

# PING PONG IN DANGEROUS GRAPHS

## Optimal Black Hole Search with Pebbles

Paola Flocchini<sup>1</sup> David Ilcinkas<sup>2</sup> Nicola Santoro<sup>3</sup>

<sup>1</sup>SITE, University of Ottawa, Canada

<sup>2</sup>CNRS, LaBRI, Universit Bordeaux I, France

<sup>3</sup>School of Computer Science, Carleton University, Canada

ALADDIN meeting  
May 30, 2008

# Problem

## Black hole in a network

**Node** blocking and **destroying any mobile agent** entering it.

Motivations :

- Site which is destroyed or dangerous for a physical robot
- Node infected by a “killer” virus
- Mute or crashed node

## Black hole search

From a safe node (homebase), a team of mobile agents has to locate the black hole(s).

Performance : nb of agents, nb of moves, time

# Problem

## Black hole in a network

**Node** blocking and **destroying any mobile agent** entering it.

Motivations :

- Site which is destroyed or dangerous for a physical robot
- Node infected by a “killer” virus
- Mute or crashed node

## Black hole search

From a safe node (homebase), a team of mobile agents has to **locate the black hole(s)**.

Performance : nb of agents, nb of moves, time

# Problem

## Black hole in a network

**Node** blocking and **destroying any mobile agent** entering it.

Motivations :

- Site which is destroyed or dangerous for a physical robot
- Node infected by a “killer” virus
- Mute or crashed node

## Black hole search

From a safe node (homebase), a team of mobile agents has to **locate the black hole(s)**.

Performance : **nb of agents**, **nb of moves**, time

# (A)synchronous

## Synchronous

Agents act simultaneously at every pulse of a global clock.

⇒ **Possible to wait** for an agent gone exploring a potentially dangerous edge.

## Asynchronous

Every action takes a finite but unbounded time.

⇒ A priori impossible to distinguish a very slow link from a link leading to the black hole.

⇒ Useless to wait for an agent.

⇒ Pb equivalent to the exploration of dangerous graphs.

# (A)synchronous

## Synchronous

Agents act simultaneously at every pulse of a global clock.

⇒ **Possible to wait** for an agent gone exploring a potentially dangerous edge.

## Asynchronous

Every action takes a **finite but unbounded time**.

⇒ A priori impossible to distinguish a **very slow link** from a **link leading to the black hole**.

⇒ **Useless to wait** for an agent.

⇒ Pb equivalent to the **exploration of dangerous graphs**.

# Necessary hypotheses (in asynchronous)

## Required knowledge

- Knowledge of the **number of black holes**  
→ Here, exactly one black hole
- Knowledge of the **number of nodes**  $n$

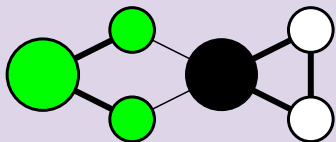
2-vertex-connected

# Necessary hypotheses (in asynchronous)

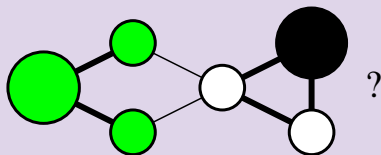
## Required knowledge

- Knowledge of the **number of black holes**  
 → Here, exactly one black hole
- Knowledge of the **number of nodes  $n$**

## 2-vertex-connected



or





# Model (still in asynchronous)

## Communication/coordination

No direct communication but

Whiteboards (memory space on each node).

ou Edge markers (mark a specific port).

ou Pebbles (node markers).

## Computational step

- Observation : sense for the presence of a pebble/message on the node.
- Interaction : put down or pick up a pebble/message.
- Action : move along an edge (or terminate).

This set of steps (forming a computational step) is executed atomically in mutual exclusion.

# Model (still in asynchronous)

## Communication/coordination

No direct communication but

Whiteboards (memory space on each node).

ou Edge markers (mark a specific port).

ou Pebbles (node markers).

## Computational step

- **Observation** : sense for the presence of a pebble/message on the node.
- **Interaction** : put down or pick up a pebble/message.
- **Action** : move along an edge (or terminate).

This set of steps (forming a computational step) is executed atomically **in mutual exclusion**.

# Related work (asynchronous, whiteboards)

## Seminal results

[Dobrev, Flocchini, Prencipe, Santoro. Dist. Comp. 2006]

	nb of agents	nb of moves
No information (besides $n$ )	$\Delta + 1$	$\Theta(n^2)$
Sense of direction	2	$\Theta(n^2)$
Complete knowledge	2	$\Theta(n \log n)$

Improvements (with complete knowledge)

[Dobrev, Flocchini, Santoro. SIROCCO 2004]

- Specific topologies (but not the ring)
- Nb of moves :  $O(n)$

[Dobrev et al. Networks 2006]

- Nb of moves :  $O(n + D \log D)$  ( $D = \text{diameter}$ )

# Related work (asynchronous, whiteboards)

## Seminal results

[Dobrev, Flocchini, Prencipe, Santoro. Dist. Comp. 2006]

	nb of agents	nb of moves
No information (besides $n$ )	$\Delta + 1$	$\Theta(n^2)$
Sense of direction	2	$\Theta(n^2)$
Complete knowledge	2	$\Theta(n \log n)$

## Improvements (with complete knowledge)

[Dobrev, Flocchini, Santoro. SIROCCO 2004]

- **Specific topologies** (but not the ring)
- Nb of moves :  $O(n)$

[Dobrev et al. Networks 2006]

- Nb of moves :  $O(n + D \log D)$  ( $D = \text{diameter}$ )

# Related work (asynchronous)

Minimal conditions on agents coordination.

Edge markers, ring, FIFO links

[Dobrev, Santoro, Shi. CIAC 2006]

- $O(1)$  edge markers in total, up to 3 per node
- Nb of moves :  $\Theta(n \log n)$

Edge markers, arbitrary graphs, FIFO links

[Dobrev, Flocchini, Kralovic, Santoro. IFIP TCS 2006]

- No information (besides  $n$ )
- Nb of agents :  $\Delta + 1$
- Nb of moves : roughly  $O(n^{10})$

Pb : Is coordination through pebbles sufficient ?

# Related work (asynchronous)

Minimal conditions on agents coordination.

## Edge markers, ring, FIFO links

[Dobrev, Santoro, Shi. CIAC 2006]

- $O(1)$  edge markers in total, up to 3 per node
- Nb of moves :  $\Theta(n \log n)$

## Edge markers, arbitrary graphs, FIFO links

[Dobrev, Flocchini, Kralovic, Santoro. IFIP TCS 2006]

- No information (besides  $n$ )
- Nb of agents :  $\Delta + 1$
- Nb of moves : roughly  $O(n^{10})$

Pb : Is coordination through pebbles sufficient ?

# Related work (asynchronous)

Minimal conditions on agents coordination.

## Edge markers, ring, FIFO links

[Dobrev, Santoro, Shi. CIAC 2006]

- $O(1)$  edge markers in total, up to 3 per node
- Nb of moves :  $\Theta(n \log n)$

## Edge markers, arbitrary graphs, FIFO links

[Dobrev, Flocchini, Kralovic, Santoro. IFIP TCS 2006]

- No information (besides  $n$ )
- Nb of agents :  $\Delta + 1$
- Nb of moves : roughly  $O(n^{10})$

Pb : Is coordination through pebbles sufficient ?

# Related work (asynchronous)

Minimal conditions on agents coordination.

## Edge markers, ring, FIFO links

[Dobrev, Santoro, Shi. CIAC 2006]

- $O(1)$  edge markers in total, up to 3 per node
- Nb of moves :  $\Theta(n \log n)$

## Edge markers, arbitrary graphs, FIFO links

[Dobrev, Flocchini, Kralovic, Santoro. IFIP TCS 2006]

- No information (besides  $n$ )
- Nb of agents :  $\Delta + 1$
- Nb of moves : roughly  $O(n^{10})$

Pb : Is **coordination through pebbles sufficient** ?



