# Lecture 12

# The Port Numbering Model

Today we're looking at a particularly weak model of computation: deterministic algorithms in the message passing model *without node identifiers*. This means that we have to specify whether (and how) nodes can tell each other apart: while nodes are anonymous, there is the question whether they can recognize if two messages originate from the same neighbor or not. We assume that they can, and model this by *port numbers*. A node of degree $\delta_v$ has a bijection $p$ from $1, \ldots, \delta_v$ to its edges. Whenever it receives a message, it "sees" the port on which it arrives, and thus knows it was sent by the node incident to the respective edge. Likewise, whenever it sends a message, it specifies the port on which it sends the message, and the other endpoint of the respective edge is the receiver of the message.

We care neither about message size nor the number of messages sent. Hence we can run an $\alpha$-synchronizer, which in turn means that it's fine to assume that the system is synchronous to begin with. Altogether, this is called the *port numbering model*. Let's wrap up how it works:

- The network is described by a simple connected graph $G = (V, E)$.
- For each node $v \in V$, there is a bijection $p_v \colon \{w \in V \mid \{v, w\} \in E\} \to 1, \ldots, \delta_v$.
- The system operates in synchronous rounds. In each round, each node $v$

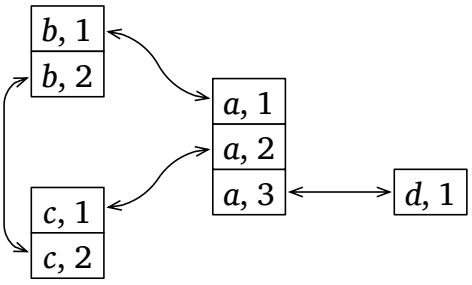


Figure 12.1: A port numbering network consisting of nodes $a, b, c, d$.

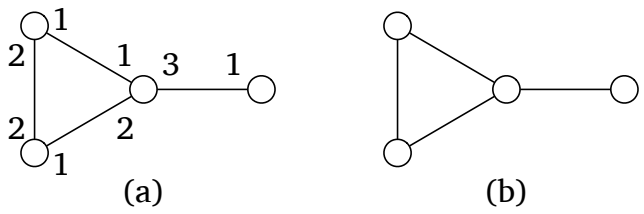


Figure 12.2: The same network represented as a labeled graph (a) and without indicating port numbers (b).

1. performs arbitrary (finite) local computations,
2. sends a message to each port $1, \ldots, \delta_v$ (sending none is ok, too), and
3. receives, for each neighbor $w$, on port $p_v(w)$ the message $w$ sent to its own port $p_w(v)$.

- As usual, nodes may be given additional inputs, and should eventually terminate and return a value so that all outputs together describe a solution of the problem at hand.

**Remarks:**

- Randomization is out of the question this time, simply because it permits to generate unique identifiers with high probability.
- We study this model primarily for understanding the relative power of the models.
- Lower bounds in the port numbering model can also be a good starting point for ones in stronger models. They are usually easier to show, and in some cases it's possible to "lift" the result to a more powerful one by using simulation (i.e., showing that at least for the considered problem the "stronger" model is not actually stronger).

## 12.1 What we can't do

Having no identifiers is quite the bummer. We cannot break symmetry, so we can basically do nothing at all.[1]

**Theorem 12.1.** *In general, it is impossible to break symmetry in the port numbering model. In particular, one cannot always*

- *solve leader election,*
- *find proper vertex or edge colorings,*
- *determine a non-empty independent set,*
- *determine a dominating set that does not contain all nodes,*

[1] Or *do* we?

