

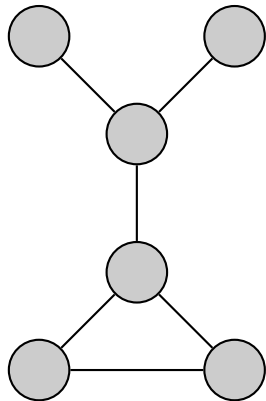
# The value of incoming message multiplicities in distributed computing

Tuomo Lempäinen

Aalto University

Logic seminar, 10th December 2014

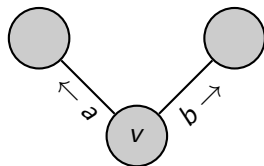
# Distributed system



A graph, whose each node

- runs the same algorithm,
- can be given a local input,
- can communicate with its neighbours,
- produces a local output.

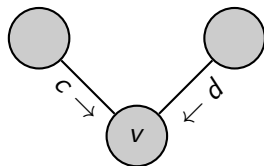
# Communications happens in synchronous rounds



In every round, each node  $v$

- 1 sends messages to its neighbours,
- 2 receives messages from its neighbours,
- 3 updates its state.

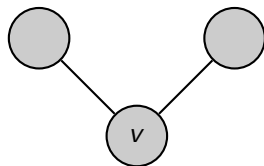
# Communications happens in synchronous rounds



In every round, each node  $v$

- 1 sends messages to its neighbours,
- 2 receives messages from its neighbours,
- 3 updates its state.

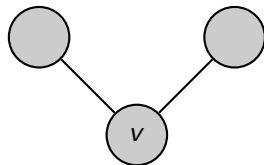
# Communications happens in synchronous rounds



In every round, each node  $v$

- 1 sends messages to its neighbours,
- 2 receives messages from its neighbours,
- 3 updates its state.

## Communications happens in synchronous rounds

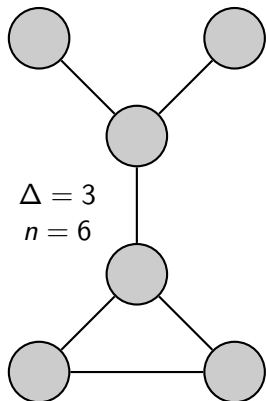


In every round, each node  $v$

- 1 sends messages to its neighbours,
- 2 receives messages from its neighbours,
- 3 updates its state.

After the final round, each node announces its own output.

## Focus on communication, not computation



The running time of an algorithm is the *number of communications rounds*.

The running time may depend on

- the maximum degree of the graph,  $\Delta$ ,
- the number of nodes,  $n$ .

## Port numbering

A *port* of a graph  $G = (V, E)$  is a pair  $(v, i)$ , where  $v \in V$  and  $i \in \{1, 2, \dots, \deg(v)\}$ . Let  $P(G)$  be the set of all ports of  $G$ . A *port numbering* of  $G$  is a bijection  $p: P(G) \rightarrow P(G)$  such that

$$p(v, i) = (u, j) \text{ for some } i \text{ and } j \text{ if and only if } \{v, u\} \in E.$$

Intuitively, if  $p(v, i) = (u, j)$ , then  $(v, i)$  is an output port of node  $v$  that is connected to an input port  $(u, j)$  of node  $u$ .



## Port numbering

A *port* of a graph  $G = (V, E)$  is a pair  $(v, i)$ , where  $v \in V$  and  $i \in \{1, 2, \dots, \deg(v)\}$ . Let  $P(G)$  be the set of all ports of  $G$ . A *port numbering* of  $G$  is a bijection  $p: P(G) \rightarrow P(G)$  such that

$$p(v, i) = (u, j) \text{ for some } i \text{ and } j \text{ if and only if } \{v, u\} \in E.$$

Intuitively, if  $p(v, i) = (u, j)$ , then  $(v, i)$  is an output port of node  $v$  that is connected to an input port  $(u, j)$  of node  $u$ .

We say that a port numbering  $p$  is *consistent* if we have

$$p(p(v, i)) = (v, i) \text{ for all } (v, i) \in P(G),$$

or, in other words, if the input port and the output port connected to the same neighbour always have the same number.

## Graph classes and local inputs

For each positive integer  $\Delta$ , denote by  $\mathcal{F}(\Delta)$  the class of all simple undirected graphs of maximum degree at most  $\Delta$ .

## Graph classes and local inputs

For each positive integer  $\Delta$ , denote by  $\mathcal{F}(\Delta)$  the class of all simple undirected graphs of maximum degree at most  $\Delta$ .

An *input* for a graph  $G = (V, E)$  is a function  $f: V \rightarrow X$ , where  $X \ni \emptyset$  is a finite set. For each  $v \in V$ , the value  $f(v)$  is called the *local input* of  $v$ .

The symbol  $\emptyset \in X$  is used to indicate “no input”.

# Algorithms as state machines

Let  $\Delta \in \mathbb{N}_+$  and let  $X$  be a 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,
- $\sigma_0: \{0, 1, \dots, \Delta\} \times X \rightarrow Y$  is a function that defines the initial state,
- $M$  is a set of messages such that  $\epsilon \in M$ ,
- $\mu: Y \times [\Delta] \rightarrow M$  is a function that constructs the outgoing messages, such that  $\mu(z, i) = \epsilon$  for all  $z \in Z$  and  $i \in [\Delta]$ ,
- $\sigma: Y \times M^\Delta \rightarrow Y$  is a function that defines the state transitions, such that  $\sigma(z, \bar{m}) = z$  for all  $z \in Z$  and  $\bar{m} \in M^\Delta$ .

The special symbol  $\epsilon \in M$  indicates “no message”.

