Robustness of the rotor-router mechanism

E. Bampas[1,2], L. Gąsieniec[3], R. Klasing[1], A. Kosowski[1,4], T. Radzik[5]

[1] LaBRI, CNRS / INRIA / Univ. of Bordeaux, France

[2] School of Elec. & Comp. Eng., National Technical Univ. of Athens, Greece

[3] Dept of Computer Science, Univ. of Liverpool, UK

[4] Dept of Algorithms and System Modeling, Gdańsk Univ. of Technology, Poland

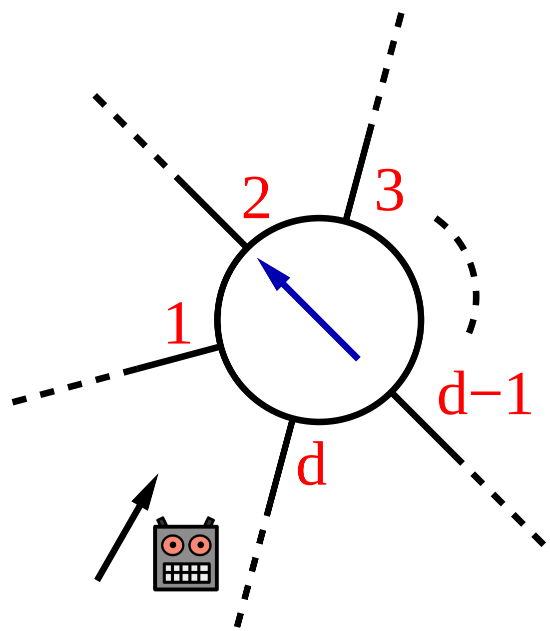
[5] Dept of Computer Science, King's College London, UK

Introduction

- Anonymous graphs (no node labels, but local port numbering)
- Exploration
- Agent with no operational memory
- Fault tolerance

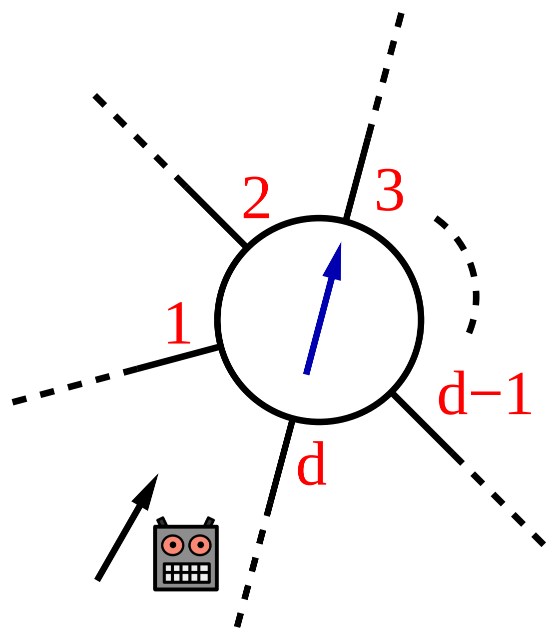
The rotor-router mechanism

- Each node is equipped with a pointer



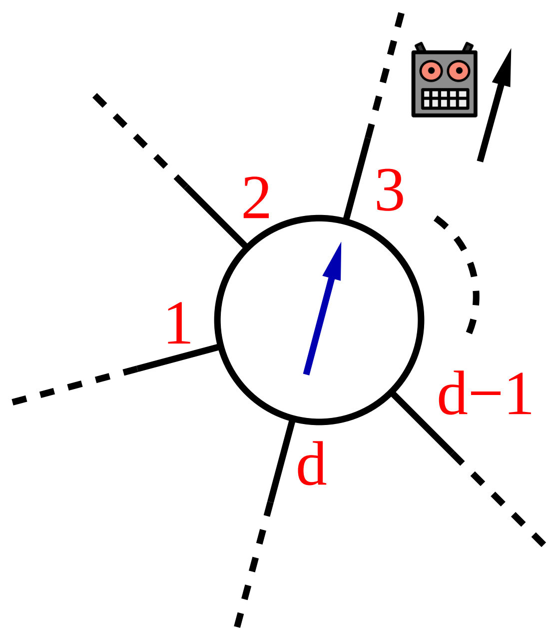
The rotor-router mechanism

- Each node is equipped with a pointer
- Incoming robot first increments the pointer,

