Distributed Computing
9 - Port Numbering Model

Mikaël Rabie
Université de Paris, IRIF
Model Definition
From LOCAL to Port-Numbering

...... 37 52 8 32 46 47 73 ......
From LOCAL to Port-Numbering
From LOCAL to Port-Numbering

…… 37  52  8  32  46  47  73  ……
From LOCAL to Port-Numbering
From LOCAL to Port-Numbering
From LOCAL to Port-Numbering
From LOCAL to Port-Numbering
Communication

0 communication round
0 communication round
1 communication round
1 communication round
A Port-Numbered Network is a triple \((V, P, p)\):

- \(V\) is the set of nodes
- \(P\) is the set of ports. \(P \subseteq V \times \mathbb{N}\)
- \(p : P \rightarrow P\) 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):
  - $init_d : I \rightarrow S$
  - $send_d : S \rightarrow M^d$ (sends message $i$ through port $i$)
  - $receive_d : S \times M^d \rightarrow M$ (receives the $d$ messages from the ports and updates the state)
    \[ \forall o \in O, \text{ we have } receive_d(o, m) = o. \]
- **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**: $f : V \rightarrow I$ (pre-labelling of the nodes)
- **Configuration**: $x : V \rightarrow S$
- **Transition**: $x_k \rightarrow x_{k+1}$
  
  For each $v \in V$ of degree $d$, if $p(v, i) = (u, j)$,
  
  $m_i$ corresponds to the $j^{th}$ 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 $k$, $x_k \rightarrow x_{k+1}$

- **End** of execution: first $k$ such that $\forall v \in V, x_k(v) \in O$
Formal Definition - Problem Solving

- **Distributed Graph Problems**: \( \Pi \) such as, for a PN Network \( N = (V, P, p) \), \( \Pi(N) \) is the set of accepted labellings.

- **Solution**: \( f : V \rightarrow O \) such as \( f \in \Pi(N) \)

- Algorithm \( \mathcal{A} \) Solves \( \Pi \) from input \( \Pi' \) on \( N \) in time \( k \) if
  - For any \( f_{\text{init}} \in \Pi' \), we have a sequence \( x_0, \ldots, x_k' \)
  - starting from \( f_{\text{init}} \) with \( k' \leq k \) and \( x_k' \in \Pi \).

- Algorithm \( \mathcal{A} \) Solves \( \Pi \) from input \( \Pi' \) on family \( \mathcal{F} \) of graphs in time \( T : \mathbb{N} \rightarrow \mathbb{N} \) if
  - For any PN Network \( N = (V, P, p) \) representing a graph \( G \in \mathcal{F} \),
  - \( \mathcal{A} \) solves \( \Pi \) from input \( \Pi' \) on \( N \) in time \( T(|V|) \)
3 Algorithms
3-coloring a path

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

Idea: Local maximums choose the smallest possible color.
3-coloring a path

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

Idea: Local maximums choose the smallest possible color.
3-coloring a path

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

Idea: Local maximums choose the smallest possible color.
3-coloring a path

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

Idea: Local maximums choose the smallest possible color.
3-coloring a path

- $\mathcal{P}$: Paths.
- $\Pi$: 3-colored paths.
- $\Pi'$: proper coloring of the path.

Idea: Local maximums choose the smallest possible color.
3-coloring a path

- $\mathcal{P}$: Paths.
- $\Pi$: 3-colored paths.
- $\Pi'$: proper coloring of the path.

Idea: Local maximums choose the smallest possible color.
3-coloring a path

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

Idea: Local maximums choose the smallest possible color.
3-coloring a path

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

Idea: Local maximums choose the smallest possible color.
Idea: Local maximums choose the smallest possible color.
3-coloring a path

- \( \mathcal{F} \): Paths.
- \( \Pi \): 3-colored paths.
- \( \Pi' \): proper coloring of the path.