# Our results

## Model

- Complete knowledge (arbitrary graphs)
- Undistinguishable **pebbles** (at most one per node)
- **Links do not need to be FIFO**

## Optimal result

- 1 pebble per agent
- 2 agents
- $\Theta(n \log n)$  moves

# Our results

## Model

- Complete knowledge (arbitrary graphs)
- Undistinguishable pebbles (at most one per node)
- Links do not need to be FIFO

## Optimal result

- 1 pebble per agent
- 2 agents
- $\Theta(n \log n)$  moves

# Case of the ring

## Model

- Two identical agents equipped with one pebble each
- Identical pebbles
- At most one pebble per node / carried by an agent
- Asynchronous ring with exactly one black hole

## Definition

- Edge of the homebase with smallest port: right
- Other direction: left

# Case of the ring

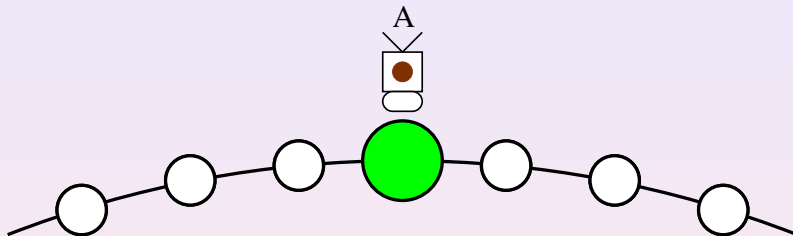
## Model

- Two identical agents equipped with one pebble each
- Identical pebbles
- At most one pebble per node / carried by an agent
- Asynchronous ring with exactly one black hole