## Execution

Let  $G = (V, E) \in \mathcal{F}(\Delta)$ , let  $p$  be a port numbering of  $G$ , let  $f: V \rightarrow X$ , and let  $\mathcal{A}$  be a distributed state machine for  $(\mathcal{F}(\Delta), X)$ .

The state of the system in round  $r \in \mathbb{N}$  is a function  $x_r: V \rightarrow Y$ , where  $x_r(v)$  is the *state* of node  $v$  in round  $r$ . To initialise the nodes, set

$$x_0(v) = \sigma_0(\deg(v), f(v)) \quad \text{for each } v \in V.$$

# Execution

Let  $G = (V, E) \in \mathcal{F}(\Delta)$ , let  $p$  be a port numbering of  $G$ , let  $f: V \rightarrow X$ , and let  $\mathcal{A}$  be a distributed state machine for  $(\mathcal{F}(\Delta), X)$ .

The state of the system in round  $r \in \mathbb{N}$  is a function  $x_r: V \rightarrow Y$ , where  $x_r(v)$  is the *state* of node  $v$  in round  $r$ . To initialise the nodes, set

$$x_0(v) = \sigma_0(\deg(v), f(v)) \quad \text{for each } v \in V.$$

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

$$\bar{a}_{r+1}(v) = (a_{r+1}(v, 1), a_{r+1}(v, 2), \dots, a_{r+1}(v, \deg(v)), \epsilon, \epsilon, \dots, \epsilon) \in M^\Delta.$$

Now we can define the new state of each node  $v \in V$  as follows:

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

## Running time

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  $x_t: V \rightarrow Y$ .

For each  $v \in V$ , the *local output* of  $v$  is  $x_t(v)$ .

# Graph problems

We study *graph problems* where

- problem instance is the communication graph (and the possible local inputs),
- the local outputs together define a solution.



# Graph problems

We study *graph problems* where

- problem instance is the communication graph (and the possible local inputs),
- the local outputs together define a solution.

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  $f: V \rightarrow X$  to a set  $\Pi_{X,Y}(G, f)$  of solutions.

Each *solution*  $S \in \Pi_{X,Y}(G, f)$  is a function  $S: V \rightarrow Y$ .

## Solving a graph problem

Let  $\Pi_{X,Y}$  be a graph problem,  $T: \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$  and  $\mathbf{A} = (\mathcal{A}_1, \mathcal{A}_2, \dots)$  such that each  $\mathcal{A}_\Delta$  is a distributed state machine for  $(\mathcal{F}(\Delta), X)$ . Algorithm  $\mathbf{A}$  solves  $\Pi_{X,Y}$  in time  $T$  if the following holds for all  $\Delta \in \mathbb{N}$ , all finite graphs  $G = (V, E) \in \mathcal{F}(\Delta)$ , all inputs  $f: V \rightarrow X$  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  $\Pi_{X,Y}(G, f)$ .

## Solving a graph problem

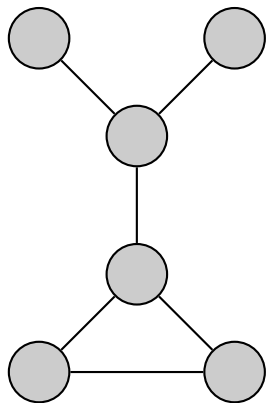
Let  $\Pi_{X,Y}$  be a graph problem,  $T: \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$  and  $\mathbf{A} = (\mathcal{A}_1, \mathcal{A}_2, \dots)$  such that each  $\mathcal{A}_\Delta$  is a distributed state machine for  $(\mathcal{F}(\Delta), X)$ . Algorithm  $\mathbf{A}$  solves  $\Pi_{X,Y}$  in time  $T$  if the following holds for all  $\Delta \in \mathbb{N}$ , all finite graphs  $G = (V, E) \in \mathcal{F}(\Delta)$ , all inputs  $f: V \rightarrow X$  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  $\Pi_{X,Y}(G, f)$ .

We say that  $\mathbf{A}$  solves  $\Pi_{X,Y}$  in time  $T$  assuming consistency if the above holds for all consistent port numberings  $p$  of  $G$ .

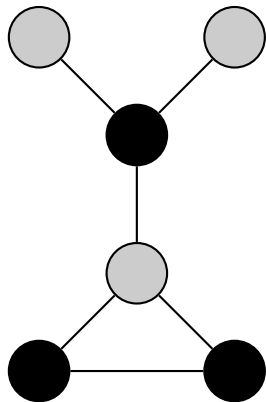
If  $T(\Delta, n)$  does not depend on  $n$ , we say that  $\mathbf{A}$  solves  $\Pi_{X,Y}$  in constant time or that  $\mathbf{A}$  is a local algorithm for  $\Pi_{X,Y}$ .

# Graph problems



Often the solution  $S: V \rightarrow Y$  is an encoding of a subset of vertices or edges of the graph.

# Graph problems

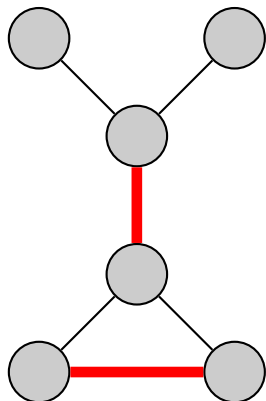


Often the solution  $S: V \rightarrow Y$  is an encoding of a subset of vertices or edges of the graph.

Example problems:

- minimum vertex cover,

# Graph problems



Often the solution  $S: V \rightarrow Y$  is an encoding of a subset of vertices or edges of the graph.

Example problems:

- minimum vertex cover,
- maximal matching.

## Variants of the model of computation

$\mathcal{VV}$  is the class of all distributed state machines (send a vector, receive a vector).

