Chapter 4a

Lower Bounds: Covering Maps

The previous chapters focused on positive results; now we will turn our attention to techniques that can be used to prove negative results. We will start with so-called covering maps—we will use covering maps to prove that many problems cannot be solved at all with deterministic port numbering algorithms.

Definition 4.1 (Covering Map)**.** *Let $G = (V, E, \{p_v\}_{v \in V})$ and $G' = (V', E', \{p'_v\}_{v \in V'})$ be port-numbered networks, and let $\phi \colon V \to V'$. We say that ϕ is a* covering map from G to G' *if the following holds:*

1. *ϕ is a surjection: $\phi(V) = V'$.*
2. *ϕ preserves degrees: $\deg_G(v) = \deg_{G'}(\phi(v))$ for all $v \in V$.*
3. *ϕ preserves connections and port numbers: $p_u(v) = i$ and $p_v(u) = j$ imply $p'_{\phi(u)}(\phi(v)) = i$ and $p'_{\phi(v)}(\phi(u)) = j$.*

We can also consider labeled networks, for example, networks with local inputs. Let $c \colon V \to X$ and $c' \colon V' \to X$ be the node labels in G and G'. We say that ϕ is a covering map from (G, c) to (G', c') if ϕ is a covering map from G to G' and the following holds:

4. *ϕ preserves labels: $c(v) = c'(\phi(v))$ for all $v \in V$.*

Remarks:

- A covering map is a topological concept that finds applications in many areas of mathematics, including graph theory.
- Figure 4.2 shows an example of a covering map.

4a.1 Covers and Executions

Now we will study covering maps from the perspective of deterministic port numbering algorithms. The basic idea is that a covering map ϕ from G to

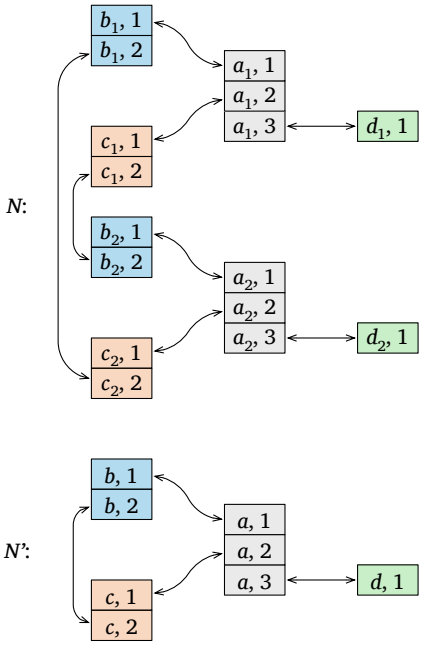

Figure 4.2: There is a covering map ϕ from N to N' that maps $a_i \mapsto a$, $b_i \mapsto b$, $c_i \mapsto c$, and $d_i \mapsto d$ for each $i \in \{1, 2\}$.

G' fools any port numbering algorithm A: a node v in G is indistinguishable from the node $\phi(v)$ in G'. Before diving into the analysis, we will formalize the execution of an algorithm in the port numbering model.

Definition 4.3 (configuration)**.** *We say that a system is fully defined (at any point during the execution) by its* configuration C*. The configuration includes the state of every node, and all messages that are in transit (sent but not yet received).*

Remarks:

- An initial configuration in a port-numbered network, denoted C_0, consists of the initial state of every node, its degree, and the messages that this node will send in the first round.
- In synchronous communication, the messages that are in transit in configuration C_i are exactly the messages that were sent by the nodes in round i.

Definition 4.4 (synchronous execution)**.** *Let $m_t(u) = (m_t(u, 1), \ldots, m_t(u, d))$ be the set of massages in transit from node u through each of the ports in round t. An execution is a sequence of configurations $C_0, C_1, C_2, \ldots$ such that C_{i+1} is reached from C_i by receiving the messages $m_t(u)$ for all nodes $u \in V$ and performing some deterministic local computation.*

Theorem 4.5. *Assume that*

1. *A is a deterministic port numbering algorithm,*
2. *$G = (V, E, \{p_v\}_{v \in V})$ and $G' = (V', E', \{p'_v\}_{v \in V'})$ are port-numbered networks, and*
3. *$\phi \colon V \to V'$ is a covering map from G to G'.*

Let

4. *$C_0, C_1, \ldots$ be the execution of A on G,*
5. *$C'_0, C'_1, \ldots$ be the execution of A on G', and*
6. *$C_r(v)$ and $C'_r(w)$ denote the r^{th} configurations of the nodes v and v' in the respective networks.*

Then for each $r = 0, 1, \ldots$ and each $v \in V$ we have $C_r(v) = C'_r(\phi(v))$.