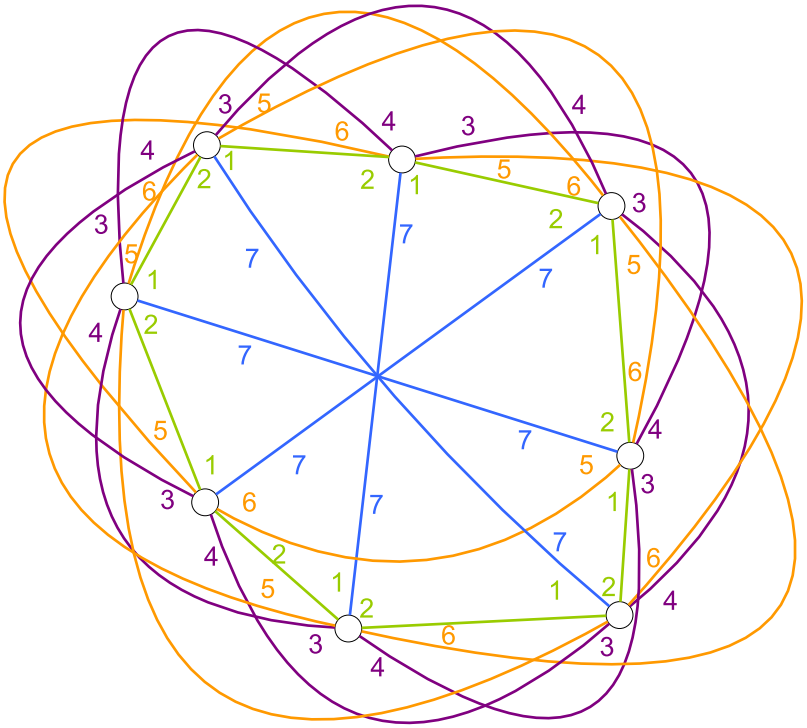

Figure 12.3: A symmetric port numbering network with 8 nodes of degree 7.

- *find a non-empty matching, or*
- *compute a minimum vertex cover.*

*This holds also when restricting to graphs of uniform degree* $\Delta$*, for any* $1 < \Delta \in \mathbb{N}$ *and* $n > \Delta$*.*

*Proof.* Assume that initially, all nodes are in the same state and all nodes have the same degree $\Delta$. Then each node will send the same message to a given port $i \in \{1, \ldots, \Delta\}$. Thus, if each port $i$ connects to the other endpoint of the corresponding edge with the same port number $j(i) \in \{1, \ldots, \Delta\}$, each node receives the same message on port $j(i)$, implying that all nodes are in the same state at the end of the first round. By induction, this shows that symmetry cannot be broken and all the above statements readily follow.

Hence, all we need to do is to show that, for any fixed $\Delta > 1$ and any $n > \Delta$, connected, simple port-numbered graphs with the following property exist: there is a bijection $b\colon \{1, \ldots, \Delta\} \to \{1, \ldots, \Delta\}$ so that for each edge $\{v, w\} \in E$, it holds that $p_v(w) = b(p_w(v))$.

Consider the following graphs and port numberings:

- For even $\Delta$, connect for each $h \in \{1, \ldots, \Delta/2\}$ node $i \in \{1, \ldots, n\}$ to node $(i + h) \bmod n$ with port number $2h - 1$ at node $i$ and port number $2h$ at node $(i + h) \bmod n$.

- For odd $\Delta$, observe that $n$ must be even (as the sum of degrees must be even). Do the same as for even $\Delta$ for $h \in \{1, \ldots, \lfloor \Delta/2 \rfloor\}$. Then add the perfect matching $\{1, \lceil n/2 \rceil\}, \{2, \lceil n/2 \rceil + 1\}, \ldots, \{\lfloor n/2 \rfloor, n\}$ with port number $\Delta$ for both endpoints of each edge.

The bijection $b$ is then given as follows

- If $\Delta$ is even:
$$b(i) = \begin{cases} i+1 & \text{if } i \text{ is odd, and} \\ i-1 & \text{if } i \text{ is even.} \end{cases}$$
- And for odd $\Delta$:
$$b(i) = \begin{cases} i+1 & \text{if } i < \Delta \text{ and } i \text{ is odd,} \\ i-1 & \text{if } i \text{ is even, and} \\ i & \text{if } i = \Delta. \end{cases}$$
□

**Remarks:**

- Of course, the list of things one cannot do in this model given in the theorem could go on forever.
- For $\Delta \leq n/2$, one can also construct bipartite graphs with $b(i) = i$ in a similar fashion. So nothing solvable in this case either? Can we do anything at all?!?