We can place different restrictions on the algorithms:

- $\mathcal{VB}$ : broadcast the same message to all neighbours:

$$\mu(y, i) = \mu(y, j) \text{ for all } i, j \in \{1, 2, \dots, \Delta\} \text{ and } y \in Y,$$

# Variants of the model of computation

$\mathcal{V}\mathcal{V}$  is the class of all distributed state machines (send a vector, receive a vector).

We can place different restrictions on the algorithms:

- $\mathcal{V}\mathcal{B}$ : broadcast the same message to all neighbours:

$$\mu(y, i) = \mu(y, j) \text{ for all } i, j \in \{1, 2, \dots, \Delta\} \text{ and } y \in Y,$$

- $\mathcal{M}\mathcal{V}$ : receive a multiset of messages:

$$\text{multiset}(\bar{a}) = \text{multiset}(\bar{b}) \Rightarrow \sigma(y, \bar{a}) = \sigma(y, \bar{b}) \text{ for all } y \in Y,$$

- $\mathcal{S}\mathcal{V}$ : receive a set of messages:

$$\text{set}(\bar{a}) = \text{set}(\bar{b}) \Rightarrow \sigma(y, \bar{a}) = \sigma(y, \bar{b}) \text{ for all } y \in Y,$$



## Variants of the model of computation

$\mathcal{VV}$  is the class of all distributed state machines (send a vector, receive a vector).

We can place different restrictions on the algorithms:

- $\mathcal{VB}$ : broadcast the same message to all neighbours:

$$\mu(y, i) = \mu(y, j) \text{ for all } i, j \in \{1, 2, \dots, \Delta\} \text{ and } y \in Y,$$

- $\mathcal{MV}$ : receive a multiset of messages:

$$\text{multiset}(\bar{a}) = \text{multiset}(\bar{b}) \Rightarrow \sigma(y, \bar{a}) = \sigma(y, \bar{b}) \text{ for all } y \in Y,$$

- $\mathcal{SV}$ : receive a set of messages:

$$\text{set}(\bar{a}) = \text{set}(\bar{b}) \Rightarrow \sigma(y, \bar{a}) = \sigma(y, \bar{b}) \text{ for all } y \in Y,$$

- $\mathcal{MB} = \mathcal{MV} \cap \mathcal{VB}$  and  $\mathcal{SB} = \mathcal{SV} \cap \mathcal{VB}$ .

## Variants of the model of computation

$$\mathbf{VV} = \{(\mathcal{A}_1, \mathcal{A}_2, \dots) : \mathcal{A}_\Delta \in \mathcal{VV} \text{ for all } \Delta\},$$

$$\mathbf{MV} = \{(\mathcal{A}_1, \mathcal{A}_2, \dots) : \mathcal{A}_\Delta \in \mathcal{MV} \text{ for all } \Delta\},$$

$$\mathbf{SV} = \{(\mathcal{A}_1, \mathcal{A}_2, \dots) : \mathcal{A}_\Delta \in \mathcal{SV} \text{ for all } \Delta\},$$

$$\mathbf{VB} = \{(\mathcal{A}_1, \mathcal{A}_2, \dots) : \mathcal{A}_\Delta \in \mathcal{VB} \text{ for all } \Delta\},$$

$$\mathbf{MB} = \{(\mathcal{A}_1, \mathcal{A}_2, \dots) : \mathcal{A}_\Delta \in \mathcal{MB} \text{ for all } \Delta\},$$

$$\mathbf{SB} = \{(\mathcal{A}_1, \mathcal{A}_2, \dots) : \mathcal{A}_\Delta \in \mathcal{SB} \text{ for all } \Delta\}.$$

# Complexity classes

Let  $P$  be the class of all graph problems.

$VV_c = \{\Pi \in P : \text{there is } \mathbf{A} \in \mathbf{VV} \text{ that solves } \Pi \text{ assuming consistency}\},$

$VV = \{\Pi \in P : \text{there is } \mathbf{A} \in \mathbf{VV} \text{ that solves } \Pi\},$

$MV = \{\Pi \in P : \text{there is } \mathbf{A} \in \mathbf{MV} \text{ that solves } \Pi\},$

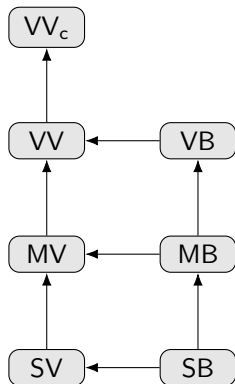
$SV = \{\Pi \in P : \text{there is } \mathbf{A} \in \mathbf{SV} \text{ that solves } \Pi\},$

$VB = \{\Pi \in P : \text{there is } \mathbf{A} \in \mathbf{VB} \text{ that solves } \Pi\},$

$MB = \{\Pi \in P : \text{there is } \mathbf{A} \in \mathbf{MB} \text{ that solves } \Pi\},$

$SB = \{\Pi \in P : \text{there is } \mathbf{A} \in \mathbf{SB} \text{ that solves } \Pi\}.$

# Containment relations between the classes

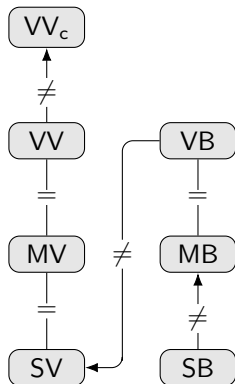


Trivial relations:

- $SV \subseteq MV \subseteq VV \subseteq VV_c$ ,
- $SB \subseteq MB \subseteq VB$ ,
- $VB \subseteq VV$ ,
- $MB \subseteq MV$ ,
- $SB \subseteq SV$ .

Non-trivial:  $SV \subseteq VB$ ?  $VB \subseteq SV$ ?

