

Euler Tour Lock-in Problem in the Rotor-Router Model

I choose pointers and you choose port numbers

Evangelos Bampas^{1,3} Leszek Gasieniec²
Nicolas Hanusse³ David Ilcinkas³
Ralf Klasing³ Adrian Kosowski^{3,4}

¹National Technical University of Athens, Greece

²University of Liverpool, UK

³CNRS / INRIA / Univ. of Bordeaux, France

⁴Gdańsk University of Technology, Poland

ANR Aladdin meeting
November 27th, 2009

Definitions

Anonymous graphs / networks

- No (used) node labeling
- **Local port numbering** at node v from 1 to $\deg(v)$

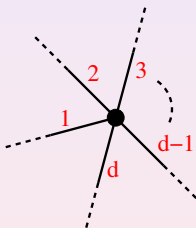
Mobile agent / robot / message / anything

Follows the rotor-router mechanism

Definitions

Anonymous graphs / networks

- No (used) node labeling
- **Local port numbering** at node v from 1 to $\deg(v)$



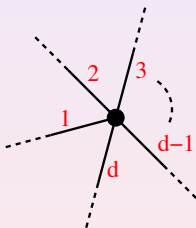
Mobile agent / robot / message / anything

Follows the rotor-router mechanism

Definitions

Anonymous graphs / networks

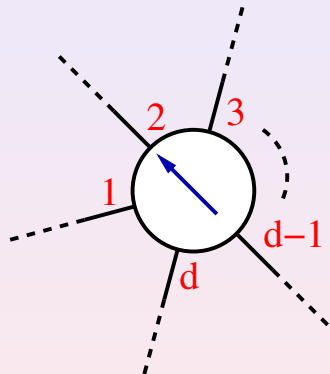
- No (used) node labeling
- **Local port numbering** at node v from 1 to $\deg(v)$



Mobile agent / robot / message / anything

Follows the **rotor-router** mechanism

Rotor-router mechanism

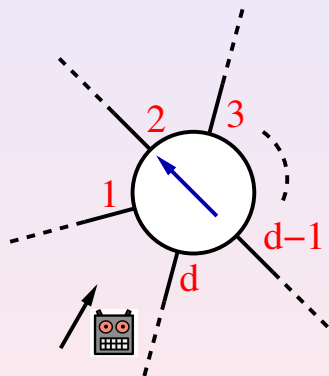


Very simple mechanism:

- Each node has a **pointer** ↗
- The agent follows the pointer and “increments” it with respect to the cyclic ordering \odot (induced by the port numbering)

Other names: Propp machine, Next-Port, Edge Ant Walk

Rotor-router mechanism

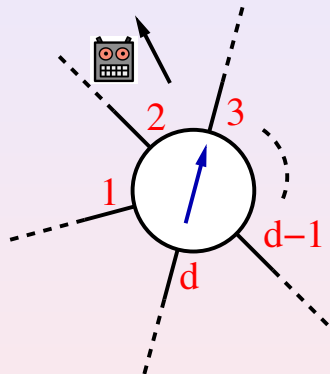


Very simple mechanism:

- Each node has a **pointer** ↗
- The agent follows the **pointer** and “increments” it with respect to the cyclic ordering \odot (induced by the port numbering)

Other names: Propp machine, Next-Port, Edge Ant Walk

Rotor-router mechanism

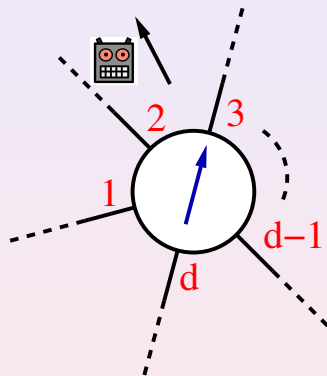


Very simple mechanism:

- Each node has a **pointer** ↗
- The agent follows the **pointer** and “increments” it with respect to the **cyclic ordering** ⌚ (induced by the port numbering)

Other names: Propp machine, Next-Port, Edge Ant Walk

Rotor-router mechanism



Very simple mechanism:

- Each node has a **pointer** ↗
- The agent follows the **pointer** and “increments” it with respect to the **cyclic ordering** ⌚ (induced by the port numbering)

Other names: Propp machine, Next-Port, Edge Ant Walk

Context

Known results

- The agent eventually **traverses each edge**
- The traversal stabilizes into an Euler tour: each edge is traversed once in each direction within a period
- Lock-in time: $\Theta(m \cdot D)$ (m : # edges, D : diameter)

Motivations / Applications

- Graph exploration (by a software agent / robot)
- Mutual exclusion
- Stabilisation of distributed processes
- Work and load balancing problems
- etc.