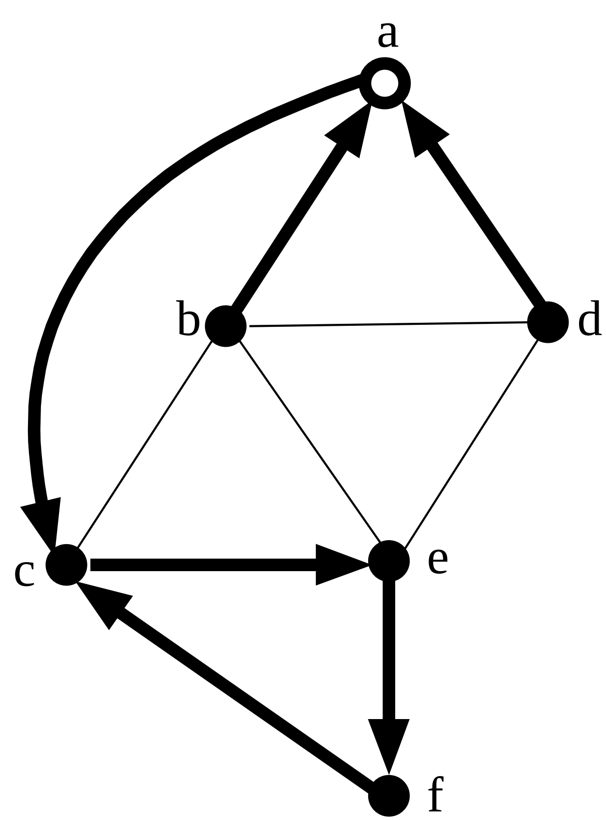


The rotor-router mechanism

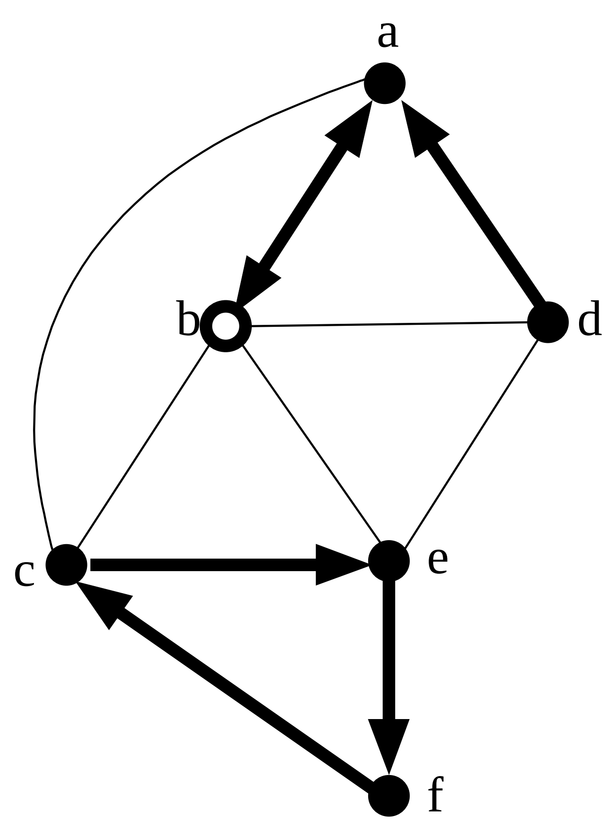
- Each node is equipped with a pointer
- Incoming robot first increments the pointer,
- then follows the pointer