## 12.2 Bipartite Matching

Let's make our life a little bit easier. We consider 2-colored graphs now, where each node has its color as input. This is still fairly natural: Think of relations such as client/server, VIP/fan, or hypergraphs,[2] where we represent each hyperedge by a node (on one side of the bipartite graph) connected to its constituent nodes (on the other side).

Now finding a maximal matching is straightforward.

**Theorem 12.2.** *On 2-colored graphs of maximum degree $\Delta$, Algorithm 27 computes a maximal matching in $2\Delta$ rounds.*

*Proof.* Each white node is incident to at most one matching edge, because in each iteration it proposes only a single edge and terminates if it is selected. Each black node is incident to at most one matching edge, because it accepts only a single proposal. Any edge will be proposed, unless its white endpoint is matched before it gets to proposing it. Any proposed edge will be accepted, unless its black endpoint is already matched. Hence, any edge that is not in the matching is adjacent to a matching edge, implying that the matching is maximal.

The time complexity is $2\Delta$, as there are $\Delta$ iterations, each consisting of one round for proposals and one for accepts. □

[2] A hypergraphs is a structure $H = (V, E)$ in which each *hyperedge* $e \in E$ is an arbitrary subset of the nodes.

**Algorithm 27** Matching in 2-colored graphs of maximum degree $\Delta$ using port numberings, code at node $v$. Nodes return their matched port or $\perp$ if none of their incident edges is in the matching.

```
1:  for i = 1, ..., Δ do
2:    if v is white then
3:      send propose to port number i
4:    else if v receives propose then
5:      send accept to minimal port j at which propose was received
6:      return j
7:    end if
8:    if v receives accept on port i then
9:      return i
10:   end if
11: end for
12: return ⊥
```

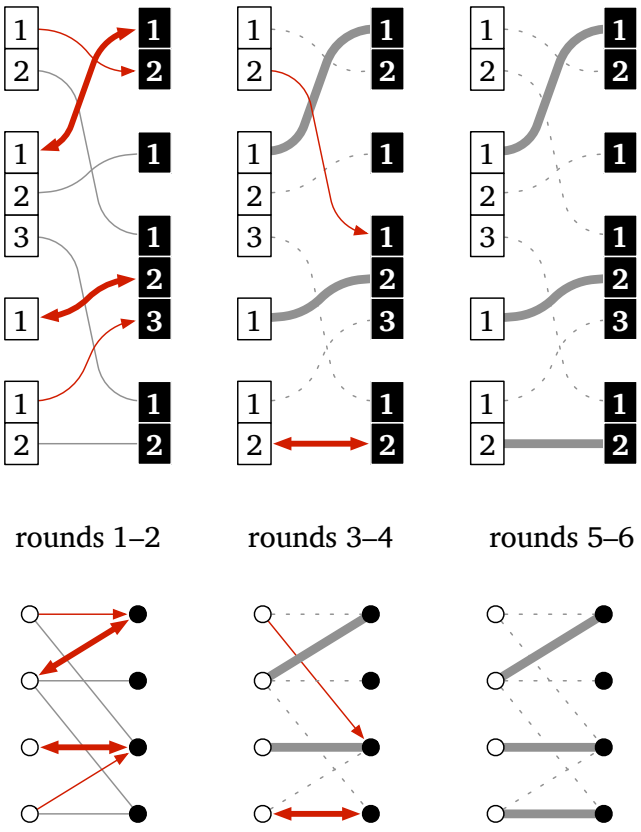


Figure 12.4: An execution of the matching algorithm. Red edges are proposed in this round, grey edges have not yet been proposed, and grey dotted lines did not make it; thick edges are matched.