## Definition

- Edge of the homebase with smallest port: right
- Other direction: left

# First steps

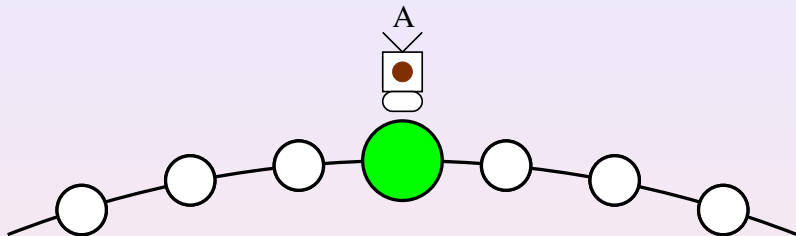


Beginning of the algorithm

**If** node is empty **then**

Go right

# First steps



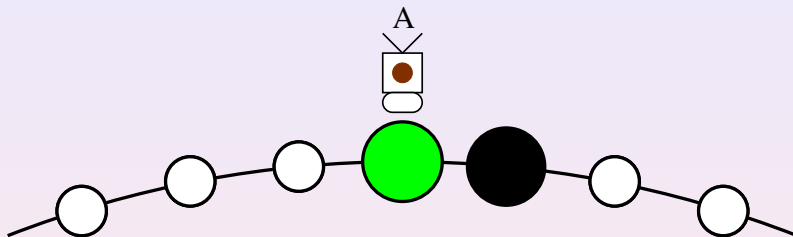
## Beginning of the algorithm

**If** node is empty **then**

Do not put down the pebble

Go right

# First steps



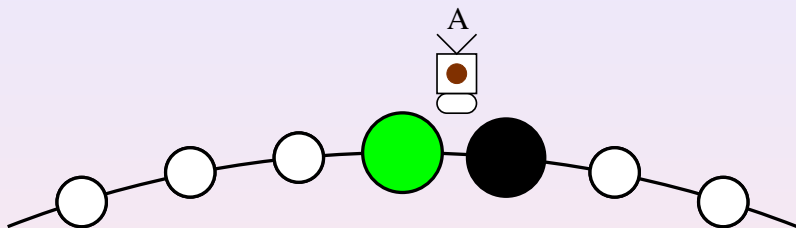
## Beginning of the algorithm

**If** node is empty **then**

Do not put down the pebble

Go right

# First steps



## Beginning of the algorithm

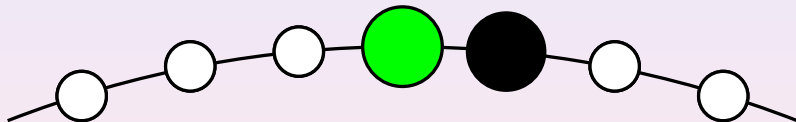
**If** node is empty **then**

Do not put down the pebble

Go right



# First steps



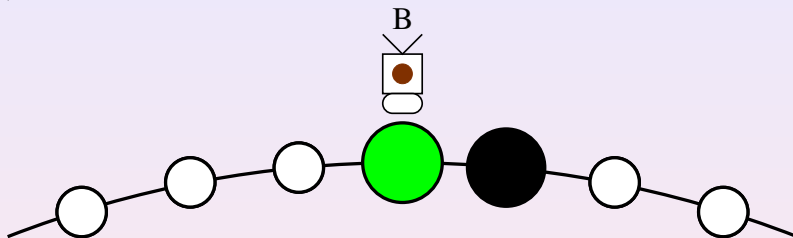
## Beginning of the algorithm

**If** node is empty **then**

Do not put down the pebble

Go right

# First steps



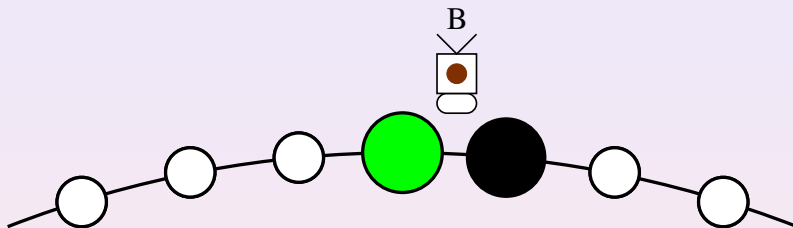
## Beginning of the algorithm

**If** node is empty **then**

Do not put down the pebble

Go right

# First steps



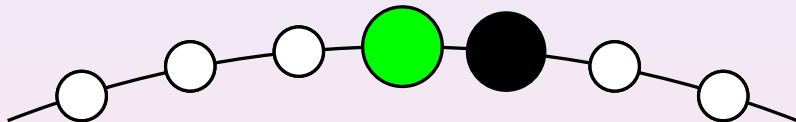
## Beginning of the algorithm

**If** node is empty **then**

Do not put down the pebble

Go right

# First steps



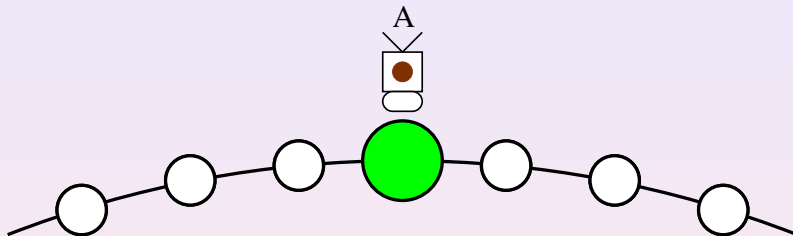
## Beginning of the algorithm

**If** node is empty **then**

Do not put down the pebble

Go right

# First steps



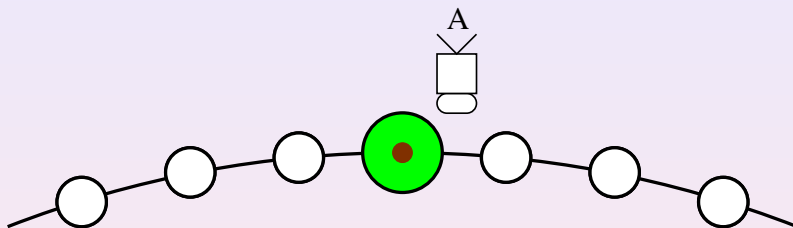
## Beginning of the algorithm

**If** node is empty **then**

Put down the pebble

Go right

# First steps



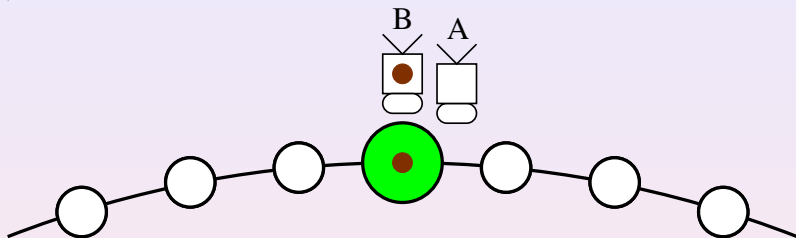
## Beginning of the algorithm

**If** node is empty **then**

Put down the pebble

Go right

# First steps



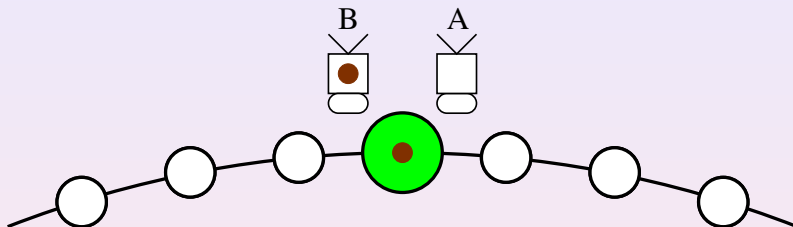
## Beginning of the algorithm

**If** node is empty **then**

Put down the pebble

Go right

# First steps

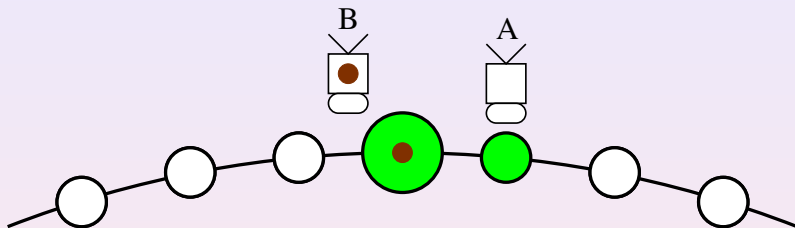


## Beginning of the algorithm

```
If node is empty then  
    Put down the pebble  
    Go right  
else Go left endif
```



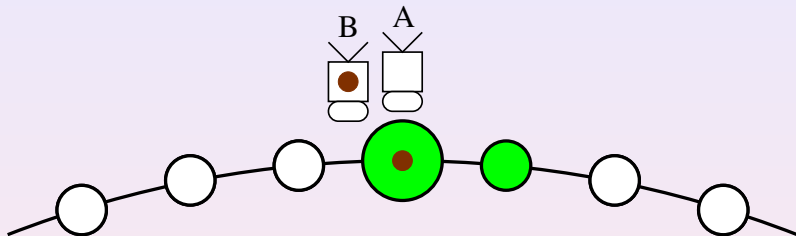
# First steps



## Beginning of the algorithm

```
If node is empty then  
    Put down the pebble  
    Go right  
else Go left endif
```

# First steps

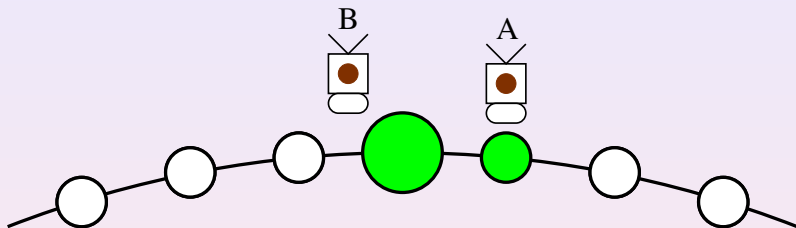


## Beginning of the algorithm

```

If node is empty then
  Put down the pebble
  Go right
else Go left endif
  
```

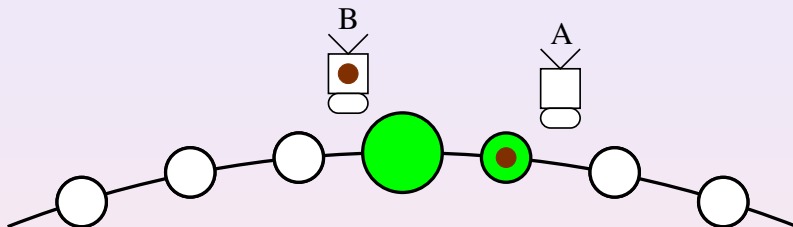
# First steps



## Beginning of the algorithm

```
If node is empty then  
    Put down the pebble  
    Go right  
else Go left endif
```

# First steps



## Beginning of the algorithm

```
If node is empty then  
    Put down the pebble  
    Go right  
else Go left endif
```

# Cautious walk

Exploration of  $e = \{u, v\}$  from  $u$  to  $v$

## Simple cautious walk

Initially: agent at  $u$  with a pebble, node without a pebble

- Put down the pebble (= warn the other agent) and go to  $v$
- Come back to  $u$
- Retrieve the pebble and return at  $v$

## Double cautious walk

Initially: agent at  $u$  with a pebble, node with a pebble

- Go to  $v$
- Put down the pebble and go back to  $u$
- Retrieve the other pebble and return at  $v$

# Cautious walk

Exploration of  $e = \{u, v\}$  from  $u$  to  $v$

## Simple cautious walk

Initially: agent at  $u$  with a pebble, node without a pebble

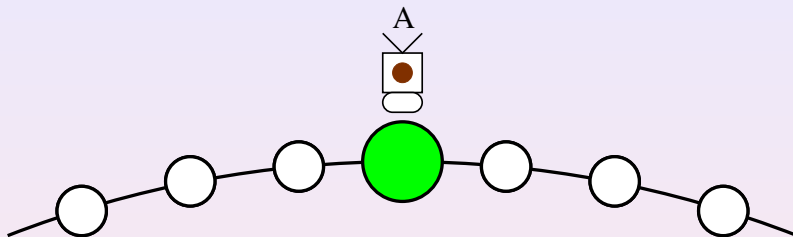
- Put down the pebble (= warn the other agent) and go to  $v$
- Come back to  $u$
- Retrieve the pebble and return at  $v$

## Double cautious walk

Initially: agent at  $u$  with a pebble, node with a pebble

- Go to  $v$
- Put down the pebble and go back to  $u$
- Retrieve the other pebble and return at  $v$

# Ping Pong algorithm

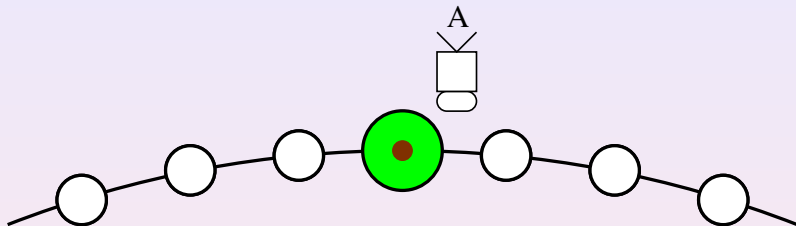


## Quick description

If the agent “controls”

- 1 pebble → Simple cautious walk to the right
- 2 pebbles → Double cautious walk to the left
- 0 pebbles → Non-cautious walk to the left

# Ping Pong algorithm



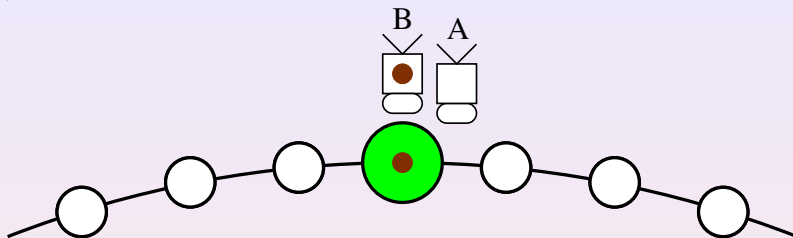
## Quick description

If the agent “controls”

- 1 pebble → Simple cautious walk to the right
- 2 pebbles → Double cautious walk to the left
- 0 pebbles → Non-cautious walk to the left



# Ping Pong algorithm

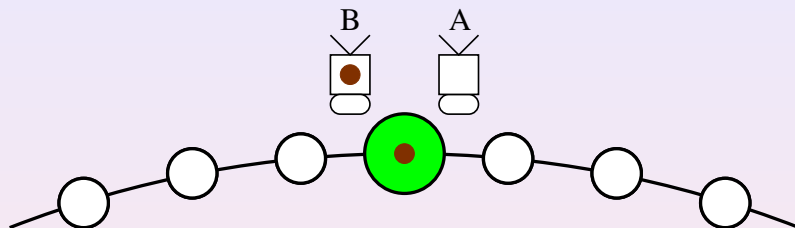


## Quick description

If the agent “controls”

