Space-optimal collaborative exploration of undirected graphs

Yann Disser Jan Hackfeld Max Klimm

Humboldt-Universität zu Berlin

23.02.2018

Berlin Mathematical School

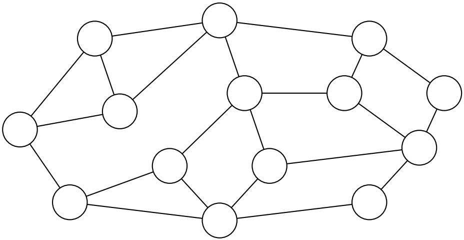
SPP 1736

Introduction and model

The octagonal Jubilee Maze at Symonds Yat - NotFromUtrecht - CC 3.0

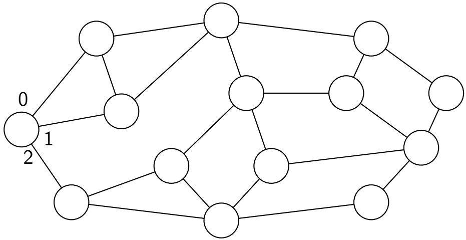
Introduction and model

- k agents in undirected, initially unknown graph



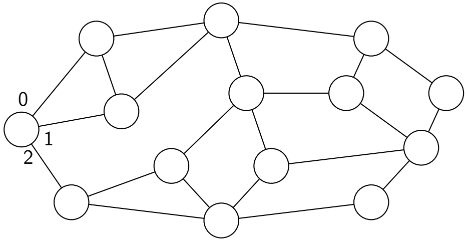
Introduction and model

- k agents in undirected, initially unknown graph
  - vertices unlabeled, edges (locally) labeled



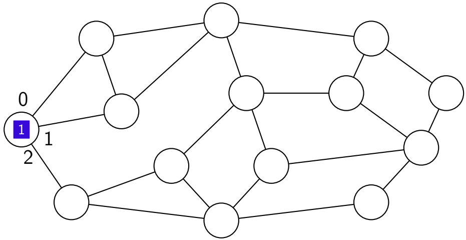
Introduction and model

- k agents in undirected, initially unknown graph
  - vertices unlabeled, edges (locally) labeled
- goal: explore graph (every edge traversed by an agent)



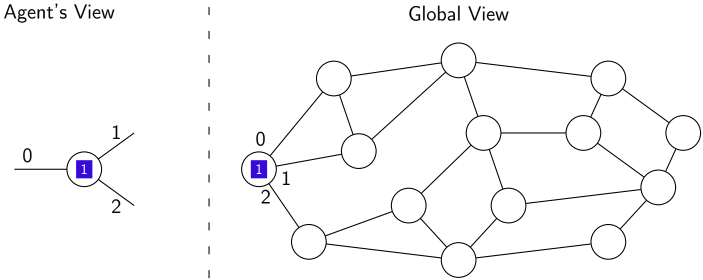
Introduction and model

- k agents in undirected, initially unknown graph
  - vertices unlabeled, edges (locally) labeled
- goal: explore graph (every edge traversed by an agent)



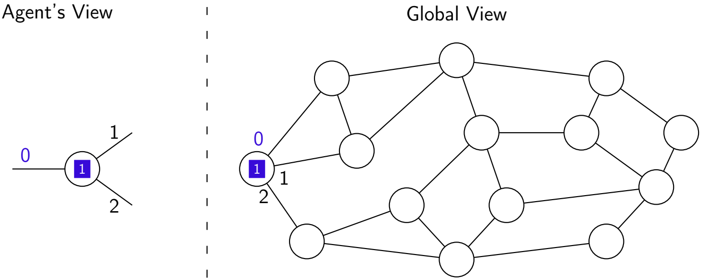
Introduction and model

- k agents in undirected, initially unknown graph
  - vertices unlabeled, edges (locally) labeled
- goal: explore graph (every edge traversed by an agent)



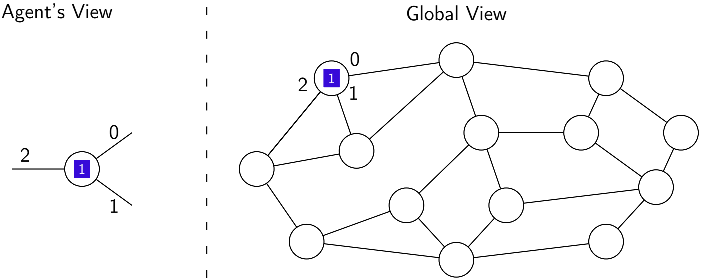
Introduction and model

- k agents in undirected, initially unknown graph
  - vertices unlabeled, edges (locally) labeled
- goal: explore graph (every edge traversed by an agent)



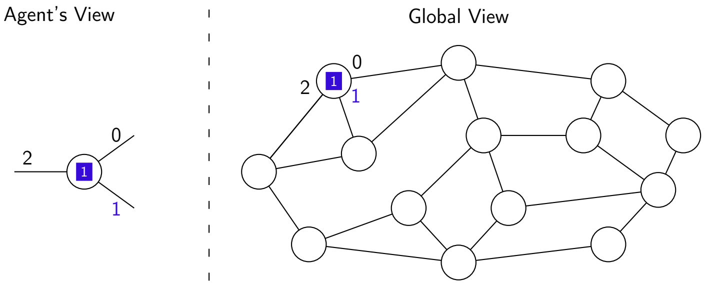
Introduction and model

- k agents in undirected, initially unknown graph
  - vertices unlabeled, edges (locally) labeled
- goal: explore graph (every edge traversed by an agent)



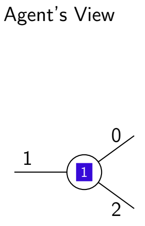
Introduction and model

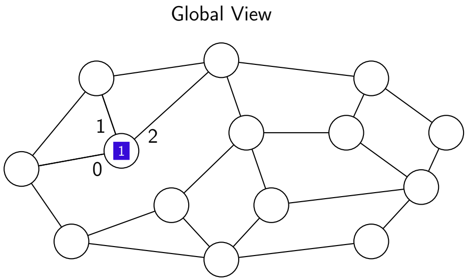
- k agents in undirected, initially unknown graph
  - vertices unlabeled, edges (locally) labeled
- goal: explore graph (every edge traversed by an agent)



Introduction and model

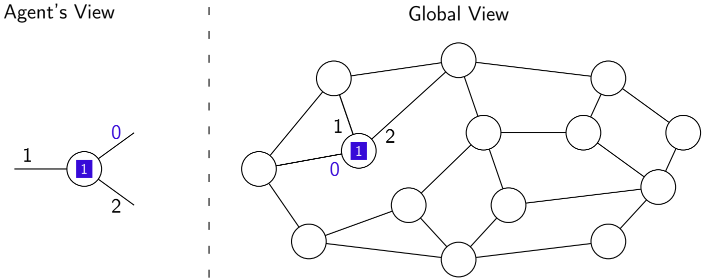
- k agents in undirected, initially unknown graph
  - vertices unlabeled, edges (locally) labeled
- goal: explore graph (every edge traversed by an agent)





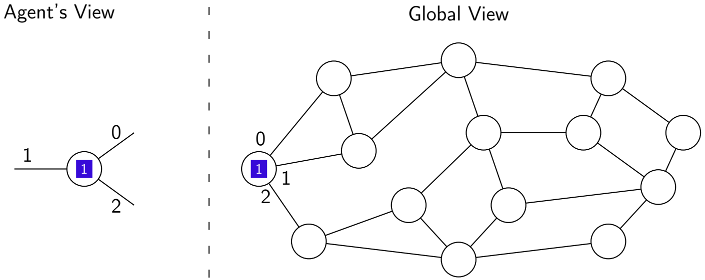
Introduction and model

- k agents in undirected, initially unknown graph
  - vertices unlabeled, edges (locally) labeled
- goal: explore graph (every edge traversed by an agent)



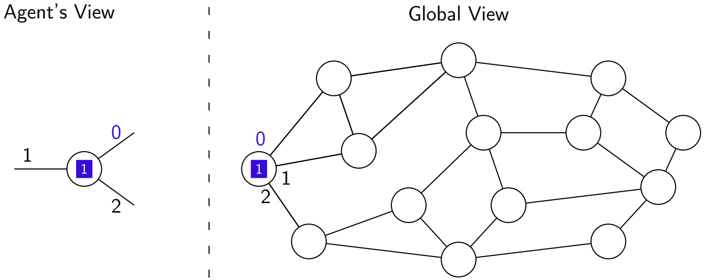
Introduction and model

- k agents in undirected, initially unknown graph
  - vertices unlabeled, edges (locally) labeled
- goal: explore graph (every edge traversed by an agent)



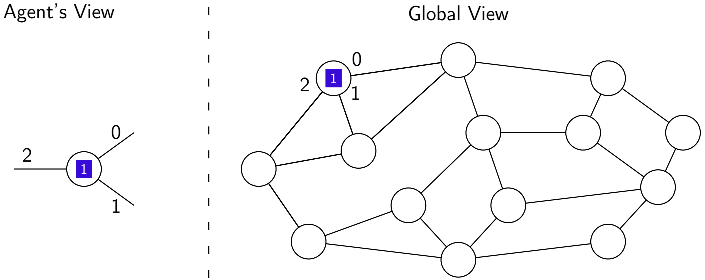
Introduction and model

- k agents in undirected, initially unknown graph
  - vertices unlabeled, edges (locally) labeled
- goal: explore graph (every edge traversed by an agent)



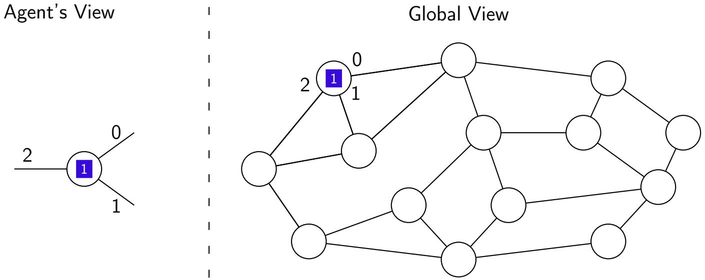
Introduction and model

- k agents in undirected, initially unknown graph
  - vertices unlabeled, edges (locally) labeled
- goal: explore graph (every edge traversed by an agent)