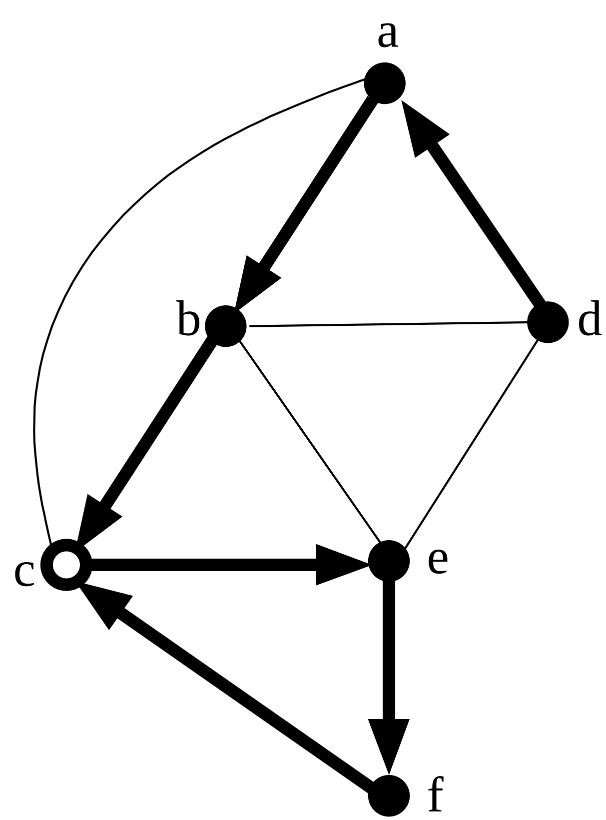
Example



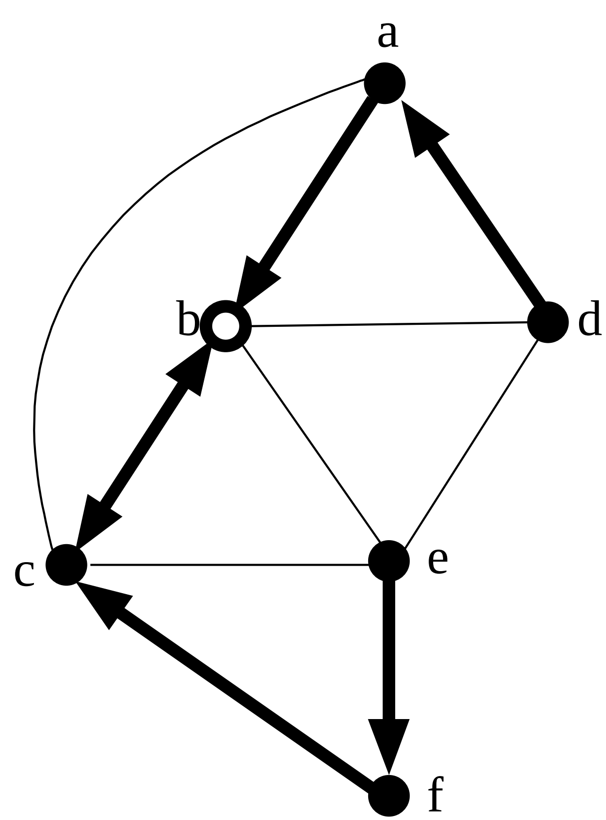
Example



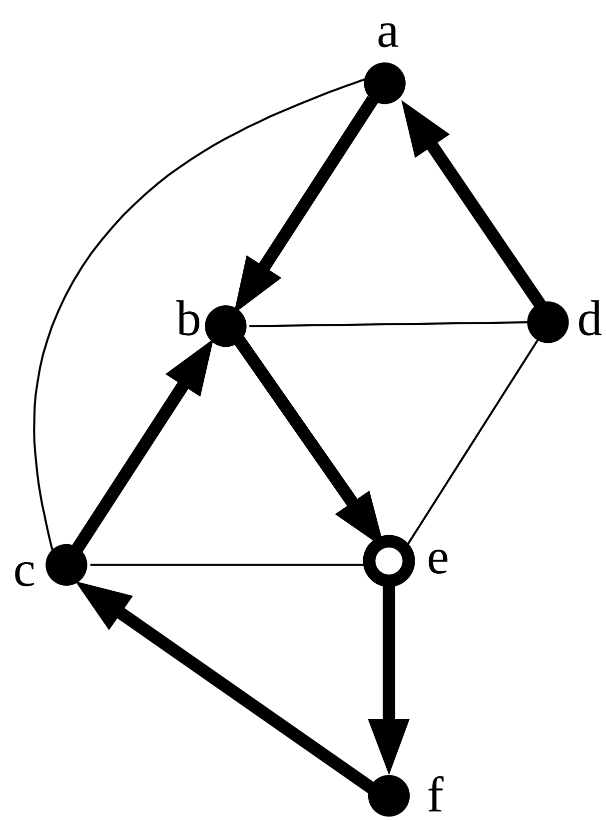
Example



Example



Example

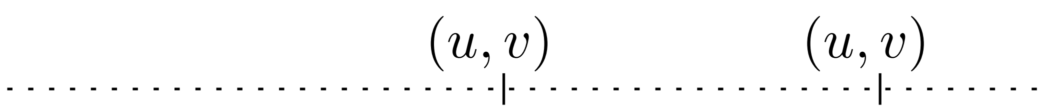


Known facts

- The agent visits each node infinitely many times
- It takes at most $2mD$ steps before the agent stabilizes into an Euler tour of the corresponding symmetric-directed graph
- After the stabilization period, the agent keeps repeating the same Euler tour
- No nested repetitions of arc traversals

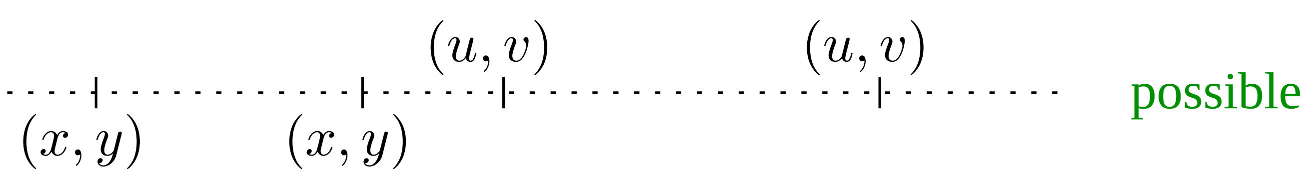
Known facts

- The agent visits each node infinitely many times
- It takes at most $2mD$ steps before the agent stabilizes into an Euler tour of the corresponding symmetric-directed graph
- After the stabilization period, the agent keeps repeating the same Euler tour
- No nested repetitions of arc traversals



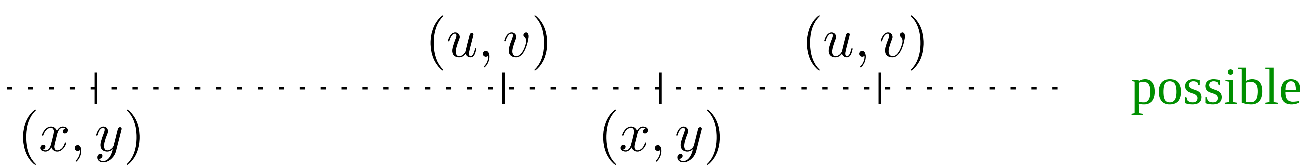
Known facts

- The agent visits each node infinitely many times
- It takes at most $2mD$ steps before the agent stabilizes into an Euler tour of the corresponding symmetric-directed graph
- After the stabilization period, the agent keeps repeating the same Euler tour
- No nested repetitions of arc traversals