Context

Known results

- The agent eventually **traverses each edge**
- The traversal stabilizes into an **Euler tour**: each edge is traversed **once in each direction** within a period
- Lock-in time: $\Theta(m \cdot D)$ (m : # edges, D : diameter)

Motivations / Applications

- Graph exploration (by a software agent / robot)
- Mutual exclusion
- Stabilisation of distributed processes
- Work and load balancing problems
- etc.

Context

Known results

- The agent eventually **traverses each edge**
- The traversal stabilizes into an **Euler tour**: each edge is traversed **once in each direction** within a period
- Lock-in time: $\Theta(m \cdot D)$ (m : # edges, D : diameter)

Motivations / Applications

- Graph exploration (by a software agent / robot)
- Mutual exclusion
- Stabilisation of distributed processes
- Work and load balancing problems
- etc.

Context

Known results

- The agent eventually **traverses each edge**
- The traversal stabilizes into an **Euler tour**: each edge is traversed **once in each direction** within a period
- Lock-in time: $\Theta(m \cdot D)$ (m : # edges, D : diameter)

Motivations / Applications

- Graph exploration (by a software agent / robot)
- Mutual exclusion
- Stabilisation of distributed processes
- Work and load balancing problems
- etc.

Related work

- Y. Afek and E. Gafni, *SIAM Journal on Computing*, 1994.
- S. Bhatt, S. Even, D. Greenberg, and R. Tayar, *Journal of Graph Algorithms and Applications*, 2002.
- J.N. Cooper and J. Spencer, *Combinatorics, Probability and Computing*, 2006.
- B. Doerr and T. Friedrich, *Combinatorics, Probability and Computing*, 2009.
- A.S. Fraenkel, *Mathematics Magazine*, 1970.
- L. Gasieniec and T. Radzik, *Proc. WG*, 2008.
- V.B. Priezzhev, D. Dhar, A. Dhar, and S. Krishnamurthy, *Physics Review Letters*, 1996.
- S. Tixeuil, *Proc. WSS*, 2001.
- V. Yanovski, I.A. Wagner, and A.M. Bruckstein, *Algorithmica*, 2003.

Related work

- Y. Afek and E. Gafni, *SIAM Journal on Computing*, 1994.
- S. Bhatt, S. Even, D. Greenberg, and R. Tayar, *Journal of Graph Algorithms and Applications*, 2002.
- J.N. Cooper and J. Spencer, *Combinatorics, Probability and Computing*, 2006.
- B. Doerr and T. Friedrich, *Combinatorics, Probability and Computing*, 2009.
- A.S. Fraenkel, *Mathematics Magazine*, 1970.
- L. Gasieniec and T. Radzik, *Proc. WG*, 2008.
- V.B. Priezzhev, D. Dhar, A. Dhar, and S. Krishnamurthy, *Physics Review Letters*, 1996.
- S. Tixeuil, *Proc. WSS*, 2001.
- **V. Yanovski, I.A. Wagner, and A.M. Bruckstein, *Algorithmica*, 2003.**

Our results

Problem

How does the **lock-in time** depend on the **initial configuration** of the ports \circlearrowleft and pointers \nearrow ?

Scenario	Worst case	Best case
Case \mathcal{P} -all	$\Theta(m)$	$\Theta(m)$
Case $\mathcal{A}(\circlearrowleft)\mathcal{P}(\nearrow)$	$\Theta(m)$	$\Theta(m)$
Case $\mathcal{P}(\nearrow)\mathcal{A}(\circlearrowleft)$	$\Theta(m \cdot \min\{\log m, D\})$	$\Theta(m)$
Case $\mathcal{A}(\nearrow)\mathcal{P}(\circlearrowleft)$	$\Theta(m \cdot D)$	$\Theta(m)$
Case $\mathcal{P}(\circlearrowleft)\mathcal{A}(\nearrow)$	$\Theta(m \cdot D)$	$\Theta(m)$ for $D \leq \sqrt{n}$
Case \mathcal{A} -all	$\Theta(m \cdot D)$	$\Theta(m \cdot D)$

\mathcal{P} =Player, \mathcal{A} =Adversary

(Subtitle: I choose pointers and you choose port numbers)

Our results

Problem

How does the **lock-in time** depend on the **initial configuration** of the ports \circ and pointers \nearrow ?