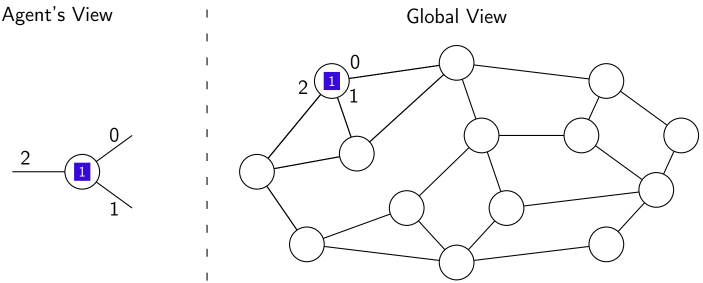
Introduction and model

- k agents in undirected, initially unknown graph
  - vertices unlabeled, edges (locally) labeled
- goal: explore graph (every edge traversed by an agent) minimizing
  - memory requirement
  - number of agents



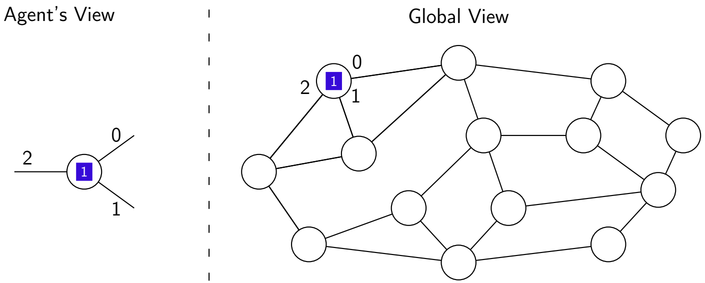
Introduction and model

- k agents in undirected, initially unknown graph
  - vertices unlabeled, edges (locally) labeled
- goal: explore graph (every edge traversed by an agent) minimizing
  - memory requirement
  - number of agents
- every transition of agent depends on:



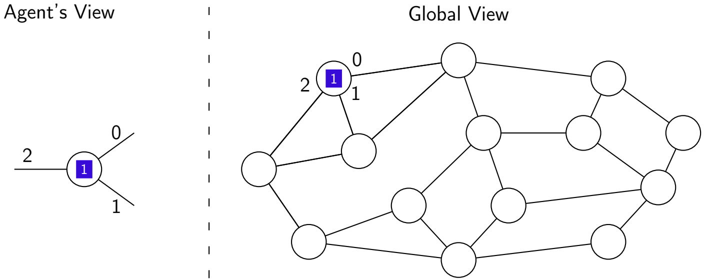
Introduction and model

- k agents in undirected, initially unknown graph
  - vertices unlabeled, edges (locally) labeled
- goal: explore graph (every edge traversed by an agent) minimizing
  - memory requirement
  - number of agents
- every transition of agent depends on:
  - label of incoming edge used last



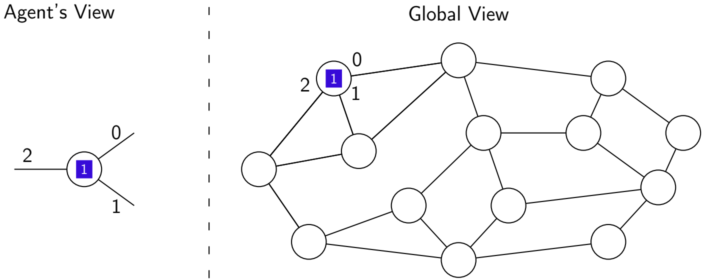
Introduction and model

- k agents in undirected, initially unknown graph
  - vertices unlabeled, edges (locally) labeled
- goal: explore graph (every edge traversed by an agent) minimizing
  - memory requirement
  - number of agents
- every transition of agent depends on:
  - label of incoming edge used last
  - degree of current vertex



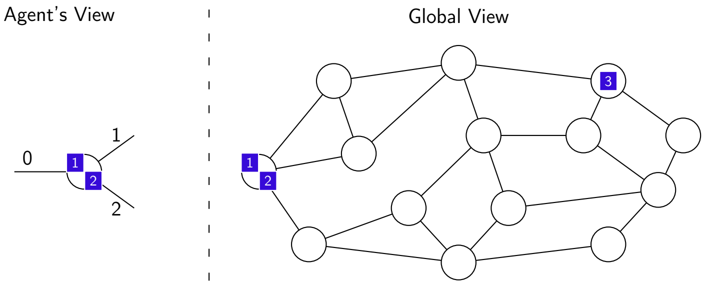
Introduction and model

- k agents in undirected, initially unknown graph
  - vertices unlabeled, edges (locally) labeled
- goal: explore graph (every edge traversed by an agent) minimizing
  - memory requirement
  - number of agents
- every transition of agent depends on:
  - label of incoming edge used last
  - degree of current vertex
  - state of the agent and of other agents at current location



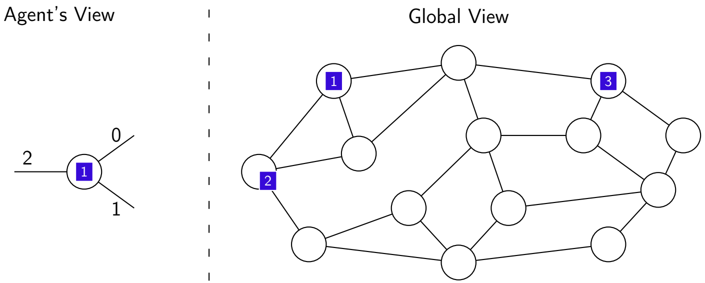
Introduction and model

- k agents in undirected, initially unknown graph
  - vertices unlabeled, edges (locally) labeled
- goal: explore graph (every edge traversed by an agent) minimizing
  - memory requirement
  - number of agents
- every transition of agent depends on:
  - label of incoming edge used last
  - degree of current vertex
  - state of the agent and of other agents at current location



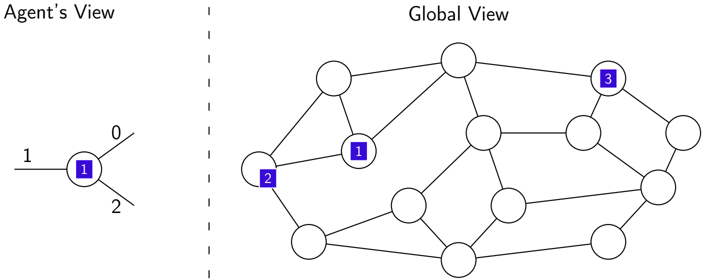
Introduction and model

- k agents in undirected, initially unknown graph
  - vertices unlabeled, edges (locally) labeled
- goal: explore graph (every edge traversed by an agent) minimizing
  - memory requirement
  - number of agents
- every transition of agent depends on:
  - label of incoming edge used last
  - degree of current vertex
  - state of the agent and of other agents at current location



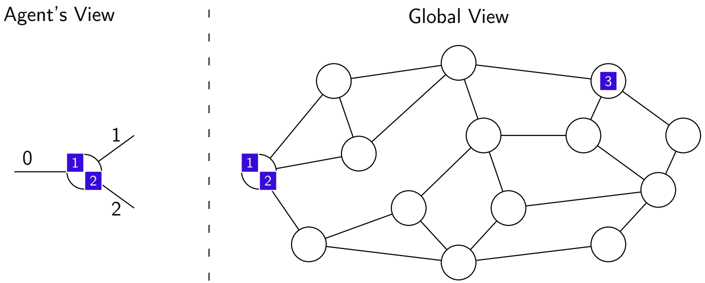
Introduction and model

- k agents in undirected, initially unknown graph
  - vertices unlabeled, edges (locally) labeled
- goal: explore graph (every edge traversed by an agent) minimizing
  - memory requirement
  - number of agents
- every transition of agent depends on:
  - label of incoming edge used last
  - degree of current vertex
  - state of the agent and of other agents at current location



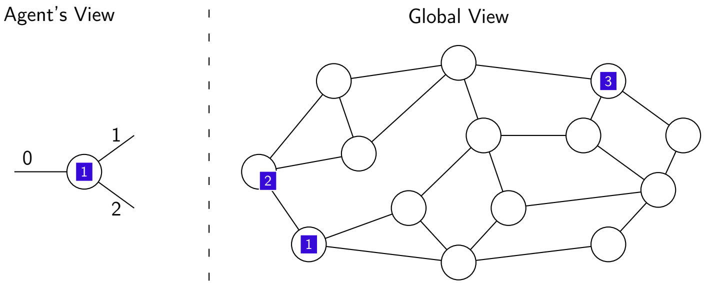
Introduction and model

- k agents in undirected, initially unknown graph
  - vertices unlabeled, edges (locally) labeled
- goal: explore graph (every edge traversed by an agent) minimizing
  - memory requirement
  - number of agents
- every transition of agent depends on:
  - label of incoming edge used last
  - degree of current vertex
  - state of the agent and of other agents at current location



Introduction and model

- k agents in undirected, initially unknown graph
  - vertices unlabeled, edges (locally) labeled
- goal: explore graph (every edge traversed by an agent) minimizing
  - memory requirement
  - number of agents
- every transition of agent depends on:
  - label of incoming edge used last
  - degree of current vertex
  - state of the agent and of other agents at current location



Known and new results

n = number of vertices of the undirected graph

Known and new results

n = number of vertices of the undirected graph

$k = 1$ agent:

Known and new results

n = number of vertices of the undirected graph

$k = 1$ agent:

- $\Omega(\log n)$ bits necessary [1]

[1] Fraigniaud, Ilcinkas, Peer, Pelc, Peleg, *TCS*, 2005.

Known and new results

n = number of vertices of the undirected graph

$k = 1$ agent:

- $\Omega(\log n)$ bits necessary [1]
- $\mathcal{O}(\log n)$ bits sufficient [2]

[1] Fraigniaud, Ilcinkas, Peer, Pelc, Peleg, *TCS*, 2005.
[2] Reingold, JACM, 2008.

Known and new results

n = number of vertices of the undirected graph

$k = 1$ agent:

- $\Omega(\log n)$ bits necessary [1]
- $\mathcal{O}(\log n)$ bits sufficient [2]

$k > 1$ agents:

[1] Fraigniaud, Ilcinkas, Peer, Pelc, Peleg, *TCS*, 2005.
[2] Reingold, JACM, 2008.

