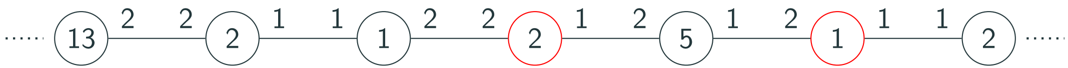
- $\mathfrak{F}$: Paths.
- Π : 3-colored paths.
- Π' : proper coloring of the path.



Idea : Local maximums choose the smallest possible color.

3-coloring a path

- $\mathfrak{F}$: Paths.
- Π : 3-colored paths.
- Π' : proper coloring of the path.



Idea : Local maximums choose the smallest possible color.

3-coloring a path

- $\mathfrak{F}$: Paths.
- Π : 3-colored paths.
- Π' : proper coloring of the path.

Idea : Local maximums choose the smallest possible color.

3-coloring a path

- $\mathfrak{F}$: Paths.
- Π : 3-colored paths.
- Π' : proper coloring of the path.

Idea : Local maximums choose the smallest possible color.

3-coloring a path

- $\mathfrak{F}$: Paths.
- Π : 3-colored paths.
- Π' : proper coloring of the path.

Idea : Local maximums choose the smallest possible color.

3-coloring Algorithm

- $S = \mathbb{N}^+$
- $I = \mathbb{N}^+$
- $O = \{1, 2, 3\}$
- $M = \mathbb{N}^+$
- $init_d(x) = x$, $\forall d \leq 2$
- $send_d(x) = x^d$, $\forall d \leq 2$
- $receive_d(x, Y) = \min(\mathbb{N} \setminus (Y \cup \{x\}))$ if $x \geq 4$ and $x > Y$, $receive_d(x, Y) = x$ otherwise.

- Complexity ?

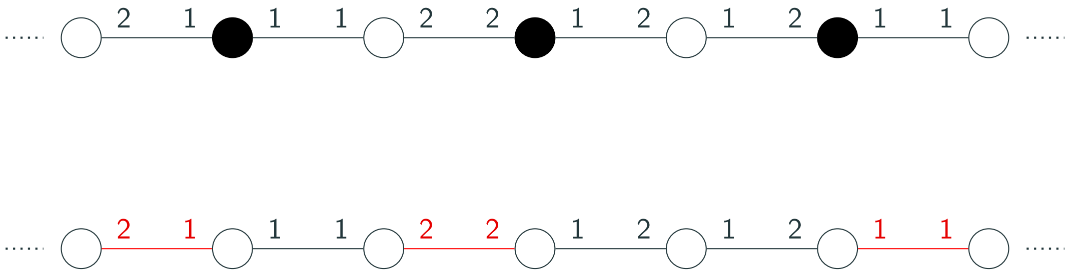
3-coloring Algorithm

- $S = \mathbb{N}^+$
- $I = \mathbb{N}^+$
- $O = \{1, 2, 3\}$
- $M = \mathbb{N}^+$
- $init_d(x) = x$, $\forall d \leq 2$
- $send_d(x) = x^d$, $\forall d \leq 2$
- $receive_d(x, Y) = \min(\mathbb{N} \setminus (Y \cup \{x\}))$ if $x \geq 4$ and $x > Y$, $receive_d(x, Y) = x$ otherwise.

