Distributed Chasing of Network Intruders

Lélia Blin[1] Pierre Fraigniaud[2] Nicolas Nisse[3]
Sandrine Vial[1]

[1]Université d'Evry, France

[2]CNRS, Université Paris-Sud, France

[3]Université Paris-Sud, France.

SIROCCO 06, Chester, 2006

Graph Searching

Goal

In a given network,

- whose edges are contaminated ;
- a team of searchers aims at clearing the network.

We want to find a strategy that clears the netwok
using the fewest searchers as possible.

Motivations

- network security,
- speleological rescue,
- ...

Graph Searching

Goal

In a given network,

- whose edges are contaminated ;
- a team of searchers aims at clearing the network.

We want to find a strategy that clears the netwok
using the fewest searchers as possible.

Motivations

- network security,
- speleological rescue,
- ...

Search Strategy, Parson. [GTC,1978]

Sequence of three basic operations,. . .

1. Place a searcher at a vertex of the graph ;
2. Move a searcher along an edge of the graph ;
3. Remove a searcher from a vertex of the graph.

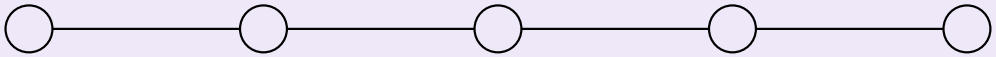
. . . that must result in clearing the graph

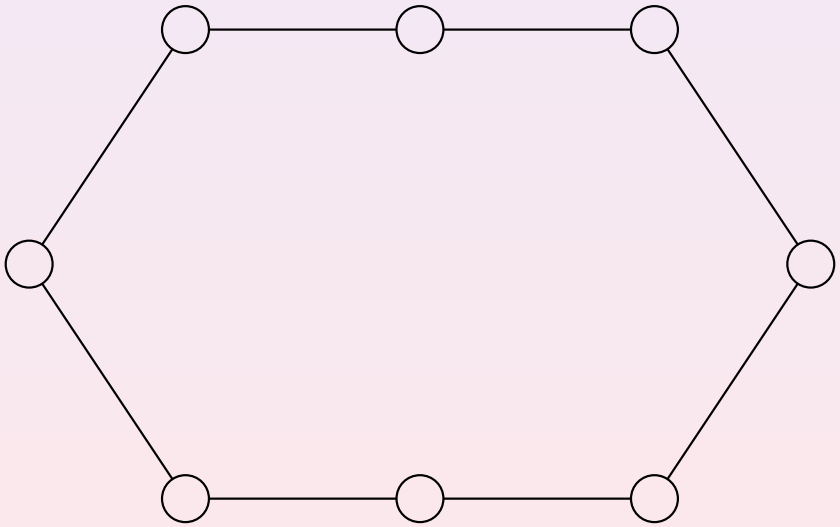
An edge is cleared when it is traversed by a searcher.

We want to minimize the number of searchers.

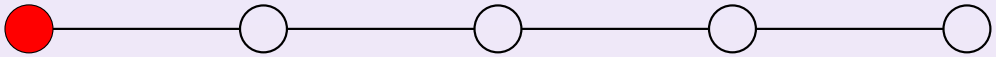
Let $s(G)$ be the smallest number of searchers required to clear a graph G.