Known and new results

n = number of vertices of the undirected graph

$k = 1$ agent:

- $\Omega(\log n)$ bits necessary [1]
- $\mathcal{O}(\log n)$ bits sufficient [2]

$k > 1$ agents:

- $\Omega(\log^*_{(2^b)} n)$ agents with b bits each necessary [3]

[1] Fraigniaud, Ilcinkas, Peer, Pelc, Peleg, *TCS*, 2005.
[2] Reingold, JACM, 2008.
[3] Rollik, Acta Informatica, 1980.

Known and new results

n = number of vertices of the undirected graph

$k = 1$ agent:

- $\Omega(\log n)$ bits necessary [1]
- $\mathcal{O}(\log n)$ bits sufficient [2]

$k > 1$ agents:

- $\Omega(\log^*_{(2^b)} n)$ agents with b bits each necessary [3]
- $\Omega(\log \log n)$ agents necessary [4]

[1] Fraigniaud, Ilcinkas, Peer, Pelc, Peleg, *TCS*, 2005.
[2] Reingold, JACM, 2008.
[3] Rollik, Acta Informatica, 1980.
[4] Disser, H., Klimm, in preperation.

Known and new results

n = number of vertices of the undirected graph

$k = 1$ agent:

- $\Omega(\log n)$ bits necessary [1]
- $\mathcal{O}(\log n)$ bits sufficient [2]

$k > 1$ agents:

- $\Omega(\log^*_{(2^b)} n)$ agents with b bits each necessary [3]
- $\Omega(\log \log n)$ agents necessary [4]
- $\mathcal{O}(\log \log n)$ agents with $O(1)$ bits sufficient [4]

[1] Fraigniaud, Ilcinkas, Peer, Pelc, Peleg, *TCS*, 2005.
[2] Reingold, JACM, 2008.
[3] Rollik, Acta Informatica, 1980.
[4] Disser, H., Klimm, in preperation.

Upper bound

Exploration algorithm - obstacles

Theorem (Disser, H., Klimm)

$\mathcal{O}(\log \log n)$ agents with $\mathcal{O}(1)$ memory each can explore any graph on at most n vertices in polynomial time.

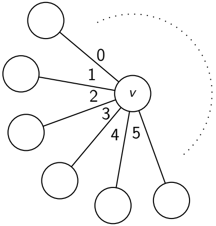
Exploration algorithm - obstacles

Theorem (Disser, H., Klimm)

$\mathcal{O}(\log\log n)$ agents with $\mathcal{O}(1)$ memory each can explore any graph on at most n vertices in polynomial time.

Issues:

- high degree vertices with constant memory?



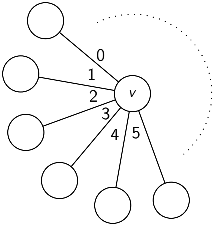
Exploration algorithm - obstacles

Theorem (Disser, H., Klimm)

$\mathcal{O}(\log\log n)$ *agents with* $\mathcal{O}(1)$ *memory each can explore any graph on at most* n *vertices in polynomial time.*

Issues:

- high degree vertices with constant memory?
  - transitions can depend on **last edge used**



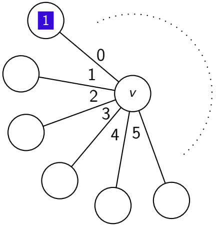
Exploration algorithm - obstacles

Theorem (Disser, H., Klimm)

$\mathcal{O}(\log\log n)$ *agents with* $\mathcal{O}(1)$ *memory each can explore any graph on at most* n *vertices in polynomial time.*

Issues:

- high degree vertices with constant memory?
  - transitions can depend on **last edge used**



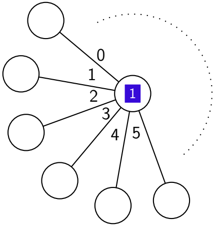
Exploration algorithm - obstacles

Theorem (Disser, H., Klimm)

$\mathcal{O}(\log\log n)$ *agents with* $\mathcal{O}(1)$ *memory each can explore any graph on at most* n *vertices in polynomial time.*

Issues:

- high degree vertices with constant memory?
  - transitions can depend on **last edge used**



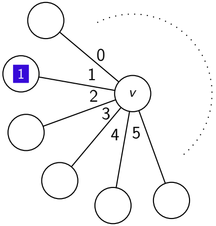
Exploration algorithm - obstacles

Theorem (Disser, H., Klimm)

$\mathcal{O}(\log \log n)$ *agents with* $\mathcal{O}(1)$ *memory each can explore any graph on at most* n *vertices in polynomial time.*

Issues:

- high degree vertices with constant memory?
  - transitions can depend on **last edge used**



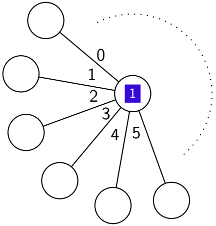
Exploration algorithm - obstacles

Theorem (Disser, H., Klimm)

$\mathcal{O}(\log\log n)$ *agents with* $\mathcal{O}(1)$ *memory each can explore any graph on at most* n *vertices in polynomial time.*

Issues:

- high degree vertices with constant memory?
  - transitions can depend on **last edge used**



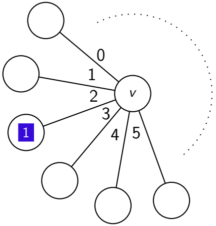
Exploration algorithm - obstacles

Theorem (Disser, H., Klimm)

$\mathcal{O}(\log\log n)$ agents with $\mathcal{O}(1)$ memory each can explore any graph on at most n vertices in polynomial time.

Issues:

- high degree vertices with constant memory?
  - transitions can depend on **last edge used**



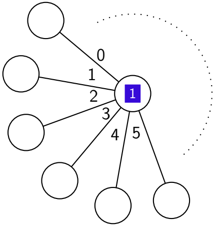
Exploration algorithm - obstacles

Theorem (Disser, H., Klimm)

$\mathcal{O}(\log\log n)$ *agents with* $\mathcal{O}(1)$ *memory each can explore any graph on at most* n *vertices in polynomial time.*

Issues:

- high degree vertices with constant memory?
  - transitions can depend on **last edge used**



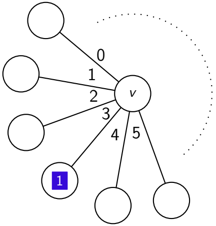
Exploration algorithm - obstacles

Theorem (Disser, H., Klimm)

$\mathcal{O}(\log\log n)$ agents with $\mathcal{O}(1)$ memory each can explore any graph on at most n vertices in polynomial time.

Issues:

- high degree vertices with constant memory?
  - transitions can depend on **last edge used**



Exploration algorithm - obstacles

Theorem (Disser, H., Klimm)

$\mathcal{O}(\log\log n)$ *agents with* $\mathcal{O}(1)$ *memory each can explore any graph on at most* n *vertices in polynomial time.*

Issues:

- high degree vertices with constant memory?
  - transitions can depend on **last edge used**
- only constant number of actions?

Exploration algorithm - obstacles

Theorem (Disser, H., Klimm)

$\mathcal{O}(\log\log n)$ *agents with* $\mathcal{O}(1)$ *memory each can explore any graph on at most* n *vertices in polynomial time.*

Issues:

- high degree vertices with constant memory?
  - transitions can depend on **last edge used**
- only constant number of actions?
  - transitions can depend on **states of other agents** at same vertex

Exploration algorithm - obstacles

Theorem (Disser, H., Klimm)

$\mathcal{O}(\log\log n)$ *agents with* $\mathcal{O}(1)$ *memory each can explore any graph on at most* n *vertices in polynomial time.*

Issues:

- high degree vertices with constant memory?
  - transitions can depend on **last edge used**
- only constant number of actions?
  - transitions can depend on **states of other agents** at same vertex

    $\Rightarrow$ k agents with $\mathcal{O}(1)$ bits of memory at least as powerful as 1 agent with $\mathcal{O}(k)$ bits of memory

Exploration algorithm - obstacles

Theorem (Disser, H., Klimm)

$\mathcal{O}(\log\log n)$ agents with $\mathcal{O}(1)$ memory each can explore any graph on at most n vertices in polynomial time.

Issues:

- high degree vertices with constant memory?
  - transitions can depend on **last edge used**
- only constant number of actions?
  - transitions can depend on **states of other agents** at same vertex
    $\Rightarrow$ k agents with $\mathcal{O}(1)$ bits of memory at least as powerful as
    1 agent with $\mathcal{O}(k)$ bits of memory

$\Theta(\log\log n)$ agents

1 2 3 4 5 ....

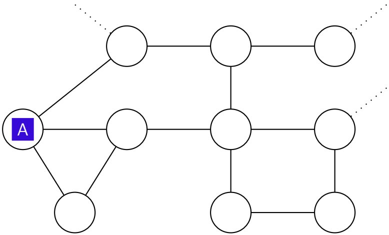
half of the agents move together

use half of the agents as markers

1 agent with $\Theta(\log\log n)$ bits

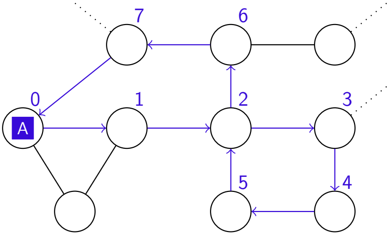
$\Theta(\log\log n)$ pebbles

Exploration algorithm - main idea



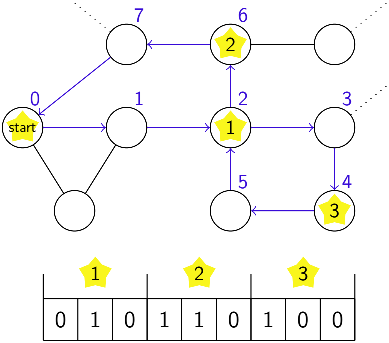
Exploration algorithm - main idea

- assume: agent A can find closed walk ω



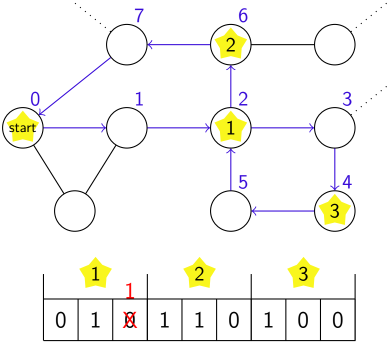
Exploration algorithm - main idea

- assume: agent A can find closed walk ω
- positions of pebbles 1 (marker agents) along ω encode memory