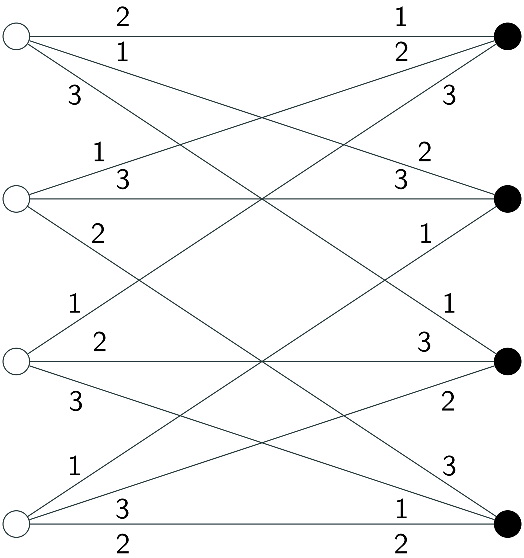
- Complexity : $O(n)$

Maximal Matching on Bipartite Graphs

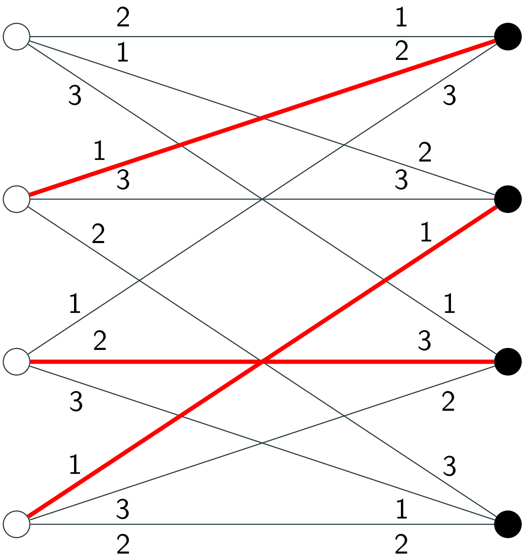
- $\mathfrak{F}$: Bipartite Graphs.
- Π : Maximal Matching.
- Π' : Black/White-coloring.



