Juho Hirvonen and Jukka Suomela:
Distributed Algorithms 2020
https://jukkasuomela.fi/da2020/ · March 14, 2025

Chapter 7
Covering Maps

Chapters 3–6 have 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 PN-algorithms.

7.1 Definition

A covering map is a topological concept that finds applications in many areas of mathematics, including graph theory. We will focus on one special case: covering maps between port-numbered networks.

Let $N = (V, P, p)$ and $N' = (V', P', p')$ be port-numbered networks, and let $\phi : V \to V'$. We say that ϕ is a *covering map from N to N'* if the following holds:

(a) ϕ is a surjection: $\phi(V) = V'$.

(b) ϕ preserves degrees: $\deg_N(v) = \deg_{N'}(\phi(v))$ for all $v \in V$.

(c) ϕ preserves connections and port numbers: $p(u,i) = (v,j)$ implies $p'(\phi(u), i) = (\phi(v), j)$.

See Figures 7.1–7.3 for examples.

We can also consider labeled networks, for example, networks with local inputs. Let $f : V \to X$ and $f' : V' \to X$. We say that ϕ is a covering

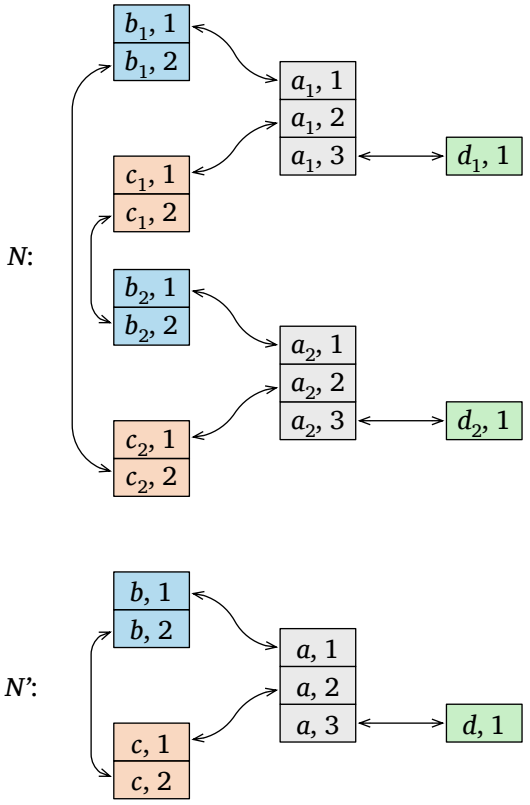

Figure 7.1: There is a covering map ϕ from N to N' that maps $a_i \mapsto a$, $b_i \mapsto b$, $c_i \mapsto c$, and $d_i \mapsto d$ for each $i \in \{1, 2\}$.

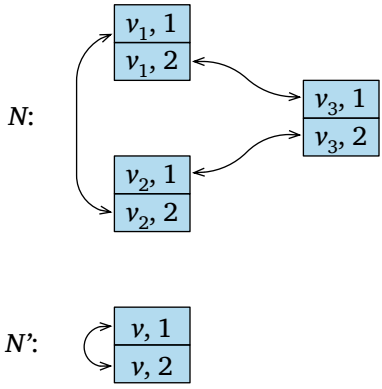

Figure 7.2: There is a covering map ϕ from N to N' that maps $v_i \mapsto v$ for each $i \in \{1,2,3\}$. Here N is a simple port-numbered network but N' is not.

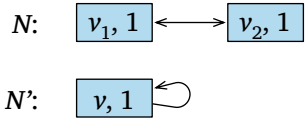

Figure 7.3: There is a covering map ϕ from N to N' that maps $v_i \mapsto v$ for each $i \in \{1,2\}$. Again, N is a simple port-numbered network but N' is not.

map from (N, f) to (N', f') if ϕ is a covering map from N to N' and the following holds:

(d) ϕ preserves labels: $f(v) = f'(\phi(v))$ for all $v \in V$.

7.2 Covers and Executions

Now we will study covering maps from the perspective of deterministic PN-algorithms. The basic idea is that a covering map ϕ from N to N' fools any PN-algorithm A: a node v in N is indistinguishable from the node $\phi(v)$ in N'.