- 1 pebble → Simple cautious walk to the right
- 2 pebbles → Double cautious walk to the left
- 0 pebbles → Non-cautious walk to the left

# Ping Pong algorithm

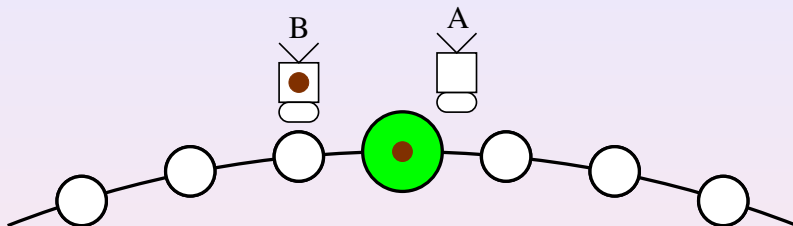


## Quick description

If the agent “controls”

- 1 pebble → Simple cautious walk to the right
- 2 pebbles → Double cautious walk to the left
- 0 pebbles → Non-cautious walk to the left

# Ping Pong algorithm

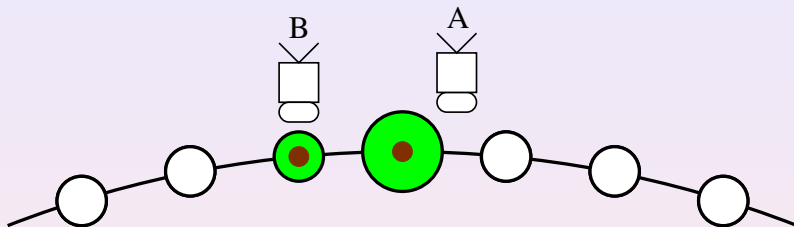


## Quick description

If the agent “controls”

- 1 pebble → Simple cautious walk to the right
- 2 pebbles → Double cautious walk to the left
- 0 pebbles → Non-cautious walk to the left

# Ping Pong algorithm

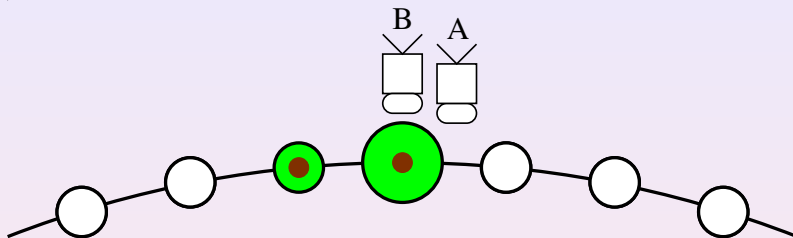


## Quick description

If the agent “controls”

- 1 pebble → Simple cautious walk to the right
- 2 pebbles → Double cautious walk to the left
- 0 pebbles → Non-cautious walk to the left

# Ping Pong algorithm

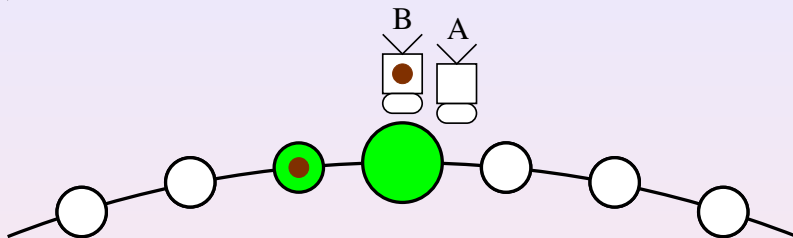


## Quick description

If the agent “controls”

- 1 pebble → **Simple** cautious walk to the **right**
- 2 pebbles → **Double** cautious walk to the **left**
- 0 pebbles → **Non-cautious** walk to the **left**

# Ping Pong algorithm

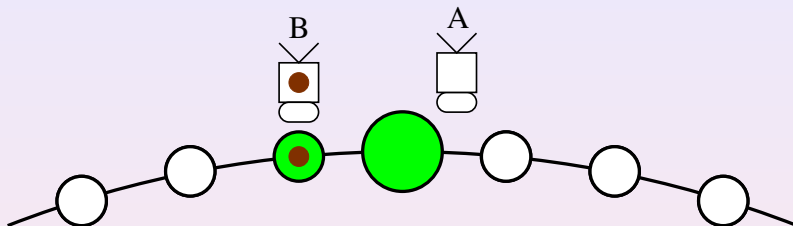


## Quick description

If the agent “controls”

- 1 pebble → Simple cautious walk to the right
- 2 pebbles → Double cautious walk to the left
- 0 pebbles → Non-cautious walk to the left

# Ping Pong algorithm

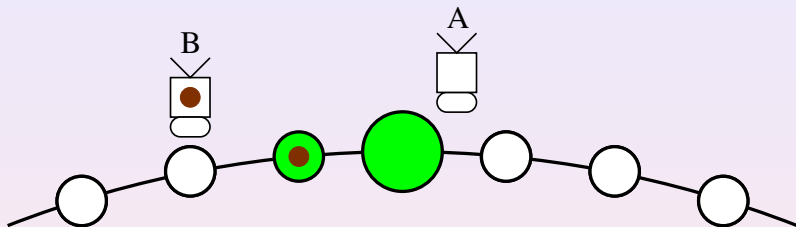


## Quick description

If the agent “controls”

- 1 pebble → Simple cautious walk to the right
- 2 pebbles → Double cautious walk to the left
- 0 pebbles → Non-cautious walk to the left

# Ping Pong algorithm



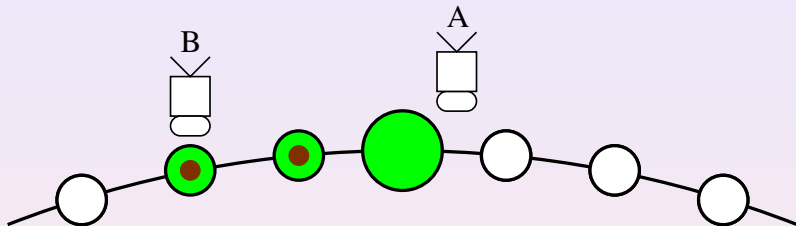
## Quick description

If the agent “controls”

- 1 pebble → Simple cautious walk to the right
- 2 pebbles → Double cautious walk to the left
- 0 pebbles → Non-cautious walk to the left



# Ping Pong algorithm

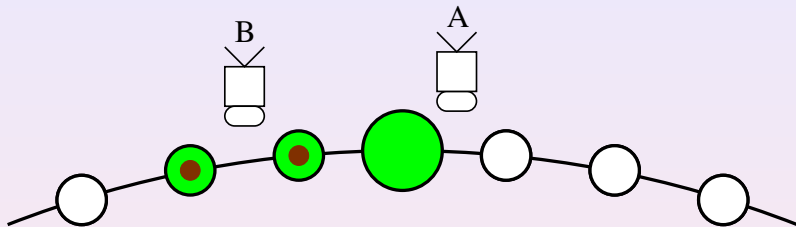


## Quick description

If the agent “controls”

- 1 pebble → Simple cautious walk to the right
- 2 pebbles → Double cautious walk to the left
- 0 pebbles → Non-cautious walk to the left

# Ping Pong algorithm

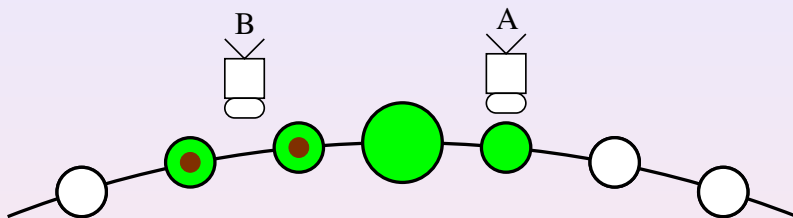


## Quick description

If the agent “controls”