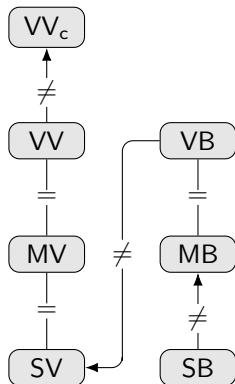
## Containment relations between the classes



Hella, Järvisalo, Kuusisto, Laurinharju, L.,  
Luosto, Suomela, Virtema (PODC 2012):

$$SB \not\subseteq MB = VB \not\subseteq SV = MV = VV \not\subseteq VV_c.$$

## Containment relations between the classes



Hella, Järvisalo, Kuusisto, Laurinharju, L., Luosto, Suomela, Virtema (PODC 2012):

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

Hella et al. also showed that constant-time variants of the classes can be characterised by certain modal logics.

# The relationship of MV and SV

Trivially  $SV \subseteq MV$ .

Hella, Järvisalo, Kuusisto, Laurinharju, L., Luosto, Suomela, Virtema (PODC 2012):

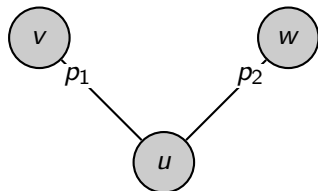
## Theorem

*Let  $\Pi$  be a graph problem and let  $T : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ . Assume that there is an algorithm  $\mathbf{A} \in \mathbf{MV}$  that solves  $\Pi$  in time  $T$ . Then there is an algorithm  $\mathbf{B} \in \mathbf{SV}$  that solves  $\Pi$  in time  $T'$ , where  $T'(n, \Delta) = T(n, \Delta) + 2\Delta - 2$ .*

It follows that  $SV = MV$ .

## Idea behind the simulation theorem

First, solve the following *simulation problem* by an  $\mathcal{SV}$ -algorithm:



If  $p_1 = p_2$ , then  
 $\text{output}(v) \neq \text{output}(w)$ .

Now the pair

(output, port number)

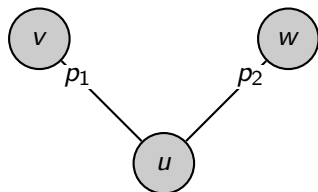
is distinct for each neighbour.

This takes  $2\Delta - 2$  communication rounds.



## Idea behind the simulation theorem

First, solve the following *simulation problem* by an  $\mathcal{SV}$ -algorithm:



If  $p_1 = p_2$ , then  
 $\text{output}(v) \neq \text{output}(w)$ .

Now the pair

(output, port number)

is distinct for each neighbour.

This takes  $2\Delta - 2$  communication rounds.

Then, simulate the  $\mathcal{MV}$ -algorithm by attaching the above pair to each message. That way we can reconstruct the message multiplicities.

## New results: Lower bounds for the simulation

Is the overhead of  $2\Delta - 2$  rounds really needed to reconstruct the message multiplicities by an  $\mathcal{SV}$ -algorithm?

### Theorem

*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  $1, 2, \dots, 2\Delta - 2$ .*

## New results: Lower bounds for the simulation

Is the overhead of  $2\Delta - 2$  rounds really needed to reconstruct the message multiplicities by an  $\mathcal{SV}$ -algorithm?

### Theorem

*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  $1, 2, \dots, 2\Delta - 2$ .*

### Theorem

*There is a graph problem  $\Pi$  that can be solved in one round by an algorithm in **MV** but that requires at least time  $T$ , where  $T(n, \Delta) \geq \Delta$  for all  $\Delta \geq 2$ , when solved by an algorithm in **SV**.*

## Notation for outgoing port numbers

If  $p(v, i) = (u, j)$ , we write  $\pi(v, u) = i$ . That is,  $\pi(v, u)$  is the number of the output port of  $v$  that is connected to  $u$ .

# How to prove that two nodes stay in the same state?

## Definition

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. An  $r$ -SV-bisimulation between nodes  $v \in V$  and  $v' \in V'$  is a sequence of binary relations  $B_r \subseteq B_{r-1} \subseteq \dots \subseteq B_0 \subseteq V \times V'$  such that the following conditions hold for  $1 \leq i \leq r$ :

- 1  $(v, v') \in B_r$ .
- 2 If  $(u, u') \in B_0$ , then  $\deg_G(u) = \deg_{G'}(u')$  and  $f(u) = f'(u')$ .
- 3 If  $(u, u') \in B_i$  and  $\{u, w\} \in E$ , then there is  $w' \in V'$  such that  $\{u', w'\} \in E'$ ,  $(w, w') \in B_{i-1}$  and  $\pi(w, u) = \pi'(w', u')$ .
- 4 If  $(u, u') \in B_i$  and  $\{u', w'\} \in E'$ , then there is  $w \in V$  such that  $\{u, w\} \in E$ ,  $(w, w') \in B_{i-1}$  and  $\pi(w, u) = \pi'(w', u')$ .

# How to prove that two nodes stay in the same state?

We say that  $v \in V$  and  $v' \in V'$  are  $r$ - $\mathcal{SV}$ -bisimilar and write  $(G, f, v, p) \stackrel{\mathcal{SV}}{\leftrightarrow}_r (G', f', v', p')$  (or simply  $v \stackrel{\mathcal{SV}}{\leftrightarrow}_r v'$ ) if there exists an  $r$ - $\mathcal{SV}$ -bisimulation between them.

## Lemma

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  $(G, f, v, p) \stackrel{\mathcal{SV}}{\leftrightarrow}_r (G', f', v', p')$  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 state of  $v$  and  $v'$  is identical in rounds  $0, 1, \dots, r$ .