Scenario	Worst case	Best case
Case \mathcal{P} -all	$\Theta(m)$	$\Theta(m)$
Case $\mathcal{A}(\circ)\mathcal{P}(\nearrow)$	$\Theta(m)$	$\Theta(m)$
Case $\mathcal{P}(\nearrow)\mathcal{A}(\circ)$	$\Theta(m \cdot \min\{\log m, D\})$	$\Theta(m)$
Case $\mathcal{A}(\nearrow)\mathcal{P}(\circ)$	$\Theta(m \cdot D)$	$\Theta(m)$
Case $\mathcal{P}(\circ)\mathcal{A}(\nearrow)$	$\Theta(m \cdot D)$	$\Theta(m)$ for $D \leq \sqrt{n}$
Case \mathcal{A} -all	$\Theta(m \cdot D)$	$\Theta(m \cdot D)$

\mathcal{P} =Player, \mathcal{A} =Adversary

(Subtitle: I choose pointers and you choose port numbers)

Our results

Problem

How does the **lock-in time** depend on the **initial configuration** of the ports \circlearrowleft and pointers \nearrow ?

Scenario	Worst case	Best case
Case \mathcal{P} -all	$\Theta(m)$	$\Theta(m)$
Case $\mathcal{A}(\circlearrowleft)\mathcal{P}(\nearrow)$	$\Theta(m)$	$\Theta(m)$
Case $\mathcal{P}(\nearrow)\mathcal{A}(\circlearrowleft)$	$\Theta(m \cdot \min\{\log m, D\})$	$\Theta(m)$
Case $\mathcal{A}(\nearrow)\mathcal{P}(\circlearrowleft)$	$\Theta(m \cdot D)$	$\Theta(m)$
Case $\mathcal{P}(\circlearrowleft)\mathcal{A}(\nearrow)$	$\Theta(m \cdot D)$	$\Theta(m)$ for $D \leq \sqrt{n}$
Case \mathcal{A} -all	$\Theta(m \cdot D)$	$\Theta(m \cdot D)$

\mathcal{P} =Player, \mathcal{A} =Adversary

(Subtitle: I choose pointers and you choose port numbers)

Player wins

Scenario	Worst case	Best case
Case \mathcal{P} -all	$\Theta(m)$	$\Theta(m)$
Case $\mathcal{A}(\odot)\mathcal{P}(\nearrow)$	$\Theta(m)$	$\Theta(m)$

Very simple algorithm

- Consider any choice of ports \odot and pointers \nearrow
- Run virtually the agent for $\Theta(m \cdot D)$ steps
- The pointers \nearrow are now correctly set again by \mathcal{P}

Player wins

Scenario	Worst case	Best case
Case \mathcal{P} -all	$\Theta(m)$	$\Theta(m)$
Case $\mathcal{A}(\circlearrowleft)\mathcal{P}(\nearrow)$	$\Theta(m)$	$\Theta(m)$

Very simple algorithm

- Consider **any** choice of ports \circlearrowleft and pointers \nearrow
- Run virtually the agent for $\Theta(m \cdot D)$ steps
- The pointers \nearrow are now correctly set again by \mathcal{P}

Player wins

Scenario	Worst case	Best case
Case \mathcal{P} -all	$\Theta(m)$	$\Theta(m)$
Case $\mathcal{A}(\circ)\mathcal{P}(\nearrow)$	$\Theta(m)$	$\Theta(m)$

Very simple algorithm

- Consider **any** choice of ports \circ and pointers \nearrow
- Run **virtually** the agent for $\Theta(m \cdot D)$ steps
- The pointers \nearrow are now correctly set again by \mathcal{P}