Without further ado, we state the main result and prove it—many applications and examples will follow.

Theorem 7.1. *Assume that*

(a) A is a deterministic PN*-algorithm with $X = \mathrm{Input}_A$,*

(b) $N = (V, P, p)$ and $N' = (V', P', p')$ are port-numbered networks,

(c) $f : V \to X$ and $f' : V' \to X$ are arbitrary functions, and

(d) $\phi : V \to V'$ is a covering map from (N, f) to (N', f').

Let

(e) $x_0, x_1, \ldots$ be the execution of A on (N, f), and

(f) $x'_0, x'_1, \ldots$ be the execution of A on (N', f').

Then for each $t = 0, 1, \ldots$ and each $v \in V$ we have $x_t(v) = x'_t(\phi(v))$.

Proof. We will use the notation of Section 3.3.2; the symbols with a prime refer to the execution of A on (N', f'). In particular, $m'_t(u', i)$ is the message received by $u' \in V'$ from port i in round t in the execution of A on (N', f'), and $m'_t(u')$ is the vector of messages received by u'.

The proof is by induction on t. To prove the base case $t = 0$, let $v \in V$, $d = \deg_N(v)$, and $v' = \phi(v)$; we have

$$x'_0(v') = \mathrm{init}_{A,d}(f'(v')) = \mathrm{init}_{A,d}(f(v)) = x_0(v).$$

For the inductive step, let $(u,i) \in P$, $(v,j) = p(u,i)$, $d = \deg_N(u)$, $\ell = \deg_N(v)$, $u' = \phi(u)$, and $v' = \phi(v)$. Let us first consider the messages sent by v and v'; by the inductive assumption, these are equal:

$$\mathrm{send}_{A,\ell}(x'_{t-1}(v')) = \mathrm{send}_{A,\ell}(x_{t-1}(v)).$$

A covering map ϕ preserves connections and port numbers: $(u,i) = p(v,j)$ implies $(u',i) = p'(v',j)$. Hence $m_t(u,i)$ is component j of $\mathrm{send}_{A,\ell}(x_{t-1}(v))$, and $m'_t(u',i)$ is component j of $\mathrm{send}_{A,\ell}(x'_{t-1}(v'))$. It follows that $m_t(u,i) = m'_t(u',i)$ and $m_t(u) = m'_t(u')$. Therefore

$$\begin{aligned} x'_t(u') &= \mathrm{receive}_{A,d}\big(x'_{t-1}(u'), m'_t(u')\big) \\ &= \mathrm{receive}_{A,d}\big(x_{t-1}(u), m_t(u)\big) = x_t(u). \end{aligned}$$ □

In particular, if the execution of A on (N,f) stops in time T, the execution of A on (N',f') stops in time T as well, and vice versa. Moreover, ϕ preserves the local outputs: $x_T(v) = x'_T(\phi(v))$ for all $v \in V$.

7.3 Examples

We will give representative examples of negative results that we can easily derive from Theorem 7.1. First, we will observe that a deterministic PN-algorithm cannot break symmetry in a cycle—unless we provide some symmetry-breaking information in local inputs.

Lemma 7.2. *Let $G = (V,E)$ be a cycle graph, let A be a deterministic* PN*-algorithm, and let f be a constant function $f : V \to \{0\}$. Then there is a simple port-numbered network $N = (V,P,p)$ such that*

(a) the underlying graph of N is G, and

(b) if A stops on (N,f), the output is a constant function $g : V \to \{c\}$ 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 as follows:

$$p : (v_1, 1) \mapsto (v_2, 2),\ (v_2, 1) \mapsto (v_3, 2),\ \ldots,$$
$$(v_{n-1}, 1) \mapsto (v_n, 2),\ (v_n, 1) \mapsto (v_1, 2).$$

See Figure 7.2 for an illustration in the case $n = 3$.

Define another port-numbered network $N' = (V', P', p')$ with $V' = \{v\}$, $P' = \{(v, 1), (v, 2)\}$, and $p(v, 1) = (v, 2)$. Let $f' : V' \to \{0\}$. Define a function $\phi : V \to V'$ by setting $\phi(v_i) = v$ for each i.