- 1 pebble → Simple cautious walk to the right
- 2 pebbles → Double cautious walk to the left
- 0 pebbles → Non-cautious walk to the left

# Ping Pong algorithm

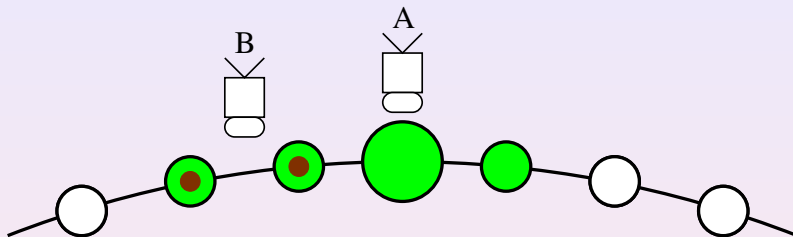


## Quick description

If the agent “controls”

- 1 pebble → Simple cautious walk to the right
- 2 pebbles → Double cautious walk to the left
- 0 pebbles → Non-cautious walk to the left

# Ping Pong algorithm

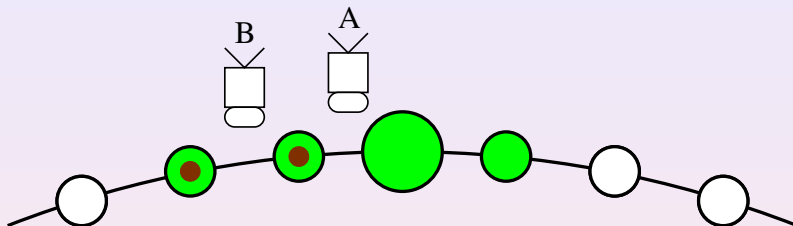


## Quick description

If the agent “controls”

- 1 pebble → Simple cautious walk to the right
- 2 pebbles → Double cautious walk to the left
- 0 pebbles → Non-cautious walk to the left

# Ping Pong algorithm

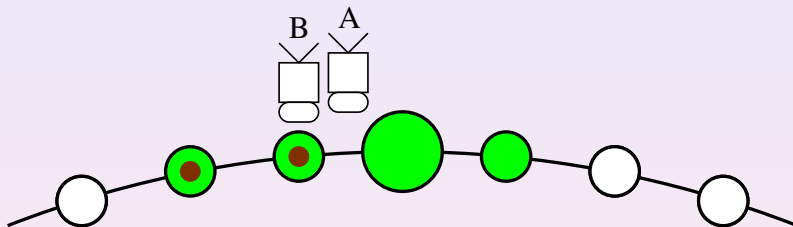


## Quick description

If the agent “controls”

- 1 pebble → Simple cautious walk to the right
- 2 pebbles → Double cautious walk to the left
- 0 pebbles → Non-cautious walk to the left

# Ping Pong algorithm

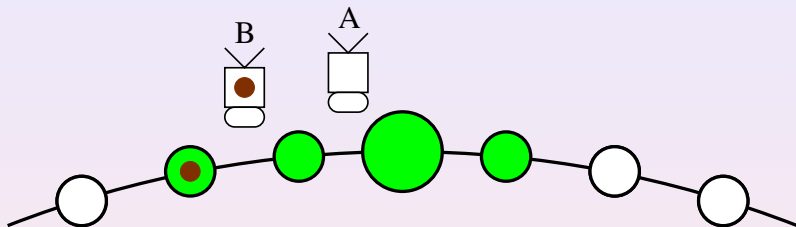


## Quick description

If the agent “controls”

- 1 pebble → Simple cautious walk to the right
- 2 pebbles → Double cautious walk to the left
- 0 pebbles → Non-cautious walk to the left

# Ping Pong algorithm

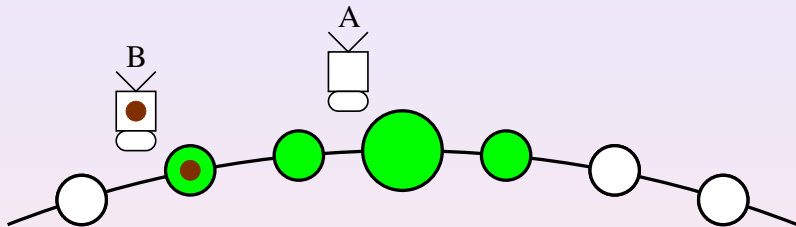


## Quick description

If the agent “controls”

- 1 pebble → Simple cautious walk to the right
- 2 pebbles → Double cautious walk to the left
- 0 pebbles → Non-cautious walk to the left

# Ping Pong algorithm



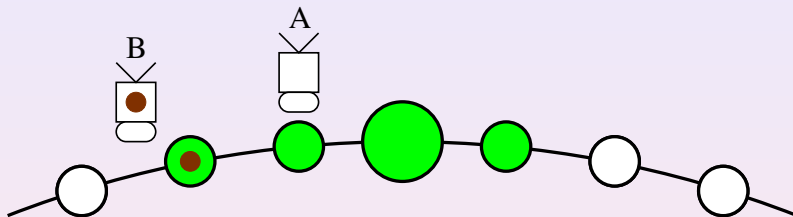
## Quick description

If the agent “controls”

- 1 pebble → Simple cautious walk to the right
- 2 pebbles → Double cautious walk to the left
- 0 pebbles → Non-cautious walk to the left



# Ping Pong algorithm

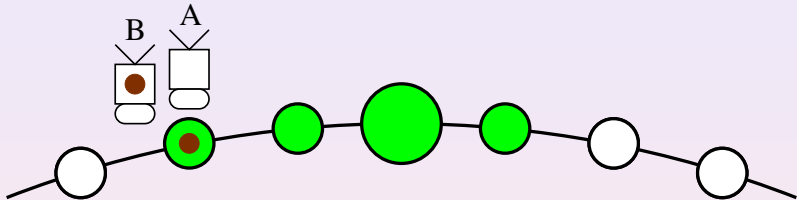


## Quick description

If the agent “controls”

- 1 pebble → Simple cautious walk to the right
- 2 pebbles → Double cautious walk to the left
- 0 pebbles → Non-cautious walk to the left

# Ping Pong algorithm

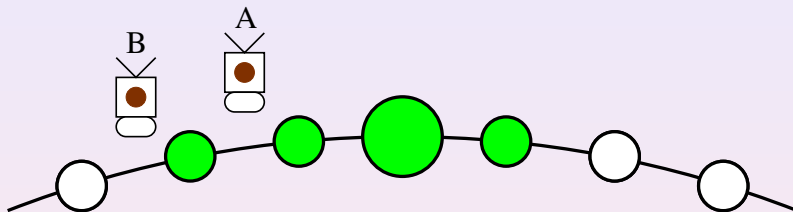


## Quick description

If the agent “controls”

- 1 pebble → Simple cautious walk to the right
- 2 pebbles → Double cautious walk to the left
- 0 pebbles → Non-cautious walk to the left

# Ping Pong algorithm

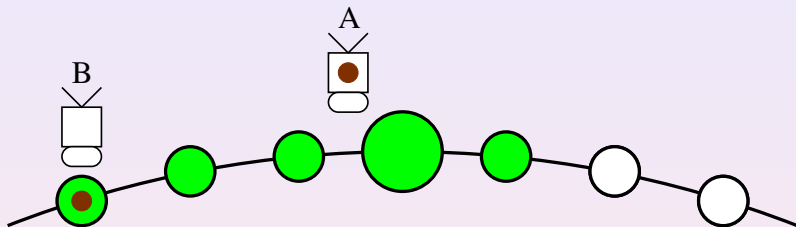


## Quick description

If the agent “controls”

- 1 pebble → Simple cautious walk to the right
- 2 pebbles → Double cautious walk to the left
- 0 pebbles → Non-cautious walk to the left

# Ping Pong algorithm

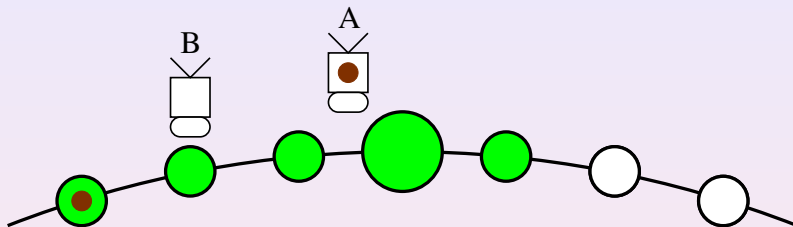


## Quick description

If the agent “controls”