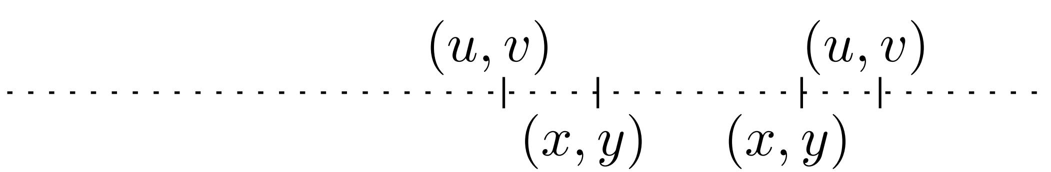
Known facts

- The agent visits each node infinitely many times
- It takes at most $2mD$ steps before the agent stabilizes into an Euler tour of the corresponding symmetric-directed graph
- After the stabilization period, the agent keeps repeating the same Euler tour
- No nested repetitions of arc traversals



Known facts

- The agent visits each node infinitely many times
- It takes at most $2mD$ steps before the agent stabilizes into an Euler tour of the corresponding symmetric-directed graph
- After the stabilization period, the agent keeps repeating the same Euler tour
- No nested repetitions of arc traversals



impossible

Related work

Priezzhev, Dhar, Dhar, Krishnamurthy: Eulerian walkers as a model of self-organized criticality. Phys. Rev. Lett. 77(25), 5079-5082 (1996)

introduction of the model

Wagner, Lindenbaum, Bruckstein: Smell as a computational resource - a lesson we can learn from the ant. ISTCS 1996, 219-230

Wagner, Lindenbaum, Bruckstein: Distributed covering by ant-robots using evaporating traces. IEEE Trans. Robot. Autom. 15, 918-933 (1999)

cover all edges in $O(nm)$ steps

Bhatt, Even, Greenberg, Tayar: Traversing directed Eulerian mazes. J. Graph Algorithms Appl. 6(2), 157-173 (2002)

enter an Euler tour in $O(nm)$ steps

Yanovski, Wagner, Bruckstein: A distributed ant algorithm for efficiently patrolling a network. Algorithmica 37(3), 165-186 (2003)

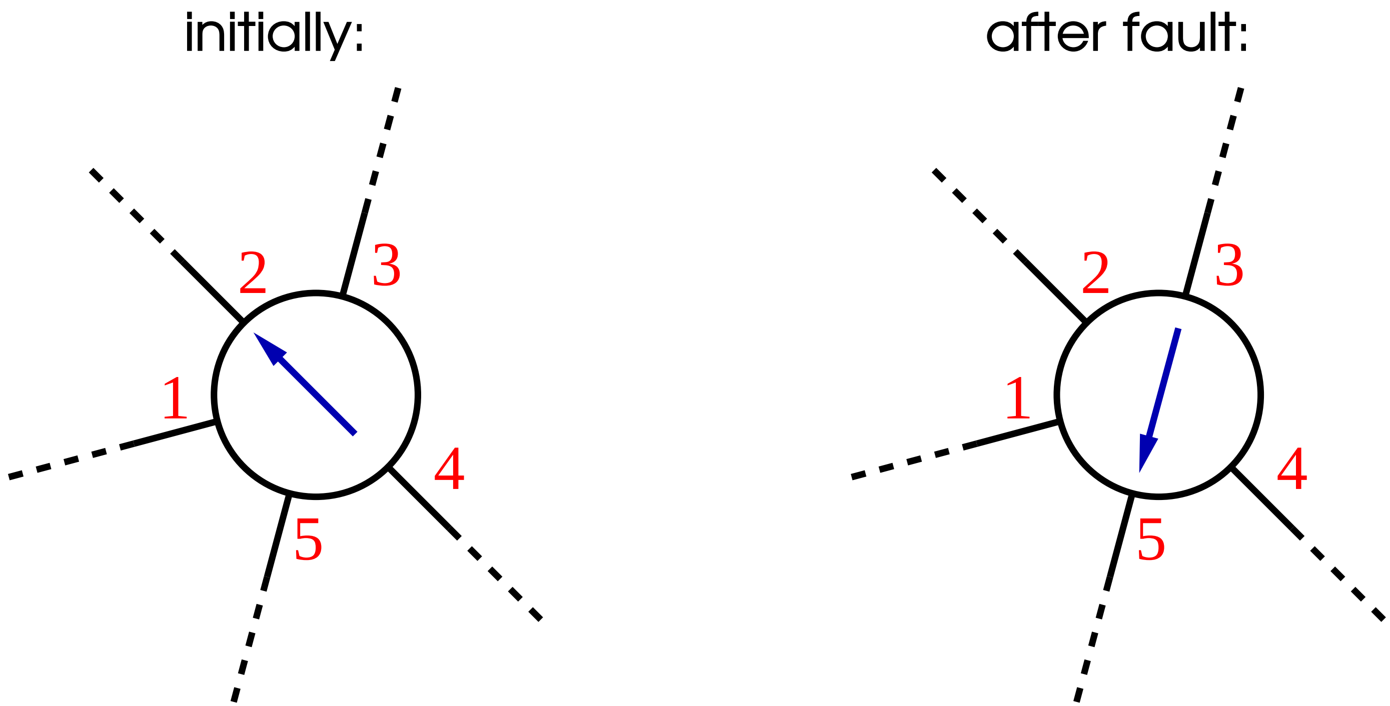
enter an Euler tour in $2mD$ steps

Main question

Assuming the rotor-router has stabilized, how long does it take to recover from potential pointer faults or dynamic changes in the graph?