# Bisimilarity

## Lemma

The  $r$ - $SV$ -bisimilarity relation  $\Leftrightarrow_r^{SV}$  is an equivalence relation 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 port numbering of  $G$  and  $v \in V$ .

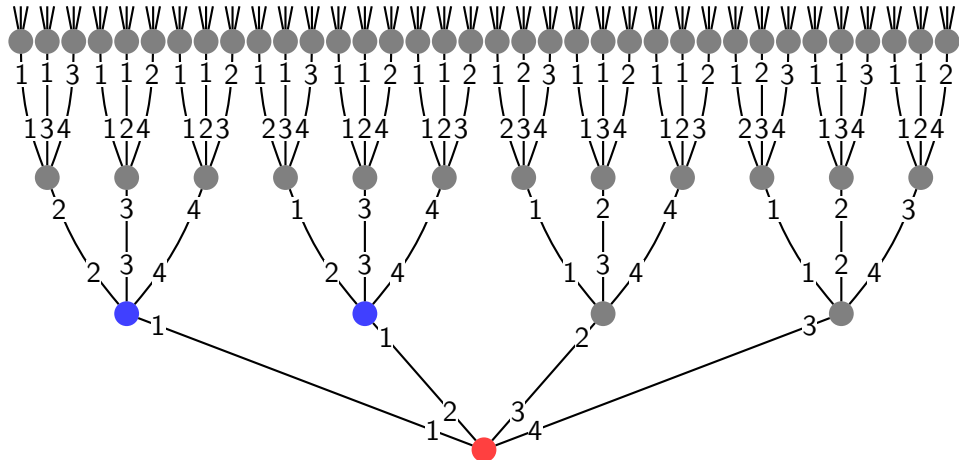
## Lemma

Let  $G = (V, E)$  and  $G' = (V', E')$  be graphs, let  $f$  and  $f'$  be inputs for  $G$  and  $G'$ , respectively, let  $p$  and  $p'$  be port numberings of  $G$  and  $G'$ , respectively, and let  $v \in V$ ,  $v' \in V'$ . Then  $(G, f, v, p) \Leftrightarrow_r^{SV} (G', f', v', p')$  iff the following conditions hold:

- 1  $(G, f, v, p) \Leftrightarrow_{r-1}^{SV} (G', f', v', p')$ .
- 2 If  $\{v, w\} \in E$ , then there is  $w' \in V'$  such that  $\{v', w'\} \in E'$ ,  $(G, f, w, p) \Leftrightarrow_{r-1}^{SV} (G', f', w', p')$  and  $\pi(w, v) = \pi'(w', v')$ .
- 3 If  $\{v', w'\} \in E'$ , then there is  $w \in V$  such that  $\{v, w\} \in E$ ,  $(G, f, w, p) \Leftrightarrow_{r-1}^{SV} (G', f', w', p')$  and  $\pi(w, v) = \pi'(w', v')$ .

# The lower bound construction $G_d$ for $d = 4$

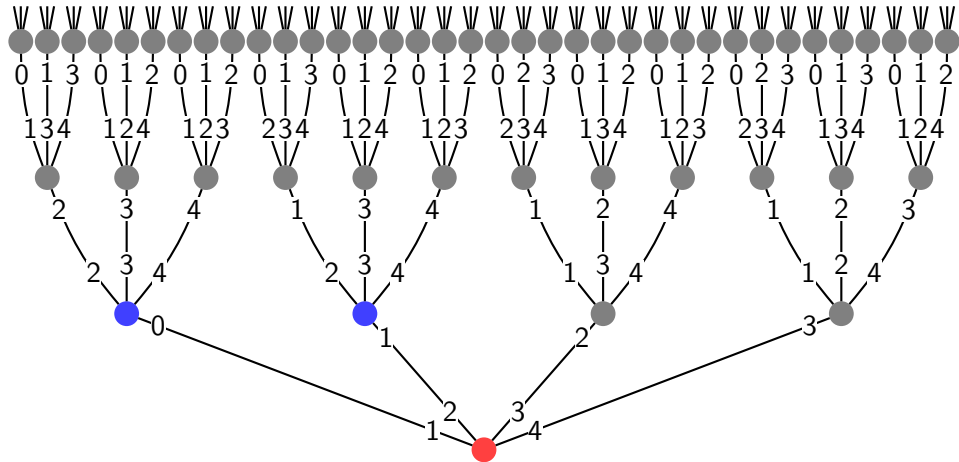
⋮





# The lower bound construction $G_d$ for $d = 4$

⋮



## Definition of the graph $G_d$

- 1  $\emptyset \in V_d$ .
- 2  $((1, 0)), ((2, 1)), ((3, 2)), ((4, 3)), \dots, ((d, d - 1)) \in V_d$ .
- 3 If  $(a_1, a_2, \dots, a_i) \in V_d$ , where  $i$  is odd and  $i < 2d$ , then  $(a_1, a_2, \dots, a_{i+1}^j) \in V_d$  for all  $j = 1, 2, \dots, d - 1$ , where  $a_{i+1}^j = (c_1^j, c_2^j)$  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

$$c_1^j = \min(\{1, 2, \dots, d\} \setminus \{b_2^+, c_1^1, c_1^2, \dots, c_1^{j-1}\}),$$

$$c_2^j = \min(\{1, 2, \dots, d\} \setminus \{b_1, c_2^1, c_2^2, \dots, c_2^{j-1}\}).$$

- 4 If  $(a_1, a_2, \dots, a_i) \in V_d$ , where  $i$  is even and  $0 < i < 2d$ , then  $(a_1, a_2, \dots, a_{i+1}^j) \in V_d$  for all  $j = 1, 2, \dots, d - 1$ , where  $a_{i+1}^j = (c_1^j, c_2^j)$  is defined as follows. Let  $(b_1, b_2) = a_i$ . Define

$$c_1^j = \min(\{1, 2, \dots, d\} \setminus \{b_2, c_1^1, c_1^2, \dots, c_1^{j-1}\}),$$

$$c_2^j = \min(\{0, 1, \dots, d - 1\} \setminus \{b_1, c_2^1, c_2^2, \dots, c_2^{j-1}\}).$$

## Definition of the graph $G_d$

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  $i \in \{0, 1, \dots\}$ .