Now we can verify that ϕ is a covering map from (N, f) to (N', f'). Assume that A stops on (N, f) and produces an output g. By Theorem 7.1, A also stops on (N', f') and produces an output g'. Let $c = g'(v)$. Now

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

for all i. □

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 7.1 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 7.2 has many immediate corollaries.

Corollary 7.3. *Let $\mathscr{F}$ be the family of cycle graphs. Then there is no deterministic* PN*-algorithm that solves any of the following problems on $\mathscr{F}$:*

(a) maximal independent set,
(b) 1.999-approximation of a minimum vertex cover,
(c) 2.999-approximation of a minimum dominating set,
(d) maximal matching,

(e) vertex coloring,
(f) weak coloring,
(g) 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 7.2.

For example, consider the case of dominating sets; other cases are similar. Assume that $G = (V, E)$ is a cycle with $3k$ nodes. Then a minimum dominating set consists of k nodes—it is sufficient to take every third node. Hence a 2.999-approximation of a minimum dominating set consists of at most $2.999k < 3k$ nodes. A solution $D = V$ violates the approximation guarantee, as D has too many nodes, while $D = \varnothing$ is not a dominating set. Hence if A outputs a constant function, it cannot produce a 2.999-approximation of a minimum dominating set. □

Lemma 7.4. *There is no deterministic* PN*-algorithm that finds a weak coloring for every* 3*-regular graph.*

Proof. Again, we are going to apply the standard technique: pick a suitable 3-regular graph G, find a port-numbered network N that has G as its underlying graph, find a smaller network N' such that we have a covering map ϕ from N to N', and apply Theorem 7.1.

However, it is not immediately obvious which 3-regular graph would be appropriate; hence we try the simplest possible case first. Let $G = (V, E)$ be the *complete graph* on four nodes: $V = \{s, t, u, v\}$, and we have an edge between any pair of nodes; see Figure 7.4. The graph is certainly 3-regular: each node is adjacent to the other three nodes.

Now it is easy to verify that the edges of G can be partitioned into a 2-factor X and a 1-factor Y. The 2-factor consists of a cycle and a 1-factor consists of disjoint edges. We can use the factors to guide the selection of port numbers in N.

In the cycle induced by X, we can choose symmetric port numbers using the same idea as what we had in the proof of Lemma 7.2; one end of each edge is connected to port 1 while the other end is connected to

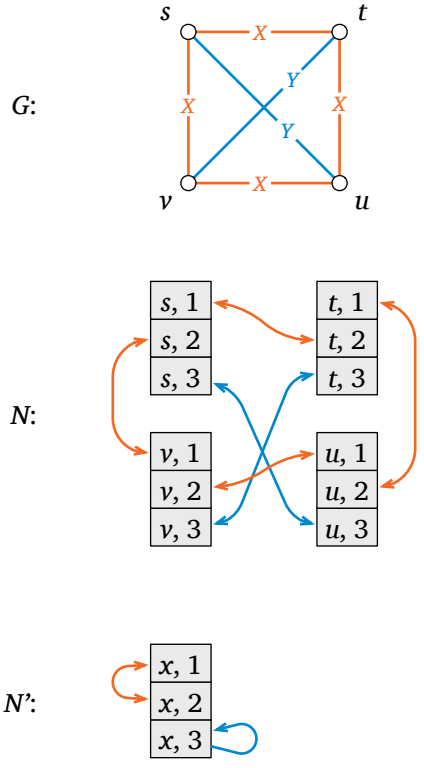

Figure 7.4: Graph G is the complete graph on four nodes. The edges of G can be partitioned into a 2-factor X and a 1-factor Y. Network N has G as its underlying graph, and there is a covering map ϕ from N to N'

port 2. For the edges of the 1-factor Y, we can assign port number 3 at each end. We have constructed the port-numbered network N that is illustrated in Figure 7.4.

Now we can verify that there is a covering map ϕ from N to N', where N' is the network with one node illustrated in Figure 7.4. Therefore in any algorithm A, if we do not have any local inputs, all nodes of N will produce the same output. However, a constant output is not a weak coloring of G. □

In the above proof, we could have also partitioned the edges of G into three 1-factors, and we could have used the 1-factorization to guide the selection of port numbers. However, the above technique is more general: there are 3-regular graphs that do not admit a 1-factorization but that can be partitioned into a 1-factor and a 2-factor.