Proof. Let $m'_r(u', i)$ be the message received by $u' \in V'$ from port i in round r in the execution of A on G', and $m'_r(u')$ is the vector of messages received by u'.

The proof is by induction on r. To prove the base case $r = 0$, let $v \in V$, $d = \deg_N(v)$, and $v' = \phi(v)$. The initial configuration $C'_0(v')$ depends on the initial input of v', its degree, and the corresponding port numbers, as the node does not have any other information before the first communication round. Observe that the initial input of v, its degree, and the port numbers are the same as for v'. Therefore, v will send the same messages as v' and thus the initial configurations $C'_0(v')$ and $C_0(v)$ are the identical. This holds for all nodes $v \in V$.

For the inductive step, assume that $C_{r-1}(v) = C'_{r-1}(\phi(v))$ for all nodes $v \in V$. Thus, also the vectors of sent messages are the same in both configurations, i.e. $m_{r-1}(v) = m'_{r-1}(\phi(v))$. Note that this equality also holds for the received messages at every node in round $r - 1$. Thus, v and $\phi(v)$ will perform the same local computation and send the same messages at the beginning or round r.

□

Remarks:

- In particular, if the execution of A on G stops in time T, the execution of A on G' stops in time T as well, and vice versa. Moreover, ϕ preserves the local outputs of the nodes in every round.

4a.2 Lower Bounds

We will give representative examples of negative results that we can easily derive from Theorem 4.5. First, we will revisit the proof of Theorem **??** from the chapter on Port Numbering, where we showed that it is not possible to compute a proper vertex coloring, the minimum vertex cover, and a non-empty matching in the port numbering model. In the following, we will differentiate between a graph G and the port-numbered network N defined on this graph. This differentiation is convenient in the context of covering maps, since we would like to talk about the port-numbered network and its cover as networks that both represent the execution of the same algorithm on the underlying graph G.

Impossibility theorem from Port Numbering revisited. We will start by considering the vertex coloring problem on a graph with two nodes that are connected by an edge. We will denote this graph as N. Observe that we can map this port-numbered network to a single node with a self-loop, see network N' in Figure 4.6. This map preserves the only port number as well as the single edge that each node has. We can now simulate any coloring algorithm in N'. If an algorithm terminates in N', it terminates with some output color c. In the original network N, this means that both nodes would terminate with the same output color c. Alternatively, the nodes might not terminate at all. In either case, any algorithm would fail to compute a proper coloring. The same argument can be applied to the minimum vertex cover problem. □

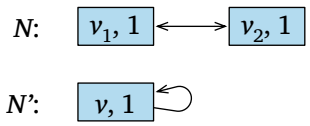

Figure 4.6: There is a covering map ϕ from N to N' that maps $v_i \mapsto v$ for each $i \in \{1, 2\}$. Again, N is a simple port-numbered network but N' is not.

Remarks:

- Recall that the non-empty matching problem can still be solved on a graph with two nodes.
- Next, we will observe that a deterministic port numbering algorithm cannot break symmetry in a cycle—unless we provide some symmetry-breaking information in local inputs.

Lemma 4.7. *Let $G = (V, E)$ be a cycle graph, let A be a deterministic port numbering algorithm. Then there is a simple port-numbered network $N = (V, E, \{p_v\}_{v \in V})$ such that*

1. *the underlying graph of N is G, and*
2. *if A stops, the output is a constant function $g \colon V \to \{c\}$ for some c.*

Proof. Label the nodes $V = \{ v_1, v_2, \ldots, v_n \}$ along the cycle so that the edges are

$$E = \big\{ \{v_1, v_2\},\ \{v_2, v_3\},\ \ldots,\ \{v_{n-1}, v_n\},\ \{v_n, v_1\} \big\}.$$

Choose the port numbering p of G as $p_{u_i}(u_{i+1}) = p_{u_n}(u_1) = 1$ and $p_{u_{i+1}}(u_i) = p_{u_1}(u_n) = 2$ for all $i \in [n-1]$. See Figure 4.8 for an illustration in the case $n = 3$.