Idea: Local maximums choose the smallest possible color.
3-coloring a path

- $\mathcal{F}$: Paths.
- $\Pi$: 3-colored paths.
- $\Pi'$: 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 $x > Y$, otherwise $receive_d(x, Y) = x$
- Complexity?
3-coloring Algorithm

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

- Complexity: \( O(n) \)
Maximal Matching on Bipartite Graphs

- $\tilde{\mathcal{F}}$: Bipartite Graphs.
- $\Pi$: Maximal Matching.
- $\Pi'$: Black/White-coloring.
Maximal Matching Algorithm

- Computer network with port numbering
- Bipartite, 2-colored graph
- $\Delta$-regular (here $\Delta = 3$)

Output: maximal matching
Maximal Matching Algorithm

Very simple algorithm

unmatched white nodes:
send proposal to port 1
**Very simple algorithm**

**unmatched white nodes:**
send *proposal* to port 1

**black nodes:**
*accept* the first proposal you get, *reject* everything else
(break ties with port numbers)
Maximal Matching Algorithm

Very simple algorithm

unmatched white nodes: send proposal to port 1

black nodes: accept the first proposal you get, reject everything else
(break ties with port numbers)
Very simple algorithm

unmatched white nodes: send *proposal* to port 2
**Maximal Matching Algorithm**

**Very simple algorithm**

**unmatched white nodes:**
send *proposal* to port 2

**black nodes:**
*accept* the first proposal you get, *reject* everything else
(break ties with port numbers)
Maximal Matching Algorithm

Very simple algorithm

unmatched white nodes: send proposal to port 2

black nodes: accept the first proposal you get, reject everything else (break ties with port numbers)
Maximal Matching Algorithm

Very simple algorithm

unmatched white nodes: send *proposal* to port 3
**Maximal Matching Algorithm**

**Very simple algorithm**

unmatched white nodes: send *proposal* to port 3

black nodes: *accept* the first proposal you get, *reject* everything else (break ties with port numbers)
Maximal Matching Algorithm

**Very simple algorithm**

- **unmatched white nodes:** send *proposal* to port 3
- **black nodes:** *accept* the first proposal you get, *reject* everything else (break ties with port numbers)
Maximal Matching Algorithm

Very simple algorithm

Finds a maximal matching in $O(\Delta)$ communication rounds

Note: running time does not depend on $n$
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 $C$.

- $\mathcal{F}$: Any graphs.
- $\Pi$: 3-approximation of Vertex-Cover.
- $\Pi'$: Nothing.
Virtual Bipartite Black/White Graph

1 ➤ Duplicate Nodes ➤ Find a Maximal Matching ➤ Take Matched Nodes

Graph
Virtual Bipartite Black/White Graph

Graph $\Rightarrow$ Duplicate Nodes
Virtual Bipartite Black/White Graph

Graph $\Rightarrow$ Duplicate Nodes $\Rightarrow$ Find a Maximal Matching
Virtual Bipartite Black/White Graph

Graph $\Rightarrow$ Duplicate Nodes $\Rightarrow$ Find a Maximal Matching $\Rightarrow$ 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

- $\mathcal{G}$: Paths.
- $\Pi$: $k$-colored paths.
- $\Pi'$: Nothing.
k-Coloring a path

- $\mathcal{F}$: Paths.
- $\Pi$: $k$-colored paths.
- $\Pi'$: Nothing.
Maximal Matching a cycle

- \( \tilde{\mathcal{F}} \) : Cycles.
- \( \Pi \) : Maximal Matching.
- \( \Pi' \) : Nothing.
Maximal Matching a cycle

- $\mathcal{F}$: Cycles.
- $\Pi$: Maximal Matching.
- $\Pi'$: Nothing.
Tools to break symmetry:

- Identifiers (LOCAL Model)
- Randomness (Simulation of LOCAL model)
- Inputs
- ...
• Jukka Suomela’s courses (https://jukkanoumela.fi/da2020/da2020-03.pdf)