If  $v = (a_1, a_2, \dots, a_i)$  and  $u = (a_1, a_2, \dots, a_{i+1})$ , where  $a_{i+1} = (b_1, b_2)$ , the outgoing port number from  $v$  to  $u$  is  $\pi_d(v, u) = b_1$  and the outgoing port number from  $u$  to  $v$  is  $\pi_d(u, v) = b_2$ .

## Pairs of separating walks (PSWs)

A *walk* is a sequence  $\bar{v} = (v_0, v_1, \dots, v_k)$  of nodes such that  $\{v_i, v_{i+1}\} \in E_d$  for all  $i = 0, 1, \dots, k - 1$ .

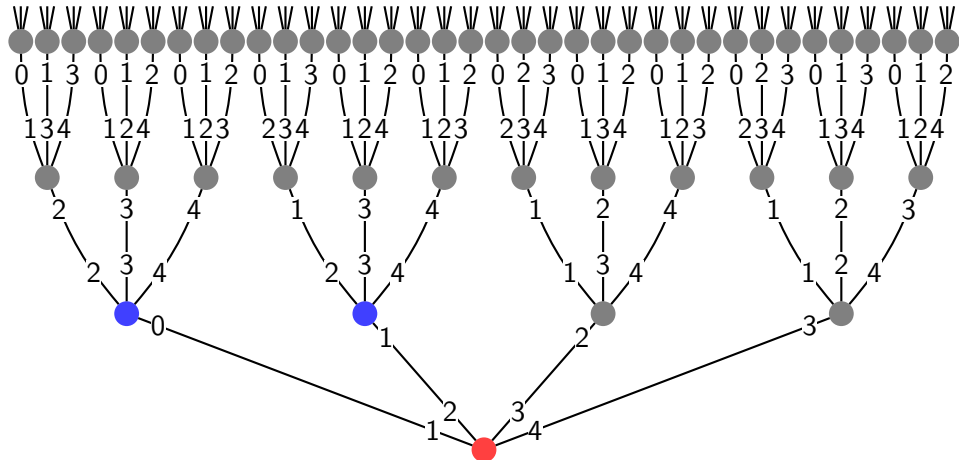
A pair  $(\bar{v}_1, \bar{v}_2)$  of walks, where  $\bar{v}_i = (v_0^i, v_1^i, \dots, v_k^i)$  for all  $i = 1, 2$ , is called a *pair of separating walks (PSW)* of length  $k$  in  $G_d$  if the following conditions hold:

- 1  $v_0^1 = ((1, 0))$  and  $v_0^2 = ((2, 1))$ .
- 2  $\pi_d(v_j^1, v_{j-1}^1) = \pi_d(v_j^2, v_{j-1}^2)$  for all  $j = 1, 2, \dots, k$ .
- 3 There is  $v_{k+1}^1 \in V_d$  with  $\{v_k^1, v_{k+1}^1\} \in E_d$  such that there is no  $v_{k+1}^2 \in V_d$  for which  $\{v_k^2, v_{k+1}^2\} \in E_d$  and  $\pi_d(v_{k+1}^1, v_k^1) = \pi_d(v_{k+1}^2, v_k^2)$ .

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  $k' < k$ .