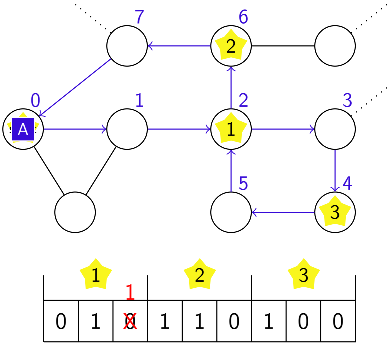
Exploration algorithm - main idea

- assume: agent A can find closed walk ω
- positions of pebbles 1 (marker agents) along ω encode memory



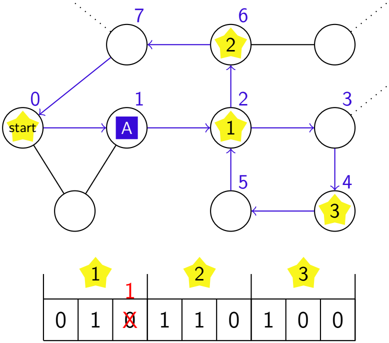
Exploration algorithm - main idea

- assume: agent A can find closed walk ω
- positions of pebbles 1 (marker agents) along ω encode memory



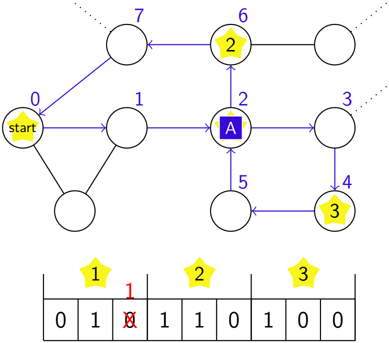
Exploration algorithm - main idea

- assume: agent A can find closed walk ω
- positions of pebbles 1 (marker agents) along ω encode memory



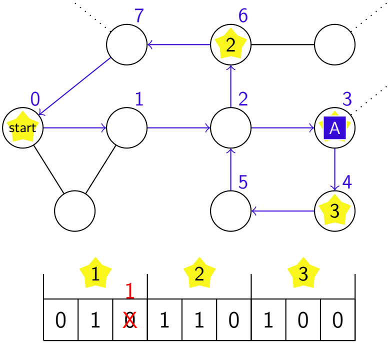
Exploration algorithm - main idea

- assume: agent A can find closed walk ω
- positions of pebbles 1 (marker agents) along ω encode memory



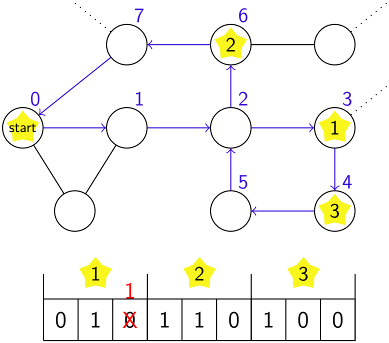
Exploration algorithm - main idea

- assume: agent A can find closed walk ω
- positions of pebbles 1 (marker agents) along ω encode memory



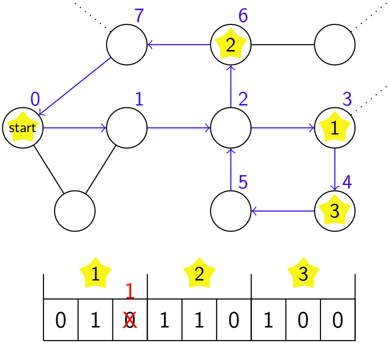
Exploration algorithm - main idea

- assume: agent A can find closed walk ω
- positions of pebbles 1 (marker agents) along ω encode memory



Exploration algorithm - main idea

- assume: agent A can find closed walk ω
- positions of pebbles 1 (marker agents) along ω encode memory

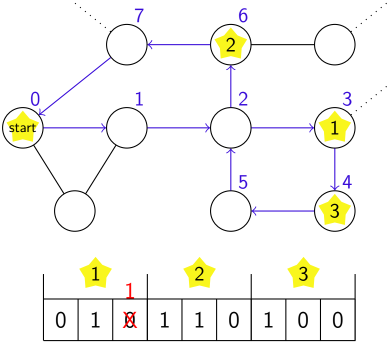


Challenges:

- vertices appearing multiple times along ω

Exploration algorithm - main idea

- assume: agent A can find closed walk ω
- positions of pebbles 1 (marker agents) along ω encode memory



Challenges:

- vertices appearing multiple times along ω
- carrying the memory along while agent traverses graph

Exploration algorithm

Theorem (Reingold, JACM '08)

Agent with $\mathcal{O}(\log n)$ bits can explore any graph with n vertices.

Exploration algorithm

Theorem (Reingold, JACM '08)

Agent with $\mathcal{O}(\log n)$ bits can explore any graph with n vertices.

Lemma (Disser, H., Klimm)

Agent with $\mathcal{O}(\log a)$ bits of memory can

- *move on closed walk*
- *visit at least $\min\{a, n\}$ vertices in any graph*

Exploration algorithm

Lemma (Disser, H., Klimm)

Agent with $\mathcal{O}(\log a)$ bits of memory can

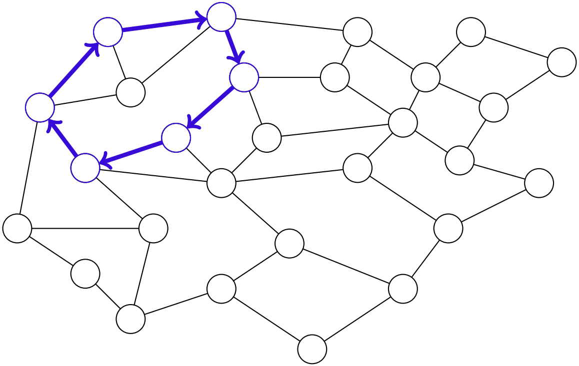
- *move on closed walk*
- *visit at least* $\min\{a, n\}$ *vertices in any graph*

Exploration algorithm

Lemma (Disser, H., Klimm)

Agent with $\mathcal{O}(\log a)$ bits of memory can

- *move on closed walk*
- *visit at least* $\min\{a, n\}$ *vertices in any graph*

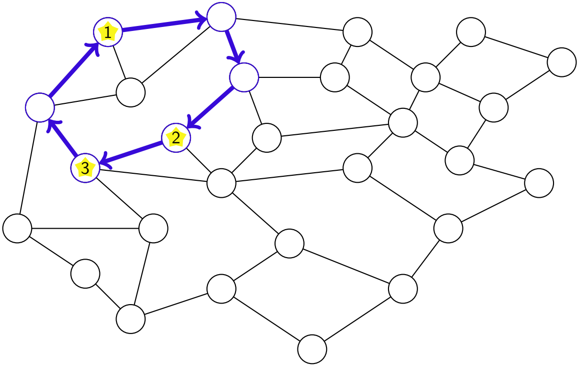


Exploration algorithm

Lemma (Disser, H., Klimm)

Agent with $\mathcal{O}(\log a)$ *bits of memory can*

- *move on closed walk*
- *visit at least* $\min\{a, n\}$ *vertices in any graph*

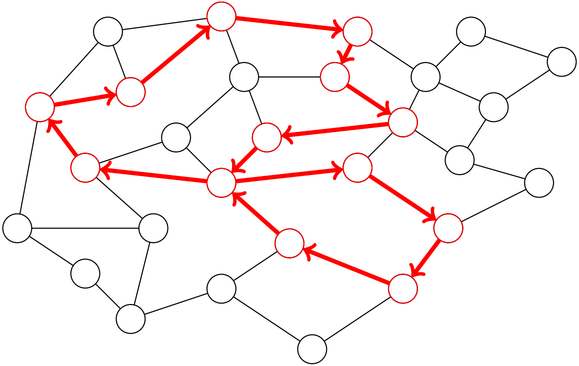


Exploration algorithm

Lemma (Disser, H., Klimm)

Agent with $\mathcal{O}(\log a)$ bits of memory can

- *move on closed walk*
- *visit at least* $\min\{a, n\}$ *vertices in any graph*

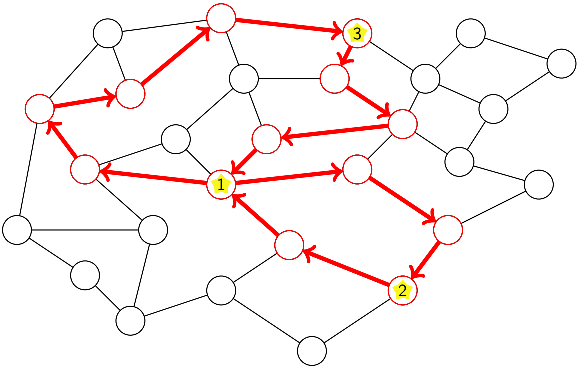


Exploration algorithm

Lemma (Disser, H., Klimm)

Agent with $\mathcal{O}(\log a)$ bits of memory can

- *move on closed walk*
- *visit at least* $\min\{a, n\}$ *vertices in any graph*

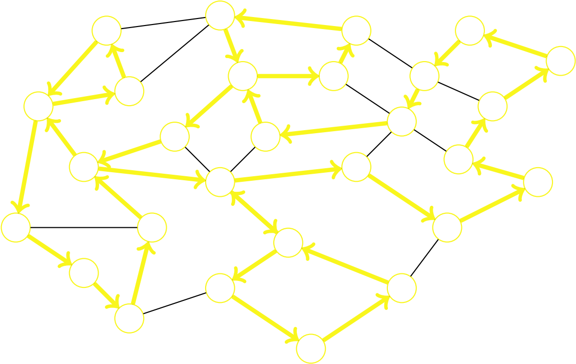


Exploration algorithm

Lemma (Disser, H., Klimm)

Agent with $\mathcal{O}(\log a)$ bits of memory can

- *move on closed walk*
- *visit at least* $\min\{a, n\}$ *vertices in any graph*



Exploration algorithm

Lemma (Disser, H., Klimm)

Agent with $\mathcal{O}(\log a)$ *bits of memory can*

- *move on closed walk*
- *visit at least* $\min\{a, n\}$ *vertices in any graph*