Define another port-numbered network $N' = (V', E', \{p'_v\}_{v \in V'})$ with $V' = \{v\}$, $E' = \{\{v, v\}\}$ and a port numbering where port 1 of v is connected to port 2 of v. Let $f' \colon V' \to \{0\}$. Define a function $\phi \colon V \to V'$ by setting $\phi(v_i) = v$ for each i.

Now we can verify that ϕ is a covering map from N to N'. Assume that A terminates on N and produces an output g. By Theorem 4.5, A also terminates on N' and produces an output g'. Let $c = g'(v)$. Now

$$g(v_i) = g'(\phi(v_i)) = g'(v) = c$$

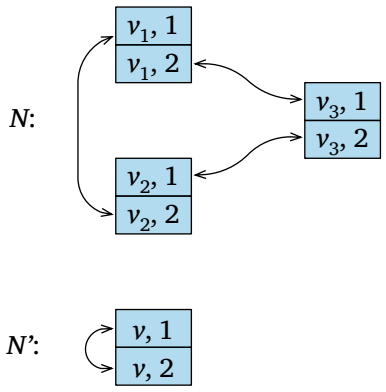

Figure 4.8: There is a covering map ϕ from N to N' that maps $v_i \mapsto v$ for each $i \in \{1, 2, 3\}$. Here N is a simple port-numbered network but N' is not.

for all i. □

Remarks:

- In the above proof, we never assumed that the execution of A on N' makes any sense—after all, N' is not even a simple port-numbered network, and there is no underlying graph. Algorithm A was never designed to be applied to such a strange network with only one node. Nevertheless, the execution of A on N' is formally well-defined, and Theorem 4.5 holds. We do not really care what A outputs on N', but the existence of a covering map can be used to prove that the output of A on N has certain properties. It may be best to interpret the execution of A on N' as a thought experiment, not as something that we would actually try to do in practice.
- Lemma 4.7 has many immediate corollaries.

Corollary 4.9. *Let $\mathcal{F}$ be the family of cycle graphs. Then there is no deterministic port numbering algorithm that solves any of the following problems on $\mathcal{F}$:*

1. *maximal independent set,*
2. *1.999-approximation of a minimum vertex cover,*
3. *maximal matching,*
4. *non-empty matching,*
5. *vertex coloring,*
6. *edge coloring.*

Proof. In each of these cases, there is a graph $G \in \mathcal{F}$ such that a constant function is not a feasible solution in the network N that we constructed in Lemma 4.7.

For example, consider the case of minimum vertex cover; other cases are similar. Assume that $G = (V, E)$ is a cycle with $2k$ nodes. Then a minimum vertex cover consists of k nodes—it is sufficient to take every second node. Hence a 1.999-approximation of a minimum dominating set consists of at most

$1.999k < 2k$ nodes. A solution $D = V$ violates the approximation guarantee, as D has too many nodes, while $D = \varnothing$ is not a vertex cover. Hence if A outputs a constant function, it cannot produce a 1.999-approximation of a minimum vertex cover.

□

Remarks:

- So far we have used only one covering map in our proofs; the following lemma gives an example of the use of more than one covering map.
- In the following, we will consider a global graph problem - the computation of a spanning tree on a cycle.