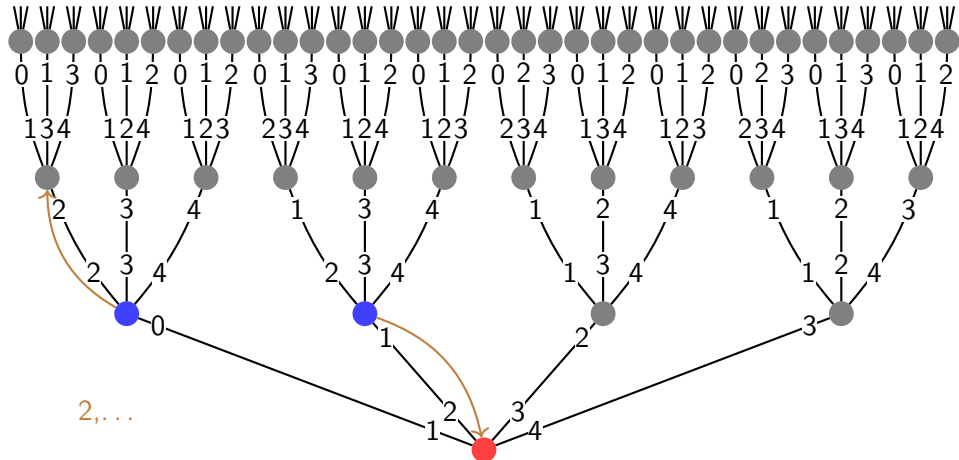
# A pair of separating walks in $G_4$

⋮



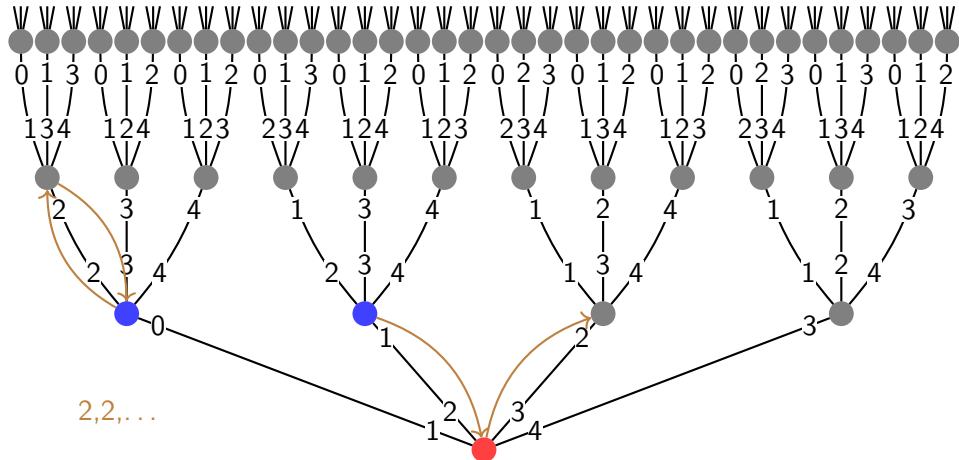
# A pair of separating walks in $G_4$

⋮



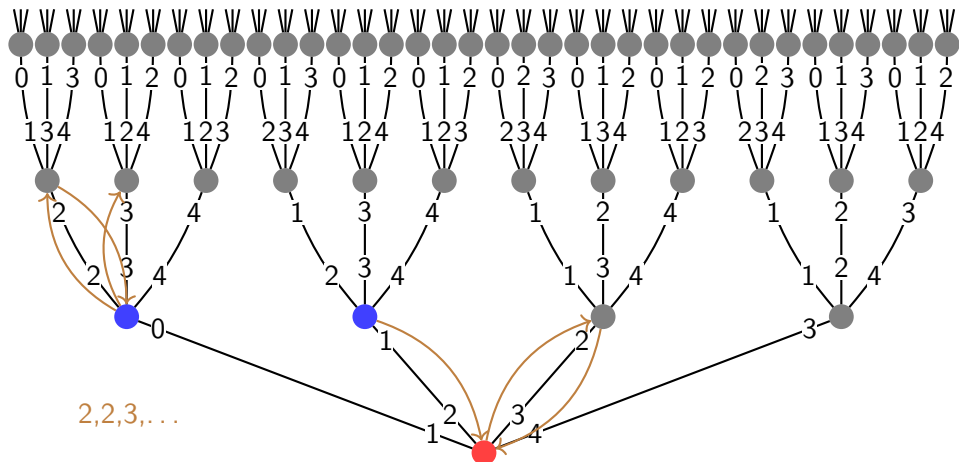
# A pair of separating walks in $G_4$

⋮



# A pair of separating walks in $G_4$

⋮



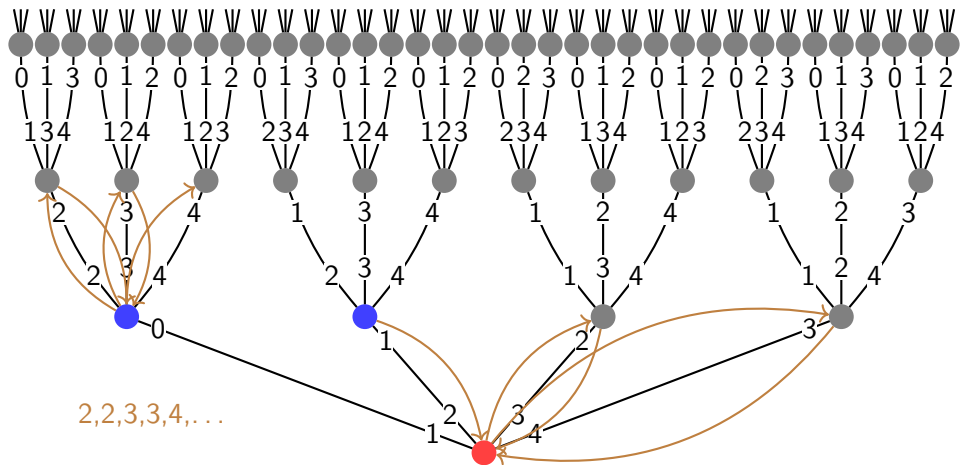
2,2,3,...





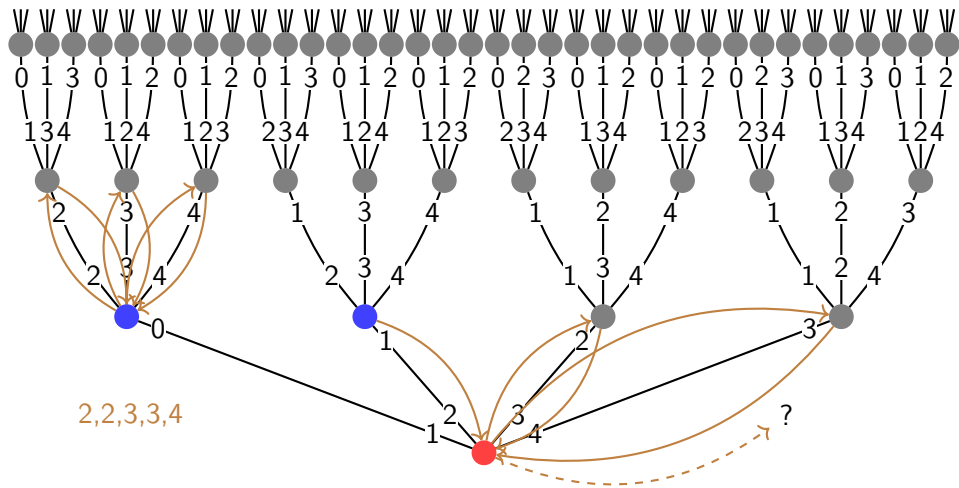
# A pair of separating walks in $G_4$

⋮



# A pair of separating walks in $G_4$

⋮



2,2,3,3,4

# Proof idea

- 1 The length of a critical PSW in  $G_d$  is at least  $2d - 3$ .
  - 1 The sequence of port numbers starts from 1 or 2.
  - 2 The numbers grow slowly along the walks.
  - 3 Eventually the sequence reaches  $d$ .
- 2 If  $k$  is the largest integer for which  $((1, 0)) \stackrel{SV}{\leftrightarrow}_k ((2, 1))$ , then there is a PSW of length  $k$  in  $G_d$ .