- Not very fast if $\Delta$ is large. However, if $\Delta$ is a constant, so is the running time of the algorithm.
- The assumption that we have a 2-coloring does some intial symmetry breaking for us. For instance, we already have the best coloring one could get, and each color is a maximal independent set (as the graph is connected).
- However, we still cannot solve leader election, regardless of how much time we spend. Based on identifiers, this is possible, so the model is still weaker than the standard message passing model!
- This feels like cheating. Let's do something without requiring a 2-coloring!

## 12.3 3-Approximating Minimum Vertex Cover

We know that we cannot solve minimum vertex cover precisely, but that's ok – it's an NP-complete problem anyway. It's even NP-hard to approximate within a constant and, assuming the unique games conjecture, NP-hard to approximate better than factor $2 - o(1)$. On the other hand, obtaining a 2-approximation is easy: just output a maximal matching!

...except that we can't do that. We cannot compute *any* non-trivial matching. We first need to transform the graph into something we can handle: a 2-colored graph. Once this goal is set, it is actually not too hard to achieve.

- Replace each node by 2 copies, a white copy and a black copy.
- For an edge $\{v, w\}$, connect the white copy of $v$ to the black copy of $w$ and the black copy of $v$ to the white copy of $w$.
- The new edges inherit their port numbers from the originals.

**Lemma 12.3.** *For a port-numbered graph $G = (V, E, \{p_v\}_{v \in V})$, denote the (port-numbered) graph constructed above by $G'$. Then the constructed port numbering on $G'$ is feasible. Moreover, $G'$ is 2-colored (by nodes being white and black), has the same maximum degree as $G$, and the port-numbering model on $G'$ can be simulated on $G$ without overhead in round complexity.*

*Proof.* All properties are straightforward. Neighbors in $G'$ have different colors by construction, the new nodes have the same degree as the originals, inheriting port numbers results in port numbers $1, \ldots, \delta_v$ for a node of degree $\delta_v$, and each node can simulate both of its copies, where communication on new edges is performed via the original edges. □

**Theorem 12.4.** *A 4-approximation to vertex cover can be computed in $\mathcal{O}(\Delta)$ rounds of the port numbering model.*

*Proof.* We construct $G'$ and simulate Algorithm 27. The algorithm can be made to terminate without knowledge of $\Delta$ by non-matched white nodes terminating once they proposed on all their ports, letting their neighbors know, and non-matched black nodes terminating once all their neighbors terminated. By

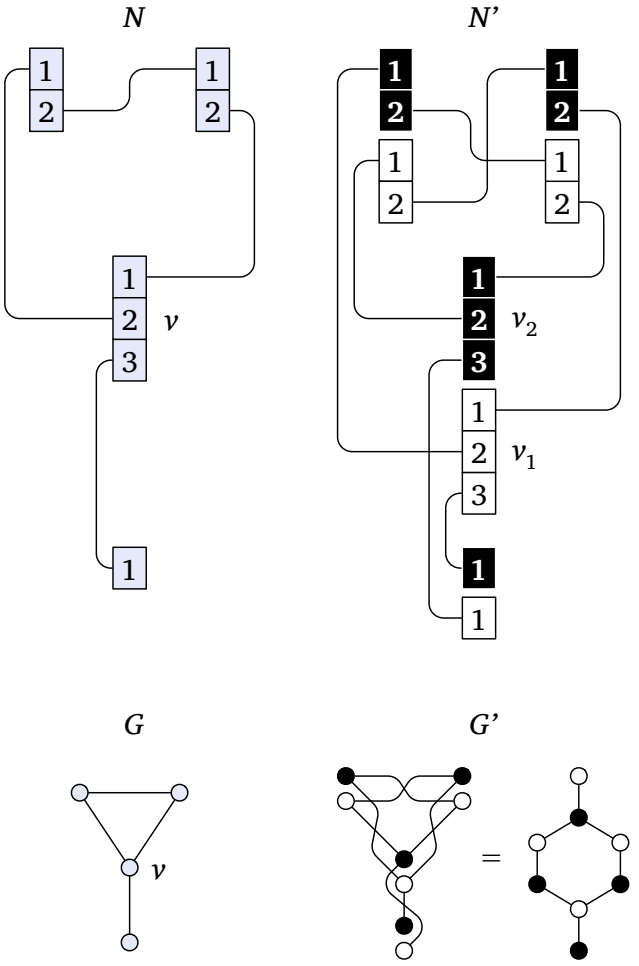


Figure 12.5: The construction of $G'$ from $G$.