Player wins

Scenario	Worst case	Best case
Case \mathcal{P} -all	$\Theta(m)$	$\Theta(m)$
Case $\mathcal{A}(\circlearrowleft)\mathcal{P}(\nearrow)$	$\Theta(m)$	$\Theta(m)$

Very simple algorithm

- Consider **any** choice of ports \circlearrowleft and pointers \nearrow
- Run **virtually** the agent for $\Theta(m \cdot D)$ steps
- The pointers \nearrow are now correctly set again by \mathcal{P}

Useful properties (known results)

Definitions

- **Phase:** Interval between two traversals of the first edge
- G_i : Graph induced by the edges traversed in Phase i
- **Saturated node:** Node whose incident edges are all traversed in both directions during the current phase

(Known) Properties

- Each arc is traversed at most once during a phase
- $G_i \subseteq G_{i+1}$
- If $v \in G_i$, then v is saturated in G_{i+1}
- $\Rightarrow G_{i+1} \supseteq \text{neighborhood}(G_i)$



Useful properties (known results)

Definitions

- **Phase:** Interval between two traversals of the first edge
- G_i : Graph induced by the edges traversed in Phase i
- **Saturated node:** Node whose incident edges are all traversed in both directions during the current phase

(Known) Properties

- Each arc is traversed at most once during a phase
- $G_i \subseteq G_{i+1}$
- If $v \in G_i$, then v is saturated in G_{i+1}
- $\Rightarrow G_{i+1} \supseteq \text{neighborhood}(G_i)$

Useful properties (known results)

Definitions

- **Phase:** Interval between two traversals of the first edge
- G_i : Graph induced by the edges traversed in Phase i
- **Saturated node:** Node whose incident edges are all traversed in both directions during the current phase

(Known) Properties

- Each arc is traversed at most once during a phase
- $G_i \subseteq G_{i+1}$
- If $v \in G_i$, then v is saturated in G_{i+1}
- $\Rightarrow G_{i+1} \supseteq \text{neighborhood}(G_i)$



Useful properties (known results)

Definitions

- **Phase:** Interval between two traversals of the first edge
- G_i : Graph induced by the edges traversed in Phase i
- **Saturated node:** Node whose incident edges are all traversed in both directions during the current phase

(Known) Properties

- Each arc is traversed **at most once** during a phase

- $G_i \subseteq G_{i+1}$

- If $v \in G_i$, then v is saturated in G_{i+1}

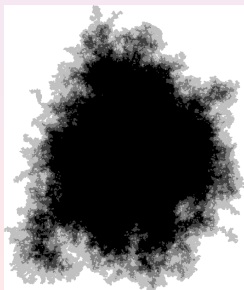
- $\Rightarrow G_{i+1} \supseteq \text{neighborhood}(G_i)$



Useful properties (known results)

Definitions

- **Phase:** Interval between two traversals of the first edge
- G_i : Graph induced by the edges traversed in Phase i
- **Saturated node:** Node whose incident edges are all traversed in both directions during the current phase



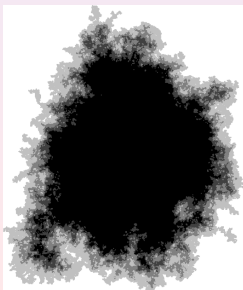
(Known) Properties

- Each arc is traversed **at most once** during a phase
- $G_i \subseteq G_{i+1}$
- If $v \in G_i$, then v is saturated in G_{i+1}
- $\Rightarrow G_{i+1} \supseteq \text{neighborhood}(G_i)$

Useful properties (known results)

Definitions

- **Phase:** Interval between two traversals of the first edge
- G_i : Graph induced by the edges traversed in Phase i
- **Saturated node:** Node whose incident edges are all traversed in both directions during the current phase



(Known) Properties

- Each arc is traversed **at most once** during a phase
- $G_i \subseteq G_{i+1}$
- If $v \in G_i$, then v is saturated in G_{i+1}

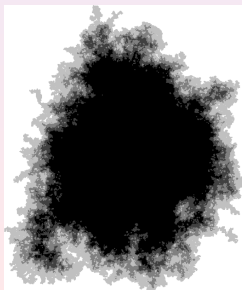
• $\Rightarrow G_{i+1} \supseteq \text{neighborhood}(G_i)$