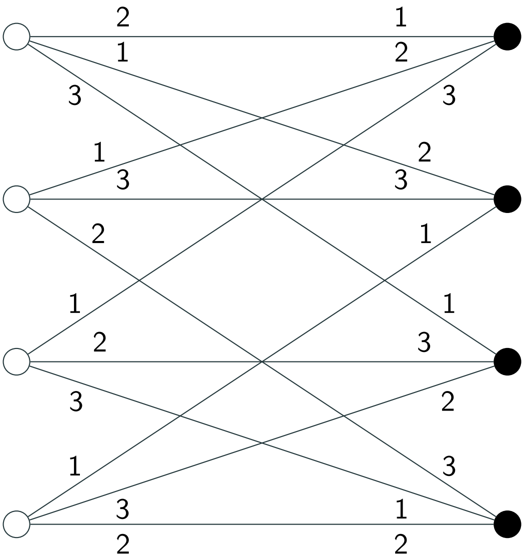
Maximal Matching Algorithm



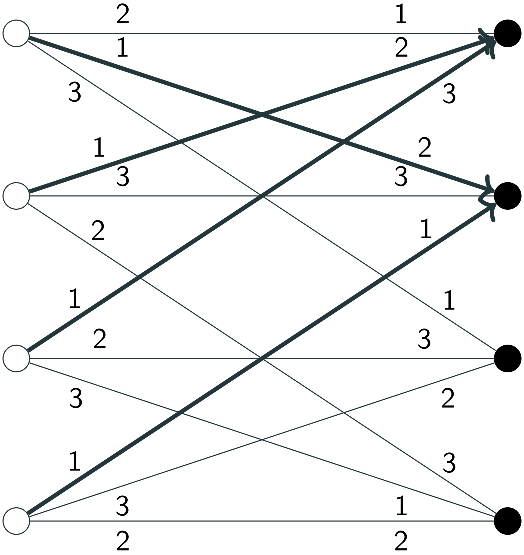
Maximal Matching Algorithm



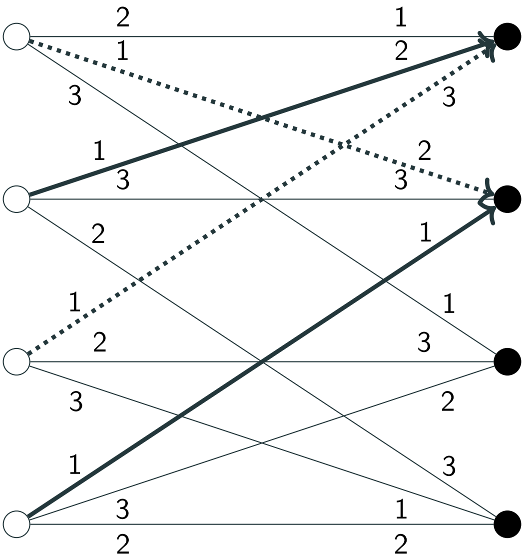
Maximal Matching Algorithm



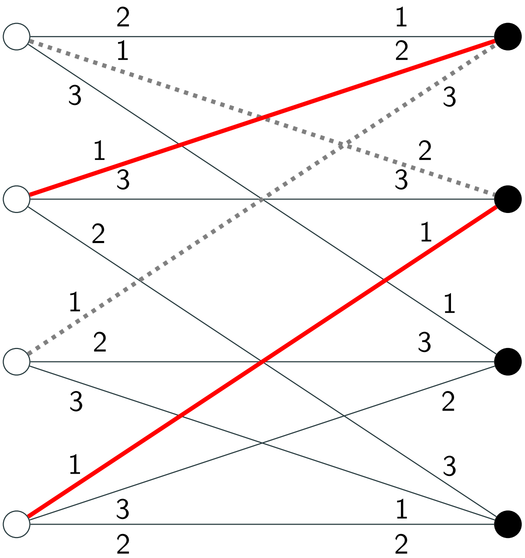
Maximal Matching Algorithm



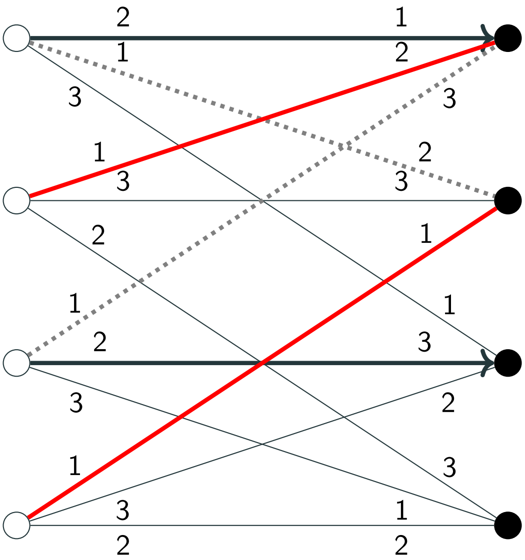
Maximal Matching Algorithm



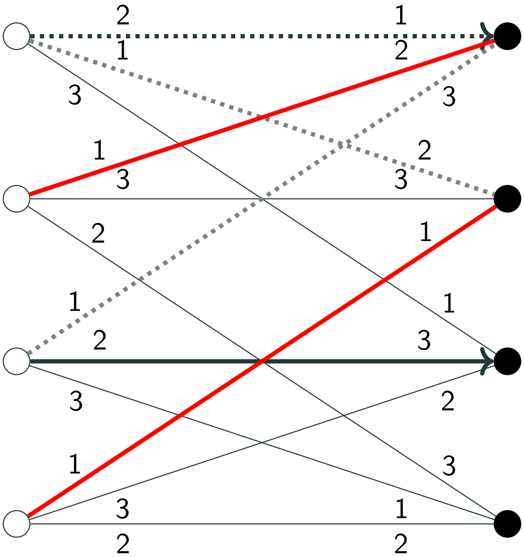
Maximal Matching Algorithm



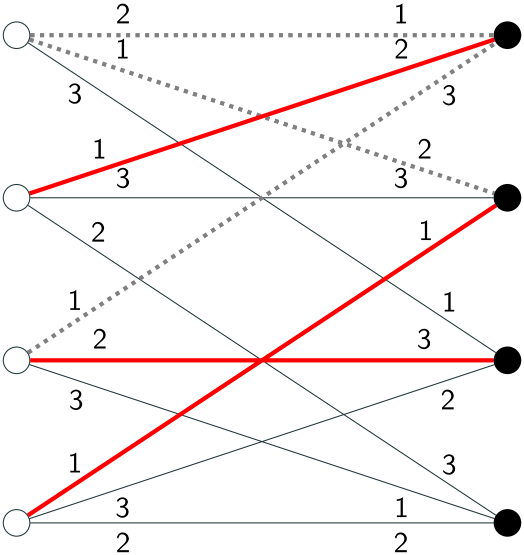
Maximal Matching Algorithm



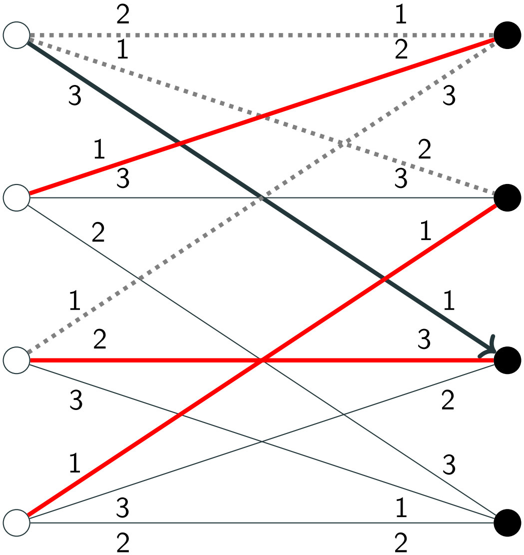