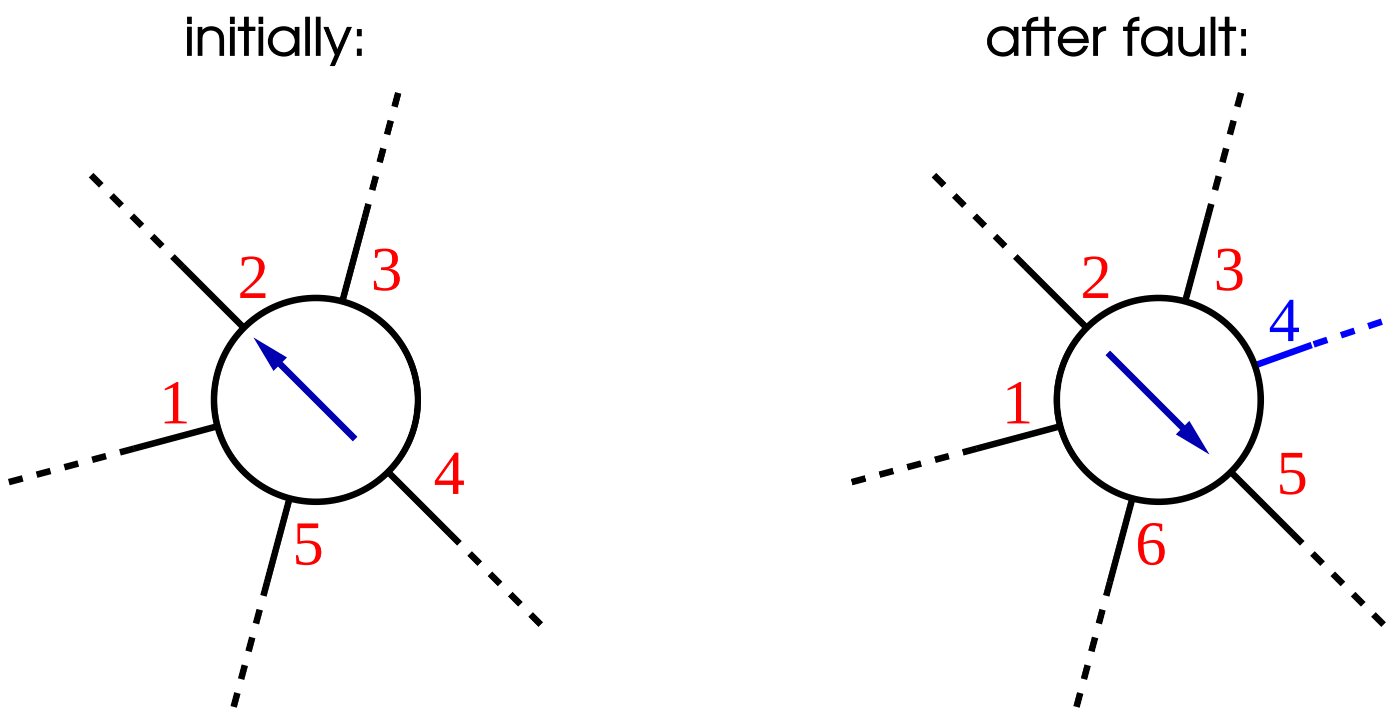
- Already known answer: the agent will enter a new Euler tour in at most $2mD$ steps
- More refined answer, e.g. in terms of number of faults?

Faults in port pointers



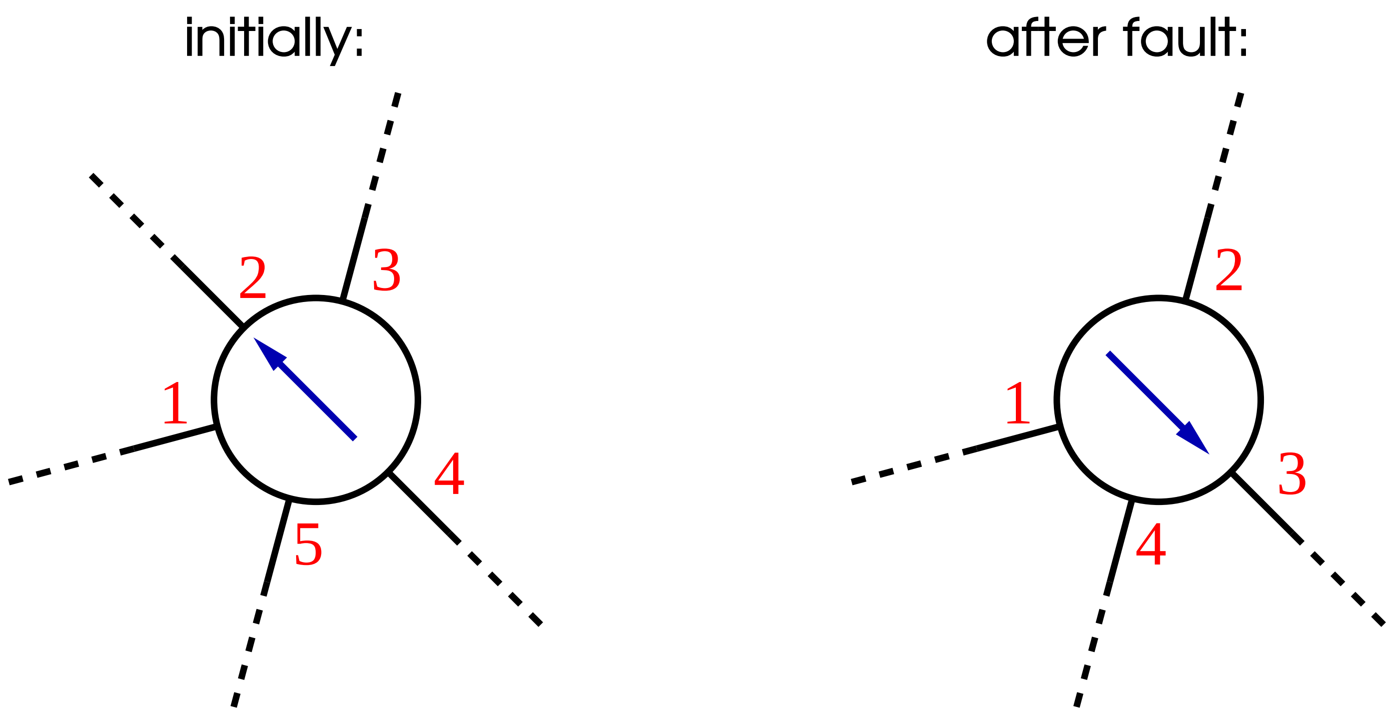
- pointers may be changed to point to arbitrary edges