Useful properties (known results)

Definitions

- **Phase:** Interval between two traversals of the first edge
- G_i : Graph induced by the edges traversed in Phase i
- **Saturated node:** Node whose incident edges are all traversed in both directions during the current phase



(Known) Properties

- Each arc is traversed **at most once** during a phase
- $G_i \subseteq G_{i+1}$
- If $v \in G_i$, then v is saturated in G_{i+1}
- $\Rightarrow G_{i+1} \supseteq \text{neighborhood}(G_i)$

Adversary wins

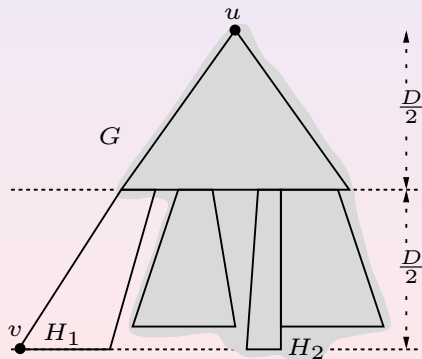
Scenario	Worst case	Best case
Case \mathcal{A} -all	$\Theta(m \cdot D)$	$\Theta(m \cdot D)$

(Sketch of the) Proof

- Let $[u, v]$ be a diameter
- Make G_1 be the larger of H_1 and H_2
- Make $G_{i+1} = \text{neighborhood}(G_i)$

Adversary wins

Scenario	Worst case	Best case
Case <i>A-all</i>	$\Theta(m \cdot D)$	$\Theta(m \cdot D)$

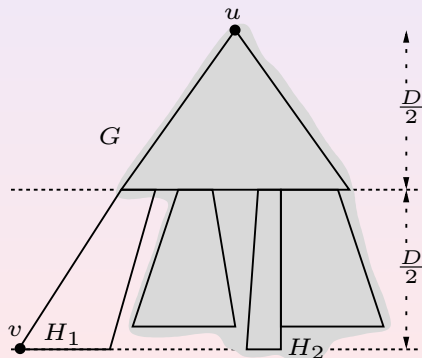


(Sketch of the) Proof

- Let $[u, v]$ be a diameter
- Make G_1 be the larger of H_1 and H_2
- Make $G_{i+1} = \text{neighborhood}(G_i)$

Adversary wins

Scenario	Worst case	Best case
Case \mathcal{A} -all	$\Theta(m \cdot D)$	$\Theta(m \cdot D)$



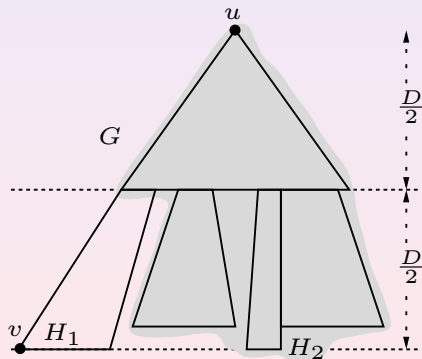
(Sketch of the) Proof

- Let $[u, v]$ be a diameter
- Make G_1 be the larger of H_1 and H_2

Make $G_{i+1} = \text{neighborhood}(G_i)$

Adversary wins

Scenario	Worst case	Best case
Case \mathcal{A} -all	$\Theta(m \cdot D)$	$\Theta(m \cdot D)$

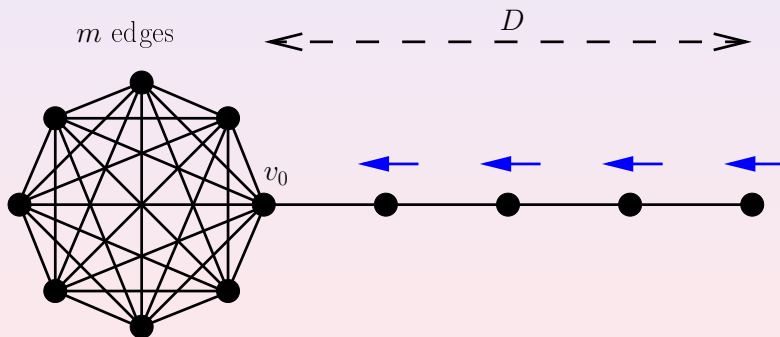


(Sketch of the) Proof

- Let $[u, v]$ be a diameter
- Make G_1 be the larger of H_1 and H_2
- Make $G_{i+1} = \text{neighborhood}(G_i)$