Theorem 12.2, Lemma 12.3, and this modification, the algorithm terminates in $\mathcal{O}(\Delta)$ rounds.

If a black or white copy of a node is incident to a matching edge in $G'$, the node is in the vertex cover, otherwise it is not. Recall that the endpoints of a maximal matching form a 2-approximate vertex cover (as shown in Corollary 5.17). Because any vertex cover of $G$ induces a vertex cover of $G'$ of at most twice the size, the returned set has at most 4 times the size of a minimum vertex cover. As all edges in $G'$ have at least one incident node in the vertex cover of $G'$, the same is true in the computed node set of $G$, i.e., we did indeed find a vertex cover of $G$. □

If we look a bit closer, there's another surprise. The result is, in fact, a 3-approximation!

**Corollary 12.5.** *The algorithm from Theorem 12.4 returns a vertex cover that is at most factor* 3 *larger than the optimum.*

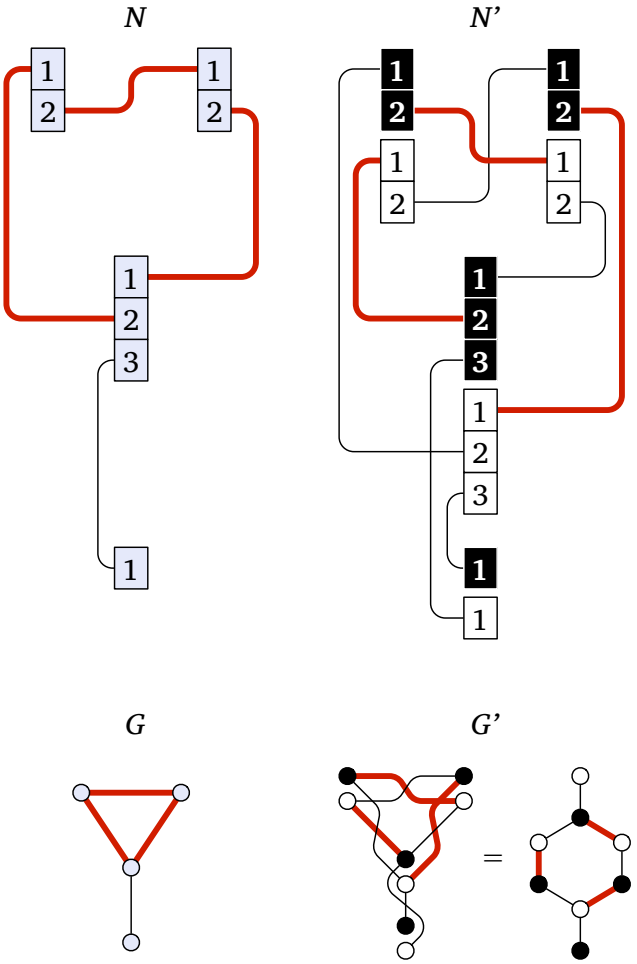


Figure 12.6: The computed matching in $G'$ and the corresponding edges in $G$.

*Proof.* Consider the originals of the matching edges in the maximal matching of $G'$. These induce a subgraph of $G$ of maximum degree 2, which is a disjoint union of paths and cycles of $k \geq 2$ nodes. To cover all edges of $G$, one needs to cover in particular these edges, and they can only be covered by the nodes on the respective paths and cycles. To cover a cycle of $k$ nodes, at least $k/2$ of its nodes must be selected. To cover a path of $k$ nodes, at least $\lfloor k/2 \rfloor$ of its nodes must be selected. As we choose all these nodes (and no others), the size of the computed vertex cover is at most factor $3 = \max_{2 \leq k \in \mathbb{N}} \{k/\lfloor k/2 \rfloor\}$ larger than the optimum. □

## What to take Home

- Even in very restricted models, some things can be done efficiently.

