Ability to Count Messages Is Worth $\Theta(\Delta)$ Rounds in Distributed Computing

Tuomo Lempiäinen

Helsinki Institute for Information Technology HIIT, Department of Computer Science, Aalto University, Finland

tuomo.lempiainen@aalto.fi

Abstract

Hella et al. (PODC 2012, *Distributed Computing* 2015) identified seven different message-passing models of distributed computing—one of which is the port-numbering model—and provided a complete classification of their computational power relative to each other. However, their method for simulating the ability to count incoming messages causes an additive overhead of $2\Delta - 2$ communication rounds, and it was not clear if this is actually optimal. In this paper we give a positive answer, by using bisimulation as our main tool: there is a matching linear-in-Δ lower bound. This closes the final gap in our understanding of the models, with respect to the number of communication rounds. By a previously identified connection to modal logic, our result has implications to the relationship between multimodal logic and graded multimodal logic.

Categories and Subject Descriptors F.1.1 [*Computation by Abstract Devices*]: Models of Computation—Relations between models, unbounded-action devices; F.1.3 [*Computation by Abstract Devices*]: Complexity Measures and Classes—Complexity hierarchies, relations among complexity classes; F.4.1 [*Mathematical Logic and Formal Languages*]: Mathematical Logic—Modal logic; G.2.2 [*Discrete Mathematics*]: Graph Theory—Trees

Keywords distributed computing, port-numbering model, local algorithms, lower bounds, bisimulation

1. Introduction

This work studies the significance of being able to count the multiplicities of identical incoming messages in distributed algorithms. We compare two models: one, in which each node receives a *set* of messages in each round, and another, in which each node receives a *multiset* of messages in each round. It has been previously shown that the latter model can be simulated in the former model by allowing an additive overhead of linear in Δ communication rounds, where Δ is the maximum degree of the graph [9]. In this work we use bisimulation arguments to show that this is optimal: in some cases, linear in Δ extra rounds are strictly necessary.

1.1 Distributed Computing

In our framework of distributed computing, each node of an undirected graph runs the same algorithm. The graph is unknown to the algorithm and serves both as the communication network and the problem instance. The nodes communicate with adjacent nodes in synchronous rounds and eventually each node produces an output. The local outputs together constitute a solution to a graph problem—for instance, in the case of the vertex cover problem, we want each node to indicate whether it is part of the vertex cover. Running time is measured by the number of communication rounds, usually as a function of the number of nodes n or the maximum degree Δ.

We focus on deterministic distributed algorithms in anonymous networks—that is, nodes do not have unique identifiers. Instead, nodes can have *port numbers*: a node v of degree $\deg(v)$ can refer to its neighbours by numbers $1, 2, \ldots, \deg(v)$. If we have such a numbering and it is *consistent*, that is, *input port* i and *output port* i are always connected to the same neighbour, we arrive at the well-known *port-numbering model* introduced by Angluin [2].

1.2 A Hierarchy of Weak Models

The models that we study are weaker variants of the port-numbering model. Hella et al. [9] defined a collection of seven models, one of which is the port-numbering model. We denote by $\mathsf{VV_c}$ the class of all *graph problems* that can be solved in this model. The following subclasses of $\mathsf{VV_c}$ correspond to the weaker variants:

- VV: Input and output ports are numbered, but not necessarily consistently.
- MV: Only output ports are numbered; nodes receive a multiset of messages.
- SV: Only output ports are numbered; nodes receive a set of messages.
- VB: Only input ports are numbered; nodes broadcast the same message to all neighbours.
- MB: Combination of the restrictions of MV and VB.
- SB: Combination of the restrictions of SV and VB.

There are some trivial containment relations between the classes, such as $\mathsf{SV} \subseteq \mathsf{MV} \subseteq \mathsf{VV} \subseteq \mathsf{VV_c}$. The trivial relations are depicted in Figure 1a. However, some classes, such as VB and SV, are seemingly orthogonal. Somewhat surprisingly, Hella et al. [9] were able to show that the classes form a linear order:

$$\mathsf{SB} \subsetneq \mathsf{MB} = \mathsf{VB} \subsetneq \mathsf{SV} = \mathsf{MV} = \mathsf{VV} \subsetneq \mathsf{VV_c}.$$

For each class, we can also define the subclass of problems solvable in constant time independent of the size of the input graph. The same containment relations hold for the constant-time versions of the classes. The relations are depicted in Figure 1b.

The equalities between classes are proved by showing that algorithms corresponding to a seemingly more powerful class can be simulated by algorithms corresponding to a seemingly weaker class. In the case of $\mathsf{SV} = \mathsf{MV}$, there is an overhead involved, whereas the rest of the simulation results do not increase the running time.

LICS '16 July 5–8, 2016, New York, NY, USA
Copyright © 2016 ACM 978-1-4503-4391-6/16/07…$15.00
DOI: http://dx.doi.org/10.1145/2933575.2934567

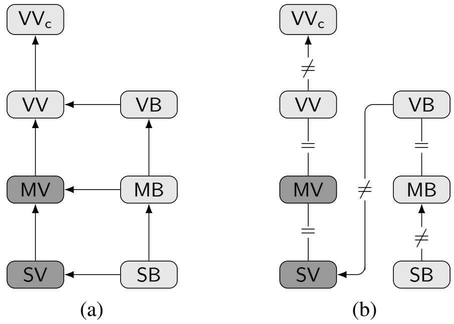

Figure 1: (a) Trivial containment relations between the problem classes. (b) The linear order obtained by Hella et al. [9].

1.3 Classes SV and MV

In this work we study further the relationship between the models that are related to the classes SV and MV. Neither of the models features incoming port numbers. The only difference is that in the case of MV, algorithms are able to count the number of neighbours that sent any particular message, while in SV this is not possible. For now we will use informally the terms SV-*algorithm* and MV-*algorithm*; a more formal definition will follow in Section 2.

Hella et al. [9] proved that any MV-algorithm can be simulated by an SV-algorithm, given that the simulating algorithm is allowed to use $2\Delta - 2$ extra communication rounds. The basic idea is that when nodes gather all available information from their radius-($2\Delta - 2$) neighbourhood, the outgoing port numbers necessarily break symmetry. Any neighbours u and w of a node v either have different outgoing port numbers towards v or see different local neighbourhoods. This symmetry-breaking information can then be used during the simulation to receive a distinct message from each neighbour.

1.4 Contributions

This work gives tight lower bounds for simulating MV-algorithms by SV-algorithms. We will prove two theorems. The first theorem is about a so-called simulation problem, that is, breaking symmetry between incoming messages. It is intended to be an exact counterpart to the upper bound result given by the simulation algorithm of Hella et al. [9].

Theorem 1. *For each $\Delta \geq 2$ there is a port-numbered graph of maximum degree Δ with nodes v, u, w, such that when executing any* SV*-algorithm in the graph, node v receives identical messages from its neighbours u and w in rounds $1, 2, \ldots, 2\Delta - 2$.*

Our second theorem gives a *graph problem* that separates MV-algorithms from SV-algorithms with respect to running time as a function of the maximum degree Δ.

Theorem 2. *There is a graph problem that can be solved in one communication round by an* MV*-algorithm, but that requires at least Δ rounds for each odd Δ and $\Delta - 1$ rounds for each even Δ, when solved by an* SV*-algorithm.*

Our results are based on a construction of a family of graphs with an intricate port numbering of certain kind. We start by proving Theorem 1 in Section 3, and then we adapt the same construction to prove Theorem 2 in Section 4.

In addition to studying the relationship between MV-algorithms and SV-algorithms, we aim to promote the use of tools from logic, in this case bisimulation, to advance the understanding of distributed computing.

1.5 Motivation and Related Work

The port-numbering model, or $\mathsf{VV_c}$, can be thought to model wired networks, whereas the model SB corresponds to fully wireless systems. Other models in the hierarchy are intermediate steps between the two extreme cases.

Models similar to MV have been studied previously under various names: output port awareness [6], wireless in input [5], mailbox [5], port-to-mailbox [18] and port-à-boîte [7]. However, most of the previous research does not give general results about graph problems, but instead focuses on individual problems or makes different assumptions about the model. To the best of our knowledge, the model SV has not been studied before the work of Hella et al. [9]. The constant-time version of SV can be seen as a special case of the distributed graph automata defined by Reiter [16], when restricted to bounded-degree graphs.

Emek and Wattenhofer [8] have considered networks of nodes with very limited computation and communication capabilities. In particular, in their model nodes can count identical messages only up to some predetermined number—this restriction can be seen as an intermediate step between our SV and MV models. They argue that this kind of restricted models will be crucial when applying distributed computing to networks of biological cells, where receiving a message corresponds to recognising the presence of some protein.

Our models have analogies also in graph exploration. The models SV and MV correspond to the case where an agent does not know from which edge it arrived to a node. This is true for *traversal sequences* [1], as opposed to *exploration sequences* [10]. If we have several agents exploring a graph, the question of whether they can count the number of identical agents in a node becomes interesting. Our lower bounds indicate that, with appropriate definitions, this ability causes a difference of linear in Δ steps in certain traversal sequences.

Hella et al. [9] identified a connection between the seven models of computation and certain variants of modal logic, in the spirit of descriptive complexity theory. In certain classes of structures, *graded multimodal logic* corresponds to MV and *multimodal logic* corresponds to SV. Thus our lower-bound result implies a new separation between multimodal logic and graded multimodal logic: when given a formula ϕ of graded multimodal logic, we can find a formula ψ of multimodal logic that is equivalent to ϕ in a certain class of structures, but in general, the modal depth $\mathrm{md}(\psi)$ of ψ has to be at least $\mathrm{md}(\phi) + \Delta - 1$. For details on modal logic, see Blackburn, de Rijke and Venema [3] or Blackburn, van Benthem and Wolter [4].

Somewhat analogously to our work, Krebs and Verbitsky [11] have studied universal covers of graphs by making use of a bisimulation version of the 2-pebble counting game. They showed that there exist graphs G and H, and their nodes u and v, respectively, such that a distributed algorithm needs at least $2n - 16\sqrt{n}$ rounds to distinguish between u and v, assuming that it is possible. Further examples of the use of logic in distributed computing have been given by Kuusisto [12, 13].

2. Preliminaries

In this section we define the models of computation and the problems we study, as well as introduce tools that will be needed in order to prove our results.

2.1 Distributed Algorithms

We define distributed algorithms as state machines. They are executed in a graph such that each node of the graph is a copy of the same state machine. Nodes can communicate with adjacent nodes.

In this work, we consider only deterministic state machines and synchronous communication in anonymous networks.

In the beginning of execution, each state machine is initialised based on the degree of the node and a possible local input given to it. Then, in each communication round, each state machine performs three operations:

(1) sends a message to each neighbour,
(2) receives a message from each neighbour,
(3) moves to a new state based on the current state and the received messages.

If the new state belongs to a set of special stopping states, the machine halts. The local output of the node is its state after halting. Next, we will define distributed systems more formally.

2.1.1 Inputs and Port Numberings

Consider an undirected graph $G = (V, E)$. An *input* for G is a function $f\colon V \to X$, where X is a finite set such that $\emptyset \in X$. For each $v \in V$, the value $f(v)$ is called the *local input* of v.

A *port* of G is a pair (v, i), where $v \in V$ is a node and $i \in [\deg(v)]$ is the number of the port. Let $P(G)$ be the set of all ports of G. A *port numbering* of G is a bijection $p\colon P(G) \to P(G)$ such that

$$p(v, i) = (u, j) \quad \text{for some } i \in [\deg(v)] \text{ and } j \in [\deg(u)]$$

if and only if $\{v, u\} \in E$. Intuitively, if $p(v, i) = (u, j)$, then (v, i) is the ith output port of node v, and it is connected to (u, j), which is the jth input port of node u.