Addition of new edges



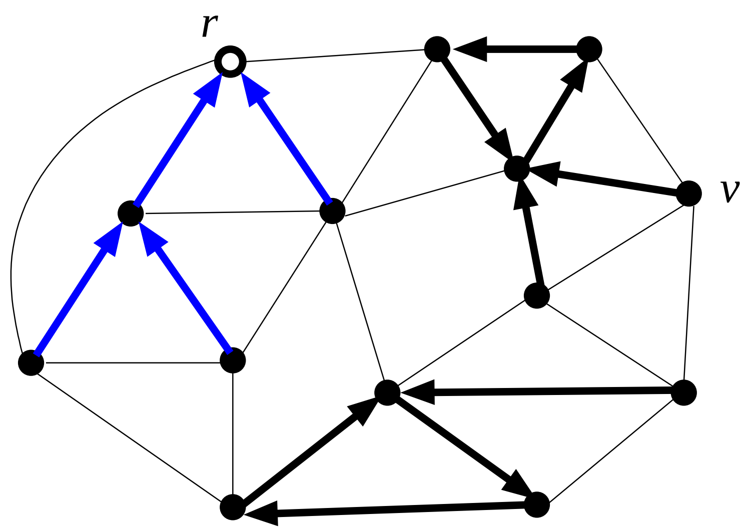
- edges may be added in arbitrary positions in the cyclic orders
- maybe combined with a pointer change

Deletion of an edge



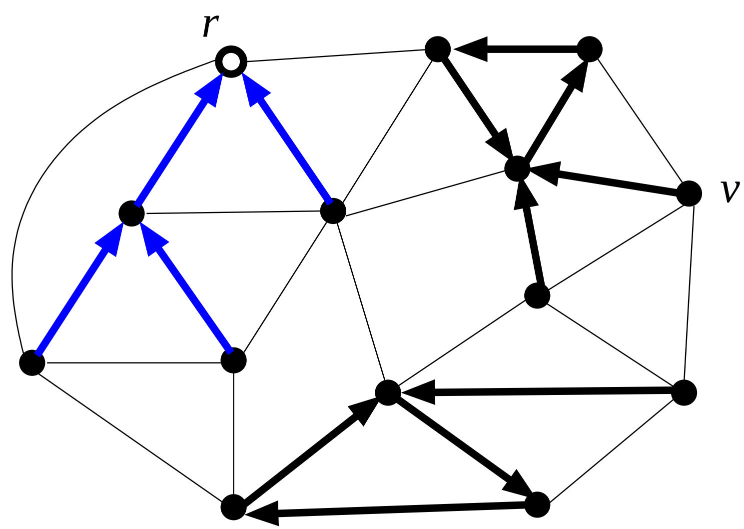
- an edge may be deleted
- maybe combined with a pointer change

Leading tree



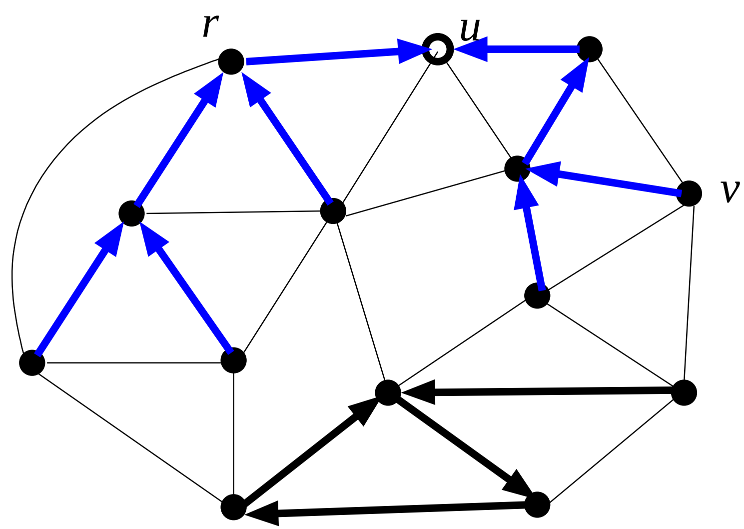
- Current set of pointers: $F = \{\pi_v : v \in V\}$
- Focus on the structure of $F_r = F \setminus \pi_r$ (r: current node)
- Component of F_r containing r is an in-bound tree rooted at r (leading tree)
- Other components: in-bound trees with root cycles