Lemma 4.10. *Let G and H be two unweighted cycle graphs. If G is a covering map of H, then it is not possible to compute a spanning tree in both G and H.*

Proof. Consider the case where $G = (V(G), E(G), \{p_v^G\}_{v \in V(G)})$ is a cycle of size n and $H = (V(H), E(H), \{p_v^H\}_{v \in V(H)})$ is a cycle of size $2n$. Similar to Lemma 4.7, label the nodes $V(G) = \{ u_1, u_2, \ldots, u_n \}$ of G along the cycle so that the edges are

$$E(G) = \big\{ \{u_1, u_2\},\ \{u_2, u_3\},\ \ldots,\ \{u_{n-1}, u_n\},\ \{u_n, u_1\} \big\}.$$

and the nodes $V(H) = \{ v_1, v_2, \ldots, v_n, w_1, w_2, \ldots, w_n \}$ of H so that the edges are

$$\begin{aligned} E(H) = \big\{ &\{v_1, v_2\},\ \ldots,\ \{v_{n-1}, v_n\},\ \{v_n, w_1\}, \\ &\{w_1, w_2\},\ \ldots,\ \{w_{n-1}, w_n\},\ \{w_n, v_1\} \big\}. \end{aligned}$$

Choose the port numbering p^G of G as $p^G_{u_i}(u_{i+1}) = p^G_{u_n}(u_1) = 1$ and $p^G_{u_{i+1}}(u_i) = p^G_{u_1}(u_n) = 2$ for all $i \in [n-1]$. Choose the port numbering p^H of H as $p^H_{v_i}(v_{i+1}) = p^H_{w_i}(w_{i+1}) = p^H_{v_n}(w_1) = p^H_{w_n}(v_1) = 1$ and $p^H_{v_{i+1}}(v_i) = p^H_{w_{i+1}}(w_i) = p^H_{w_1}(v_n) = p^H_{v_1}(w_n) = 2$ for all $i \in [n-1]$.

Observe that there is a covering map from H to G where the nodes v_i and w_i are mapped to $u_i, i \in [n]$, the edges v_i, v_{i+1} and w_i, w_{i+1} are mapped to u_i, u_{i+1}, and the edges v_n, v_1 and w_n, w_1 are mapped to u_n, u_1.

Let C_0 be the initial configuration given to the spanning tree algorithm that is running on G, and C_k its final configuration. Observe that the final configuration C_k contains a node labeling that determines the spanning tree of G. A spanning tree in a cycle with $|V|$ nodes has exactly $|V|-1$ edges. Consider an edge $e = u, v$ that is not in the spanning tree of C_k. There exist two edges $e_1 = v_i, v_j$ and $e_2 = w_i, w_j$ in H that are mapped to the edge $e' = u_1, u_2$ by the covering map. Assume now that we run the same algorithm on H. We can use Theorem 4.5 to show that neither e_1 nor e_1 will be chosen as spanning tree edges of H. This leads to a contradiction, as a spanning tree of H must contain $2n - 1$ edges. □

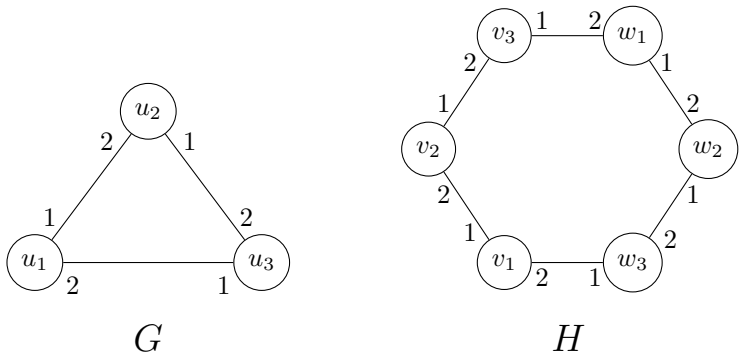

Figure 4.11: There is a covering map ϕ from H to G that maps $v_i, w_i \mapsto u_i$, $i \in \{1, 2, 3\}$. Assume that algorithm A computed the spanning tree of G such that it does not contain the edge $\{u_1, u_2\}$. Then, when executed on H, A does not include the edges $\{v_1, v_2\}$ and $\{w_1, w_2\}$, thus splitting the graph in two trees.

Remarks:

- Figure 4.11 shows an example of the proof of Lemma 4.10 on two cycles of length 3 and 6.
- The above counter-example also works in the weighted case, and if the nodes of G and H have additional inputs, such as a 3-coloring.