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.

Lemma 7.5. *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* PN*-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 7.2; 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 (see the exercises). Therefore we proceed as follows. Construct a one-node network N' as in the proof of Lemma 7.2, 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 7.5. The local inputs are assumed to be all zeros.

Let A be a PN-algorithm, and let c be the output of the only node of N'. If we apply Theorem 7.1 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

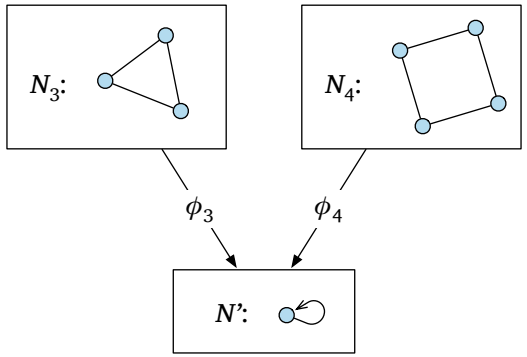

Figure 7.5: The structure of the proof of Lemma 7.5.

Theorem 7.1 to ϕ_4, we learn that all nodes of N_4 also output $c = 3$, and hence A cannot solve Π on $\mathcal{F}$. □

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

7.4 Quiz

Let $G = (V, E)$ be a graph. A set $X \subseteq V$ is a *k-tuple dominating set* if for every $v \in V$ we have $|\mathrm{ball}_G(v, 1) \cap X| \geq k$. Consider the problem of finding a minimum 2-tuple dominating set in *cycles*. What is the best (i.e. smallest) approximation ratio we can achieve in the PN model?

7.5 Exercises

We use the following definition in the exercises. A graph G is *homogeneous* if there are port-numbered networks N and N' and a covering map ϕ from N to N' such that N is simple, the underlying graph of N is

G, and N' has only one node. For example, Lemma 7.2 shows that all cycle graphs are homogeneous.

Exercise 7.1 (finding port numbers)**.** Consider the graph G and network N' illustrated in Figure 7.6. Find a simple port-numbered network N such that N has G as the underlying graph and there is a covering map from N to N'.

Exercise 7.2 (homogeneity)**.** Assume that G is homogeneous and it contains a node of degree at least two. Give several examples of graph problems that cannot be solved with any deterministic PN-algorithm in any family of graphs that contains G.

Exercise 7.3 (regular and homogeneous)**.** Show that the following graphs are homogeneous:

(a) graph G illustrated in Figure 7.7,

(b) graph G illustrated in Figure 7.6.

▷ *hint A*

Exercise 7.4 (complete graphs)**.** Recall that we say that a graph $G = (V, E)$ is *complete* if for all nodes $u, v \in V$, $u \neq v$, there is an edge $\{u, v\} \in E$. Show that

(a) any $2k$-regular graph is homogeneous,

(b) any complete graph with $2k$ nodes has a 1-factorization,

(c) any complete graph is homogeneous.

Exercise 7.5 (dominating sets)**.** Let $\Delta \in \{2, 3, \dots\}$, let $\epsilon > 0$, and let $\mathcal{F}$ consist of all graphs of maximum degree at most Δ. Show that it is possible to find a $(\Delta + 1)$-approximation of a minimum dominating set in constant time in family $\mathcal{F}$ with a deterministic PN-algorithm. Show that it is not possible to find a $(\Delta + 1 - \epsilon)$-approximation with a deterministic PN-algorithm.