- 1 pebble → Simple cautious walk to the right
- 2 pebbles → Double cautious walk to the left
- 0 pebbles → Non-cautious walk to the left

# Ping Pong algorithm

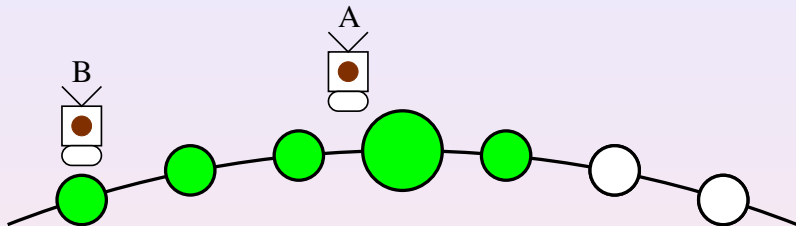


## Quick description

If the agent “controls”

- 1 pebble → Simple cautious walk to the right
- 2 pebbles → Double cautious walk to the left
- 0 pebbles → Non-cautious walk to the left

# Ping Pong algorithm

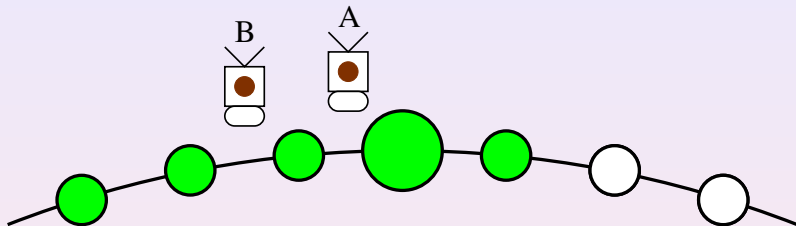


## Quick description

If the agent “controls”

- 1 pebble → Simple cautious walk to the right
- 2 pebbles → Double cautious walk to the left
- 0 pebbles → Non-cautious walk to the left

# Ping Pong algorithm

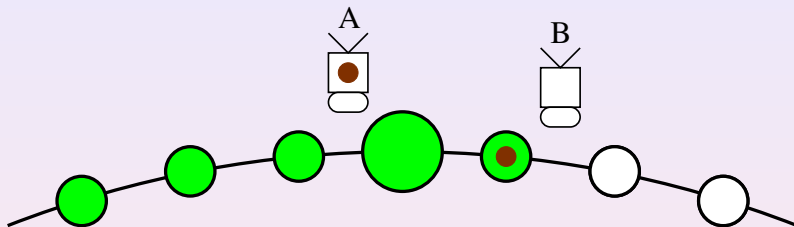


## Quick description

If the agent “controls”

- 1 pebble → Simple cautious walk to the right
- 2 pebbles → Double cautious walk to the left
- 0 pebbles → Non-cautious walk to the left

# Ping Pong algorithm



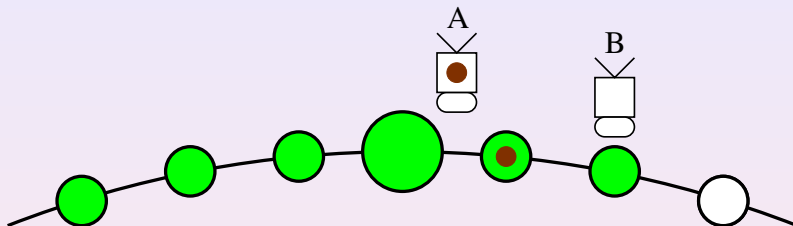
## Quick description

If the agent “controls”

- 1 pebble → Simple cautious walk to the right
- 2 pebbles → Double cautious walk to the left
- 0 pebbles → Non-cautious walk to the left



# Ping Pong algorithm

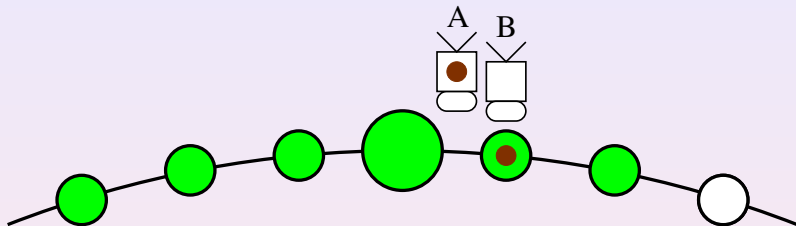


## Quick description

If the agent “controls”

- 1 pebble → Simple cautious walk to the right
- 2 pebbles → Double cautious walk to the left
- 0 pebbles → Non-cautious walk to the left

# Ping Pong algorithm

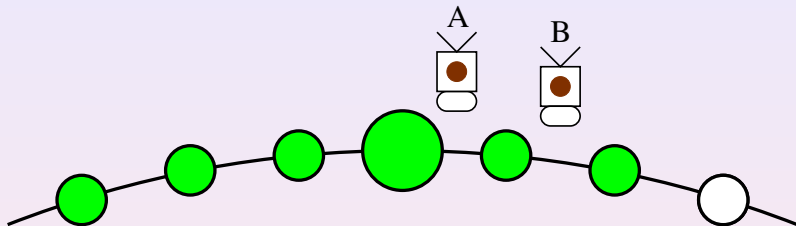


## Quick description

If the agent “controls”

- 1 pebble → **Simple** cautious walk to the **right**
- 2 pebbles → **Double** cautious walk to the **left**
- 0 pebbles → **Non-cautious** walk to the **left**

# Ping Pong algorithm

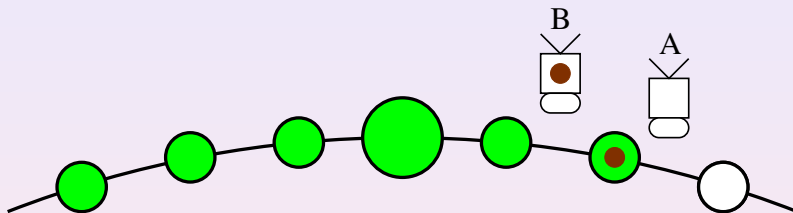


## Quick description

If the agent “controls”

- 1 pebble → Simple cautious walk to the right
- 2 pebbles → Double cautious walk to the left
- 0 pebbles → Non-cautious walk to the left

# Ping Pong algorithm



## Quick description

If the agent “controls”

- 1 pebble → Simple cautious walk to the right
- 2 pebbles → Double cautious walk to the left
- 0 pebbles → Non-cautious walk to the left

# Ping Pong algorithm

⇒ The algorithm is **correct** and uses  $O(n^2)$  moves

## Quick description

If the agent “controls”

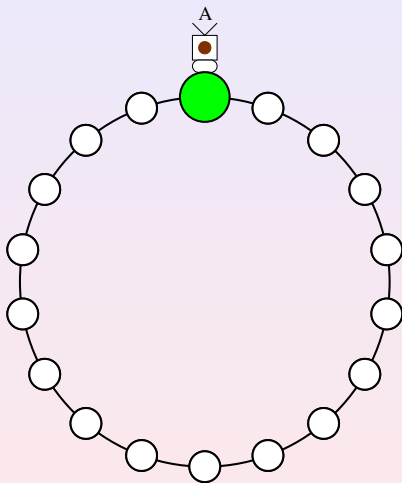
- 1 pebble → **Simple** cautious walk to the **right**
- 2 pebbles → **Double** cautious walk to the **left**
- 0 pebbles → **Non-cautious** walk to the **left**