Lemma 4.12. *Let $\mathcal{F} = \{ G_3, G_4 \}$, where G_3 is the cycle graph with 3 nodes, and G_4 is the cycle graph with 4 nodes. There is no deterministic port numbering algorithm that solves the following problem Π on $\mathcal{F}$: in $\Pi(G_3)$ all nodes output 3 and in $\Pi(G_4)$ all nodes output 4.*

Proof. We again apply the construction of Lemma 4.7; for each $i \in \{3, 4\}$, let N_i be the symmetric port-numbered network that has G_i as the underlying graph.

Now it would be convenient if we could construct a covering map from N_4 to N_3; however, this is not possible. Therefore we proceed as follows. Construct a one-node network N' as in the proof of Lemma 4.7, construct the covering map ϕ_3 from N_3 to N', and construct the covering map ϕ_4 from N_4 to N'; see Figure 4.13. The local inputs are assumed to be all zeros.

Let A be a port numbering algorithm, and let c be the output of the only node of N'. If we apply Theorem 4.5 to ϕ_3, we conclude that all nodes of N_3 output c; if A solves Π on G_3, we must have $c = 3$. However, if we apply Theorem 4.5 to ϕ_4, we learn that all nodes of N_4 also output $c = 3$, and hence A cannot solve Π on $\mathcal{F}$. □

Remarks:

- We have learned that a deterministic port numbering algorithm cannot determine the length of a cycle. In particular, a deterministic port numbering algorithm cannot determine if the underlying graph is bipartite.

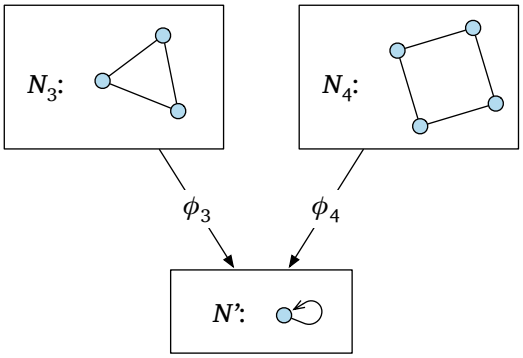

Figure 4.13: The structure of the proof of Lemma 4.12.

4a.3 Bibliographic Notes

The use of covering maps in the context of distributed algorithms was introduced by Angluin [Ang80]. In the beginning, the main focus of covering maps has been on global problems, such as leader election [Ang80], spanning trees, topology recognition [YK96], or computation of functions such as AND, SUM, XOR, and non-constant functions [MW86, DG87, ASW88, AS91]. More recently, covering maps have also been used to classify more local problems, such as edge dominating sets, weak coloring, and maximal fractional matching [Suo10, ÅPR+10, GHS14].

Bibliography

[Ang80] Dana Angluin. Local and global properties in networks of processors. In *Proc. 12th Annual ACM Symposium on Theory of Computing (STOC 1980)*, 1980.

[ÅPR+10] Matti Åstrand, Valentin Polishchuk, Joel Rybicki, Jukka Suomela, and Jara Uitto. Local algorithms in (weakly) coloured graphs, 2010.

[AS91] Hagit Attiya and Marc Snir. Better computing on the anonymous ring. *Journal of Algorithms*, 12(2):204–238, 1991.

[ASW88] Hagit Attiya, Marc Snir, and Manfred K. Warmuth. Computing on an anonymous ring. *J. ACM*, 35(4), oct 1988.

[DG87] Pavol Duris and Zvi Galil. Two lower bounds in asynchronous distributed computation. In *28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*, pages 326–330, 1987.

[GHS14] Mika Göös, Juho Hirvonen, and Jukka Suomela. Linear-in-delta lower bounds in the local model. In *Proceedings of the 2014 ACM Symposium on Principles of Distributed Computing*, PODC '14, 2014.

[MW86] Shlomo Moran and Manfred Warmuth. Gap theorems for distributed computing. In *Proceedings of the Fifth Annual ACM Symposium on Principles of Distributed Computing*, PODC '86, 1986.

[Suo10] Jukka Suomela. Distributed algorithms for edge dominating sets. In *Proc. 29th Annual ACM Symposium on Principles of Distributed Computing (PODC 2010)*, pages 365–374, 2010.

[YK96] Masafumi Yamashita and Tsunehiko Kameda. Computing on anonymous networks: part I—characterizing the solvable cases. *IEEE Transactions on Parallel and Distributed Systems*, 7(1):69–89, 1996.