Maximal Matching Algorithm



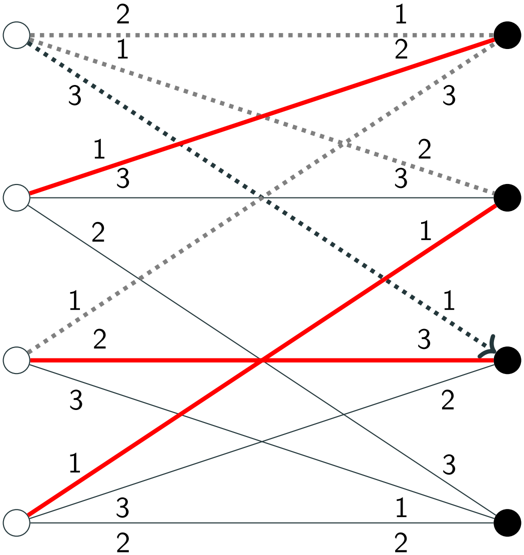
Maximal Matching Algorithm



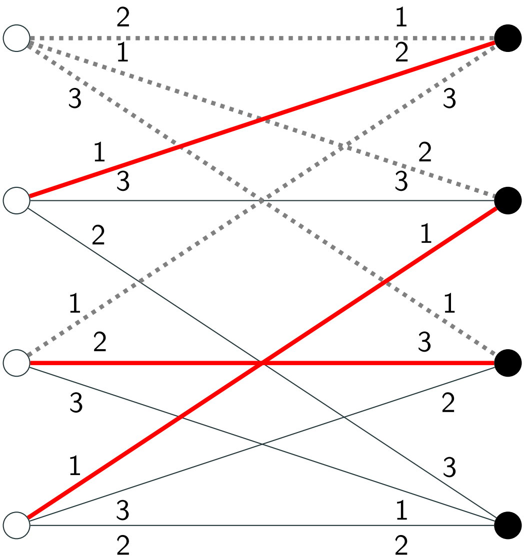
Maximal Matching Algorithm



Maximal Matching Algorithm



Maximal Matching Algorithm

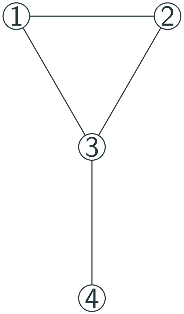


Vertex Cover 3-approximation

Vertex Cover : $C \subseteq V$ such that each edge $e = \{u, v\} \in V^2$ has at least one endpoint in C.

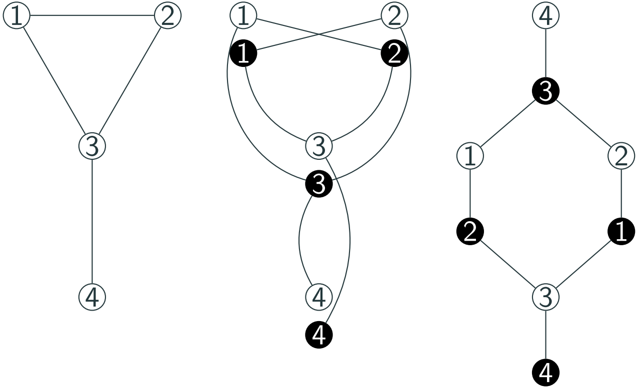
- $\mathfrak{F}$: Any graphs.
- Π : 3-approximation of Vertex-Cover.
- Π' : Nothing.