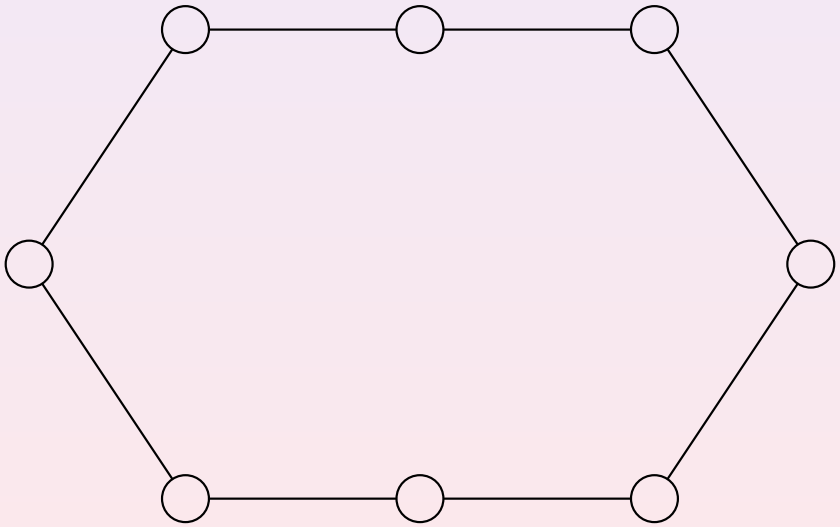
Simple Examples : Path and Ring



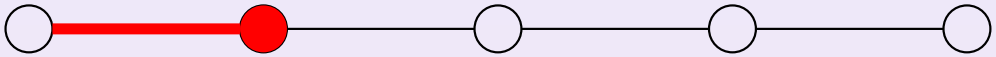


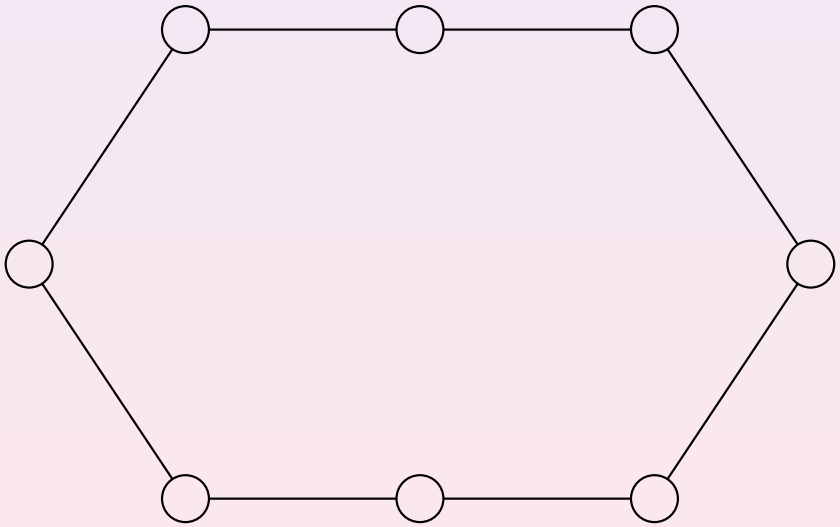
Simple Examples : Path and Ring



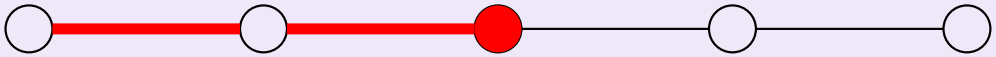


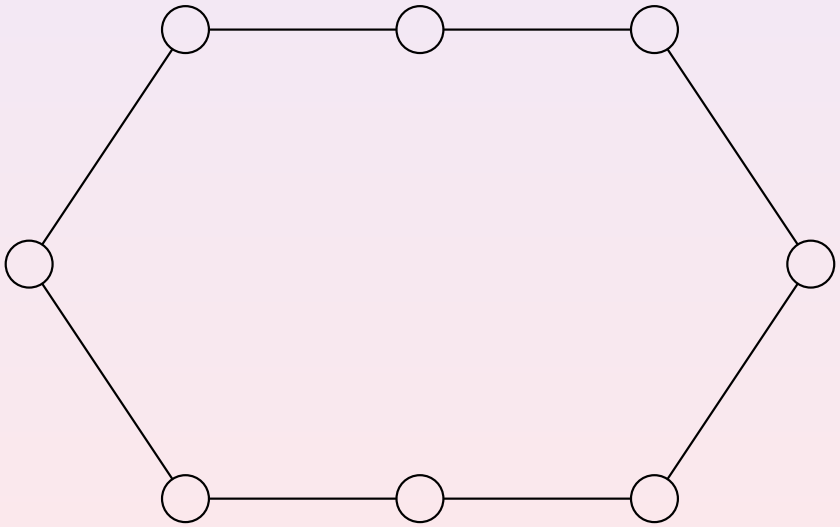
Simple Examples : Path and Ring