Algorithm:

memory = constant # bits

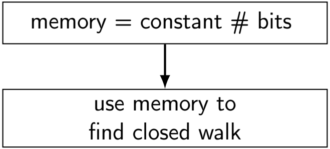
Exploration algorithm

Lemma (Disser, H., Klimm)

Agent with $\mathcal{O}(\log a)$ bits of memory can

- *move on closed walk*
- *visit at least $\min\{a, n\}$ vertices in any graph*

Algorithm:



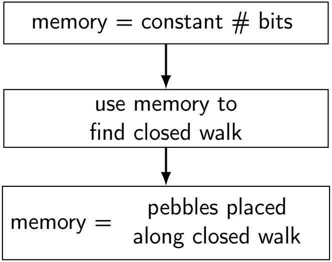
Exploration algorithm

Lemma (Disser, H., Klimm)

Agent with $\mathcal{O}(\log a)$ bits of memory can

- *move on closed walk*
- *visit at least* $\min\{a, n\}$ *vertices in any graph*

Algorithm:



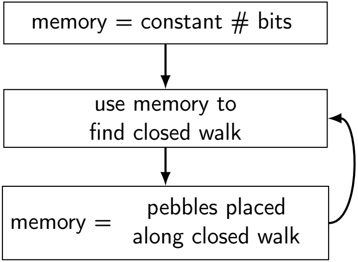
Exploration algorithm

Lemma (Disser, H., Klimm)

Agent with $\mathcal{O}(\log a)$ bits of memory can

- *move on closed walk*
- *visit at least $\min\{a, n\}$ vertices in any graph*

Algorithm:



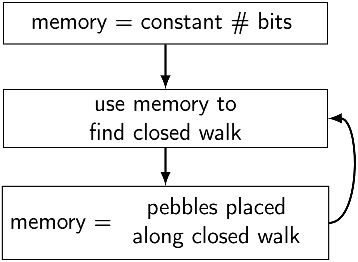
Exploration algorithm

Lemma (Disser, H., Klimm)

Agent with $\mathcal{O}(\log a)$ bits of memory can

- *move on closed walk*
- *visit at least $\min\{a, n\}$ vertices in any graph*

Algorithm:



Key properties:

1. memory doubles in each step

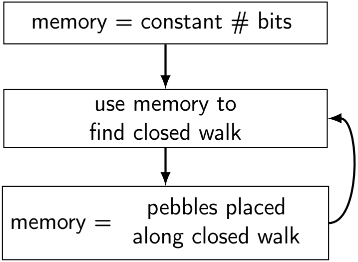
Exploration algorithm

Lemma (Disser, H., Klimm)

Agent with $\mathcal{O}(\log a)$ bits of memory can

- *move on closed walk*
- *visit at least* $\min\{a, n\}$ *vertices in any graph*

Algorithm:



Key properties:

1. memory doubles in each step
   $\Rightarrow$ starting with c bits of memory yields $c \cdot 2^{\log \log n} = c \log n$ bits after $\log \log n$ steps

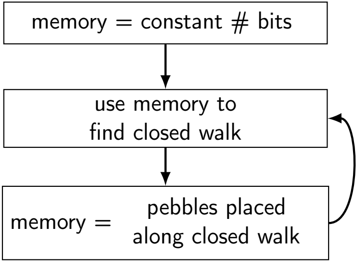
Exploration algorithm

Lemma (Disser, H., Klimm)

Agent with $\mathcal{O}(\log a)$ bits of memory can

- *move on closed walk*
- *visit at least* $\min\{a, n\}$ *vertices in any graph*

Algorithm:



Key properties:

1. memory doubles in each step
   $\Rightarrow$ starting with c bits of memory
   yields $c \cdot 2^{\log \log n} = c \log n$ bits
   after $\log \log n$ steps
   $\Rightarrow$ $\log \log n$ steps sufficient
   for exploring graph

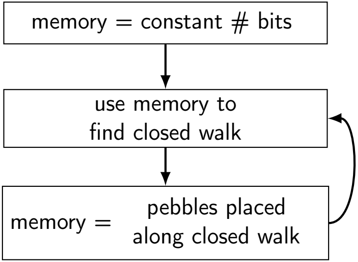
Exploration algorithm

Lemma (Disser, H., Klimm)

Agent with $\mathcal{O}(\log a)$ bits of memory can

- *move on closed walk*
- *visit at least* $\min\{a, n\}$ *vertices in any graph*

Algorithm:



Key properties:

1. memory doubles in each step
   $\Rightarrow$ starting with c bits of memory yields $c \cdot 2^{\log \log n} = c \log n$ bits after $\log \log n$ steps
   $\Rightarrow$ $\log \log n$ steps sufficient for exploring graph
2. each step needs constant # pebbles

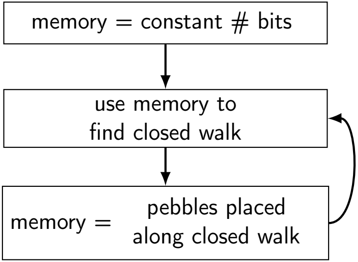
Exploration algorithm

Lemma (Disser, H., Klimm)

Agent with $\mathcal{O}(\log a)$ bits of memory can

- *move on closed walk*
- *visit at least $\min\{a, n\}$ vertices in any graph*

Algorithm:



Key properties:

1. memory doubles in each step
   $\Rightarrow$ starting with c bits of memory
   yields $c \cdot 2^{\log\log n} = c \log n$ bits
   after $\log\log n$ steps
   $\Rightarrow$ $\log\log n$ steps sufficient
   for exploring graph
2. each step needs constant # pebbles
   $\Rightarrow$ $\mathcal{O}(\log\log n)$ pebbles needed

Lower bound

Lower bound - building block

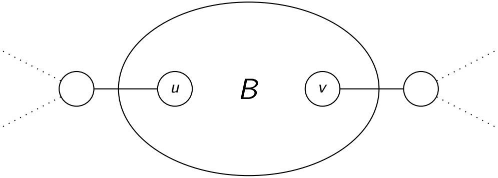
$\mathcal{A} =$ set of k agents with s states each

Lower bound - building block

$\mathcal{A}$ = set of k agents with s states each

Definition (Barrier)

B is ***r*-barrier** for $\mathcal{A} \Leftrightarrow$ no subset of $\mathcal{A}$ of at most r agents can traverse B from u to v (or vice versa)

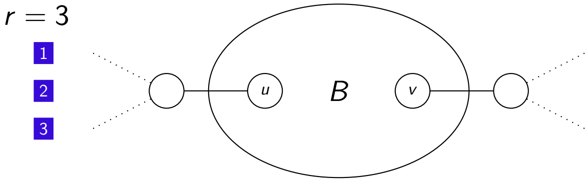


Lower bound - building block

$\mathcal{A}$ = set of k agents with s states each

Definition (Barrier)

B is ***r*-barrier** for $\mathcal{A} \Leftrightarrow$ no subset of $\mathcal{A}$ of at most r agents can traverse B from u to v (or vice versa)

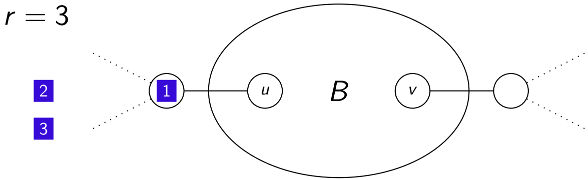


Lower bound - building block

$\mathcal{A}$ = set of k agents with s states each

Definition (Barrier)

B is ***r*-barrier** for $\mathcal{A} \Leftrightarrow$ no subset of $\mathcal{A}$ of at most r agents can traverse B from u to v (or vice versa)



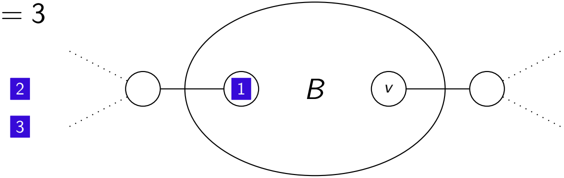
Lower bound - building block

$\mathcal{A}$ = set of k agents with s states each

Definition (Barrier)

B is ***r*-barrier** for $\mathcal{A}$ $\Leftrightarrow$ no subset of $\mathcal{A}$ of at most r agents can traverse B from u to v (or vice versa)

$r = 3$



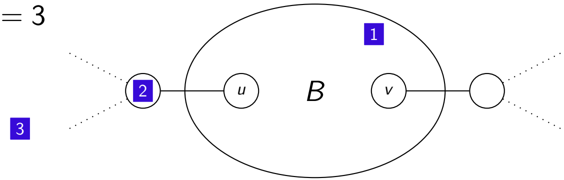
Lower bound - building block

$\mathcal{A}$ = set of k agents with s states each

Definition (Barrier)

B is ***r*-barrier** for $\mathcal{A}$ $\Leftrightarrow$ no subset of $\mathcal{A}$ of at most r agents can traverse B from u to v (or vice versa)

$r = 3$



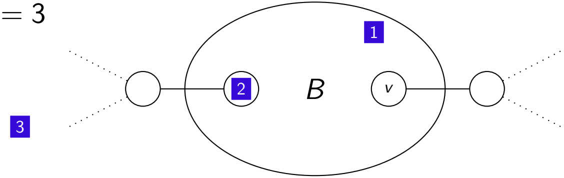
Lower bound - building block

$\mathcal{A}$ = set of k agents with s states each

Definition (Barrier)

B is ***r*-barrier** for $\mathcal{A} \Leftrightarrow$ no subset of $\mathcal{A}$ of at most r agents can traverse B from u to v (or vice versa)

$r = 3$



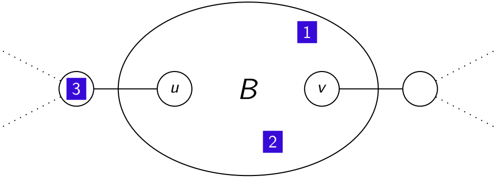
Lower bound - building block

$\mathcal{A}$ = set of k agents with s states each

Definition (Barrier)

B is **r-barrier** for $\mathcal{A}$ $\Leftrightarrow$ no subset of $\mathcal{A}$ of at most r agents can traverse B from u to v (or vice versa)