When analysing lower-bound constructions, we will find the following generalisation of port numbers useful. Let N be an arbitrary set. Assume that for each $v \in V$, $I_v \subseteq N$ and $O_v \subseteq N$ are subsets of size $\deg(v)$. Now, a *generalised input port* is a pair (v, i), where $v \in V$ and $i \in I_v$, and a *generalised output port* is a pair (v, o), where $v \in V$ and $o \in O_v$. A *generalised port numbering* p is then a bijection from the set of generalised output ports to the set of generalised input ports such that

$$p(v, o) = (u, i) \quad \text{for some } o \in O_v \text{ and } i \in I_u$$

if and only if $\{v, u\} \in E$.

2.1.2 State Machines

For each positive integer Δ, denote by $\mathcal{F}(\Delta)$ the class of all simple undirected graphs of maximum degree at most Δ. Let $X \ni \emptyset$ be a finite set of local inputs. A *distributed state machine* for $(\mathcal{F}(\Delta), X)$ is a tuple $\mathcal{A} = (Y, Z, \sigma_0, M, \mu, \sigma)$, where

- Y is a set of states,
- $Z \subseteq Y$ is a finite set of stopping states,
- $\sigma_0\colon \{0, 1, \dots, \Delta\} \times X \to Y$ is a function that defines the initial state,
- M is a set of messages such that $\epsilon \in M$,
- $\mu\colon Y \times [\Delta] \to M$ is a function that constructs the outgoing messages, such that $\mu(z, i) = \epsilon$ for all $z \in Z$ and $i \in [\Delta]$,
- $\sigma\colon Y \times M^\Delta \to Y$ is a function that defines the state transitions, such that $\sigma(z, \overline{m}) = z$ for all $z \in Z$ and $\overline{m} \in M^\Delta$.

The special symbol $\epsilon \in M$ indicates "no message" and $\emptyset$ indicates "no input".

2.1.3 Executions

Let $G = (V, E) \in \mathcal{F}(\Delta)$ be a graph, let p be a port numbering of G, let $f\colon V \to X$ be an input for G, and let $\mathcal{A}$ be a distributed state machine for $(\mathcal{F}(\Delta), X)$. Then we can define the *execution* of $\mathcal{A}$ in (G, f, p) as follows.

The state of the system in round $r \in \mathbb{N}$ is represented as a function $x_r\colon V \to Y$, where $x_r(v)$ is the *state* of node v in round r. To initialise the nodes, set $x_0(v) = \sigma_0(\deg(v), f(v))$ for each $v \in V$.

Then, assume that x_r is defined for some $r \in \mathbb{N}$. Let $(u, j) \in P(G)$ and $(v, i) = p(u, j)$. Now, node v receives the message $a_{r+1}(v, i) = \mu(x_r(u), j)$ from its port (v, i) in round $r + 1$. For each $v \in V$, we define a vector of length Δ consisting of messages received by node v in round $r + 1$ and the symbol ϵ:

$$\overline{a}_{r+1}(v) = (a_{r+1}(v, 1), \dots, a_{r+1}(v, \deg(v)), \epsilon, \dots, \epsilon),$$

where the padding with the special symbol ϵ is to simplify our notation so that $\overline{a}_{r+1}(v) \in M^\Delta$. Now we can define the new state of each node $v \in V$ as follows:

$$x_{r+1}(v) = \sigma(x_r(v), \overline{a}_{r+1}(v)).$$

Let $t \in \mathbb{N}$. If $x_t(v) \in Z$ for all $v \in V$, we say that $\mathcal{A}$ *stops in time t* in (G, f, p). The *running time* of $\mathcal{A}$ in (G, f, p) is the smallest t for which this holds. If $\mathcal{A}$ stops in time t in (G, f, p), the *output* of $\mathcal{A}$ in (G, f, p) is $x_t\colon V \to Y$. For each $v \in V$, the *local output* of v is $x_t(v)$.

We define the *execution* of $\mathcal{A}$ in (G, p) to be the execution of $\mathcal{A}$ in (G, f, p), where f is the unique function $f\colon V \to \{\emptyset\}$.

2.1.4 Algorithm Classes

So far, we have defined only a single model of computation. However, our aim in this work is to investigate the relationships between two variants of the model. To this end, we will now introduce two different restrictions to the definition of a state machine.

Given a vector $\overline{a} = (a_1, a_2, \dots, a_\Delta) \in M^\Delta$, define

$$\begin{aligned} \mathrm{set}(\overline{a}) &= \{a_1, a_2, \dots, a_\Delta\}, \\ \mathrm{multiset}(\overline{a}) &= \{(m, n) : m \in M, n = |\{i \in [\Delta] : m = a_i\}|\}. \end{aligned}$$

That is, $\mathrm{set}(\overline{a})$ discards the ordering and multiplicities of the elements of $\overline{a}$, while $\mathrm{multiset}(\overline{a})$ discards only the ordering.

Now we can define the classes $\mathcal{SV}$ and $\mathcal{MV}$ of state machines. The class $\mathcal{SV}$ consists of all distributed state machines $\mathcal{A} = (Y, Z, \sigma_0, M, \mu, \sigma)$ such that

$$\mathrm{set}(\overline{a}) = \mathrm{set}(\overline{b}) \quad \text{implies} \quad \sigma(y, \overline{a}) = \sigma(y, \overline{b})$$

for all $y \in Y$. Similarly, the class $\mathcal{MV}$ consists of all distributed state machines $\mathcal{A} = (Y, Z, \sigma_0, M, \mu, \sigma)$ such that

$$\mathrm{multiset}(\overline{a}) = \mathrm{multiset}(\overline{b}) \quad \text{implies} \quad \sigma(y, \overline{a}) = \sigma(y, \overline{b})$$

for all $y \in Y$.

The idea here is that for state machines in $\mathcal{MV}$, the state transitions are invariant with respect to the order of incoming messages; in practice, nodes receive the messages in a multiset. In $\mathcal{SV}$, nodes receive the messages in a set, which means that the state transitions are invariant with respect to both the order and multiplicities of incoming messages.

We will later find use for the following definitions for infinite sequences of state machines, where Δ will be used as an upper bound for the maximum degree of graphs:

$$\begin{aligned} \mathbf{MV} &= \{(\mathcal{A}_1, \mathcal{A}_2, \dots) : \mathcal{A}_\Delta \in \mathcal{MV} \text{ for all } \Delta\}, \\ \mathbf{SV} &= \{(\mathcal{A}_1, \mathcal{A}_2, \dots) : \mathcal{A}_\Delta \in \mathcal{SV} \text{ for all } \Delta\}. \end{aligned}$$

From now on, both distributed state machines $\mathcal{A} \in \mathcal{MV} \cup \mathcal{SV}$ and sequences of distributed state machines $\mathbf{A} \in \mathbf{MV} \cup \mathbf{SV}$ will be referred to as *algorithms*. The precise meaning should be clear from the notation.

2.2 Graph Problems

Let X and Y be finite nonempty sets. A *graph problem* is a function $\Pi_{X,Y}$ that maps each undirected simple graph $G = (V, E)$ and each input $f\colon V \to X$ to a set $\Pi_{X,Y}(G, f)$ of solutions. Each *solution* $S \in \Pi_{X,Y}(G, f)$ is a function $S\colon V \to Y$. We handle problems without local input by setting $X = \{\emptyset\}$. One can see that our definition covers a large selection of typical distributed computing problems, such as those where the task is to find a subset or colouring of vertices.

Let $\Pi_{X,Y}$ be a graph problem, $T\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ a function and $\mathbf{A} = (\mathcal{A}_1, \mathcal{A}_2, \dots)$ a sequence such that each $\mathcal{A}_\Delta$ is a distributed state machine for $(\mathcal{F}(\Delta), X)$. We say that $\mathbf{A}$ *solves* $\Pi_{X,Y}$ *in time* T if the following conditions hold for all $\Delta \in \mathbb{N}$, all finite graphs $G = (V, E) \in \mathcal{F}(\Delta)$, all inputs $f\colon V \to X$ and all port numberings p of G:

(1) $\mathcal{A}_\Delta$ stops in time $T(\Delta, |V|)$ in (G, f, p).
(2) The output of $\mathcal{A}_\Delta$ in (G, f, p) is in $\Pi_{X,Y}(G, f)$.

If there exists a function $T\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ such that $\mathbf{A}$ solves $\Pi_{X,Y}$ in time T, we say that $\mathbf{A}$ *solves* $\Pi_{X,Y}$ or that $\mathbf{A}$ *is an algorithm for* $\Pi_{X,Y}$. If the value $T(\Delta, n)$ does not depend on n, that is, if we have $T(\Delta, n) = T'(\Delta)$ for some function $T'\colon \mathbb{N} \to \mathbb{N}$, we say that $\mathbf{A}$ *solves* $\Pi_{X,Y}$ *in constant time* or that $\mathbf{A}$ *is a local algorithm for* $\Pi_{X,Y}$.

2.2.1 Problem Classes

Now we are ready to define complexity classes based on our different notions of algorithms:

- MV consists of problems Π such that there is an algorithm $\mathbf{A} \in \mathbf{MV}$ that solves Π.
- SV consists of problems Π such that there is an algorithm $\mathbf{A} \in \mathbf{SV}$ that solves Π.

For both classes, we can also define their constant-time variants $\mathsf{MV}(1)$ and $\mathsf{SV}(1)$ that are subclasses of MV and SV restricted to problems solvable in constant time.

Observe that it follows trivially from the definitions of the algorithm classes that $\mathsf{SV} \subseteq \mathsf{MV}$ and $\mathsf{SV}(1) \subseteq \mathsf{MV}(1)$. It was shown by Hella et al. [9] that we actually have $\mathsf{SV} = \mathsf{MV}$ and $\mathsf{SV}(1) = \mathsf{MV}(1)$.

2.3 Bisimulation

In this section we introduce bisimulation—and in particular, its finite approximation, r-bisimulation—which we will need when proving lower-bound results in Sections 3 and 4. Simply put, a bisimulation is a relation between two structures such that related elements have identical local information and equivalent relations to other elements. For more details on bisimulation in general, see Blackburn, de Rijke and Venema [3].

Hella et al. [9] demonstrated the use of bisimulation in distributed computing by capturing the models of computation by modal logics. Here we take a self-contained approach by showing directly that bisimilarity implies indistinguishability by distributed algorithms.

The general concept of bisimulation can be adapted to take into account the different amounts of information that is available to algorithms in each model. We will need only one variant in this work, the one corresponding to the class $\mathcal{SV}$.

Definition 3. Let $G = (V, E)$ and $G' = (V', E')$ be graphs, let f and f' be inputs for G and G', respectively, and let p and p' be generalised port numberings of G and G', respectively. We define r-$\mathcal{SV}$-bisimilarity recursively. As a base case, we say that nodes $v \in V$ and $v' \in V'$ are 0-$\mathcal{SV}$-*bisimilar* if $\deg_G(v) = \deg_{G'}(v')$ and $f(v) = f'(v')$. For $r \in \mathbb{N}_+$, we say that $v \in V$ and $v' \in V'$ are r-$\mathcal{SV}$-*bisimilar* if the following conditions hold:

(B1) Nodes v and v' are 0-$\mathcal{SV}$-bisimilar.
(B2) If $\{v, w\} \in E$, then there is $w' \in V'$ with $\{v', w'\} \in E'$ such that w and w' are $(r-1)$-$\mathcal{SV}$-bisimilar, and $p(w, a) = (v, b)$ and $p'(w', a) = (v', c)$ hold for some a, b, c.
(B3) If $\{v', w'\} \in E'$, then there is $w \in V$ with $\{v, w\} \in E$ such that w and w' are $(r-1)$-$\mathcal{SV}$-bisimilar, and $p(w, a) = (v, b)$ and $p'(w', a) = (v', c)$ hold for some a, b, c.