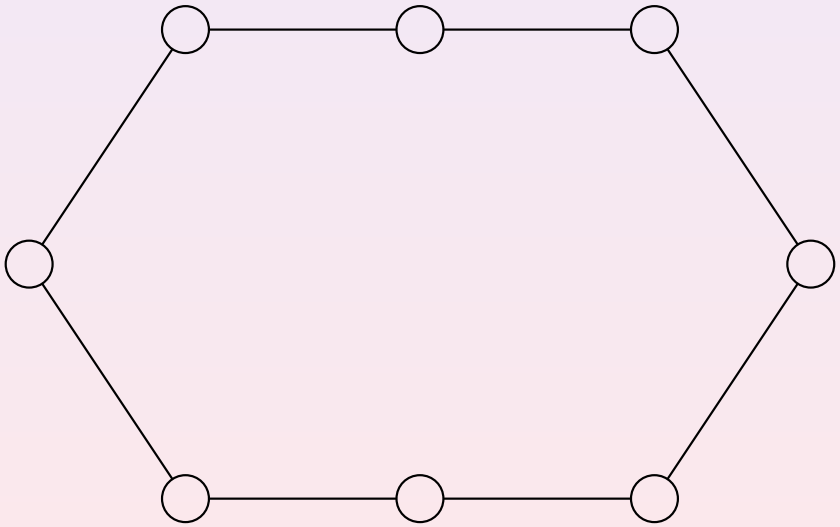


Simple Examples : Path and Ring



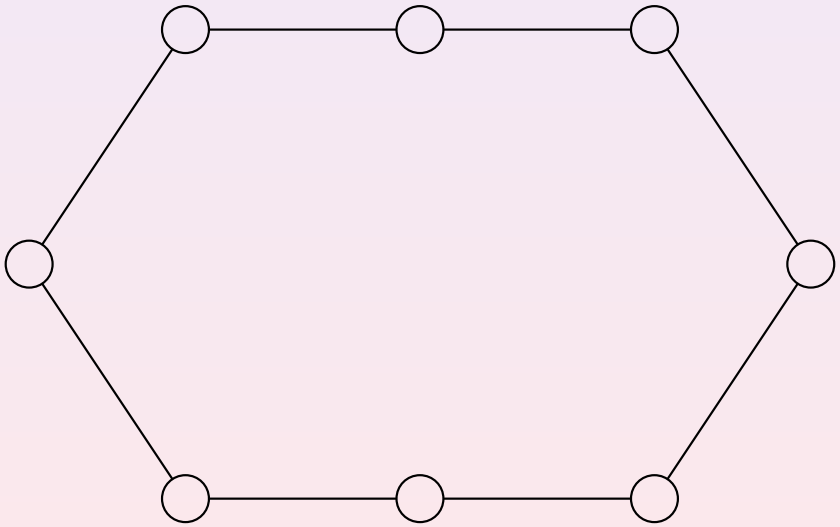


Simple Examples : Path and Ring



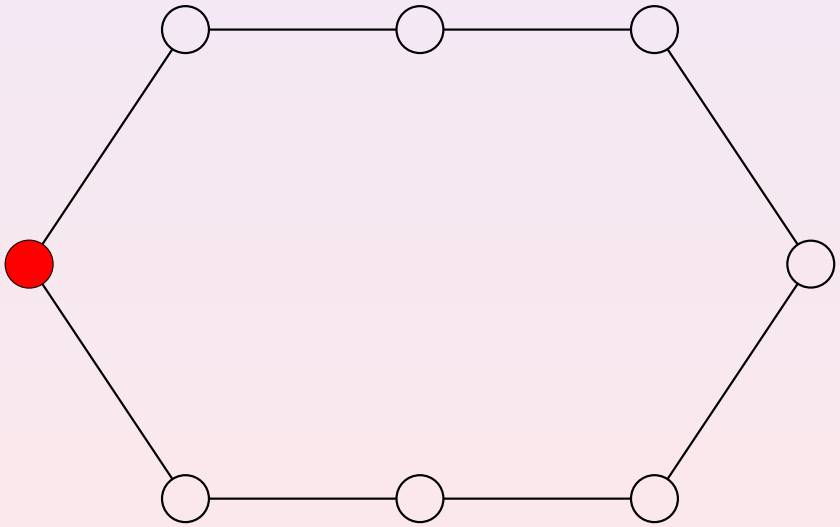
Simple Examples : Path and Ring

s(Path)=1



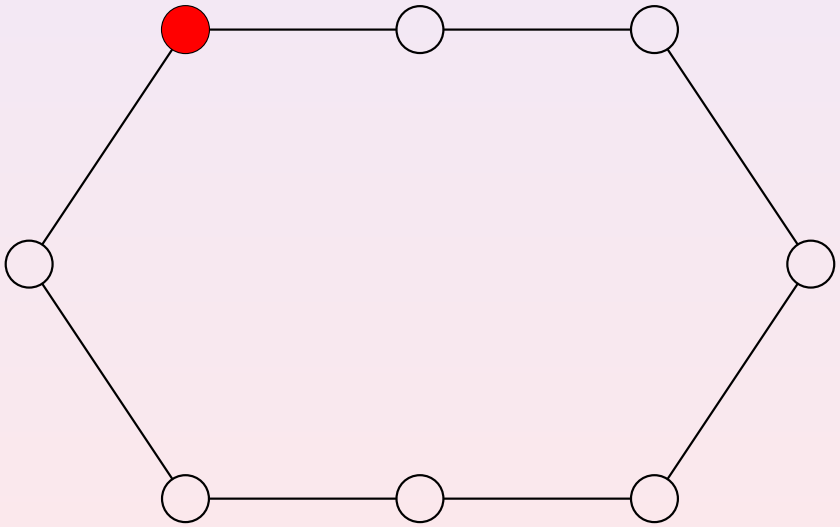
Simple Examples : Path and Ring