Properties of the leading tree (I)



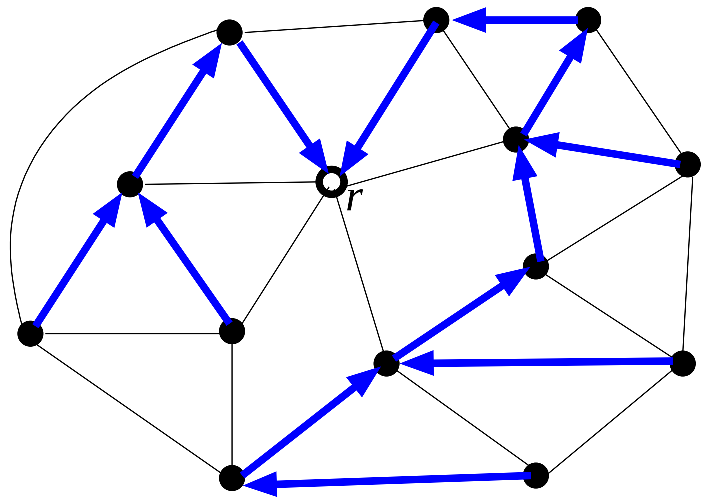
- A node never leaves the leading tree
- A node v joins the leading tree at the moment the agent visits an ancestor of v
- Thm. All nodes in the neighborhood of the leading tree are visited within the next $2m$ steps

Properties of the leading tree (I)



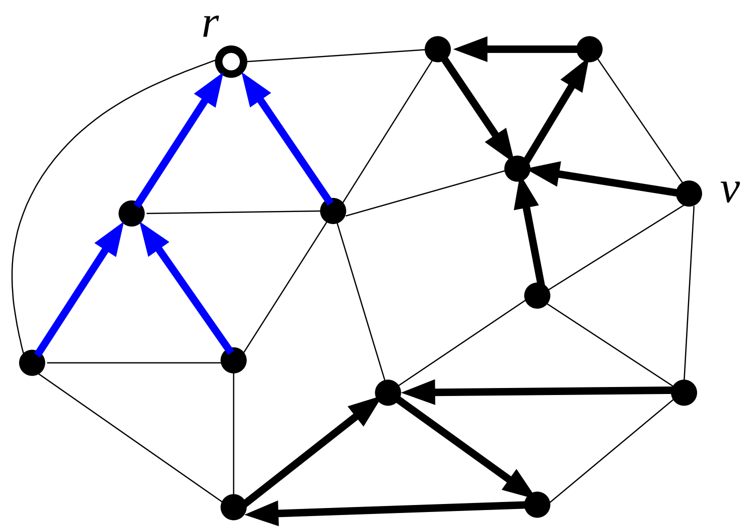
- A node never leaves the leading tree
- A node v joins the leading tree at the moment the agent visits an ancestor of v
- Thm. All nodes in the neighborhood of the leading tree are visited within the next $2m$ steps

Properties of the leading tree (II)



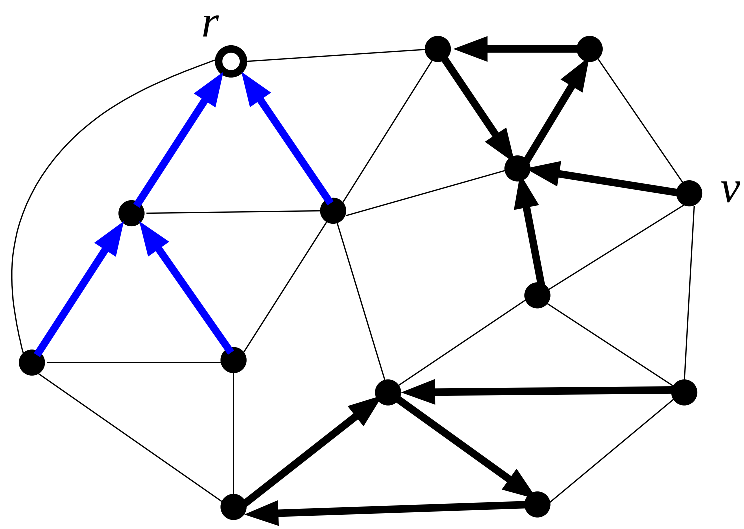
- Cor. Assuming the leading tree spans all nodes, an Euler tour is traversed in the next $2m$ steps
- The inverse also holds
- Def. Stabilization time: the first step when the leading tree spans all nodes

Main theorem

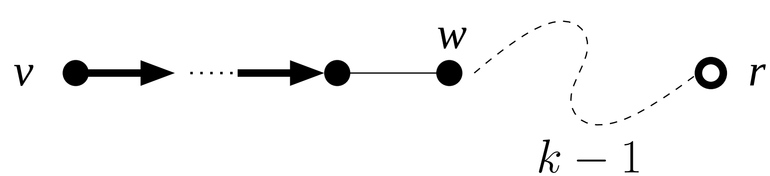


- Def. Length of an arc e is 0 iff $e \in F_r$, otherwise 1
- Thm. If the distance from a node v to the current node r is k, then node v joins the leading tree within $2km$ steps