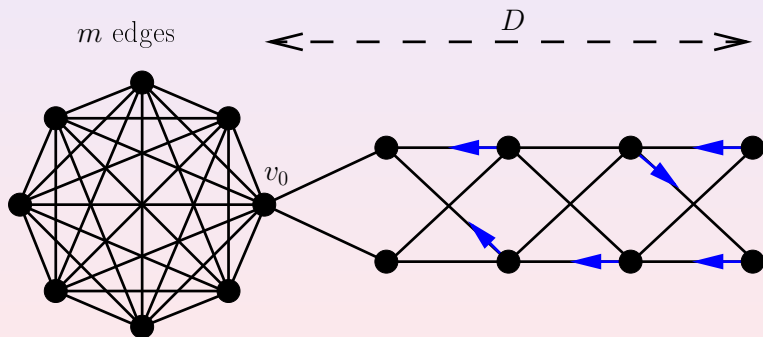
Case $\mathcal{A}(\nearrow)\mathcal{P}(\circlearrowleft)$

Scenario	Worst case	Best case
Case $\mathcal{A}(\nearrow)\mathcal{P}(\circlearrowleft)$	$\Theta(m \cdot D)$	$\Theta(m)$



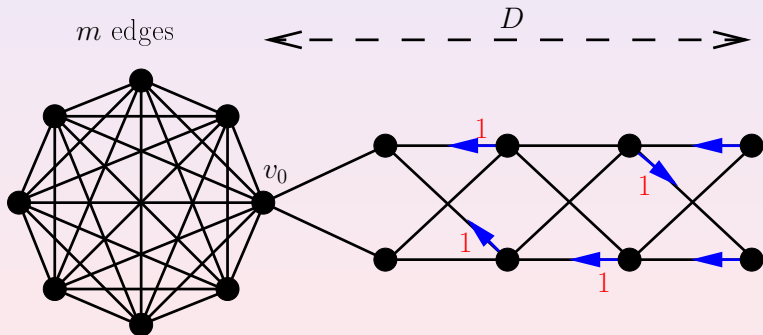
Case $\mathcal{A}(\nearrow)\mathcal{P}(\circlearrowleft)$

Scenario	Worst case	Best case
Case $\mathcal{A}(\nearrow)\mathcal{P}(\circlearrowleft)$	$\Theta(m \cdot D)$	$\Theta(m)$



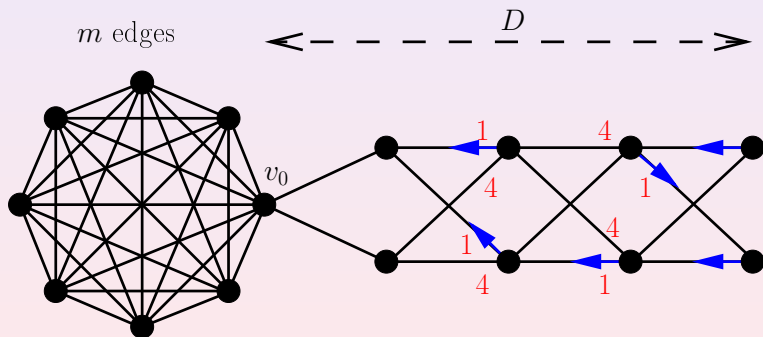
Case $\mathcal{A}(\nearrow)\mathcal{P}(\circlearrowleft)$

Scenario	Worst case	Best case
Case $\mathcal{A}(\nearrow)\mathcal{P}(\circlearrowleft)$	$\Theta(m \cdot D)$	$\Theta(m)$