s(Path)=1



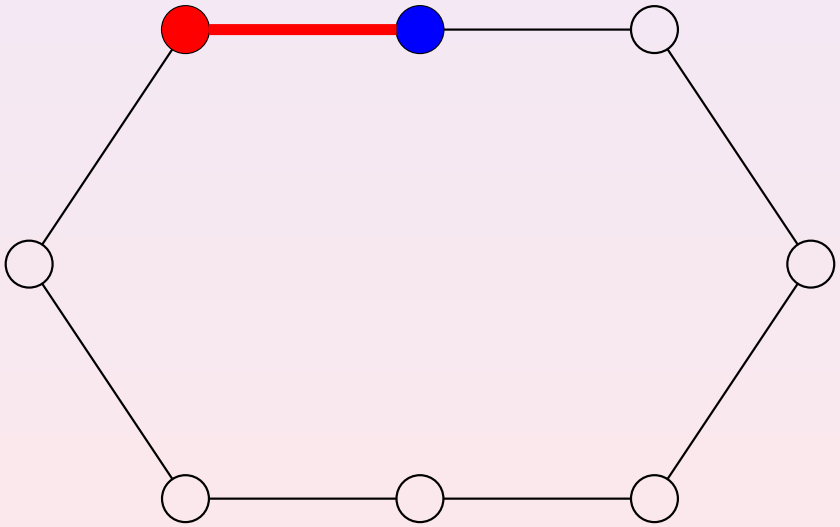
Simple Examples : Path and Ring

s(Path)=1



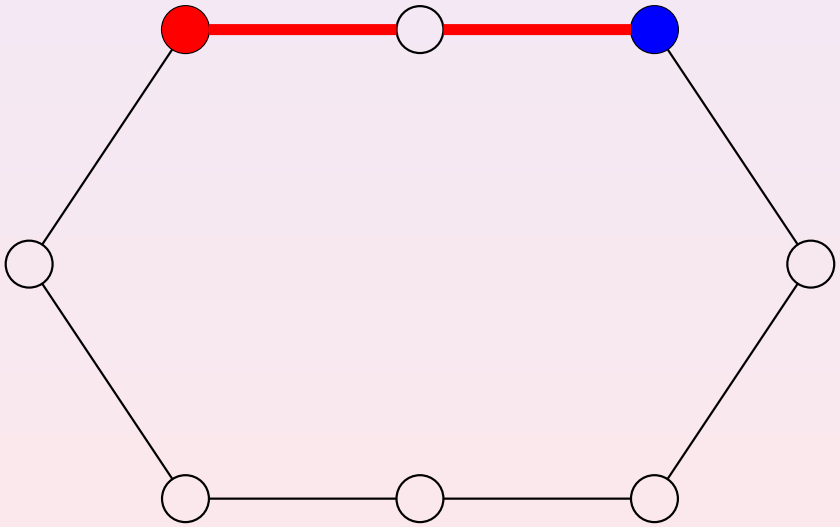
Simple Examples : Path and Ring

s(Path)=1



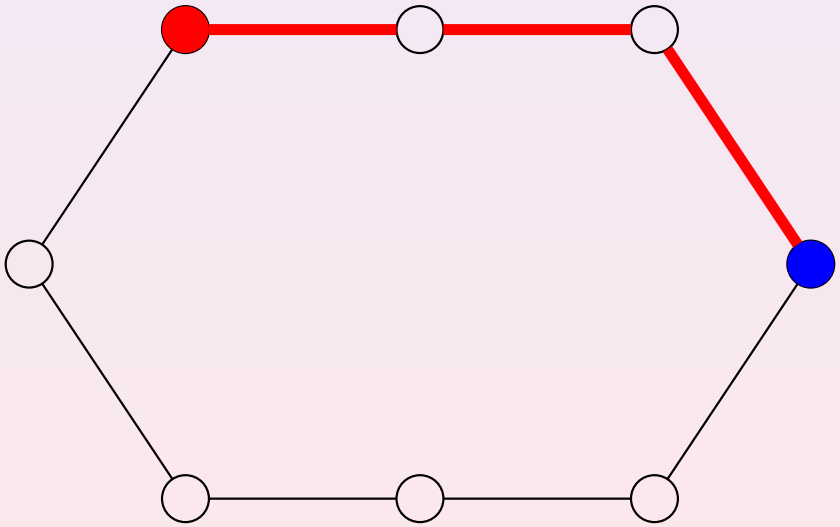
Simple Examples : Path and Ring

s(Path)=1



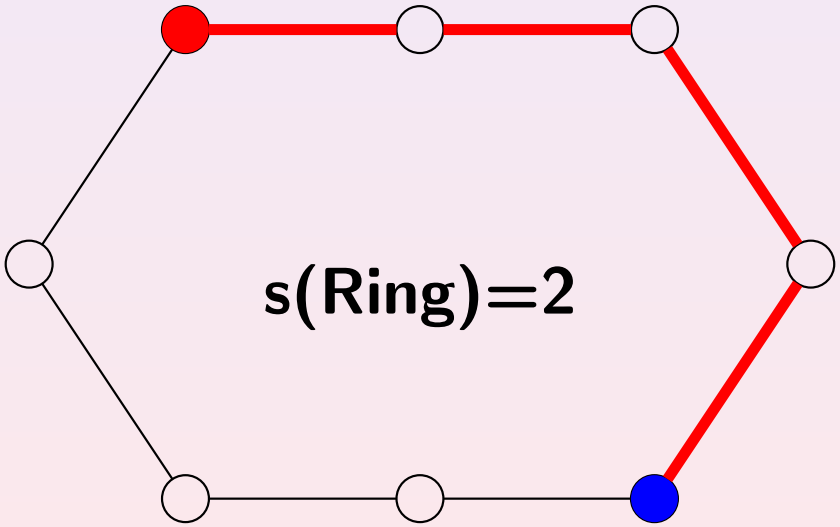
Simple Examples : Path and Ring

s(Path)=1



Simple Examples : Path and Ring

s(Path)=1



s(Ring)=2

Drawbacks of Parson's model :

In the standard settings :

1. the topology of the graph is *known* ;
2. a search strategy is performed in a sequential *synchronous* way ;
3. searchers can be placed anywhere in the graph.