▷ *hint B*

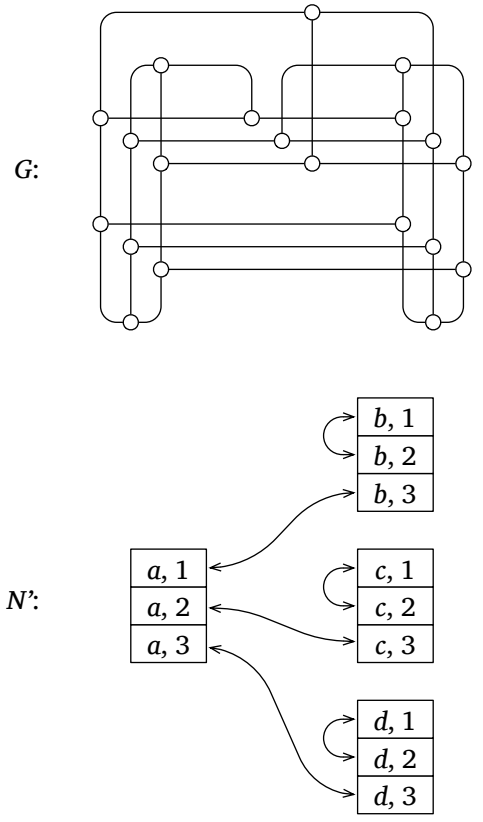

Figure 7.6: Graph G and network N' for Exercises 7.1 and 7.3b.

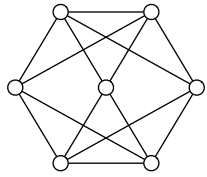

Figure 7.7: Graph G for Exercise 7.3a.

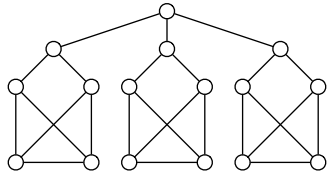

Figure 7.8: Graph G for Exercise 7.7.

Exercise 7.6 (covers with covers)**.** What is the connection between covering maps and the vertex cover 3-approximation algorithm in Section 3.6?

⋆ **Exercise 7.7** (3-regular and not homogeneous)**.** Consider the graph G illustrated in Figure 7.8.

(a) Show that G is not homogeneous.

(b) Present a deterministic PN-algorithm A with the following property: if N is a simple port-numbered network that has G as the underlying graph, and we execute A on N, then A stops and produces an output where at least one node outputs 0 and at least one node outputs 1.

(c) Find a simple port-numbered network N that has G as the underlying graph, a port-numbered network N', and a covering map ϕ from N to N' such that N' has the smallest possible number of nodes.

▷ *hint C*

⋆ **Exercise 7.8** (covers and connectivity)**.** Assume that $N = (V, P, p)$ and $N' = (V', P', p')$ are simple port-numbered networks such that there is a covering map ϕ from N to N'. Let G be the underlying graph of network N, and let G' be the underlying graph of network N'.

(a) Is it possible that G is connected and G' is not connected?

(b) Is it possible that G is not connected and G' is connected?

⋆ **Exercise 7.9** (k-fold covers)**.** Let $N = (V, P, p)$ and $N' = (V', P', p')$ be simple port-numbered networks such that the underlying graphs of N and N' are connected, and assume that $\phi : V \to V'$ is a covering map from N to N'. Prove that there exists a positive integer k such that the following holds: $|V| = k|V'|$ and for each node $v' \in V'$ we have $|\phi^{-1}(v')| = k$. Show that the claim does not necessarily hold if the underlying graphs are not connected.

7.6 Bibliographic Notes

The use of covering maps in the context of distributed algorithm was introduced by Angluin [1]. The general idea of Exercise 7.7 can be traced back to Yamashita and Kameda [5], while the specific construction in Figure 7.8 is from Bondy and Murty's textbook [3, Figure 5.10]. Parts of exercises 7.1, 7.3, 7.4, and 7.5 are inspired by our work [2, 4].

7.7 Bibliography

[1] Dana Angluin. Local and global properties in networks of processors. In *Proc. 12th Annual ACM Symposium on Theory of Computing (STOC 1980)*, 1980. doi:10.1145/800141.804655.

[2] Matti Åstrand, Valentin Polishchuk, Joel Rybicki, Jukka Suomela, and Jara Uitto. Local algorithms in (weakly) coloured graphs, 2010. arXiv:1002.0125.

[3] John A. Bondy and U. S. R. Murty. *Graph Theory with Applications*. North-Holland, 1976.

[4] 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. doi:10.1145/1835698.1835783.

[5] 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. doi:10.1109/71.481599.

7.8 Hints

A. (a) Apply the result of Exercise 2.8. (b) Find a 1-factor.

B. For the lower bound, use the result of Exercise 7.4c.

C. Show that if a 3-regular graph is homogeneous, then it has a 1-factor. Show that G does not have any 1-factor.