$r = 3$



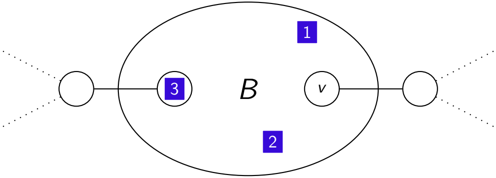
Lower bound - building block

$\mathcal{A}$ = set of k agents with s states each

Definition (Barrier)

B is ***r*-barrier** for $\mathcal{A} \Leftrightarrow$ no subset of $\mathcal{A}$ of at most r agents can traverse B from u to v (or vice versa)

$r = 3$



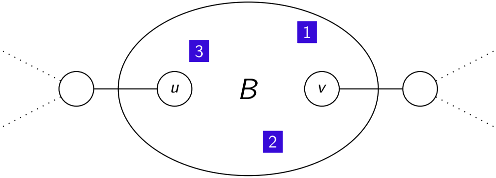
Lower bound - building block

$\mathcal{A}$ = set of k agents with s states each

Definition (Barrier)

B is ***r*-barrier** for $\mathcal{A} \Leftrightarrow$ no subset of $\mathcal{A}$ of at most r agents can traverse B from u to v (or vice versa)

$r = 3$



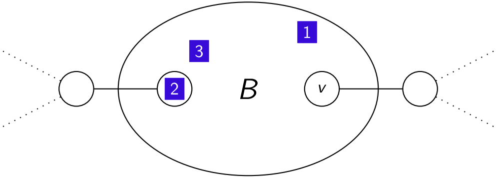
Lower bound - building block

$\mathcal{A}$ = set of k agents with s states each

Definition (Barrier)

B is ***r*-barrier** for $\mathcal{A} \Leftrightarrow$ no subset of $\mathcal{A}$ of at most r agents can traverse B from u to v (or vice versa)

$r = 3$



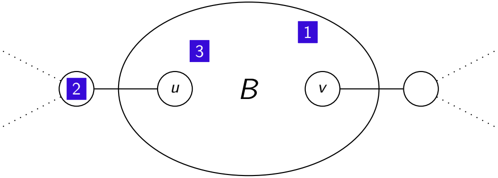
Lower bound - building block

$\mathcal{A}$ = set of k agents with s states each

Definition (Barrier)

B is ***r*-barrier** for $\mathcal{A} \Leftrightarrow$ no subset of $\mathcal{A}$ of at most r agents can traverse B from u to v (or vice versa)

$r = 3$



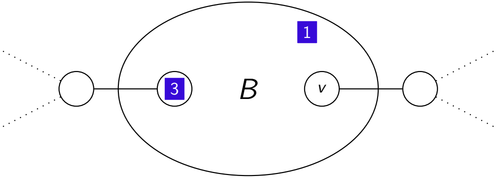
Lower bound - building block

$\mathcal{A}$ = set of k agents with s states each

Definition (Barrier)

B is ***r*-barrier** for $\mathcal{A} \Leftrightarrow$ no subset of $\mathcal{A}$ of at most r agents can traverse B from u to v (or vice versa)

$r = 3$



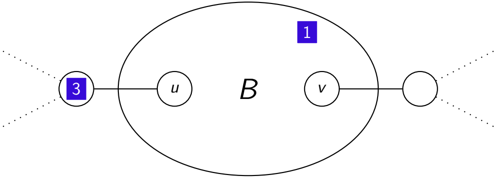
Lower bound - building block

$\mathcal{A}$ = set of k agents with s states each

Definition (Barrier)

B is ***r*-barrier** for $\mathcal{A} \Leftrightarrow$ no subset of $\mathcal{A}$ of at most r agents can traverse B from u to v (or vice versa)

$r = 3$



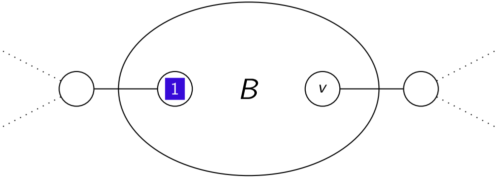
Lower bound - building block

$\mathcal{A}$ = set of k agents with s states each

Definition (Barrier)

B is ***r*-barrier** for $\mathcal{A} \Leftrightarrow$ no subset of $\mathcal{A}$ of at most r agents can traverse B from u to v (or vice versa)

$r = 3$



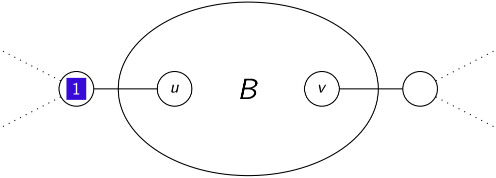
Lower bound - building block

$\mathcal{A}$ = set of k agents with s states each

Definition (Barrier)

B is ***r*-barrier** for $\mathcal{A}$ $\Leftrightarrow$ no subset of $\mathcal{A}$ of at most r agents can traverse B from u to v (or vice versa)

$r = 3$



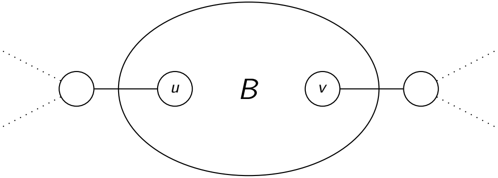
Lower bound - building block

$\mathcal{A}$ = set of k agents with s states each

Definition (Barrier)

B is ***r*-barrier** for $\mathcal{A} \Leftrightarrow$ no subset of $\mathcal{A}$ of at most r agents can traverse B from u to v (or vice versa)

$r = 3$



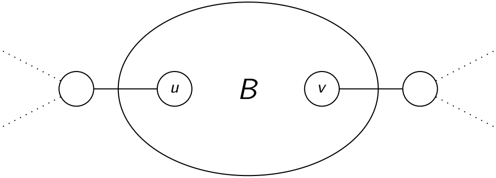
Lower bound - building block

$\mathcal{A}$ = set of k agents with s states each

Definition (Barrier)

B is ***r*-barrier** for $\mathcal{A} \Leftrightarrow$ no subset of $\mathcal{A}$ of at most r agents can traverse B from u to v (or vice versa)

$r = 3$



Observation

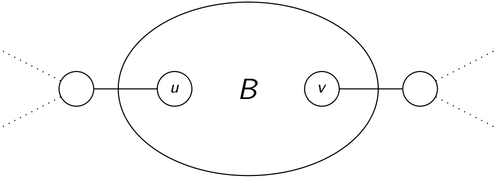
Lower bound - building block

$\mathcal{A}$ = set of k agents with s states each

Definition (Barrier)

B is ***r*-barrier** for $\mathcal{A}$ $\Leftrightarrow$ no subset of $\mathcal{A}$ of at most r agents can traverse B from u to v (or vice versa)

$r = 3$



Observation

- k-barrier for $\mathcal{A}$ yields graph that agents do **not** explore

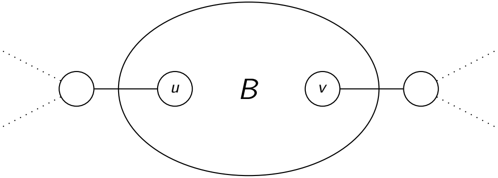
Lower bound - building block

$\mathcal{A}$ = set of k agents with s states each

Definition (Barrier)

B is ***r*-barrier** for $\mathcal{A} \Leftrightarrow$ no subset of $\mathcal{A}$ of at most r agents can traverse B from u to v (or vice versa)

$r = 3$

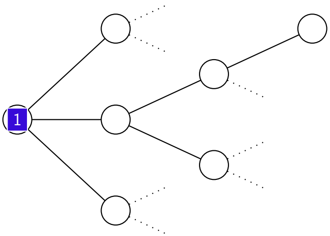


Observation

- k-barrier for $\mathcal{A}$ yields graph that agents do **not** explore
- Size of k-barrier $\Rightarrow$ lower bound on # agents and states per agent

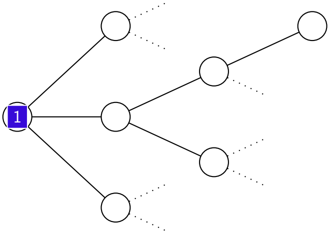
Construction of 1-barrier [Fraigniaud et al., TCS '06]

- consider 1 agent in infinite 3-regular tree



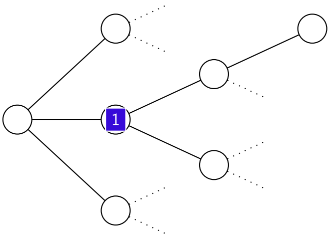
Construction of 1-barrier [Fraigniaud et al., TCS '06]

- consider 1 agent in infinite 3-regular tree
  $\Rightarrow$ state determines next vertex



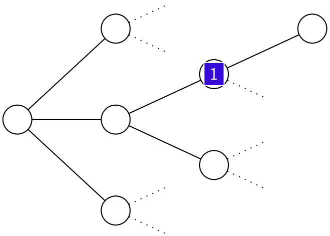
Construction of 1-barrier [Fraigniaud et al., TCS '06]

- consider 1 agent in infinite 3-regular tree
  $\Rightarrow$ state determines next vertex



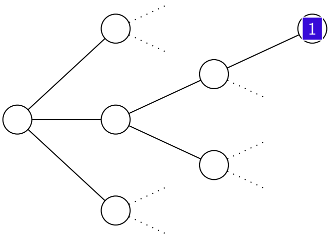
Construction of 1-barrier [Fraigniaud et al., TCS '06]

- consider 1 agent in infinite 3-regular tree
  $\Rightarrow$ state determines next vertex



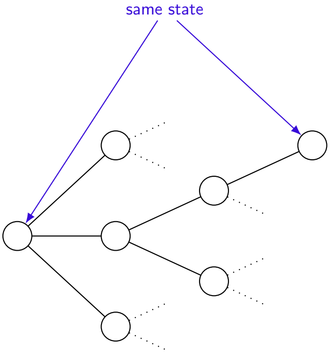
Construction of 1-barrier [Fraigniaud et al., TCS '06]

- consider 1 agent in infinite 3-regular tree
  $\Rightarrow$ state determines next vertex



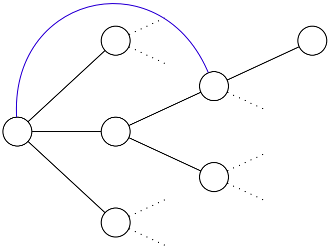
Construction of 1-barrier [Fraigniaud et al., TCS '06]