In a real network :

1. searchers have no knowledge about the topology ;
2. networks are asynchronous ;
3. searchers cannot be teleported,
   communications must be secure,
   $\Rightarrow$ the clear part must induce a connected subgraph.

Monotone connected graph searching (1)

Monotone connected search strategy

- **connectedness :** At any step of the strategy, the clear part must induce a connected subgraph.
- **monotony :** No recontamination ever occurs. Once an edge has been cleared, it remains clear until the end.

Monotone connected search number

Let **mcs**(G) be the smallest number of searchers required to clear the graph in a monotone connected manner.

Related Works : Cost of connectedness

In terms of number of searchers

- For any tree T, $\mathbf{s}(T) \leq \mathbf{mcs}(T) \leq 2\,\mathbf{s}(T) - 2$ (tight).
  Barrière, Fraigniaud, Santoro and Thilikos. [WG, 2003]
- For any graph G, $\mathbf{s}(G) \leq \mathbf{mcs}(G) \leq (1 + \log n)\,\mathbf{s}(G)$
  Fraigniaud and Nisse [LATIN, 2006]

Polynomial algorithms in specific topologies

- Trees.
  Barrière, Flocchini, Fraigniaud and Santoro. [SPAA, 2002]
- Hypercubes.
  Flocchini, Huang and Luccio. [IPDPS, 2005]
- Chordal rings, Tori, Meshes...

Related Works : Cost of connectedness

In terms of number of searchers

- For any tree T, $\mathbf{s}(T) \leq \mathbf{mcs}(T) \leq 2\,\mathbf{s}(T) - 2$ (tight).
  Barrière, Fraigniaud, Santoro and Thilikos. [WG, 2003]
- For any graph G, $\mathbf{s}(G) \leq \mathbf{mcs}(G) \leq (1 + \log n)\,\mathbf{s}(G)$
  Fraigniaud and Nisse [LATIN, 2006]

Polynomial algorithms in specific topologies

- Trees.
  Barrière, Flocchini, Fraigniaud and Santoro. [SPAA, 2002]
- Hypercubes.
  Flocchini, Huang and Luccio. [IPDPS, 2005]
- Chordal rings, Tori, Meshes...

Our Result

For any connected, asynchronous, and anonymous network G, and any $v_0 \in V(G)$, we propose a distributed algorithm that enables clearing G using searchers starting from v_0, and initially unaware of G.

Monotone connected graph searching (2)

Alternative definition

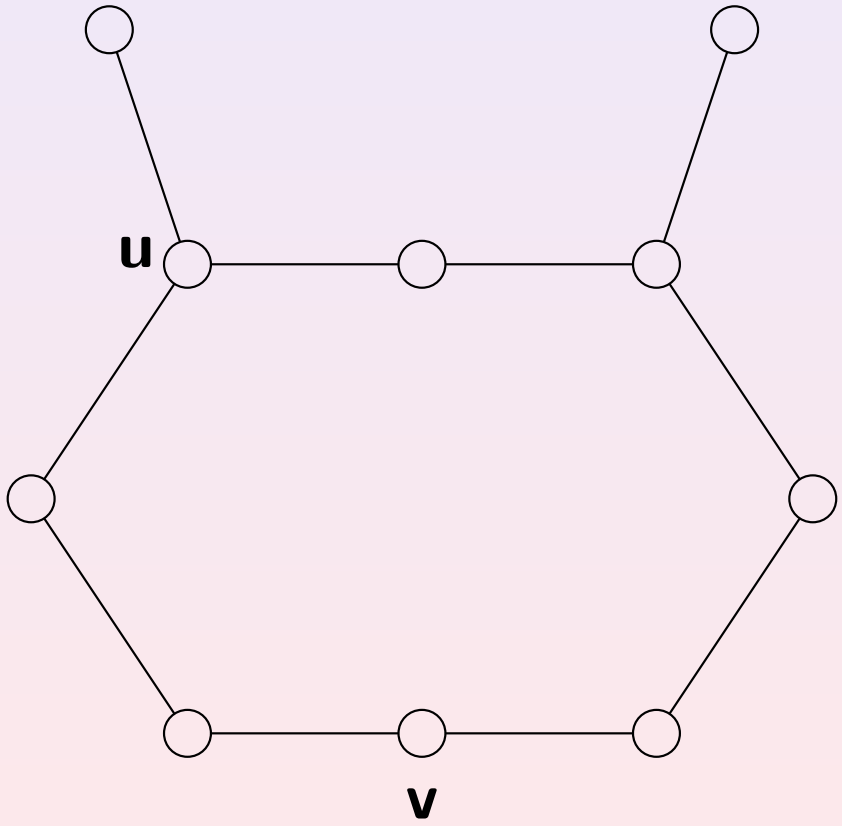
$v_0 \in V(G)$ is the homebase of the searchers.