# Enhanced Ping Pong algorithm

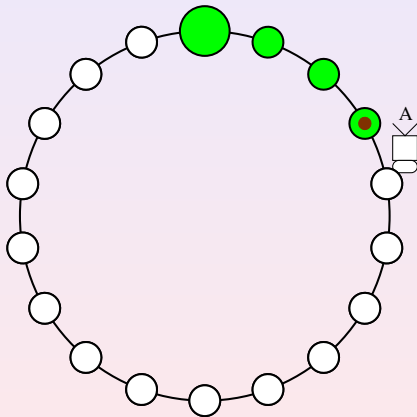
## Quick description

- **Execute Ping Pong until** agents “meet” at least two/three nodes from the homebase (**until the safe zone becomes sufficiently large**)
- **While** still an unexplored node **do**
- **Divide the unexplored part in two**
- Each agent explores an half
- The first agent to finish its part goes and helps the other agent to explore the other part
- **When** an agent finds the pebble of the other
  - **it moves the pebble by one node toward the center**
  - and restart the loop

# Example of an execution

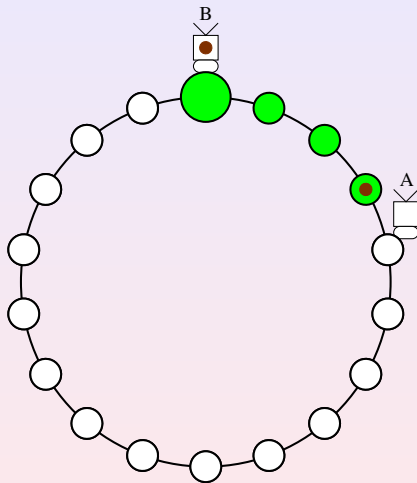


# Example of an execution

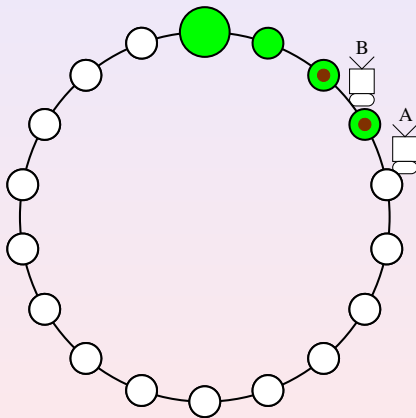




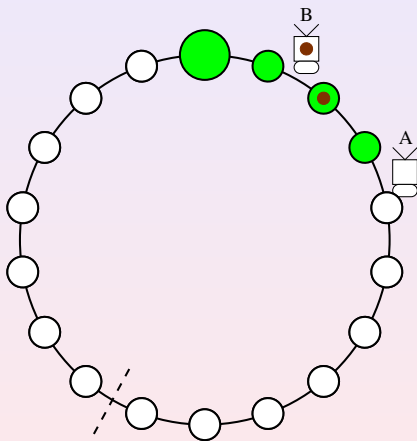
# Example of an execution



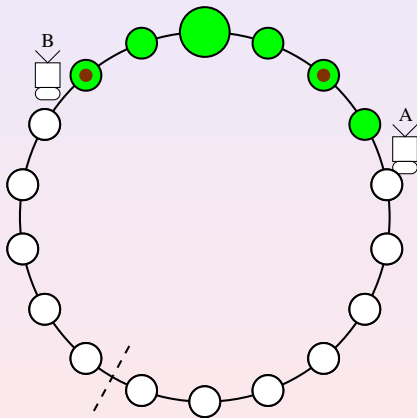
# Example of an execution



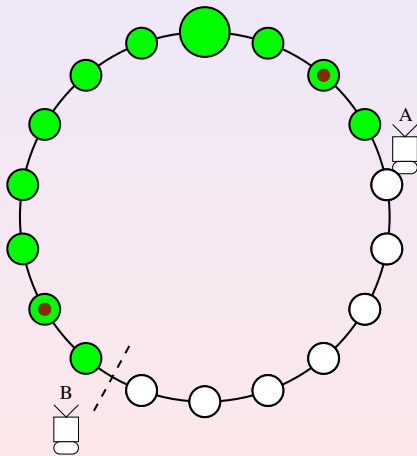
# Example of an execution



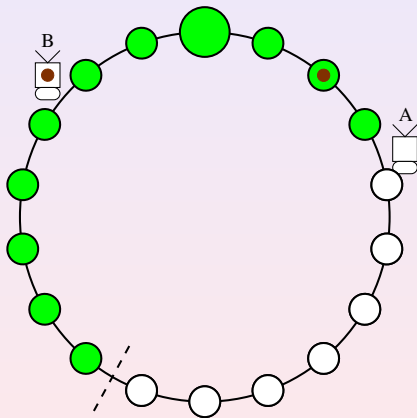
# Example of an execution



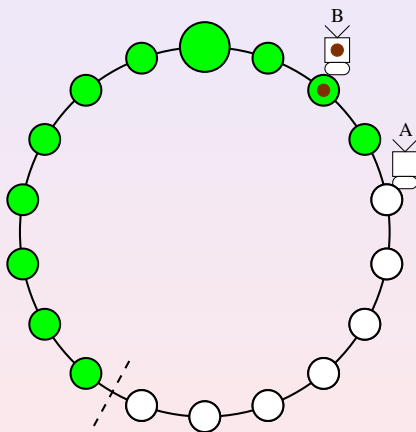
# Example of an execution



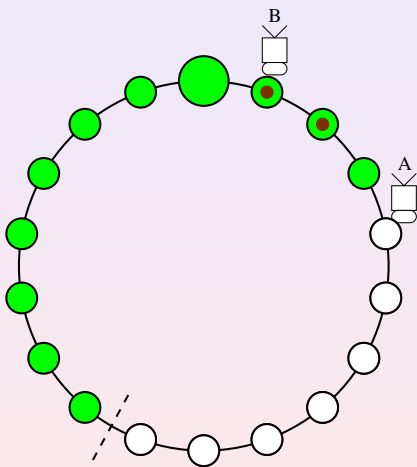
# Example of an execution



# Example of an execution

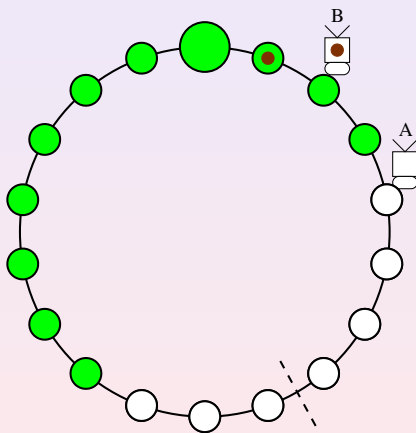


# Example of an execution

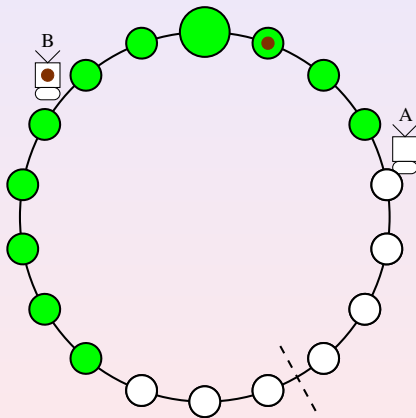




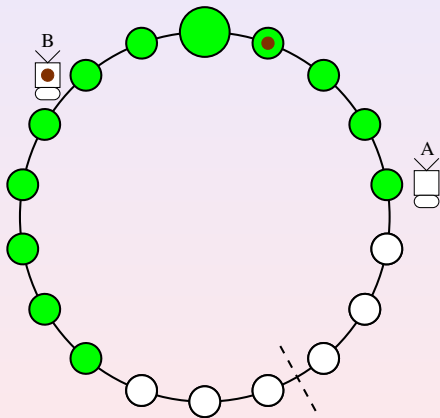
# Example of an execution



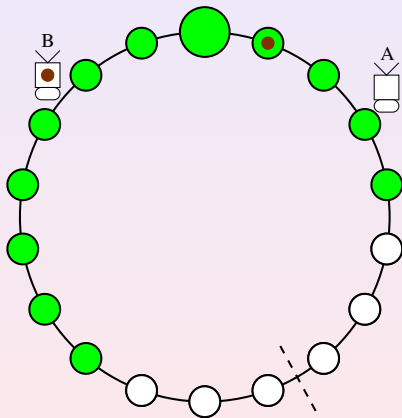
# Example of an execution



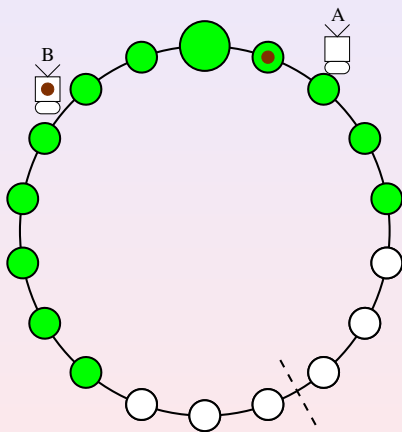
# Example of an execution



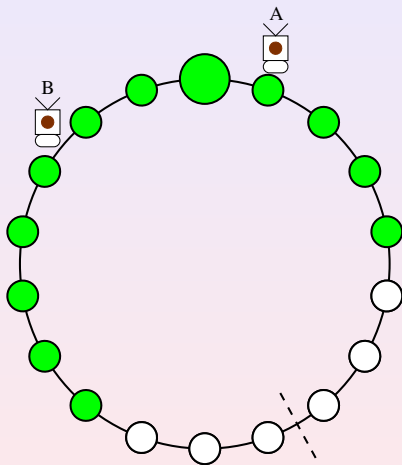
# Example of an execution



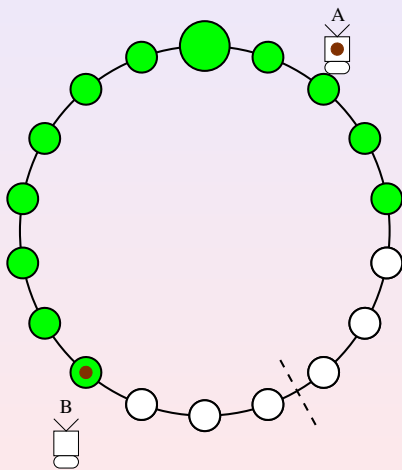
# Example of an execution



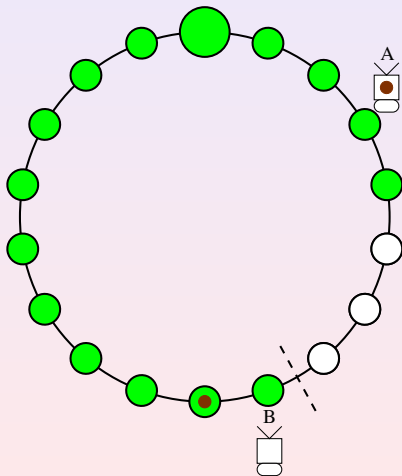
# Example of an execution



# Example of an execution

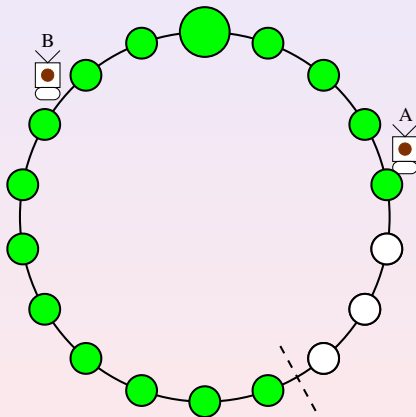


# Example of an execution

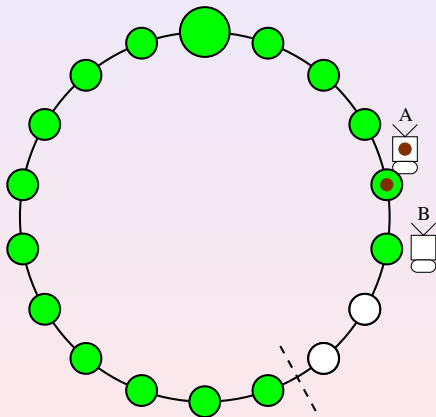




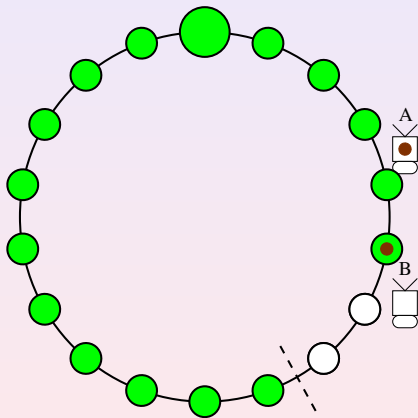
# Example of an execution



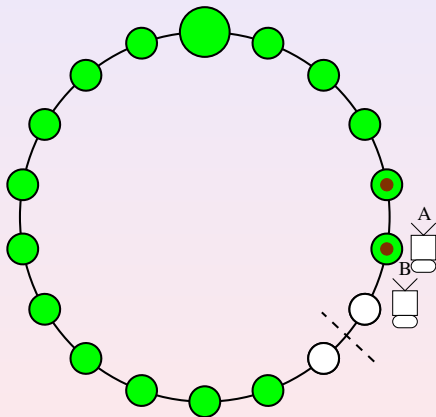
# Example of an execution



# Example of an execution



# Example of an execution



# Analysis of Enhanced Ping Pong

## Analysis

Between two halving

- Roughly **half of the nodes** are **explored**
- Agents do at most  **$O(n)$  moves**

## Result

- 1 pebble per agent
- 2 agents
- $\Theta(n \log n)$  moves

# Analysis of Enhanced Ping Pong

## Analysis

Between two halving

- Roughly **half of the nodes** are **explored**
- Agents do at most  **$O(n)$  moves**

## Result

- **1** pebble per agent
- **2** agents
- **$\Theta(n \log n)$  moves**

# Extension to arbitrary graphs

## Basic principle

Adaptation of Enhanced Ping Pong by **analogy with the ring**