Virtual Bipartite Black/White Graph



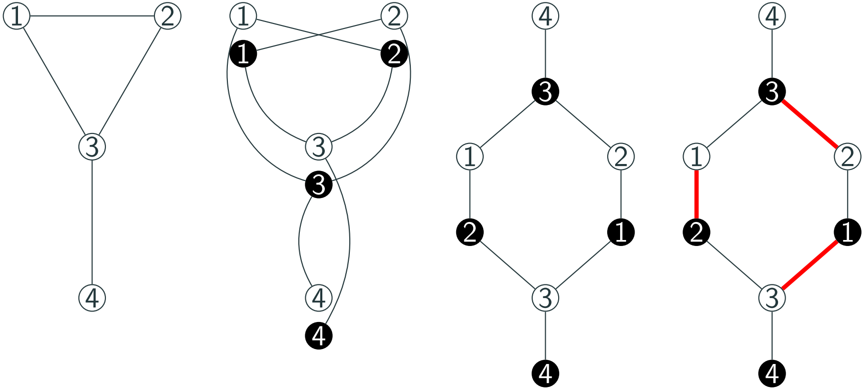
Graph

Virtual Bipartite Black/White Graph



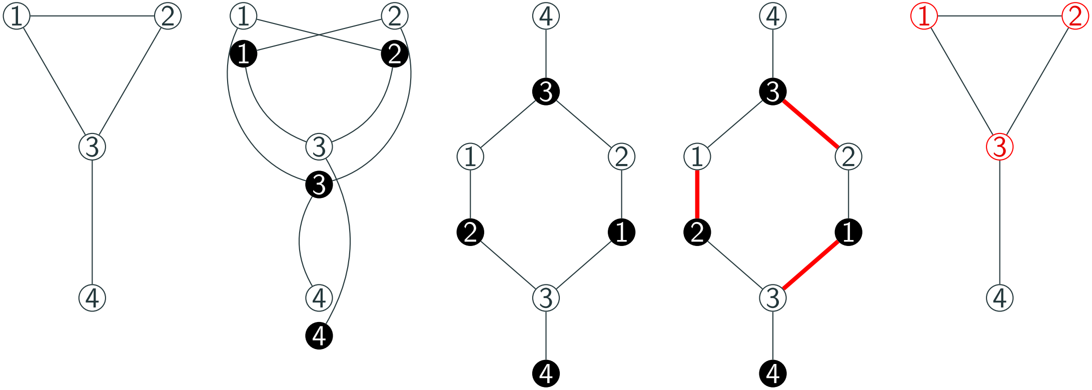
Graph ⇒ Duplicate Nodes

Virtual Bipartite Black/White Graph



Graph ⇒ Duplicate Nodes ⇒ Find a Maximal Matching

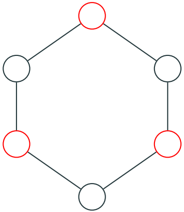
Virtual Bipartite Black/White Graph

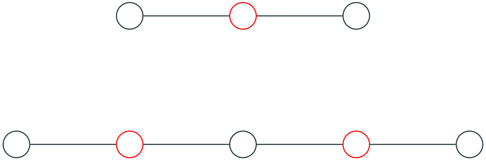


Graph ⇒ Duplicate Nodes ⇒ Find a Maximal Matching ⇒ Take Matched Nodes

Analysis

- C is a Vertex Cover :
  - In the virtual graph, each edge touches an edge of the Maximal Matching
  - In the virtual graph, each edge touches a node in C
  - In the graph, each edge touches a node in C
- It is a 3-approximation of any Vertex Cover C^* :
  - The Maximal Matching in the virtual graph forms cycles and paths in the graph
  - C^* is a Vertex Cover of those cycles and paths
  - Any Vertex Cover of a cycle uses at least half of the nodes
  - Any Vertex Cover of a path uses at least a third of the nodes





Impossibility

k-Coloring a path

- $\mathfrak{F}$: Paths.
- Π : k-colored paths.
- Π' : Nothing.

k-Coloring a path

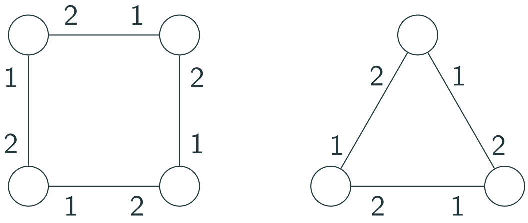
- $\mathfrak{F}$: Paths.
- Π : k-colored paths.
- Π' : Nothing.

Maximal Matching a cycle

- $\mathfrak{F}$: Cycles.
- Π : Maximal Matching.
- Π' : Nothing.

Maximal Matching a cycle

- $\mathfrak{F}$: Cycles.
- Π : Maximal Matching.
- Π' : Nothing.



Break Symmetry

Tools to break symmetry :

- Identifiers (LOCAL Model)
- Randomness (Simulation of LOCAL model)
- Inputs
- ...

Exercise

Design an algorithm for the Maximal Matching in a k-colored graph.

Bibliography

- Jukka Suomela's courses (https ://jukkasuomela.fi/da2020/da2020-03.pdf)
- Dana Angluin. **Local and global properties in networks of processors**. In Proc. 12th Annual ACM Symposium on Theory of Computing (STOC 1980), 1980.
- Michal Hanckowiak, Michal Karonski, and Alessandro Panconesi. **On the distributed complexity of computing maximal matchings**. In Proc. 9th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 1998), 1998.
- Valentin Polishchuk and Jukka Suomela. **A simple local 3-approximation algorithm for vertex cover**. Information Processing Letters, 109(12) :642–645, 2009.