If $v \in V$ and $v' \in V'$ are r-$\mathcal{SV}$-*bisimilar*, we write $(G, f, v, p) \leftrightarrows_r^{\mathcal{SV}} (G', f', v', p')$—or simply $v \leftrightarrows_r^{\mathcal{SV}} v'$, if the graphs, inputs and generalised port numberings are clear from the context.

It is easy to show by induction that if $v \leftrightarrows_r^{\mathcal{SV}} v'$ holds for some r, then $v \leftrightarrows_t^{\mathcal{SV}} v'$ holds for all $t = 0, 1, \dots, r$. The following three lemmas enable us to apply bisimulation to distributed algorithms; the proofs are straightforward induction arguments and can be found in the full version of this work [15]. Our first lemma shows that r-bisimilarity entails indistinguishability by distributed algorithms up to running time r.

Lemma 4. *Let $G = (V, E)$ and $G' = (V', E')$ be graphs, let f and f' be inputs for G and G', respectively, and let p and p' be port numberings of G and G', respectively. If $(G, f, v, p) \leftrightarrows_r^{\mathcal{SV}} (G', f', v', p')$ for some $r \in \mathbb{N}$, $v \in V$ and $v' \in V'$, then for all algorithms $\mathcal{A} \in \mathcal{SV}$ we have $x_t(v) = x'_t(v')$ for all $t = 0, 1, \dots, r$, that is, the states of v and v' are identical in rounds $0, 1, \dots, r$.*

It is quite easy to show that r-$\mathcal{SV}$-bisimilarity is an equivalence relation. Since we will only need transitivity in this work, the following lemma suffices.

Lemma 5. *The r-$\mathcal{SV}$-bisimilarity relation $\leftrightarrows_r^{\mathcal{SV}}$ is transitive in the class of quadruples (G, f, v, p), where $G = (V, E)$ is a graph, f is an input for G, p is a generalised port numbering of G and $v \in V$.*

Finally, when given a generalised port numbering and a bisimilarity result, we need to be able to introduce an ordinary port numbering in order to actually apply the result to distributed algorithms. The following lemma shows that we can do this.

Lemma 6. *Let $G = (V, E)$ and $G' = (V', E')$ be graphs, let f and f' be inputs for G and G', respectively, and let p and p' be generalised port numberings of G and G', respectively, with port numbers taken from a set N. Suppose that q and q' are port numberings of G and G', respectively, such that $p(v, i) = (u, j)$ implies $q(v, g(i)) = (u, g(j))$ and $p'(v, i) = (u, j)$ implies $q'(v, g(i)) = (u, g(j))$ for some function $g\colon N \to \mathbb{N}_+$. Then $(G, f, v, p) \leftrightarrows_r^{\mathcal{SV}} (G', f', v', p')$ implies $(G, f, v, q) \leftrightarrows_r^{\mathcal{SV}} (G', f', v', q')$ for all $v \in V$ and $v' \in V'$.*

Traditionally, the technique of local views [17] has often been used in distributed computing to show lower-bound and impossibility results. We advocate for bisimulation as a worthwhile alternative—the proof of Lemma 4 is particularly straightforward and thus demonstrates the naturalness of the concept. While bisimulation was used by Hella et al. [9], to our knowledge the present work is the first to apply the finite version, r-bisimulation, in the context of distributed computing.

3. A Lower Bound for the Simulation Overhead

Let us begin by restating the result that we will prove in this section.

Theorem 1. *For each $\Delta \geq 2$ there is a graph $G = (V, E) \in \mathcal{F}(\Delta)$, a port numbering p of G and nodes $v, u, w \in V$ such that when executing any algorithm $\mathcal{A} \in \mathcal{SV}$ in (G, p), node v receives identical messages from its neighbours u and w in rounds $1, 2, \dots, 2\Delta - 2$.*

To prove Theorem 1, we define for each $d = 2, 3, \dots$ a graph $G_d = (V_d, E_d)$ of maximum degree d. The graph itself is just a rooted tree, but it gives rise to a port numbering with certain properties. We construct the graph so that two neighbours u and w of the root node are in a sense as symmetrical as possible—with the exception that they have the same outgoing port number towards the root, while the root obviously cannot have the same port number towards them. By symmetrical we mean that if a node u' in the neighbourhood of u has a neighbour with outgoing port number i towards u', then there is a corresponding node v' in the neighbourhood of v with a neighbour having port number i towards v', and vice versa. Intuitively, one can start by assigning the port numbers 1 and 2 to the connections between the root and its two neighbours, and then add a new node whenever needed to satisfy the above notion of symmetry, until the maximum degree d is reached.

The set V_d of nodes consists of sequences of pairs (i, j), where $i, j \in \{0, 1, \dots, d\}$ will serve as a basis for port numbers, as we will see later. Each sequence can be thought as a path leading from the root to the node itself. The fundamental idea of the definition is that we construct the graph one level of nodes at a time, starting from the root, and assign generalised port numbers to each edge of a node by choosing the smallest numbers that have not yet been taken. The choice depends slightly on whether the level in question is even or odd—due to the unavoidable asymmetry in the root. Note that at this stage we will use the number 0 in our construction, but it will be replaced by 1 when defining the actual port numbering.

We define the set V_d of nodes recursively as follows:

(G1) $\emptyset \in V_d$.

(G2) $((1,0)), ((2,1)), ((3,2)), ((4,3)), \dots, ((d, d-1)) \in V_d$.

(G3) If $(a_1, a_2, \dots, a_i) \in V_d$, where i is odd and $i \le 2d-1$, then $(a_1, a_2, \dots, a_i, a^j_{i+1}) \in V_d$ for all $j = 1, 2, \dots, d-1$, where $a^j_{i+1} = (c^j_1, c^j_2)$ is defined as follows. Let $(b_1, b_2) = a_i$ and $b^+_2 = 1$ if $b_2 = 0$, $b^+_2 = b_2$ otherwise. Define

$$c^j_1 = \min(\{1, 2, \dots, d\} \setminus \{b^+_2, c^1_1, c^2_1, \dots, c^{j-1}_1\}),$$
$$c^j_2 = \min(\{1, 2, \dots, d\} \setminus \{b_1, c^1_2, c^2_2, \dots, c^{j-1}_2\}).$$

(G4) If $(a_1, a_2, \dots, a_i) \in V_d$, where i is even and $2 \le i \le 2d-2$, then $(a_1, a_2, \dots, a_i, a^j_{i+1}) \in V_d$ for all $j = 1, 2, \dots, d-1$, where $a^j_{i+1} = (c^j_1, c^j_2)$ is defined as follows. Let $(b_1, b_2) = a_i$. Define

$$c^j_1 = \min(\{1, 2, \dots, d\} \setminus \{b_2, c^1_1, c^2_1, \dots, c^{j-1}_1\}),$$
$$c^j_2 = \min(\{0, 1, \dots, d-1\} \setminus \{b_1, c^1_2, c^2_2, \dots, c^{j-1}_2\}).$$

The set E_d of edges consists of all pairs $\{v, u\}$, where $v = (a_1, a_2, \dots, a_i) \in V_d$ and $u = (a_1, a_2, \dots, a_i, a_{i+1}) \in V_d$ for some $i \in \{0, 1, \dots\}$. Figures 2 and 3 below provide illustrations of the graphs G_3 and G_6.

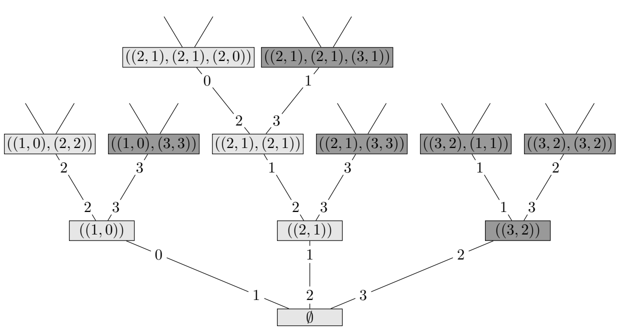

Figure 2: A part of the graph G_3, with named nodes. The nodes with a light background are contained already in the graph G_2, while the nodes with a dark background are only in G_d for $d \ge 3$.

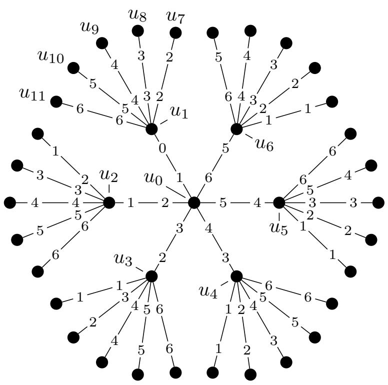

Figure 3: The radius-2 neighbourhood of node $u_0 = \emptyset$ of graph G_6. We have for example $u_1 = ((1, 0))$, $u_2 = ((2, 1))$ and $u_7 = ((1, 0), (2, 2))$.

Consider nodes $v = (a_1, a_2, \dots, a_i)$ and $u = (a_1, a_2, \dots, a_{i+1})$, where $a_{i+1} = (b_1, b_2)$. The values b_1 and b_2 serve as generalised port numbers for the edge $\{v, u\}$. We define $p_d(v, b_1) = (u, b_2)$ and $p_d(u, b_2) = (v, b_1)$. The incoming port numbers will be irrelevant in this proof, since we only consider algorithms in the classes $\mathcal{SV}$ and $\mathcal{MV}$. Thus, we will mostly use the notation $\pi_d(v, u) = b_1$ and $\pi_d(u, v) = b_2$ to denote the outgoing port numbers.

If $v = (a_1, a_2, \dots, a_i)$ and $u = (a_1, a_2, \dots, a_{i+1})$, we say that node v is the *parent* of node u and that u is a *child* of v. We say that the node v is *even* if i is even and *odd* if i is odd. If $a_i = (b_1, b_2)$, we call (b_1, b_2) the *type* of node v.

A *walk* is a sequence $\overline{v} = (v_0, v_1, \dots, v_k)$ of nodes such that $\{v_i, v_{i+1}\} \in E_d$ for all $i = 0, 1, \dots, k-1$. A pair $(\overline{v}_1, \overline{v}_2)$ of walks, where $\overline{v}_i = (v^i_0, v^i_1, \dots, v^i_k)$ for all $i = 1, 2$, and $k \le 2d-3$, is called a *pair of compatible walks (PCW) of length k in G_d* if the following two conditions hold:

(W1) $v^1_0 = ((1,0))$ and $v^2_0 = ((2,1))$.

(W2) $\pi_d(v^1_j, v^1_{j-1}) = \pi_d(v^2_j, v^2_{j-1})$ for all $j = 1, 2, \dots, k$.

A PCW $(\overline{v}_1, \overline{v}_2)$ is called a *pair of separating walks (PSW)*, if $\overline{v}_1$ can be extended in such a way that $\overline{v}_2$ cannot be extended and still remain compatible, that is, if the following holds:

(W3) There is $v^1_{k+1} \in V_d$ with $\{v^1_k, v^1_{k+1}\} \in E_d$ such that there is no $v^2_{k+1} \in V_d$ for which $\{v^2_k, v^2_{k+1}\} \in E_d$ and $\pi_d(v^1_{k+1}, v^1_k) = \pi_d(v^2_{k+1}, v^2_k)$.

We say that a pair of separating walks of length k in G_d is *critical* if there does not exist a pair of separating walks of length k' in G_d for any $k' < k$.

Consider the graph G_6 in Figure 3. One example of a PSW in G_6 is the pair $(\overline{v}_1, \overline{v}_2)$, where $\overline{v}_1 = (u_1, u_7, u_1, u_8, u_1, u_9, u_1, u_{10}, u_1, u_{11})$ and $\overline{v}_2 = (u_2, u_0, u_3, u_0, u_4, u_0, u_5, u_0, u_6, u_0)$. Here the corresponding sequence of generalised port numbers given by π_6 is 2, 2, 3, 3, 4, 4, 5, 5, 6. That is, the walks are going back and forth between one node and its neighbours—but this does not hold for PSWs in general. Observe that now node u_{11} has a neighbour u_1 with $\pi_6(u_1, u_{11}) = 6$, but node u_0 does not have such a neighbour.