- Order of exploration: DFS of spanning trees (one for the “left” direction, one for the “right” direction)
- **$i$ -th node of the ring** in one direction  $\leftrightarrow$   **$i$ -th explored node** of the graph in this direction
- **Edge  $\{i, i + 1\}$**   $\leftrightarrow$  **shortest safe path** from  $i$  to  $i + 1$

## Challenges

- One edge traversal in the ring  $\leftrightarrow$  several edge traversals in the graph
- Modification of the pair of trees at each halving

# Extension to arbitrary graphs

## Basic principle

Adaptation of Enhanced Ping Pong by **analogy with the ring**

- Order of exploration: DFS of spanning trees (one for the “left” direction, one for the “right” direction)
- **$i$ -th node of the ring** in one direction  $\leftrightarrow$   **$i$ -th explored node** of the graph in this direction
- **Edge  $\{i, i + 1\}$**   $\leftrightarrow$  **shortest safe path** from  $i$  to  $i + 1$

## Challenges

- **One** edge traversal in the ring  $\leftrightarrow$  **several** edge traversals in the graph
- Modification of the pair of trees at each halving



# Modification and analysis

## Modification

If possible, use **shortcuts**

(go from  $i$  to  $j$  without necessarily going through nodes  $i + 1, i + 2, \dots$ )

## Analysis

- Average length of a ring edge is constant
- Precise analysis of Enhanced Ping Pong: each edge is traversed at most  $O(\log n)$  times

## Result

- 1 pebble per agent
- 2 agents
- $\Theta(n \log n)$  moves

# Modification and analysis

## Modification

If possible, use **shortcuts**

(go from  $i$  to  $j$  without necessarily going through nodes  $i + 1, i + 2, \dots$ )

## Analysis

- **Average length** of a ring edge is **constant**
- Precise analysis of Enhanced Ping Pong: **each edge is traversed at most  $O(\log n)$  times**

## Result

- 1 pebble per agent
- 2 agents
- $\Theta(n \log n)$  moves

# Modification and analysis

## Modification

If possible, use **shortcuts**

(go from  $i$  to  $j$  without necessarily going through nodes  $i + 1, i + 2, \dots$ )

## Analysis

- **Average length** of a ring edge is **constant**
- Precise analysis of Enhanced Ping Pong: **each edge is traversed at most  $O(\log n)$  times**

## Result

- **1** pebble per agent
- **2** agents
- **$\Theta(n \log n)$**  moves

# Conclusion and perspectives

## Conclusion

**Equivalence**, for black hole search, between

- Coordination through whiteboards
- Coordination through pebbles

## Perspectives

**For which other mobile agents problems** is this equivalence true?