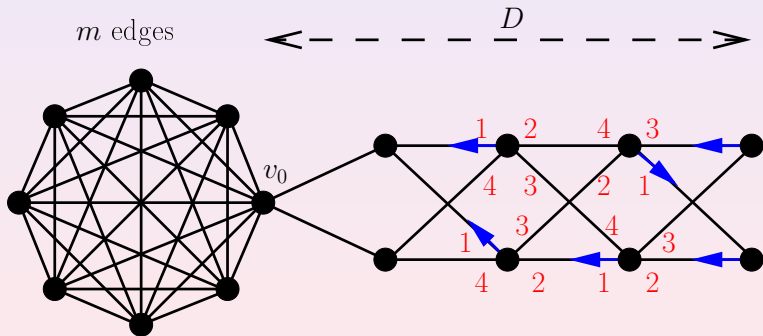
Case $\mathcal{A}(\nearrow)\mathcal{P}(\circlearrowleft)$

Scenario	Worst case	Best case
Case $\mathcal{A}(\nearrow)\mathcal{P}(\circlearrowleft)$	$\Theta(m \cdot D)$	$\Theta(m)$



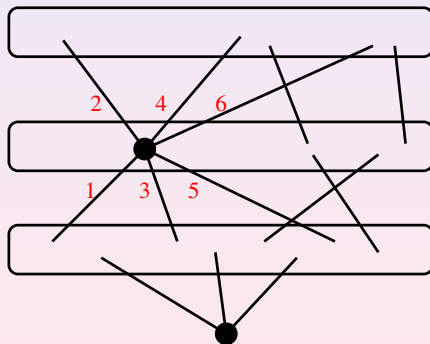
Case $\mathcal{A}(\nearrow)\mathcal{P}(\circlearrowleft)$

Scenario	Worst case	Best case
Case $\mathcal{A}(\nearrow)\mathcal{P}(\circlearrowleft)$	$\Theta(m \cdot D)$	$\Theta(m)$



Case $\mathcal{P}(\circ)\mathcal{A}(\nearrow)$

Scenario	Worst case	Best case
Case $\mathcal{P}(\circ)\mathcal{A}(\nearrow)$	$\Theta(m \cdot D)$	$\Theta(m)$ for $D \leq \sqrt{n}$



Conclusion and perspectives

Scenario	Worst case	Best case
Case \mathcal{P} -all	$\Theta(m)$	$\Theta(m)$
Case $\mathcal{A}(\circ)\mathcal{P}(\nearrow)$	$\Theta(m)$	$\Theta(m)$
Case $\mathcal{P}(\nearrow)\mathcal{A}(\circ)$	$\Theta(m \cdot \min\{\log m, D\})$	$\Theta(m)$
Case $\mathcal{A}(\nearrow)\mathcal{P}(\circ)$	$\Theta(m \cdot D)$	$\Theta(m)$
Case $\mathcal{P}(\circ)\mathcal{A}(\nearrow)$	$\Theta(m \cdot D)$	$\Theta(m)$ for $D \leq \sqrt{n}$
Case \mathcal{A} -all	$\Theta(m \cdot D)$	$\Theta(m \cdot D)$

Open problem

- Does there exist any graph of **large diameter** with lock-in time $O(m)$ in Case $\mathcal{P}(\circ)\mathcal{A}(\nearrow)$?

What if ports \circ and/or pointers \nearrow are set uniformly at random?

Conclusion and perspectives

Scenario	Worst case	Best case
Case \mathcal{P} -all	$\Theta(m)$	$\Theta(m)$
Case $\mathcal{A}(\circ)\mathcal{P}(\nearrow)$	$\Theta(m)$	$\Theta(m)$
Case $\mathcal{P}(\nearrow)\mathcal{A}(\circ)$	$\Theta(m \cdot \min\{\log m, D\})$	$\Theta(m)$
Case $\mathcal{A}(\nearrow)\mathcal{P}(\circ)$	$\Theta(m \cdot D)$	$\Theta(m)$
Case $\mathcal{P}(\circ)\mathcal{A}(\nearrow)$	$\Theta(m \cdot D)$	$\Theta(m)$ for $D \leq \sqrt{n}$
Case \mathcal{A} -all	$\Theta(m \cdot D)$	$\Theta(m \cdot D)$

Open problem

- Does there exist any graph of **large diameter** with lock-in time $O(m)$ in Case $\mathcal{P}(\circ)\mathcal{A}(\nearrow)$?
- What if ports \circ and/or pointers \nearrow are set uniformly **at random**?

Thank You
for your attention