As we will see later, the fact that the sequence grows only by small increments and eventually reaches the parameter d is actually a general property of PSWs; this is one of the crucial ideas behind our proof. While one can show with moderate effort that the generalised port numbers increase at most linearly along the walks, showing a tight upper bound for the increase—and thus a tight lower bound for the length of the walks—appears to be considerably non-trivial.

The outline of the proof is as follows. First, we will prove auxiliary results concerning the graphs G_d and PSWs. These will enable us to obtain a tight lower bound for the length of PSWs. Then, we will show that this lower bound entails bisimilarity of the nodes $((1,0))$ and $((2,1))$ up to the respective distance. The first four lemmas follow quite easily from the definition of the graphs; see the full version of this work [15] for proofs.

Lemma 7. *For each d, we have $\deg(v) \in \{1, d\}$ for all $v \in V_d$, and thus $G_d \in \mathcal{F}(d)$. Additionally, G_d is a subgraph of G_{d+1}.*

Lemma 8. *Let $v \in V_d$ and $a \in \{0, 1, \ldots, d\}$. Then there is at most one node $u \in V_d$ such that $\{v, u\} \in E_d$ and $\pi_d(u, v) = a$.*

A consequence of Lemma 8 is that in a walk, the successor of each node is uniquely determined by the port number from the successor to the node.

Lemma 9. *Let $v = (a_1, a_2, \ldots, a_i) \in V_d$, where $i < 2d$. If v is odd, then for all $a \in \{1, 2, \ldots, d\}$ there exists $u \in V_d$ such that $\{v, u\} \in E_d$ and $\pi_d(u, v) = a$. If v is even, then either for all $a \in \{0, 1, \ldots, d-1\}$ or for all $a \in \{0, 1, \ldots, d-2, d\}$ there exists $u \in V_d$ such that $\{v, u\} \in E_d$ and $\pi_d(u, v) = a$. In the case of even v and $a = d$, node u is the parent of node v.*

Lemma 9 implies that in a PSW, the last nodes of each walk must be even. Furthermore, one of the last nodes v must have a parent u with $\pi_d(u, v) = d$. It follows that we must have $v \in V_d \setminus V_{d-1}$.

Our next lemma reflects the fact that when going from G_d to G_{d+1}, old nodes can only get new children, not new parents, and that new port numbers are necessarily large.

Lemma 10. *Let $\{v, u\} \in E_{d+1} \setminus E_d$ be such that $v \in V_d$. Then u is a child of v. If v is odd, then $\pi_{d+1}(v, u) = \pi_{d+1}(u, v) = d+1$. If v is even, then $\pi_{d+1}(v, u) = d+1$ and $\pi_{d+1}(u, v) \in \{d-1, d\}$.*

With the above observations out of the way, we now go forward with more powerful results.

If the walks of a PSW go to child nodes through edges with similar port numbers, they end up in subtrees that are isomorphic up to a depth larger than the number of steps left in the walks. Thus they eventually have to return from the subtrees. This makes them longer than they need to be, as the following lemma shows.

Lemma 11. *Let $(\overline{v}_1, \overline{v}_2)$, where $\overline{v}_i = (v^i_0, v^i_1, \ldots, v^i_k)$ for all $i = 1, 2$, be a PSW in G_d. If for some $\ell \in \{0, 1, \ldots, k-1\}$ the node $v^i_{\ell+1}$ is a child of node v^i_ℓ for all $i = 1, 2$, and we have $\pi_d(v^1_\ell, v^1_{\ell+1}) = \pi_d(v^2_\ell, v^2_{\ell+1})$, then $(\overline{v}_1, \overline{v}_2)$ is not a critical PSW in G_d.*

Proof. Suppose that for all $m = \ell+2, \ell+3, \ldots, k$ we have $v^1_m \neq v^1_\ell$ or $v^2_m \neq v^2_\ell$. By assumption, $v^1_{\ell+1}$ and $v^2_{\ell+1}$ are of the same type. Consider the definition of G_d. Now it is easy to show by induction on m that nodes v^1_m and v^2_m are of the same type for all $m = \ell+1, \ell+2, \ldots, k$. Since $k \leq 2d-3$ by the definition of a PSW, both v^1_k and v^2_k have child nodes. It follows that if v^1_{k+1} is a neighbour of v^1_k, there is a neighbour v^2_{k+1} of v^2_k such that $\pi_d(v^1_{k+1}, v^1_k) = \pi_d(v^2_{k+1}, v^2_k)$. Thus $(\overline{v}_1, \overline{v}_2)$ is not a PSW in G_d, a contradiction.

Now $v^1_m = v^1_\ell$ and $v^2_m = v^2_\ell$ for some $m \in \{\ell+2, \ell+3, \ldots, k\}$. Let

$$\overline{v}'_i = (v^i_0, v^i_1, \ldots, v^i_\ell, v^i_{m+1}, v^i_{m+2}, \ldots, v^i_k)$$

for all $i = 1, 2$. Then $(\overline{v}'_1, \overline{v}'_2)$ is a PSW of length $k - m + \ell \leq k - (\ell+2) + \ell = k - 2 < k$ in G_d and hence $(\overline{v}_1, \overline{v}_2)$ is not critical. □

Next we show that we can extend a PSW by adding two nodes to each walk to get a new PSW in a larger graph.

Lemma 12. *Let $(\overline{v}_1, \overline{v}_2)$ be a PSW of length k in G_d. Then there is a PSW of length $k+2$ in G_{d+1}.*

Proof. Let $\overline{v}_i = (v^i_0, v^i_1, \ldots, v^i_k)$ for all $i = 1, 2$. By definition, there is a neighbour $u \in V_d$ of v^1_k such that for each neighbour $w \in V_d$ of v^2_k we have $\pi_d(u, v^1_k) \neq \pi_d(w, v^2_k)$. Lemma 9 implies that v^1_k and v^2_k are even, $\pi_d(u, v^1_k) \in \{d-1, d\}$, and there is a neighbour $w \in V_d$ of v^2_k for which $\pi_d(w, v^2_k) \in \{d-1, d\} \setminus \{\pi_d(u, v^1_k)\}$. That is, we have $\pi_d(u, v^1_k) = d$ or $\pi_d(w, v^2_k) = d$. Without loss of generality, we can assume $\pi_d(u, v^1_k) = d$ and thus $\pi_d(w, v^2_k) = d-1$.

Lemma 7 implies that $\deg_{G_d}(u) = \deg_{G_d}(v^2_k) = d$ and $\deg_{G_{d+1}}(u) = \deg_{G_{d+1}}(v^2_k) = d+1$. Hence there are nodes $x, y \in V_{d+1} \setminus V_d$ such that $\{u, x\} \in E_{d+1} \setminus E_d$ and $\{v^2_k, y\} \in E_{d+1} \setminus E_d$. Note that $u, v^2_k \in V_d$, u is odd and v^2_k is even. It follows from Lemma 10 that $\pi_{d+1}(u, x) = \pi_{d+1}(x, u) = d+1$, $\pi_{d+1}(v^2_k, y) = d+1$ and $\pi_{d+1}(y, v^2_k) \in \{d-1, d\}$. Since $\pi_{d+1}(w, v^2_k) = \pi_d(w, v^2_k) = d-1$ and $w \neq y$, Lemma 8 implies that $\pi_{d+1}(y, v^2_k) = d$.

Now we can extend the walks $\overline{v}_1$ and $\overline{v}_2$. Set $\overline{v}'_1 = (v^1_0, v^1_1, \ldots, v^1_k, u, x)$ and $\overline{v}'_2 = (v^2_0, v^2_1, \ldots, v^2_k, y, v^2_k)$. We have $\pi_{d+1}(u, v^1_k) = d = \pi_{d+1}(y, v^2_k)$ and $\pi_{d+1}(x, u) = d+1 = \pi_{d+1}(v^2_k, y)$, as required. Furthermore, node x has neighbour u for which $\pi_{d+1}(u, x) = d+1$. Suppose that there is a neighbour u' of v^2_k for which $\pi_{d+1}(u', v^2_k) = d+1$. Now Lemma 9 implies that u' is the parent of v^2_k. But since $v^2_k \in V_d$, we have also $u' \in V_d$, and hence $\pi_{d+1}(u', v^2_k) \leq d$, a contradiction. Similarly, node v^2_k has neighbour y for which $\pi_{d+1}(y, v^2_k) = d$, but $\pi_{d+1}(u, x) = d+1$ together with Lemma 9 implies that there is no neighbour y' of x for which $\pi_{d+1}(y', x) = d$. This shows that $(\overline{v}'_1, \overline{v}'_2)$ is a PSW of length $k+2$ in G_{d+1}. □

As the following lemma points out, the second-to-last node of at least one walk in a PSW does not belong to any smaller construction, that is, it is a new node when going from G_{d-1} to G_d.

Lemma 13. *Let $(\overline{v}_1, \overline{v}_2)$, where $\overline{v}_i = (v^i_0, v^i_1, \ldots, v^i_k)$ for all $i = 1, 2$, be a critical PSW in G_d. Then we have $v^i_{k-1} \in V_d \setminus V_{d-1}$ for some $i \in \{1, 2\}$.*

Proof. Lemma 9 implies that v^1_k and v^2_k are even, and for some $i \in \{1, 2\}$ node v^i_k has a parent u such that $\pi_d(u, v^i_k) = d$. If $v^i_k \in V_{d-1}$, then also $u \in V_{d-1}$ and hence $\pi_d(u, v^i_k) \leq d-1$, a contradiction. Therefore $v^i_k \in V_d \setminus V_{d-1}$.

Suppose that $v^j_{k-1} \in V_{d-1}$ for all $j = 1, 2$. Since $v^i_k \in V_d \setminus V_{d-1}$, we have $\{v^i_{k-1}, v^i_k\} \in E_d \setminus E_{d-1}$. Lemma 10 implies that v^i_k is a child of v^i_{k-1} and $\pi_d(v^i_{k-1}, v^i_k) = \pi_d(v^i_k, v^i_{k-1}) = d$. Let $j \in \{1, 2\} \setminus \{i\}$. As $\pi_d(v^j_k, v^j_{k-1}) = \pi_d(v^i_k, v^i_{k-1}) = d$, we have $\{v^j_{k-1}, v^j_k\} \in E_d \setminus E_{d-1}$ and thus v^j_k is a child of v^j_{k-1} and $\pi_d(v^j_{k-1}, v^j_k) = \pi_d(v^j_k, v^j_{k-1}) = d$. Now it follows from Lemma 11 that $(\overline{v}_1, \overline{v}_2)$ is not a critical PSW in G_d, a contradiction. □

Lemma 14 basically states the negation of condition (W3), with the additional observation that the walks of a PCW have symmetrical roles.