- consider 1 agent in infinite 3-regular tree
  $\Rightarrow$ state determines next vertex
- state repeats after finite number of steps



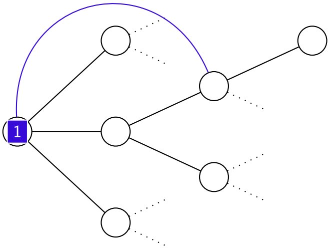
Construction of 1-barrier [Fraigniaud et al., TCS '06]

- consider 1 agent in infinite 3-regular tree
  $\Rightarrow$ state determines next vertex
- state repeats after finite number of steps $\rightsquigarrow$ close loop



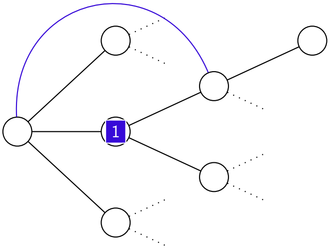
Construction of 1-barrier [Fraigniaud et al., TCS '06]

- consider 1 agent in infinite 3-regular tree
  $\Rightarrow$ state determines next vertex
- state repeats after finite number of steps $\rightsquigarrow$ close loop



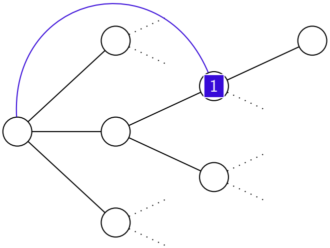
Construction of 1-barrier [Fraigniaud et al., TCS '06]

- consider 1 agent in infinite 3-regular tree
  $\Rightarrow$ state determines next vertex
- state repeats after finite number of steps $\rightsquigarrow$ close loop



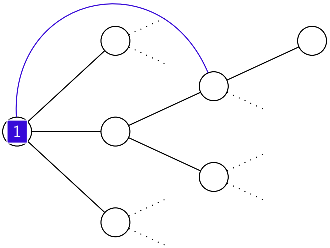
Construction of 1-barrier [Fraigniaud et al., TCS '06]

- consider 1 agent in infinite 3-regular tree
  $\Rightarrow$ state determines next vertex
- state repeats after finite number of steps $\rightsquigarrow$ close loop



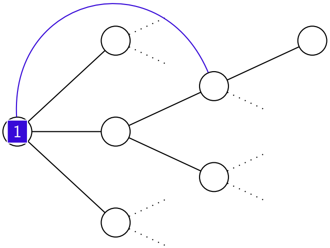
Construction of 1-barrier [Fraigniaud et al., TCS '06]

- consider 1 agent in infinite 3-regular tree
  $\Rightarrow$ state determines next vertex
- state repeats after finite number of steps $\rightsquigarrow$ close loop



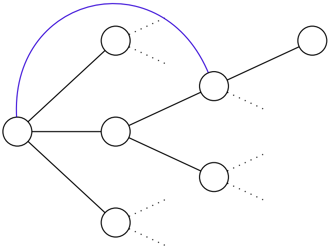
Construction of 1-barrier [Fraigniaud et al., TCS '06]

- consider 1 agent in infinite 3-regular tree
  $\Rightarrow$ state determines next vertex
- state repeats after finite number of steps $\rightsquigarrow$ close loop
  $\Rightarrow$ barrier for fixed starting state with $\mathcal{O}(s)$ vertices



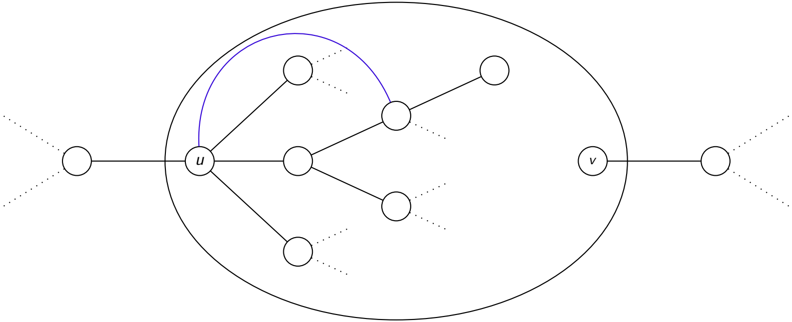
Construction of 1-barrier [Fraigniaud et al., TCS '06]

- consider 1 agent in infinite 3-regular tree
  $\Rightarrow$ state determines next vertex
- state repeats after finite number of steps $\rightsquigarrow$ close loop
  $\Rightarrow$ barrier for fixed starting state with $\mathcal{O}(s)$ vertices
- repeat construction for all k agents and all s states of every agent



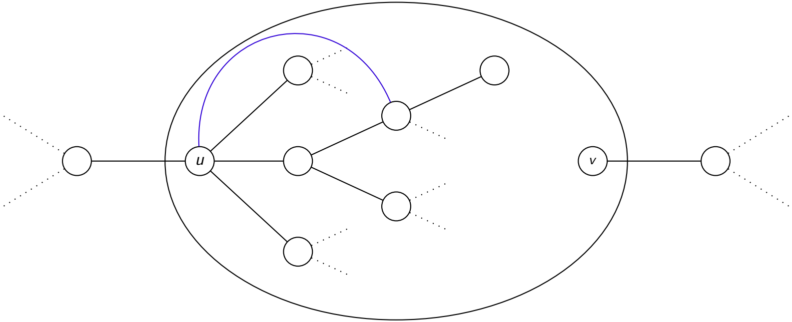
Construction of 1-barrier [Fraigniaud et al., TCS '06]

- consider 1 agent in infinite 3-regular tree
  $\Rightarrow$ state determines next vertex
- state repeats after finite number of steps $\rightsquigarrow$ close loop
  $\Rightarrow$ barrier for fixed starting state with $\mathcal{O}(s)$ vertices
- repeat construction for all k agents and all s states of every agent



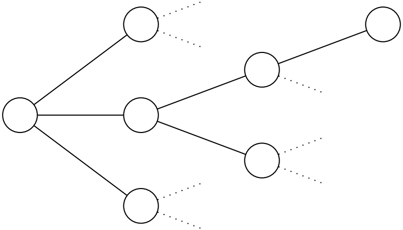
Construction of 1-barrier [Fraigniaud et al., TCS '06]

- consider 1 agent in infinite 3-regular tree
  $\Rightarrow$ state determines next vertex
- state repeats after finite number of steps $\rightsquigarrow$ close loop
  $\Rightarrow$ barrier for fixed starting state with $\mathcal{O}(s)$ vertices
- repeat construction for all k agents and all s states of every agent
  $\Rightarrow$ 1-barrier with $\mathcal{O}(k \cdot s^2)$ vertices



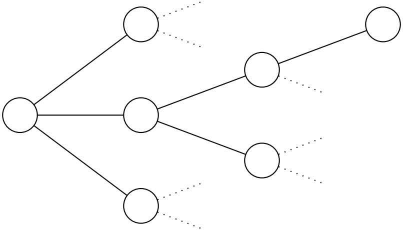
Lower bound - recursive construction

- r agent in infinite 3-regular tree



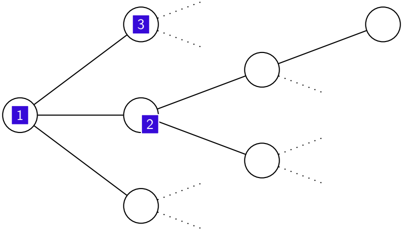
Lower bound - recursive construction

- r agent in infinite 3-regular tree
  - configuration (= states + relative positions of agents) determines next vertex



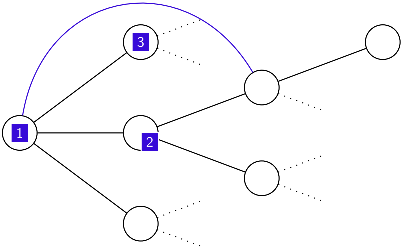
Lower bound - recursive construction

- r agent in infinite 3-regular tree
  - configuration (= states + relative positions of agents) determines next vertex



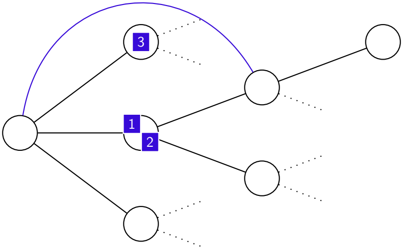
Lower bound - recursive construction

- r agent in infinite 3-regular tree
  - configuration (= states + relative positions of agents) determines next vertex



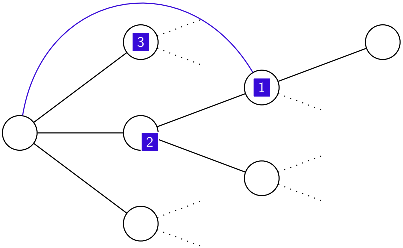
Lower bound - recursive construction

- r agent in infinite 3-regular tree
  - configuration (= states + relative positions of agents) determines next vertex



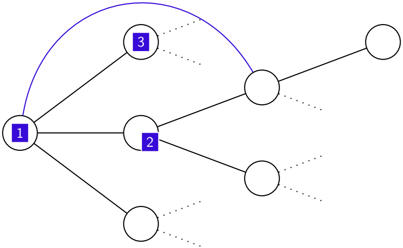
Lower bound - recursive construction

- r agent in infinite 3-regular tree
  - configuration (= states + relative positions of agents) determines next vertex



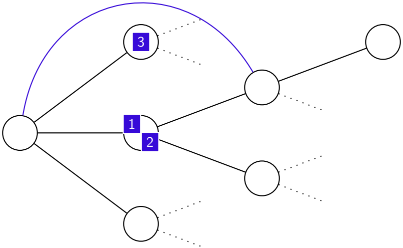
Lower bound - recursive construction

- r agent in infinite 3-regular tree
  - configuration (= states + relative positions of agents) determines next vertex



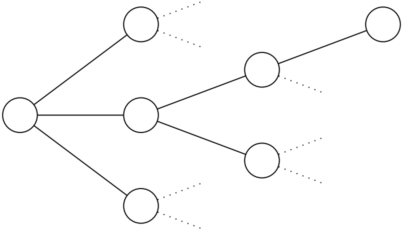
Lower bound - recursive construction

- r agent in infinite 3-regular tree
  - configuration (= states + relative positions of agents) determines next vertex