Main theorem



- Def. Length of an arc e is 0 iff $e \in F_r$, otherwise 1
- Thm. If the distance from a node v to the current node r is k, then node v joins the leading tree within $2km$ steps

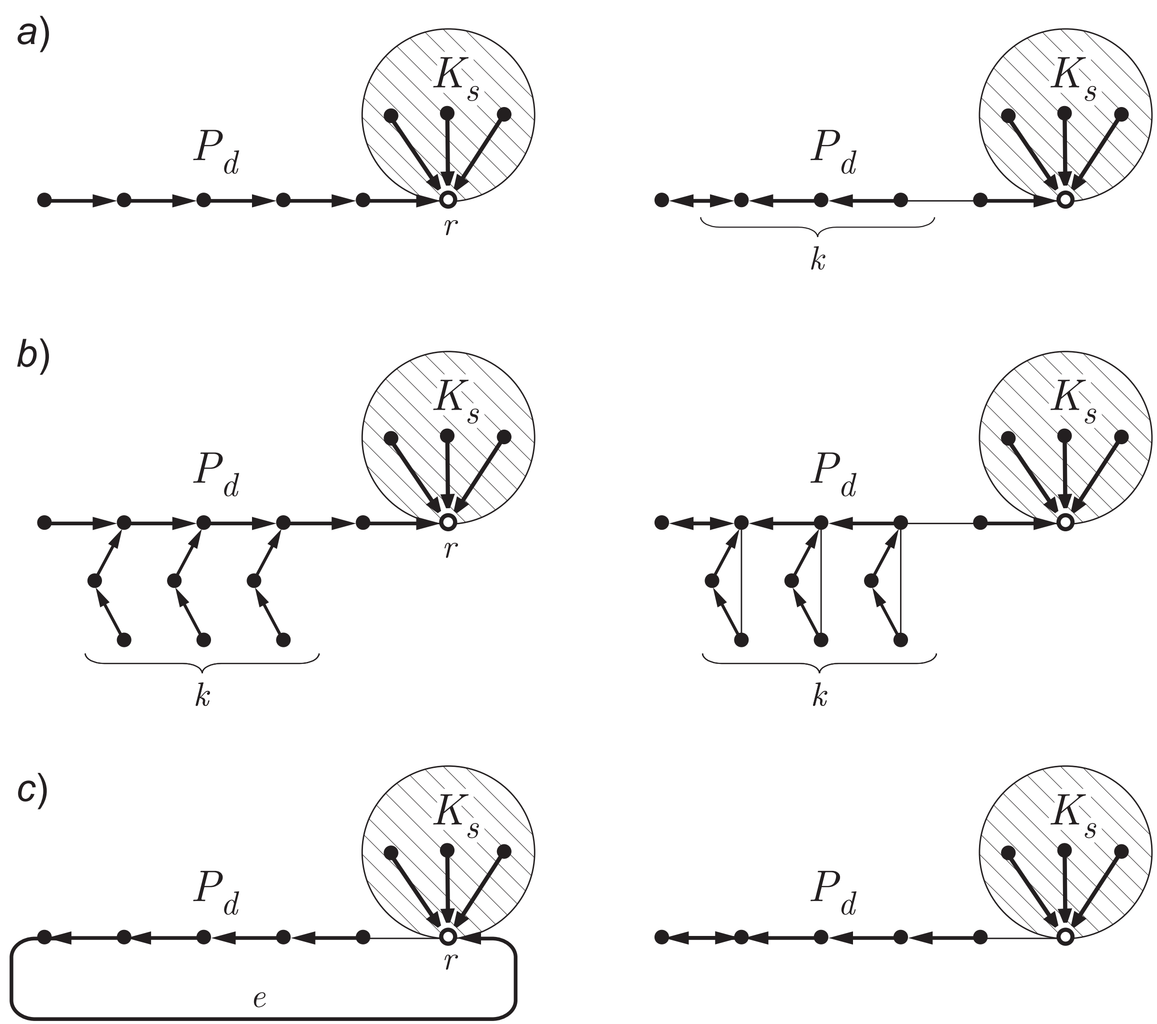


Robustness properties

Assuming the rotor-router has stabilized to some Euler tour:

- Thm. The system recovers from k pointer faults within $2m \min\{k, D\}$ steps
- Thm. The system recovers from k edge additions within $2m \min\{2k, D\}$ steps
- Thm. If the deletion of some edge e does not disconnect the graph, then the system recovers from the deletion of e within $2\gamma m$ steps (γ is the length of the smallest cycle containing e)

Lower bounds



Conclusions and open problems

- Structural properties of the rotor-router
- Application to bounding recovery time from faults/dynamic changes
- Analyze different fault models? (e.g., changes in local port orders)
- Extend the leading tree concept to the case of multiple agents?

Conclusions and open problems

- Structural properties of the rotor-router
- Application to bounding recovery time from faults/dynamic changes
- Analyze different fault models? (e.g., changes in local port orders)
- Extend the leading tree concept to the case of multiple agents?

Thank you