Lemma 14. *Let $(\overline{v}_1, \overline{v}_2)$, where $\overline{v}_i = (v^i_0, v^i_1, \ldots, v^i_k)$ for all $i = 1, 2$, be a PCW in G_d. If $(\overline{v}_1, \overline{v}_2)$ is not a PSW in G_d, then for each neighbour $v^1_{k+1} \in V_d$ of v^1_k there is a neighbour $v^2_{k+1} \in V_d$ of v^2_k such that $\pi_d(v^1_{k+1}, v^1_k) = \pi_d(v^2_{k+1}, v^2_k)$, and vice versa.*

Proof. Since $(\overline{v}_1, \overline{v}_2)$ is not a PSW, condition (W3) does not hold. This is equivalent to the first claim. For the second claim, assume that v^2_{k+1} is a neighbour of v^2_k. Suppose that there is no neighbour v^1_{k+1} of v^1_k such that $\pi_d(v^1_{k+1}, v^1_k) = \pi_d(v^2_{k+1}, v^2_k)$. Now it follows from

Lemma 7 and Lemma 9 that v_k^1 and v_k^2 are even and $\pi_d(v_{k+1}^2, v_k^2) \in \{d-1, d\}$. We also obtain from Lemma 9 that there is a neighbour u of v_k^1 for which $\pi_d(u, v_k^1) \in \{d-1, d\} \setminus \{\pi_d(v_{k+1}^2, v_k^2)\}$. Now u is a neighbour of v_k^1 such that there is no neighbour w of v_k^2 for which $\pi_d(u, v_k^1) = \pi_d(w, v_k^2)$, a contradiction. □

Now we are ready to prove the following lemma, which is the main ingredient of the proof of Theorem 1. The underlying idea is that the generalised port numbers along the walks of a PSW can only grow slowly—roughly by one every two steps. Put otherwise, each prefix of a critical PSW must be contained in a subgraph G_d for a sufficiently small value of d.

Lemma 15. *Let $(\overline{v}_1, \overline{v}_2)$, where $\overline{v}_i = (v_0^i, v_1^i, \ldots, v_k^i)$ for all $i = 1, 2$, be a critical PSW in G_d. Then $(\overline{v}'_1, \overline{v}'_2)$, where $\overline{v}'_i = (v_0^i, v_1^i, \ldots, v_{k-2}^i)$ for all $i = 1, 2$, is a PSW in G_{d-1}.*

Proof. First, suppose that $\{v_\ell^i, v_{\ell+1}^i\} \in E_{d-1}$ for all $i = 1, 2$ and $\ell = 0, 1, \ldots, k-3$ but that $(\overline{v}'_1, \overline{v}'_2)$ is not a PSW in G_{d-1}. Assume that $\{v_{k-2}^i, v_{k-1}^i\} \in E_{d-1}$ for some $i \in \{1, 2\}$ and let $j \in \{1, 2\} \setminus \{i\}$. It follows from Lemma 14 that there is a neighbour $u \in V_{d-1}$ of v_{k-2}^j such that $\pi_{d-1}(u, v_{k-2}^j) = \pi_{d-1}(v_{k-1}^i, v_{k-2}^i)$. Now Lemma 8 implies that $u = v_{k-1}^j$ and hence we have $v_{k-1}^i, v_{k-1}^j \in V_{d-1}$. Then we can use Lemma 13 to obtain that $(\overline{v}_1, \overline{v}_2)$ is not a critical PSW in G_d, a contradiction.

Let us then assume that $\{v_{k-2}^i, v_{k-1}^i\} \in E_d \setminus E_{d-1}$ for all $i = 1, 2$. As $v_{k-2}^i \in V_{d-1}$ for all $i = 1, 2$, Lemma 10 implies that v_{k-1}^i is a child of v_{k-2}^i and $\pi_d(v_{k-2}^1, v_{k-1}^1) = d = \pi_d(v_{k-2}^2, v_{k-1}^2)$ for all $i = 1, 2$. But now we can apply Lemma 11 to see that $(\overline{v}_1, \overline{v}_2)$ is not a critical PSW in G_d, a contradiction. We have now shown that if $\{v_\ell^i, v_{\ell+1}^i\} \in E_{d-1}$ for all $i = 1, 2$ and $\ell = 0, 1, \ldots, k-3$, then $(\overline{v}'_1, \overline{v}'_2)$ is a PSW in G_{d-1}.

Then, suppose that $\{v_\ell^i, v_{\ell+1}^i\} \in E_d \setminus E_{d-1}$ for some $i \in \{1, 2\}$ and $\ell \in \{0, 1, \ldots, k-3\}$. Let m be the smallest value of ℓ for which this holds. Let $j \in \{1, 2\} \setminus \{i\}$. If m is even, then the node $v_m^i \in V_{d-1}$ is odd, and by Lemma 10 we have that $\pi_d(v_m^i, v_{m+1}^i) = \pi_d(v_{m+1}^i, v_m^i) = d$ and that v_{m+1}^i is a child of v_m^i. Since $\pi_d(v_{m+1}^j, v_m^j) = \pi_d(v_{m+1}^i, v_m^i) = d$, we obtain $\{v_m^j, v_{m+1}^j\} \in E_d \setminus E_{d-1}$. As $v_m^j \in V_{d-1}$ is odd, Lemma 10 yields that $\pi_d(v_m^j, v_{m+1}^j) = \pi_d(v_{m+1}^j, v_m^j) = d$ and that v_{m+1}^j is a child of v_m^j. Lemma 11 then implies that $(\overline{v}_1, \overline{v}_2)$ is not a critical PSW in G_d, a contradiction.

To complete the proof, assume that m is odd. Recall that $\{v_m^i, v_{m+1}^i\} \in E_d \setminus E_{d-1}$. If also $\{v_m^j, v_{m+1}^j\} \in E_d \setminus E_{d-1}$, we can again use Lemma 10 to get that v_{m+1}^i and v_{m+1}^j are children of v_m^i and v_m^j, respectively, and that $\pi_d(v_m^i, v_{m+1}^i) = d = \pi_d(v_m^j, v_{m+1}^j)$. Now Lemma 11 yields a contradiction. If $\{v_m^j, v_{m+1}^j\} \in E_{d-1}$, let $\overline{v}''_\ell = (v_0^\ell, v_1^\ell, \ldots, v_m^\ell)$ for all $\ell = 1, 2$. The pair $(\overline{v}''_1, \overline{v}''_2)$ is a PSW in G_{d-1}, because otherwise by using a similar argument as above we would obtain that $\{v_m^i, v_{m+1}^i\} \in E_{d-1}$, a contradiction. But now we can use Lemma 12 to get a PSW of length $m + 2 \le (k-3) + 2 = k - 1$ in G_d, which contradicts the criticality of $(\overline{v}_1, \overline{v}_2)$. □

Having proved Lemma 15, the following result now follows by induction; see the full version of this work [15] for a proof.

Lemma 16. *Let $(\overline{v}_1, \overline{v}_2)$ be a PSW of length k in G_d. Then $k \ge 2d - 3$.*

Now we just need to show that the lower bound for the length of PSWs implies bisimilarity up to the respective distance, and we are mostly done.

Lemma 17. *We have $((1,0)) \underline{\leftrightarrow}_{2d-3}^{\mathcal{SV}} ((2,1))$, that is, the nodes $((1,0))$ and $((2,1))$ of G_d are $(2d-3)$-$\mathcal{SV}$-bisimilar.*

Proof. If we have $((1,0)) \underline{\leftrightarrow}_k^{\mathcal{SV}} ((2,1))$ for arbitrarily large k, the claim is clearly true. Otherwise, let k be the largest integer for which we have $((1,0)) \underline{\leftrightarrow}_k^{\mathcal{SV}} ((2,1))$. We will show that $k \ge 2d - 3$.

Let $v_0^1 = ((1,0))$ and $v_0^2 = ((2,1))$. Suppose then that $\ell \in \{0, 1, \ldots, k-1\}$ and that v_ℓ^1 and v_ℓ^2 have been defined. Furthermore, suppose that $k - \ell$ is the largest integer m for which $v_\ell^1 \underline{\leftrightarrow}_m^{\mathcal{SV}} v_\ell^2$ holds. If for each neighbour u of v_ℓ^1 there was a neighbour w of v_ℓ^2, and vice versa, such that $u \underline{\leftrightarrow}_{k-\ell}^{\mathcal{SV}} w$ and $\pi_d(u, v_\ell^1) = \pi_d(w, v_\ell^2)$, then by Definition 3 we would have $v_\ell^1 \underline{\leftrightarrow}_{k-\ell+1}^{\mathcal{SV}} v_\ell^2$, a contradiction. Thus for some $i \in \{1, 2\}$ and $j \in \{1, 2\} \setminus \{i\}$ there is a neighbour u of v_ℓ^i such that there is no neighbour w of v_ℓ^j for which the given condition holds. However, since $v_\ell^i \underline{\leftrightarrow}_{k-\ell}^{\mathcal{SV}} v_\ell^j$, we can choose neighbour w so that $u \underline{\leftrightarrow}_{k-\ell-1}^{\mathcal{SV}} w$ and $\pi_d(u, v_\ell^i) = \pi_d(w, v_\ell^j)$. Now we can define $v_{\ell+1}^i = u$ and $v_{\ell+1}^j = w$. We have shown that $k - \ell - 1 = k - (\ell + 1)$ is the largest integer m for which $v_{\ell+1}^i \underline{\leftrightarrow}_m^{\mathcal{SV}} v_{\ell+1}^j$ holds.

The above recursive definition yields a pair $(\overline{v}_1, \overline{v}_2)$ of walks, where $\overline{v}_i = (v_0^i, v_1^i, \ldots, v_k^i)$ for all $i = 1, 2$. Clearly conditions (W1) and (W2) hold. Additionally, we know that $k - k = 0$ is the largest integer m for which we have $v_k^1 \underline{\leftrightarrow}_m^{\mathcal{SV}} v_k^2$. However, if $k \le 2d - 3$, then for each neighbour u of v_k^1 and w of v_k^2 we have $\deg(u) = \deg(w)$ and hence $u \underline{\leftrightarrow}_0^{\mathcal{SV}} w$. It follows that for some $i \in \{1, 2\}$ and $j \in \{1, 2\} \setminus \{i\}$ there is a neighbour u of v_k^i such that there is no neighbour w of v_k^j for which $\pi_d(u, v_k^i) = \pi_d(w, v_k^j)$. If $i = 1$ and $j = 2$, this is equivalent to condition (W3). Otherwise, we use Lemma 7 and Lemma 9 to swap the roles of i and j in a similar manner as in the proof of Lemma 14.

In conclusion, we have shown that if $k \le 2d - 3$, then $(\overline{v}_1, \overline{v}_2)$ is a PSW of length k in G_d. Now Lemma 16 implies that $k = 2d - 3$. If $k > 2d - 3$, the claim is trivially true. □

To prove Theorem 1, we want the root node $\emptyset$ to receive the same messages from its neighbours $((1,0))$ and $((2,1))$. Lemma 17 shows that they are $(2d-3)$-$\mathcal{SV}$-bisimilar, but this is not enough: they also need to have identical outgoing port numbers towards node $\emptyset$. We will now define a port numbering of G_d based on the generalised port numbering p_d. Let $f \colon \{0, 1, \ldots, d\} \to [d]$ be a function such that $f(0) = 1$ and $f(i) = i$ for $i \ge 1$. Assume that $p_d(v, i) = (u, j)$ for some nodes v, u and port numbers i, j. If neither v nor u is a leaf node, we define $p'_d(v, f(i)) = (u, f(j))$. If v is a leaf node, we define $p'_d(v, 1) = (u, f(j))$ and if u is a leaf node, we define $p'_d(v, f(i)) = (u, 1)$. Due to the fact that in rule (G3) of the definition of G_d we used b_2^+ instead of b_2, no node has both 0 and 1 as port numbers in p_d. It follows that p'_d is a bijection from the set of input ports to the set of output ports, and the set of outgoing as well as incoming port numbers for each node v is $\{1, 2, \ldots, \deg(v)\}$. Observe that $p'_d(((1,0)), 1) = (\emptyset, 1)$ and $p'_d(((2,1)), 1) = (\emptyset, 2)$. Now we can apply Lemma 6 to see that the $(2d-3)$-$\mathcal{SV}$-bisimilarity still holds, that is, we have $(G_d, ((1,0)), p'_d) \underline{\leftrightarrow}_{2d-3}^{\mathcal{SV}} (G_d, ((2,1)), p'_d)$. Note that since the distance from $((1,0))$ and $((2,1))$ to the leaf nodes is $2d - 1$, the fact that we have to use the port number 1 for the leaf nodes does not affect the $(2d-3)$-$\mathcal{SV}$-bisimilarity.