- Initially, any searcher is placed at v_0.
- One single operation is allowed : move a searcher along an edge if it does not implie recontamination.

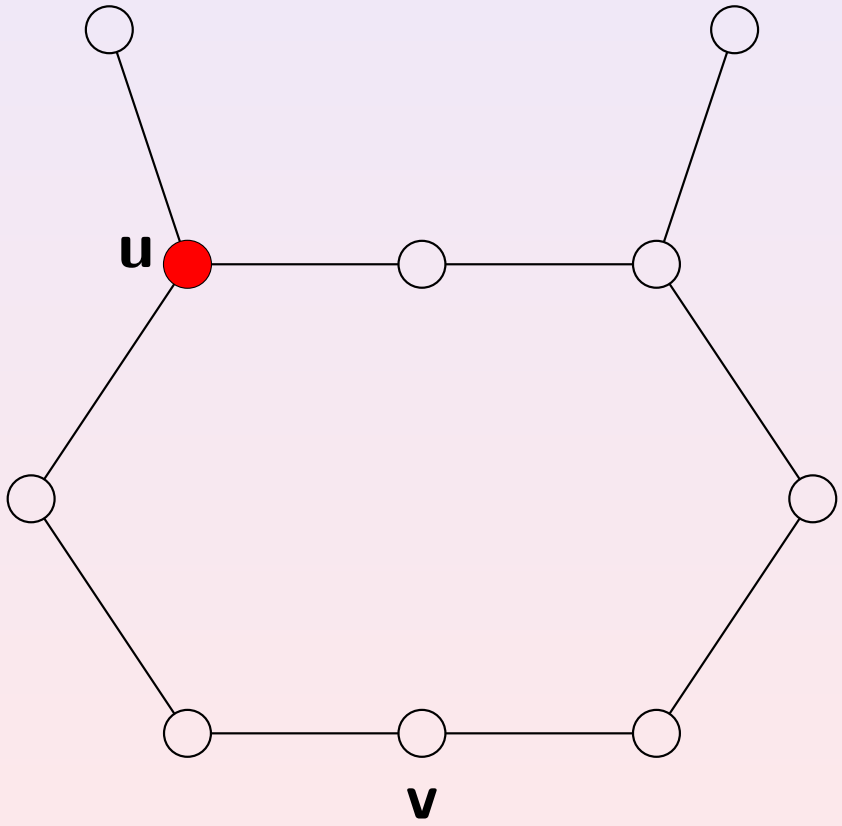
Remark

The homebase remains clear during the whole strategy.