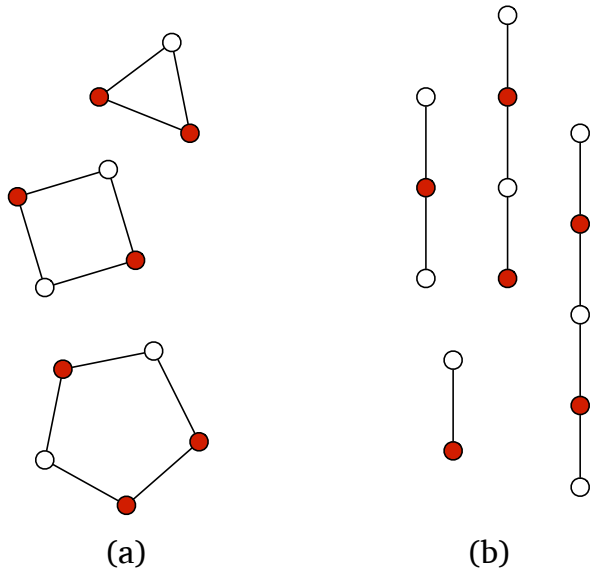


Figure 12.7: Examples for minimum vertex covers of cycles (a) and paths (b).

- Frequently, the resulting algorithms are very clean and simple, making them easy to implement.
- Such models tend to highlight what makes stronger models stronger. This can be useful in finding lower bounds or better algorithms, or just in pointing out that, e.g., one very important use of having node identifiers is the ability to execute Cole-Vishkin.
- Basic concepts such as simulation and indistinguishability are the name of the game also here.

## Bibliographic Notes

The presented vertex cover algorithm is due to Polishchuk and Suomela [PS09]. Using Cole-Vishkin *on edge weights*, it is possible to obtain a 2-approximation of the weighted version of the problem [ÅS10]. On the negative side, one cannot hope for anything better than a 2-approximation in the port-numbering model: in an even cycle an optimum solution chooses half of the nodes, but a symmetric port numbering causes all nodes to be selected. Even if identifiers and randomization are available, a classic construction shows that any distributed algorithm finding a reasonable approximation requires $\Omega(\sqrt{\log n})$ and $\Omega(\log \Delta)$ rounds.[3] Recently, it has been shown that there is some constant $\delta > 0$ so that finding a $(1+\delta)$-approximation cannot be done in $o(\log n)$ rounds [GS12], even if the graph it 2-colored and has degree 3! Note that this an unconditional lower

[3]This is the same construction. Choosing the maximum feasible value of $\Delta$ for a given value of $n$ yields the $\Omega(\sqrt{\log n})$ bound.

bound. It does not depend on P≠NP or similar assumptions, but arises from locality issues.

All figures but Figure 12.1 are courtesy of Jukka Suomela and under a creative commons license.[4] Large parts of today's lecture are my own narrative of a part of Jukka's course on deterministic distributed algorithms. A reference to his survey of local algorithms [Suo13] is also in order; a local algorithm is a distributed algorithm whose running time is bounded by a constant.

# Bibliography

[ÅS10] Matti Åstrand and Jukka Suomela. Fast Distributed Approximation Algorithms for Vertex Cover and Set Cover in Anonymous Networks. In *Proc. 22nd Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 294–302, 2010.

[GS12] Mika Göös and Jukka Suomela. No Sublogarithmic-time Approximation Scheme for Bipartite Vertex Cover. In *Proc. 26th Conference on Distributed Computing (DISC)*, pages 181–194, 2012.

[PS09] Valentin Polishchuk and Jukka Suomela. A Simple Local 3-approximation Algorithm for Vertex Cover. *Information Processing Letters*, 109(12):642–645, 2009.

[Suo13] Jukka Suomela. Survey of Local Algorithms. *ACM Computing Surveys*, 45(2):24:1–24:40, March 2013.