Let $\mathcal{A} \in \mathcal{SV}$ be an arbitrary algorithm and $\Delta \ge 2$. Let $G = G_\Delta$, $p = p'_\Delta$, $v = \emptyset$, $u = ((1,0))$ and $w = ((2,1))$. Consider the execution of $\mathcal{A}$ in (G, p). Lemma 4 implies that the state of $\mathcal{A}$ in the nodes u and w is identical in each round $r = 0, 1, \ldots, 2\Delta - 3$. Furthermore, we have $\pi(u, v) = 1 = \pi(w, v)$. It follows that u and w send the same message to node v in each round $r + 1 = 1, 2, \ldots, 2\Delta - 2$. This concludes the proof of Theorem 1.

4. Separation by a Graph Problem

Theorem 1 shows that the simulation algorithm is optimal in a certain sense. However, since we are interested in graph problems, we want to separate the classes $\mathcal{SV}$ and $\mathcal{MV}$ by one. The following theorem states that we can do this, and the lower bound is still linear in Δ.

Theorem 2. *There is a graph problem Π that can be solved in one round by an algorithm in $\mathbf{MV}$ but that requires at least time T, where $T(n, \Delta) \geq \Delta$ for all odd $\Delta \in \mathbb{N}_+$ and $T(n, \Delta) \geq \Delta - 1$ for all even $\Delta \in \mathbb{N}_+$, when solved by an algorithm in $\mathbf{SV}$.*

Let us first define formally the graph problem Π. We will be working with graphs where each node is given as a local input one of three colours: black (B), white (W) or grey (G). For each graph (G, f) with local input from the set $\{\mathsf{B}, \mathsf{W}, \mathsf{G}\}$, the set $\Pi(G, f)$ of solutions consists of mappings $S\colon V \to \{\mathsf{B}, \mathsf{W}, \mathsf{G}\}$ such that for each $v \in V$, $S(v)$ is one of the local inputs having the highest multiplicity among the neighbours of v. For example, if node v has four neighbours of colour B, four neighbours of colour W and two neighbours of colour G, then for each solution S we have $S(v) = \mathsf{B}$ or $S(v) = \mathsf{W}$. See Figure 4 below for an illustration of two small problem instances. Our proof is based on generalising the depicted construction to higher node degrees.

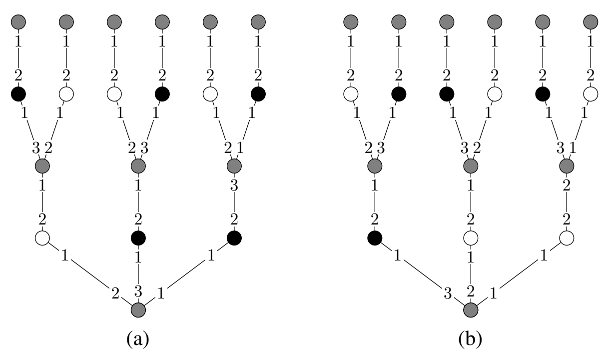

Figure 4: An $\mathcal{MV}$-algorithm can find out in one communication round whether it is run in the root node of graph (a) or graph (b), while any $\mathcal{SV}$-algorithm needs at least three communication rounds.

There is an algorithm in $\mathbf{MV}$—and, in fact, in $\mathbf{MB}$—that solves problem Π in only one communication round: Each node broadcasts its own colour to all its neighbours. Then, each node counts the multiplicity of each message it received and outputs the one with the highest multiplicity. Showing that this cannot be solved by any algorithm in $\mathbf{SV}$ in less than Δ communication rounds will require somewhat more work. Luckily, we can handle the most tricky part of the proof by making use of the proof of Theorem 1 in a black-box manner.

We start by defining for each $d = 2, 3, \ldots$ two graphs, $H_{\mathsf{B},d} = (V_{\mathsf{B},d}, E_{\mathsf{B},d})$ and $H_{\mathsf{W},d} = (V_{\mathsf{W},d}, E_{\mathsf{W},d})$. The constructions can be seen as extensions of the graph G_d defined earlier, but now each node is coloured with one of the three colours: black (B), white (W) or grey (G). Colours B and W can be thought of as complements of each other; we write $\overline{\mathsf{B}} = \mathsf{W}$ and $\overline{\mathsf{W}} = \mathsf{B}$. Again, we define $V_{\mathsf{B},d}$ recursively:

(H1) $\emptyset \in V_{\mathsf{B},d}$.

(H2) $((1, 0, \mathsf{B})), ((2, 1, \mathsf{B})), \ldots, ((d, d-1, \mathsf{B})) \in V_{\mathsf{B},d}$.

(H3) $((2, 1, \mathsf{W})), ((3, 2, \mathsf{W})), \ldots, ((d, d-1, \mathsf{W})) \in V_{\mathsf{B},d}$.

(H4) If $(a_1, a_2, \ldots, a_i) \in V_{\mathsf{B},d}$, where i is odd and $i \leq 2d-1$, then $(a_1, a_2, \ldots, a_i, a_{i+1}^j) \in V_{\mathsf{B},d}$ for all $j = 1, 2, \ldots, d-1$, where $a_{i+1}^j = (c_1^j, c_2^j, \mathsf{G})$ is defined as follows. Let $(b_1, b_2, C) = a_i$, where $C \in \{\mathsf{B}, \mathsf{W}\}$, and $b_2^+ = 1$ if $b_2 = 0$, $b_2^+ = b_2$ otherwise. Define
$$c_1^j = \min(\{1, 2, \ldots, d\} \setminus \{b_2^+, c_1^1, c_1^2, \ldots, c_1^{j-1}\}),$$
$$c_2^j = \min(\{1, 2, \ldots, d\} \setminus \{b_1, c_2^1, c_2^2, \ldots, c_2^{j-1}\}).$$

(H5) If $(a_1, a_2, \ldots, a_i) \in V_{\mathsf{B},d}$, where i is even and $2 \leq i \leq 2d-2$, then $(a_1, a_2, \ldots, a_i, a_{i+1}^j) \in V_{\mathsf{B},d}$ for all $j = 1, 2, \ldots, d-1$, where $a_{i+1}^j = (c_1^j, c_2^j, C)$ is defined as follows. Let $(d_1, d_2, C) = a_{i-1}$, where $C \in \{\mathsf{B}, \mathsf{W}\}$, and $(b_1, b_2, \mathsf{G}) = a_i$. Define
$$c_1^j = \min(\{1, 2, \ldots, d\} \setminus \{b_2, c_1^1, c_1^2, \ldots, c_1^{j-1}\}),$$
$$c_2^j = \min(\{0, 1, \ldots, d-1\} \setminus \{b_1, c_2^1, c_2^2, \ldots, c_2^{j-1}\}).$$

(H6) If $(a_1, a_2, \ldots, a_i) \in V_{\mathsf{B},d}$, where i is even and $2 \leq i \leq 2d-2$, then $(a_1, a_2, \ldots, a_i, a_{i+1}^j) \in V_{\mathsf{B},d}$ for all $j = 1, 2, \ldots, d-1$, where $a_{i+1}^j = (c_1^j, c_2^j, \overline{C})$ is defined as follows. Let $(d_1, d_2, C) = a_{i-1}$, where $C \in \{\mathsf{B}, \mathsf{W}\}$. Define
$$c_1^j = \min(\{2, 3, \ldots, d\} \setminus \{c_1^1, c_1^2, \ldots, c_1^{j-1}\}),$$
$$c_2^j = \min(\{1, 2, \ldots, d-1\} \setminus \{c_2^1, c_2^2, \ldots, c_2^{j-1}\}).$$

The set $E_{\mathsf{B},d}$ of edges consists of all pairs $\{v, u\}$, where $v = (a_1, a_2, \ldots, a_i) \in V_{\mathsf{B},d}$ and $u = (a_1, a_2, \ldots, a_i, a_{i+1}) \in V_{\mathsf{B},d}$ for some $i \in \{0, 1, \ldots\}$. The sets $V_{\mathsf{W},d}$ and $E_{\mathsf{W},d}$ are given by the same definition by replacing every occurrence of B with W and vice versa. By rearranging the branches of the trees, we observe that actually the only difference between $H_{\mathsf{B},d}$ and $H_{\mathsf{W},d}$ is the colours in the branch that starts with the node $((1, 0, C))$.

In this proof we work with the graphs $H_{\mathsf{B},d}$ and $H_{\mathsf{W},d}$ for a fixed value of d. Hence, to simplify notation, we will write H_{B} and H_{W} from now on.

We define colourings $f_{\mathsf{B}}\colon V_{\mathsf{B}} \to \{\mathsf{B}, \mathsf{W}, \mathsf{G}\}$ and $f_{\mathsf{W}}\colon V_{\mathsf{W}} \to \{\mathsf{B}, \mathsf{W}, \mathsf{G}\}$ as follows. If $v = (a_1, a_2, \ldots, a_i) \in V_C$ for some $C \in \{\mathsf{B}, \mathsf{W}\}$ and $i \geq 1$, and we have $a_i = (b_1, b_2, C')$, set $f_C(v) = C'$. If $v = \emptyset \in V_C$, set $f_C(v) = \mathsf{G}$. Notice that for each solution $S \in \Pi(H_{\mathsf{B}}, f_{\mathsf{B}})$ we have $S(\emptyset) = \mathsf{B}$ and for each solution $S \in \Pi(H_{\mathsf{W}}, f_{\mathsf{W}})$ we have $S(\emptyset) = \mathsf{W}$.

Our port numbers are pairs (a, C), where $a \in \{0, 1, \ldots, d\}$ and $C \in \{\mathsf{B}, \mathsf{W}, \mathsf{G}\}$. Generalised port numberings p_{B} and p_{W} for H_{B} and H_{W}, respectively, are defined as follows. Let $v = (a_1, a_2, \ldots, a_i)$ and $u = (a_1, a_2, \ldots, a_{i+1})$, where $a_{i+1} = (b_1, b_2, C)$, be nodes. Note that $f_{\mathsf{B}}(u) = f_{\mathsf{W}}(u) = C$. If $C \in \{\mathsf{B}, \mathsf{W}\}$, define
$$p_{\mathsf{B}}(v, (b_1, C)) = p_{\mathsf{W}}(v, (b_1, C)) = (u, (b_2, \mathsf{G})),$$
$$p_{\mathsf{B}}(u, (b_2, \mathsf{G})) = p_{\mathsf{W}}(u, (b_2, \mathsf{G})) = (v, (b_1, C)).$$

If $C = \mathsf{G}$, let $C' = f_{\mathsf{B}}(v) = f_{\mathsf{W}}(v)$ and define
$$p_{\mathsf{B}}(v, (b_1, \mathsf{G})) = p_{\mathsf{W}}(v, (b_1, \mathsf{G})) = (u, (b_2, C')),$$
$$p_{\mathsf{B}}(u, (b_2, C')) = p_{\mathsf{W}}(u, (b_2, C')) = (v, (b_1, \mathsf{G})).$$

Next we will define induced subgraphs $\hat{H}_{\mathsf{B}}$ and $\hat{H}_{\mathsf{W}}$ of H_{B} and H_{W}, respectively. For $C \in \{\mathsf{B}, \mathsf{W}\}$, the vertex set $\hat{V}_C$ of $\hat{H}_C$ consists of all vertices $(a_1, a_2, \ldots, a_i) \in V_C$ such that $f_C((a_1, a_2, \ldots, a_j)) \in \{C, \mathsf{G}\}$ for all $j \in \{0, 1, \ldots, i\}$. That is, a node v of H_C is in the subgraph $\hat{H}_C$ if and only if each node in the unique path from the root node $\emptyset$ to node v is either grey or of colour C. For each $v = (a_1, a_2, \ldots, a_i) \in V_C$ we denote the corresponding node of $\hat{H}_C$ by $\hat{v} = (a_1, a_2, \ldots, a_i) \in \hat{V}_C$.