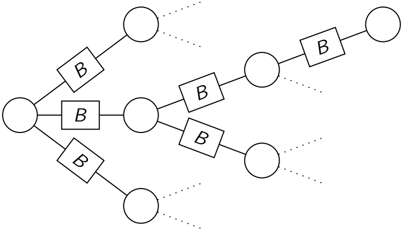
Lower bound - recursive construction

- r agent in infinite 3-regular tree
  - configuration (= states + relative positions of agents) determines next vertex
- Idea: replace edges by $(r-1)$-barrier B



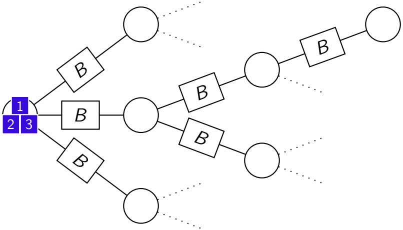
Lower bound - recursive construction

- r agent in infinite 3-regular tree
  - configuration (= states + relative positions of agents) determines next vertex
- Idea: replace edges by $(r-1)$-barrier B



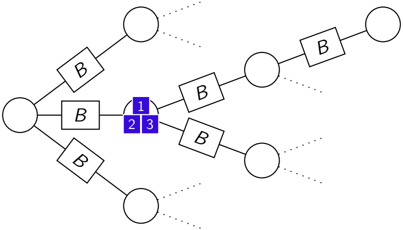
Lower bound - recursive construction

- r agent in infinite 3-regular tree
  - configuration (= states + relative positions of agents) determines next vertex
- Idea: replace edges by $(r-1)$-barrier B



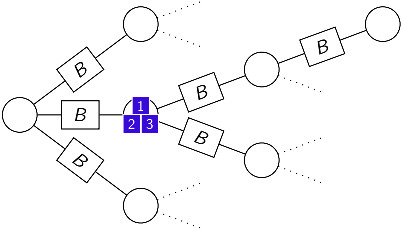
Lower bound - recursive construction

- r agent in infinite 3-regular tree
  - configuration (= states + relative positions of agents) determines next vertex
- Idea: replace edges by $(r-1)$-barrier B



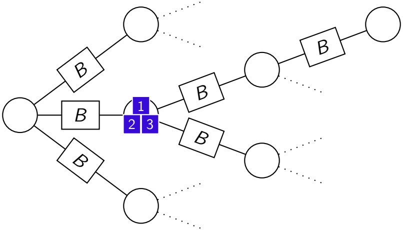
Lower bound - recursive construction

- r agent in infinite 3-regular tree
  - configuration (= states + relative positions of agents) determines next vertex
- Idea: replace edges by $(r-1)$-barrier B
  - all r agents close together



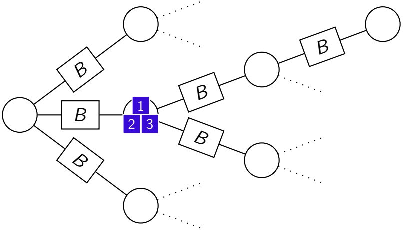
Lower bound - recursive construction

- r agent in infinite 3-regular tree
  - configuration (= states + relative positions of agents) determines next vertex
- Idea: replace edges by $(r-1)$-barrier B
  - all r agents close together
  - configuration repeats



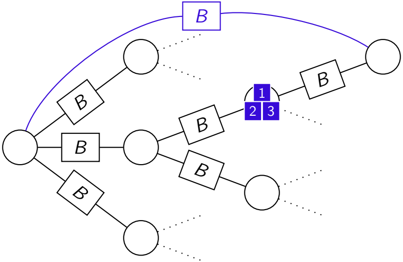
Lower bound - recursive construction

- r agent in infinite 3-regular tree
  - configuration (= states + relative positions of agents) determines next vertex
- Idea: replace edges by $(r-1)$-barrier B
  - all r agents close together
  - configuration repeats $\rightsquigarrow$ close loop (as for 1-barrier)



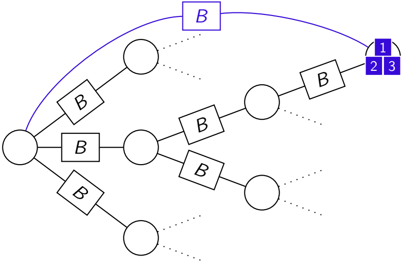
Lower bound - recursive construction

- r agent in infinite 3-regular tree
  - configuration (= states + relative positions of agents) determines next vertex
- Idea: replace edges by $(r-1)$-barrier B
  - all r agents close together
  - configuration repeats $\rightsquigarrow$ close loop (as for 1-barrier)



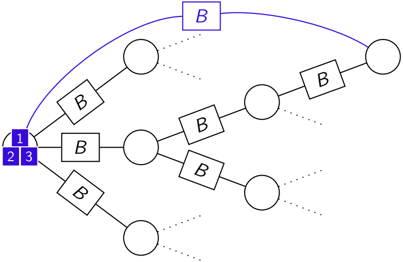
Lower bound - recursive construction

- r agent in infinite 3-regular tree
  - configuration (= states + relative positions of agents) determines next vertex
- Idea: replace edges by $(r-1)$-barrier B
  - all r agents close together
  - configuration repeats $\rightsquigarrow$ close loop (as for 1-barrier)



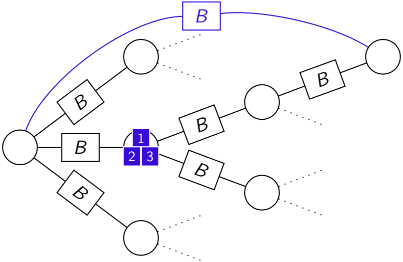
Lower bound - recursive construction

- r agent in infinite 3-regular tree
  - configuration (= states + relative positions of agents) determines next vertex
- Idea: replace edges by $(r-1)$-barrier B
  - all r agents close together
  - configuration repeats $\rightsquigarrow$ close loop (as for 1-barrier)



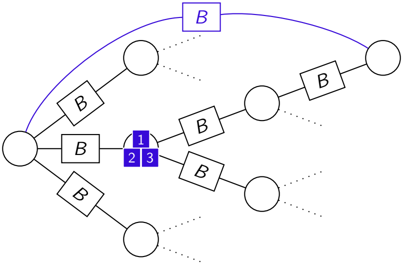
Lower bound - recursive construction

- r agent in infinite 3-regular tree
  - configuration (= states + relative positions of agents) determines next vertex
- Idea: replace edges by $(r-1)$-barrier B
  - all r agents close together
  - configuration repeats $\rightsquigarrow$ close loop (as for 1-barrier)



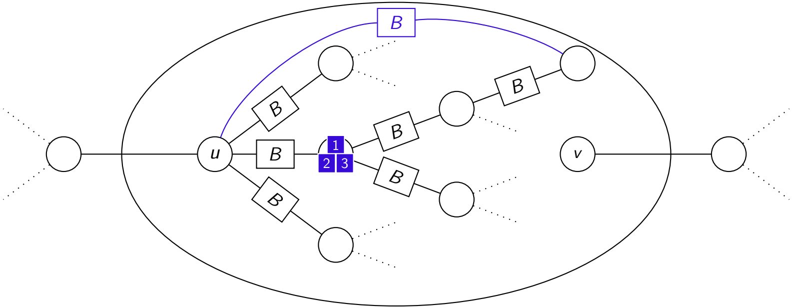
Lower bound - recursive construction

- r agent in infinite 3-regular tree
  - configuration (= states + relative positions of agents) determines next vertex
- Idea: replace edges by $(r-1)$-barrier B
  - all r agents close together
  - configuration repeats $\rightsquigarrow$ close loop (as for 1-barrier)
  - repeat construction for all $\binom{k}{r}$ subsets of r agents and all configs



Lower bound - recursive construction

- r agent in infinite 3-regular tree
  - configuration (= states + relative positions of agents) determines next vertex
- Idea: replace edges by $(r-1)$-barrier B
  - all r agents close together
  - configuration repeats $\rightsquigarrow$ close loop (as for 1-barrier)
  - repeat construction for all $\binom{k}{r}$ subsets of r agents and all configs



Lower bound

Theorem (Disser, H., Klimm)

For any set $\mathcal{A}$ of k agents with s states each,
there is a graph with $\mathcal{O}(s^{10^k})$ vertices that $\mathcal{A}$ does not explore.

Lower bound

Theorem (Disser, H., Klimm)

For any set $\mathcal{A}$ of k agents with s states each, there is a graph with $\mathcal{O}(s^{10^k})$ vertices that $\mathcal{A}$ does not explore.

Theorem (Disser, H., Klimm)

$\Omega(\log\log n)$ *agents are necessary to explore any n vertex graph, if each agent has $\mathcal{O}((\log n)^{1-\varepsilon})$ bits of memory for $\varepsilon > 0$.*

Summary

Exploration of any graph with n vertices by 1 agent requires

- $\Omega(\log n)$ bits of memory
  [Fraigniaud, Ilcinkas, Peer, Pelc, Peleg, TCS '05]
- $\mathcal{O}(\log n)$ bits of memory
  [Reingold, JACM '08]

Summary

Exploration of any graph with n vertices by 1 agent requires

- $\Omega(\log n)$ bits of memory
  [Fraigniaud, Ilcinkas, Peer, Pelc, Peleg, TCS '05]
- $\mathcal{O}(\log n)$ bits of memory
  [Reingold, JACM '08]

Exploration of any undirected graph with n vertices by k agents requires

- $\Omega(\log \log n)$ agents if each agent has at most $\mathcal{O}((\log n)^{1-\varepsilon})$ bits of memory for $\varepsilon > 0$
- $\mathcal{O}(\log \log n)$ agents with $\mathcal{O}(1)$ bits of memory

Summary

Exploration of any graph with n vertices by 1 agent requires

- $\Omega(\log n)$ bits of memory
  [Fraigniaud, Ilcinkas, Peer, Pelc, Peleg, TCS '05]
- $\mathcal{O}(\log n)$ bits of memory
  [Reingold, JACM '08]

Exploration of any undirected graph with n vertices by k agents requires

- $\Omega(\log \log n)$ agents if each agent has at most $\mathcal{O}((\log n)^{1-\varepsilon})$ bits of memory for $\varepsilon > 0$
- $\mathcal{O}(\log \log n)$ agents with $\mathcal{O}(1)$ bits of memory

Thank you!