# Definitions

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  $v$ .

### Lemma

*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  $G_{d+1}$ .*

## Easy observations

### Lemma

*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

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

## Easy observations

### Lemma

*Let  $v = (a_1, a_2, \dots, a_i) \in V_d$ , where  $i < 2d$ . If  $v$  is odd, then for all  $a \in \{1, 2, \dots, d\}$  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  $a \in \{0, 1, \dots, d-1\}$  or for all  $a \in \{0, 1, \dots, d-2, d\}$  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  $v$ .*



## Easy observations

### Lemma

Let  $v = (a_1, a_2, \dots, a_i) \in V_d$ , where  $i < 2d$ . If  $v$  is odd, then for all  $a \in \{1, 2, \dots, d\}$  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  $a \in \{0, 1, \dots, d-1\}$  or for all  $a \in \{0, 1, \dots, d-2, d\}$  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

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  $\pi_{d+1}(u, v) \in \{d-1, d\}$ .

# Walks in isomorphic subtrees

## Lemma

Let  $(\bar{v}_1, \bar{v}_2)$ , where  $\bar{v}_i = (v_0^i, v_1^i, \dots, v_k^i)$  for some  $k \leq 2d - 3$  and all  $i = 1, 2$ , be a PSW in  $G_d$ . If for some  $\ell \in \{0, 1, \dots, k - 1\}$  the node  $v_{\ell+1}^i$  is a child of node  $v_\ell^i$  for all  $i = 1, 2$ , and we have  $\pi_d(v_\ell^1, v_{\ell+1}^1) = \pi_d(v_\ell^2, v_{\ell+1}^2)$ , then  $(\bar{v}_1, \bar{v}_2)$  is not a critical PSW in  $G_d$ .

### Lemma

*Let  $(\bar{v}_1, \bar{v}_2)$  be a PSW of length  $k \leq 2d - 3$  in  $G_d$ . Then there is a PSW of length  $k + 2$  in  $G_{d+1}$ .*

Second-to-last node is in  $V_d \setminus V_{d-1}$

### Lemma

*Let  $(\bar{v}_1, \bar{v}_2)$ , where  $\bar{v}_i = (v_0^i, v_1^i, \dots, v_k^i)$  for some  $k \leq 2d - 3$  and all  $i = 1, 2$ , be a critical PSW in  $G_d$ . Then we have  $v_{k-1}^i \in V_d \setminus V_{d-1}$  for some  $i \in \{1, 2\}$ .*

## Pair of walks that is not a PSW

### Lemma

Let  $(\bar{v}_1, \bar{v}_2)$ , where  $\bar{v}_i = (v_0^i, v_1^i, \dots, v_k^i)$  for some  $k \leq 2d - 3$  and all  $i = 1, 2$ , be a pair of walks in  $G_d$  such that conditions (1) and (2) hold. If  $(\bar{v}_1, \bar{v}_2)$  is not a PSW in  $G_d$ , then for each neighbour  $v_{k+1}^1 \in V_d$  of  $v_k^1$  there is a neighbour  $v_{k+1}^2 \in V_d$  of  $v_k^2$  such that  $\pi_d(v_{k+1}^1, v_k^1) = \pi_d(v_{k+1}^2, v_k^2)$ , and vice versa.

# The main lemma

## Lemma

Let  $(\bar{v}_1, \bar{v}_2)$ , where  $\bar{v}_i = (v_0^i, v_1^i, \dots, v_k^i)$  for some  $k \leq 2d - 3$  and all  $i = 1, 2$ , be a critical PSW in  $G_d$ . Then  $(\bar{v}'_1, \bar{v}'_2)$ , where  $\bar{v}'_i = (v_0^i, v_1^i, \dots, v_{k-2}^i)$  for all  $i = 1, 2$ , is a PSW in  $G_{d-1}$ .

## Minimum length of a PSW and bisimilarity

### Lemma

*Let  $(\bar{v}_1, \bar{v}_2)$  be a PSW of length  $k \leq 2d - 3$  in  $G_d$ . Then  $k \geq 2d - 3$ .*

## Minimum length of a PSW and bisimilarity

### Lemma

Let  $(\bar{v}_1, \bar{v}_2)$  be a PSW of length  $k \leq 2d - 3$  in  $G_d$ . Then  $k \geq 2d - 3$ .

### Lemma

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



# Conclusion

$\mathcal{MV}$ : Send a vector, receive a multiset.

$\mathcal{SV}$ : Send a vector, receive a set.

Previously:

- It is possible to simulate  $\mathcal{MV}$  in  $\mathcal{SV}$  by using  $2\Delta - 2$  extra rounds.

This work:

- $2\Delta - 2$  rounds are necessary to solve the simulation problem.
- There is a graph problem for which the difference in running time between  $\mathcal{MV}$  and  $\mathcal{SV}$  is  $\Delta - 1$  rounds.

The thesis *A Classification of Weak Models of Distributed Computing* is available at <http://hdl.handle.net/10138/144214>.