For each $C \in \{\mathsf{B}, \mathsf{W}\}$, define a mapping $g_C\colon \hat{V}_C \to V_d$ as follows. Assume $\hat{v} = (a_1, a_2, \ldots, a_i) \in \hat{V}_C$, where $a_j =$

(b_1^j, b_2^j, C_j) for each j. Now set $g_C(\hat{v}) = (a'_1, a'_2, \dots, a'_i)$, where $a'_j = (b_1^j, b_2^j)$ for each j. By observing that the subgraph $\hat{H}_C$ is given by the rules (H1), (H2), (H4) and (H5) in the definition of H_C, and how they correspond to the rules (G1)–(G4) in the definition of G_d, one can see that g_C is a bijection, and in fact an isomorphism, between $\hat{H}_C$ and G_d. We can use g_C to move bisimilarity results from G_d to $\hat{H}_C$, as the following lemma shows.

Lemma 18. *Let $C \in \{\mathsf{B}, \mathsf{W}\}$, $r \in \mathbb{N}$ and $\hat{v}, \hat{u} \in \hat{V}_C$. If $g_C(\hat{v}) \leftrightarrow_r^{\mathcal{SV}} g_C(\hat{u})$ and $f_C(\hat{v}) = f_C(\hat{u})$, then $\hat{v} \leftrightarrow_r^{\mathcal{SV}} \hat{u}$.*

Proof. The proof is by induction on r. Given the inductive hypothesis and conditions (B1)–(B3) of Definition 3 for $g_C(\hat{v})$ and $g_C(\hat{u})$, it is quite straightforward to check that the conditions also hold for $\hat{v}$ and $\hat{u}$. □

Next, we will define a partial mapping $f_{v,u}\colon V_C \to V_C$ for each pair of grey nodes $\hat{v}$ and $\hat{u}$ in $\hat{H}_C$. Assume that $v = (a_1, a_2, \dots, a_i)$ and $u = (b_1, b_2, \dots, b_j)$. If $v' = (a_1, a_2, \dots, a_i, c_1, c_2, \dots, c_{i'}) \in V_C$ for some $c_1, c_2, \dots, c_{i'}$, and we have

$$f_C((a_1, a_2, \dots, a_i, c_1)) = \overline{C}$$

as well as

$$u' = (b_1, b_2, \dots, b_j, c_1, c_2, \dots, c_{i'}) \in V_C,$$

then we define $f_{v,u}(v') = u'$. The idea here is that the subtrees of H_C that have the nodes v and u as their roots and that are not contained in the subgraph $\hat{H}_C$ (except for the root nodes) are isomorphic (up to a certain distance). The mapping $f_{v,u}$ is a partial isomorphism between such subtrees, as one can quite easily check. In what follows, we will use $f_{v,u}$ to show that the r-$\mathcal{SV}$-bisimilarity of the nodes $((1, 0, C))$ and $((2, 1, C))$ in $\hat{H}_C$ can be extended to the supergraph H_C.

For each $C \in \{\mathsf{B}, \mathsf{W}\}$, denote the nodes $\emptyset$, $((1, 0, C))$ and $((2, 1, C))$ of H_C by v_C, u_C and w_C, respectively. In accordance with our previously introduced notation, denote the corresponding nodes of the subgraph $\hat{H}_C$ by $\hat{v}_C$, $\hat{u}_C$ and $\hat{w}_C$.

Lemma 19. *Let $\hat{v}, \hat{u} \in \hat{V}_C$ be grey nodes and let $t \in \mathbb{N}$ be such that $v \leftrightarrow_t^{\mathcal{SV}} u$. If $w \in \operatorname{dom}(f_{v,u})$, $\operatorname{dist}(w, v_C) < 2d - t$ and $\operatorname{dist}(f_{v,u}(w), v_C) < 2d - t$, then $w \leftrightarrow_t^{\mathcal{SV}} f_{v,u}(w)$.*

Proof. We proceed by induction on t. The base case $t = 0$ is straightforward: Since $\operatorname{dist}(w, v^C) < 2d$ and $\operatorname{dist}(f_{v,u}(w), v^C) < 2d$, we have $\deg(w) = \deg(f_{v,u}(w))$. Additionally, observe that we have $f_C(w) = f_C(f_{v,u}(w))$. It follows that we have $w \leftrightarrow_0^{\mathcal{SV}} f_{v,u}(w)$.

For the inductive case, assume that the claim holds for $t = s$ and that $v \leftrightarrow_{s+1}^{\mathcal{SV}} u$. If $w = v$, then $f_{v,u}(w) = u$ and we have nothing to prove. Hence, assume $w \neq v$. Denote the neighbours of w by $w_1, w_2, \dots, w_k$. Then the neighbours of $f_{v,u}(w)$ are $f_{v,u}(w_i)$, $i = 1, 2, \dots, k$. We have $w_i \in \operatorname{dom}(f_{v,u})$ for all i. Additionally, since $\operatorname{dist}(w, v_C) < 2d-(s+1)$ and $\operatorname{dist}(f_{v,u}(w), v_C) < 2d-(s+1)$, we have $\operatorname{dist}(w_i, v_C) < 2d - s$ and $\operatorname{dist}(f_{v,u}(w_i), v_C) < 2d - s$ for all i. Now the inductive hypothesis implies that $w \leftrightarrow_s^{\mathcal{SV}} f_{v,u}(w)$ and $w_i \leftrightarrow_s^{\mathcal{SV}} f_{v,u}(w_i)$ for all i. Additionally, it follows immediately from the definition of $f_{v,u}$ that we have $\pi_C(w_i, w) = \pi_C(f_{v,u}(w_i), f_{v,u}(w))$ for all i. Now by Definition 3 we have $w \leftrightarrow_{s+1}^{\mathcal{SV}} f_{v,u}(w)$. Hence the claim holds for $t = s + 1$. □

Lemma 20. *Let $t \in \mathbb{N}$ and let $\hat{v}, \hat{u} \in \hat{V}_C$ be such that $\operatorname{dist}(\hat{v}, \hat{v}_C) < 2d - t$ and $\operatorname{dist}(\hat{u}, \hat{v}_C) < 2d - t$. If $\hat{v} \leftrightarrow_t^{\mathcal{SV}} \hat{u}$, then $v \leftrightarrow_t^{\mathcal{SV}} u$.*

Proof. We prove the claim by induction on t. The base case $t = 0$ is easy: If $\hat{v} \leftrightarrow_0^{\mathcal{SV}} \hat{u}$, then $f_C(\hat{v}) = f_C(\hat{u})$, and thus $f_C(v) = f_C(u)$. As v and u are of the same colour and neither of them is a leaf node, $\deg(v) = \deg(u)$. Hence $v \leftrightarrow_0^{\mathcal{SV}} u$.

For the inductive step, assume that the claim holds for $t = s$ and that $\hat{v} \leftrightarrow_{s+1}^{\mathcal{SV}} \hat{u}$, where $\operatorname{dist}(\hat{v}, \hat{v}_C) < 2d - (s + 1)$ and $\operatorname{dist}(\hat{u}, \hat{v}_C) < 2d - (s + 1)$. Denote the neighbours of $\hat{v}$ and $\hat{u}$ by $\hat{v}_1, \hat{v}_2, \dots, \hat{v}_d$ and $\hat{u}_1, \hat{u}_2, \dots, \hat{u}_d$, respectively. We have $\hat{v} \leftrightarrow_s^{\mathcal{SV}} \hat{u}$, and by definition, for each $\hat{v}_i$ there is $\hat{u}_{j_i}$ such that $\hat{v}_i \leftrightarrow_s^{\mathcal{SV}} \hat{u}_{j_i}$ and $\pi_C(\hat{v}_i, \hat{v}) = \pi_C(\hat{u}_{j_i}, \hat{u})$, and vice versa. We have $\operatorname{dist}(\hat{v}_i, \hat{v}_C) < 2d - s$ and $\operatorname{dist}(\hat{u}_i, \hat{v}_C) < 2d - s$ for all i. Now the inductive hypothesis implies that $v \leftrightarrow_s^{\mathcal{SV}} u$, $v_i \leftrightarrow_s^{\mathcal{SV}} u_{j_i}$ for all i and $v_{i_j} \leftrightarrow_s^{\mathcal{SV}} u_j$ for all j.

Since $v \leftrightarrow_s^{\mathcal{SV}} u$, nodes v and u are of the same colour. If they are of colour C, they do not have neighbours other than $v_1, v_2, \dots, v_d$ and $u_1, u_2, \dots, u_d$, respectively. Then it follows from the definition that $v \leftrightarrow_{s+1}^{\mathcal{SV}} u$. Otherwise, v and u are grey, and in addition to v_i and u_i, $i = 1, 2, \dots, d$, they have neighbours generated by rule (H3) or rule (H6). Denote those neighbours by $v'_1, v'_2, \dots, v'_{d-1}$ and $u'_1, u'_2, \dots, u'_{d-1}$, respectively, such that we have $f_{v,u}(v'_i) = u'_i$ for all i. Observe that $\operatorname{dist}(v'_i, v_C) < 2d-s$ and $\operatorname{dist}(u'_i, v_C) < 2d-s$ for all i. Now Lemma 19 shows that $v'_i \leftrightarrow_s^{\mathcal{SV}} u'_i$ for all i. In addition, the definition of $f_{v,u}$ implies that $\pi_C(v'_i, v) = \pi_C(u'_i, u)$ for all i. We have shown that conditions (B2) and (B3) hold also for the additional neighbours, and consequently $v \leftrightarrow_{s+1}^{\mathcal{SV}} u$. Hence the claim is true for $t = s + 1$. □

Now we can combine our previous results to obtain bisimilarity between certain nodes in the graph H_C for each $C \in \{\mathsf{B}, \mathsf{W}\}$. Lemma 17 shows that $((1, 0)) \leftrightarrow_{2d-3}^{\mathcal{SV}} ((2, 1))$, where $((1, 0))$ and $((2, 1))$ are nodes in the graph G_d. Observe that $g_C(\hat{u}_C) = ((1, 0))$ and $g_C(\hat{w}_C) = ((2, 1))$. Now Lemma 18 implies that $\hat{u}_C \leftrightarrow_{2d-3}^{\mathcal{SV}} \hat{w}_C$. We have $\operatorname{dist}(\hat{u}_C, \hat{v}_C) = 1 < 2d-(2d-3)$ and $\operatorname{dist}(\hat{w}_C, \hat{v}_C) = 1 < 2d - (2d - 3)$. Hence it follows from Lemma 20 that $u_C \leftrightarrow_{2d-3}^{\mathcal{SV}} w_C$, where u_C and w_C are neighbours of v_C in the graph H_C.

As in the proof of Theorem 1, we define a port numbering p'_C for each $C \in \{\mathsf{B}, \mathsf{W}\}$ based on the generalised port numbering p_C. Again, we need to preserve bisimilarity as well as have identical outgoing port numbers from nodes u_C and w_C towards node v_C. Define function f from the set of all generalised ports of H_C to $[2d - 1]$ as follows: $f(1, \mathsf{B}) = f(1, \mathsf{W}) = 1$, $f(i, \mathsf{B}) = 2i - 1$ and $f(i, \mathsf{W}) = 2i-2$ for all $i = 2, 3, \dots, d$, $f(0, \mathsf{G}) = 1$ and $f(i, \mathsf{G}) = i$ for all $i = 1, 2, \dots, d$. Then, if $p_C(v, a) = (u, b)$ for some non-leaf nodes v, u and port numbers a, b, set $p'_C(v, f(a)) = (u, f(b))$. Again, use the port number 1 for the leaf nodes. Without too much effort, one can check that p'_C is indeed a valid port numbering of H_C, and that we have $\pi'_C(u_C, v_C) = 1 = \pi'_C(w_C, v_C)$. Lemma 6 implies that $(H_C, f_C, u_C, p'_C) \leftrightarrow_{2d-3}^{\mathcal{SV}} (H_C, f_C, w_C, p'_C)$.

To reach our ultimate goal, we need to define one more mapping. Define $h\colon V_{\mathsf{B}} \to V_{\mathsf{W}}$ as follows: if $v = (a_1, a_2, \dots, a_i) \in V_{\mathsf{B}}$, where $i \geq 1$ and $a_1 = (b_1, b_2, C)$ for some $b_1 \geq 2$, set $h(v) = u$, where $u = (a_1, a_2, \dots, a_i) \in V_{\mathsf{W}}$. Additionally, set $h(v_{\mathsf{B}}) = v_{\mathsf{W}}$. Thus, there is one subtree starting from a child of v_{B}, the one having the node $u_{\mathsf{B}} = ((1, 0, \mathsf{B}))$ as its root, that is excluded from the domain of h. Similarly, the subtree having $u_{\mathsf{W}} = ((1, 0, \mathsf{W}))$ as its root is excluded from the range of h. See Figure 5 for an illustration of the situation.

Lemma 21. *Let $v \in V_{\mathsf{B}}$ and $u \in V_{\mathsf{W}}$ be nodes such that $h(v) = u$. Then for all $t = 0, 1, \dots, 2d - 2$ we have $(H_{\mathsf{B}}, f_{\mathsf{B}}, v, p'_{\mathsf{B}}) \leftrightarrow_t^{\mathcal{SV}} (H_{\mathsf{W}}, f_{\mathsf{W}}, u, p'_{\mathsf{W}})$.*

Proof. We prove the claim by induction on t. The base case $t = 0$ is trivial: if $h(v) = u$, then by definition of h we have $\deg(v) = \deg(u)$ and $f_{\mathsf{B}}(v) = f_{\mathsf{W}}(u)$ and therefore $v \leftrightarrow_0^{\mathcal{SV}} u$.

For the inductive step, suppose that the claim holds for $t = s < 2d - 2$. Consider two arbitrary nodes $v \in V_{\mathsf{B}}$ and $u \in V_{\mathsf{W}}$ such that $h(v) = u$. By the inductive hypothesis we have $v \leftrightarrow_s^{\mathcal{SV}} u$. If

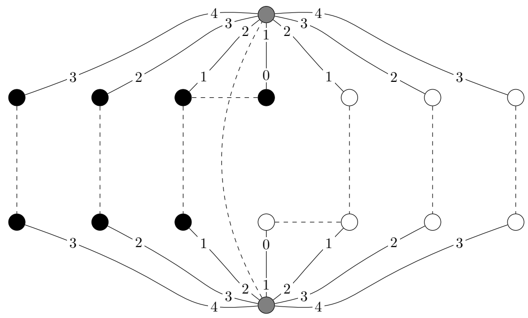

Figure 5: Graphs $H_{\mathsf{B},4}$ and $H_{\mathsf{W},4}$ up to distance one from the root nodes. The dashed lines represent r-$\mathcal{SV}$-bisimilarity between nodes.

$v \neq v_{\mathsf{B}}$, all the neighbours of v are in the domain of h and all the neighbours of u are in the range of h. Furthermore, if w is a neighbour of v, we have $\pi'_{\mathsf{B}}(w, v) = \pi'_{\mathsf{W}}(h(w), u)$, and by the inductive hypothesis, $w \leftrightarrows_s^{\mathcal{SV}} h(w)$. Now Definition 3 implies that $v \leftrightarrows_{s+1}^{\mathcal{SV}} u$.

If $v = v_{\mathsf{B}}$, v has one neighbour that is not in $\operatorname{dom}(h)$. That neighbour is $u_{\mathsf{B}} = ((1, 0, \mathsf{B}))$. Similarly, $h(v) = v_{\mathsf{W}}$ has one neighbour that is not in the range of h, namely $u_{\mathsf{W}} = ((1, 0, \mathsf{W}))$. However, as shown above, we have $u_{\mathsf{B}} \leftrightarrows_{2d-3}^{\mathcal{SV}} w_{\mathsf{B}}$, and thus $u_{\mathsf{B}} \leftrightarrows_s^{\mathcal{SV}} w_{\mathsf{B}}$. Since we have also $w_{\mathsf{B}} \leftrightarrows_s^{\mathcal{SV}} h(w_{\mathsf{B}})$, Lemma 5 implies that $u_{\mathsf{B}} \leftrightarrows_s^{\mathcal{SV}} h(w_{\mathsf{B}})$. Additionally, we have

$$\pi'_{\mathsf{B}}(u_{\mathsf{B}}, v) = \pi'_{\mathsf{B}}(w_{\mathsf{B}}, v) = \pi'_{\mathsf{B}}(h(w_{\mathsf{B}}), u).$$

Similarly, we have $u_{\mathsf{W}} \leftrightarrows_s^{\mathcal{SV}} w_{\mathsf{W}}$ and $w_{\mathsf{W}} \leftrightarrows_s^{\mathcal{SV}} h^{-1}(w_{\mathsf{W}})$, from which we get $u_{\mathsf{W}} \leftrightarrows_s^{\mathcal{SV}} h^{-1}(w_{\mathsf{W}})$. Additionally,

$$\pi'_{\mathsf{W}}(u_{\mathsf{W}}, u) = \pi'_{\mathsf{W}}(w_{\mathsf{W}}, u) = \pi'_{\mathsf{W}}(h^{-1}(w_{\mathsf{W}}), v).$$

We have shown that conditions (B1)–(B3) hold even if considering also neighbours not handled by the mapping h, and consequently we have $v \leftrightarrows_{s+1}^{\mathcal{SV}} u$. Thus the claim holds for $t = s + 1$. □

Let $d \geq 2$ and $\Delta = 2d - 1$. Then $H_{\mathsf{B},d}, H_{\mathsf{W},d} \in \mathcal{F}(\Delta)$. Let $\mathcal{A} \in \mathcal{SV}$ be any algorithm with a running time at most $\Delta - 1 = 2d - 2$. Consider the execution of $\mathcal{A}$ in the nodes $v_{\mathsf{B}} \in V_{\mathsf{B},d}$ and $v_{\mathsf{W}} \in V_{\mathsf{W},d}$. Now Lemma 21 together with Lemma 4 implies that $\mathcal{A}$ produces the same output in v_{B} and v_{W}. Recall that for any valid solutions $S \in \Pi(H_{\mathsf{B},d}, f_{\mathsf{B}})$ and $S' \in \Pi(H_{\mathsf{W},d}, f_{\mathsf{W}})$ we have $S(v_{\mathsf{B}}) \neq S'(v_{\mathsf{W}})$. Hence $\mathcal{A}$ does not solve the problem Π. This concludes the proof of Theorem 2.

Remark 22. Note that we could define a similar problem without local inputs, by encoding the colours in the structure of the graph. One way to do this is to add one new neighbour to each black node and two new neighbours to each white node. If $d \geq 3$, this does not increase the maximum degree of the graph. Then we could define the set of solutions to consist of, for example, mappings S such that $S(v) = 1$ if node v has an odd number of neighbours of an odd degree and $S(v) = 0$ otherwise. However, for illustrative purposes, it was beneficial the use a colouring instead.

Acknowledgments

This paper is based on the author's master's thesis [14]. The author would like to thank Jukka Suomela for guidance and feedback, Lauri Hella and Juha Kontinen for examining the thesis, and finally, the anonymous reviewers for their helpful comments. A minor part of the research was conducted while the author was employed at the University of Helsinki. The full version of this work is available online [15].

References

[1] R. Aleliunas, R. M. Karp, R. J. Lipton, L. Lovász, and C. Rackoff. Random walks, universal traversal sequences, and the complexity of maze problems. In *Proc. 20th Annual Symposium on Foundations of Computer Science (FOCS 1979)*, pages 218–223. IEEE, 1979. doi:10.1109/SFCS.1979.34.

[2] D. Angluin. Local and global properties in networks of processors. In *Proc. 12th Annual ACM Symposium on Theory of Computing (STOC 1980)*, pages 82–93. ACM, 1980. doi:10.1145/800141.804655.

[3] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*, volume 53 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, UK, 2001.

[4] P. Blackburn, J. van Benthem, and F. Wolter, editors. *Handbook of Modal Logic*, volume 3 of *Studies in Logic and Practical Reasoning*. Elsevier, Amsterdam, NL, 2007.

[5] P. Boldi and S. Vigna. Computing vector functions on anonymous networks. In *Proc. 4th Colloquium on Structural Information and Communication Complexity (SIROCCO 1997)*, pages 201–214. Carleton Scientific, 1997.

[6] P. Boldi, S. Shammah, S. Vigna, B. Codenotti, P. Gemmell, and J. Simon. Symmetry breaking in anonymous networks: characterizations. In *Proc. 4th Israel Symposium on the Theory of Computing and Systems (ISTCS 1996)*, pages 16–26. IEEE, 1996.

[7] J. Chalopin. *Algorithmique Distribuée, Calculs Locaux et Homomorphismes de Graphes*. PhD thesis, LaBRI, Université Bordeaux 1, 2006.

[8] Y. Emek and R. Wattenhofer. Stone age distributed computing. In *Proc. 32nd Annual ACM Symposium on Principles of Distributed Computing (PODC 2013)*, pages 137–146. ACM, 2013. doi:10.1145/2484239.2484244. arXiv:1202.1186.

[9] L. Hella, M. Järvisalo, A. Kuusisto, J. Laurinharju, T. Lempiäinen, K. Luosto, J. Suomela, and J. Virtema. Weak models of distributed computing, with connections to modal logic. *Distributed Computing*, 28(1):31–53, 2015. doi:10.1007/s00446-013-0202-3. arXiv:1205.2051.

[10] M. Koucký. Universal traversal sequences with backtracking. *Journal of Computer and System Sciences*, 65(4):717–726, 2002. doi:10.1016/S0022-0000(02)00023-5.

[11] A. Krebs and O. Verbitsky. Universal covers, color refinement, and two-variable counting logic: lower bounds for the depth. In *Proc. 30th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2015)*, pages 689–700. IEEE, 2015. doi:10.1109/LICS.2015.69. arXiv:1407.3175.

[12] A. Kuusisto. Modal logic and distributed message passing automata. In *Proc. 22nd EACSL Annual Conference on Computer Science Logic (CSL 2013)*, pages 452–468. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2013. doi:10.4230/LIPIcs.CSL.2013.452.

[13] A. Kuusisto. Infinite networks, halting and local algorithms. In *Proc. 5th International Symposium on Games, Automata, Logics and Formal Verification (GandALF 2014)*, pages 147–160, 2014. doi:10.4204/EPTCS.161.14.

[14] T. Lempiäinen. A classification of weak models of distributed computing. Master's thesis, Department of Mathematics and Statistics, University of Helsinki, 2014. http://hdl.handle.net/10138/144214.

[15] T. Lempiäinen. Ability to count messages is worth $\Theta(\Delta)$ rounds in distributed computing, 2015. Manuscript. arXiv:1505.02322.

[16] F. Reiter. Distributed graph automata. In *Proc. 30th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2015)*, pages 192–201. IEEE, 2015. doi:10.1109/LICS.2015.27. arXiv:1404.6503.

[17] M. Yamashita and T. 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.

[18] M. Yamashita and T. Kameda. Leader election problem on networks in which processor identity numbers are not distinct. *IEEE Transactions on Parallel and Distributed Systems*, 10(9):878–887, 1999. doi:10.1109/71.798313.