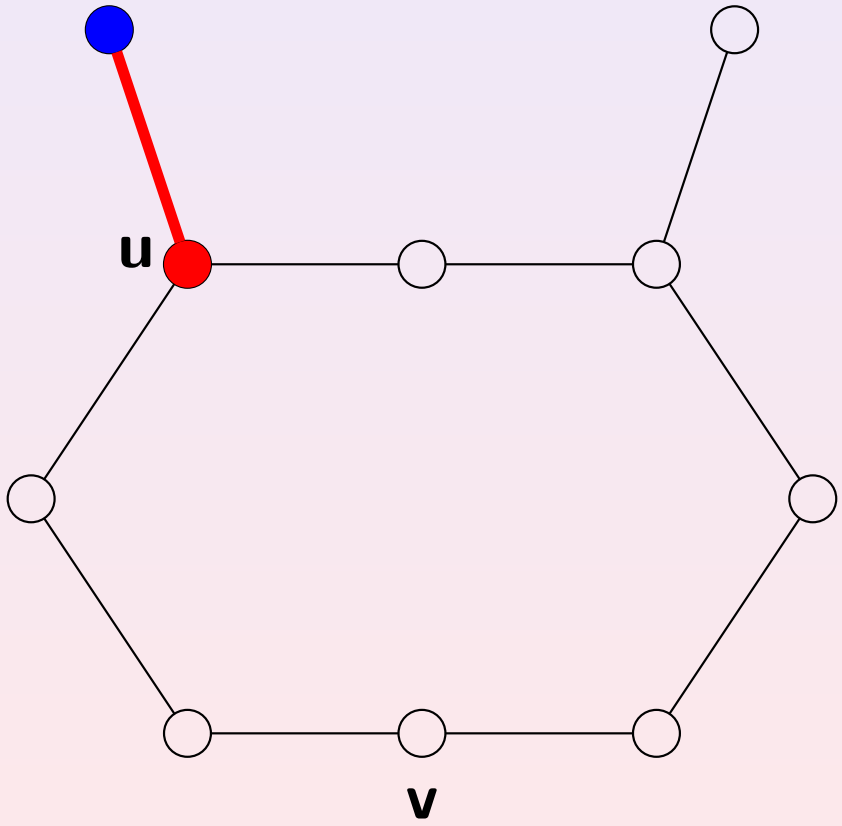
A simple example



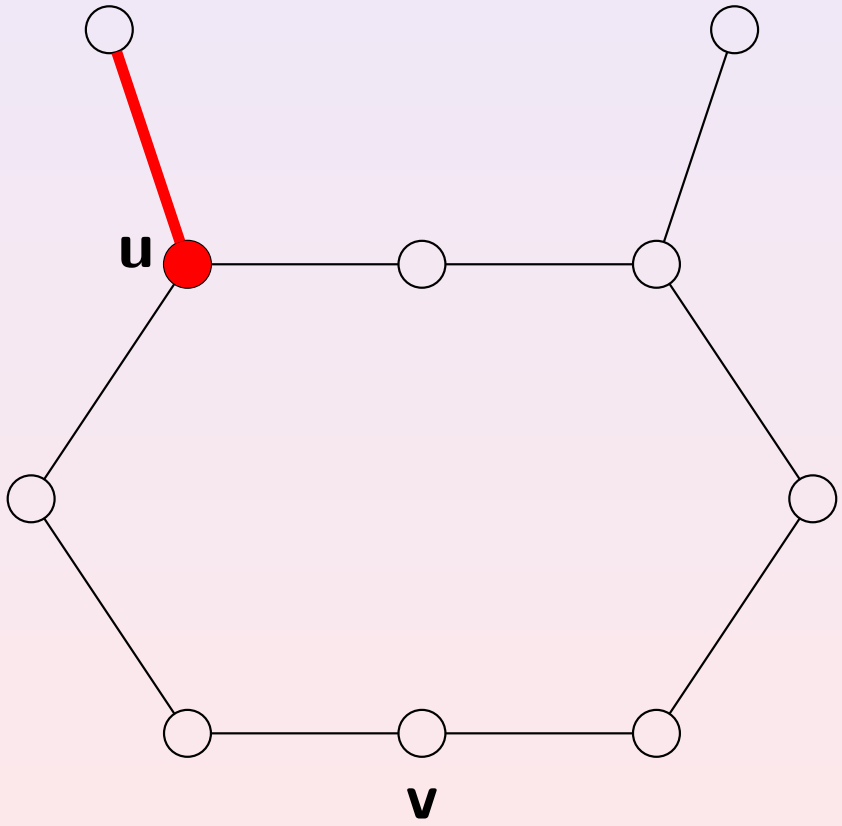
A simple example



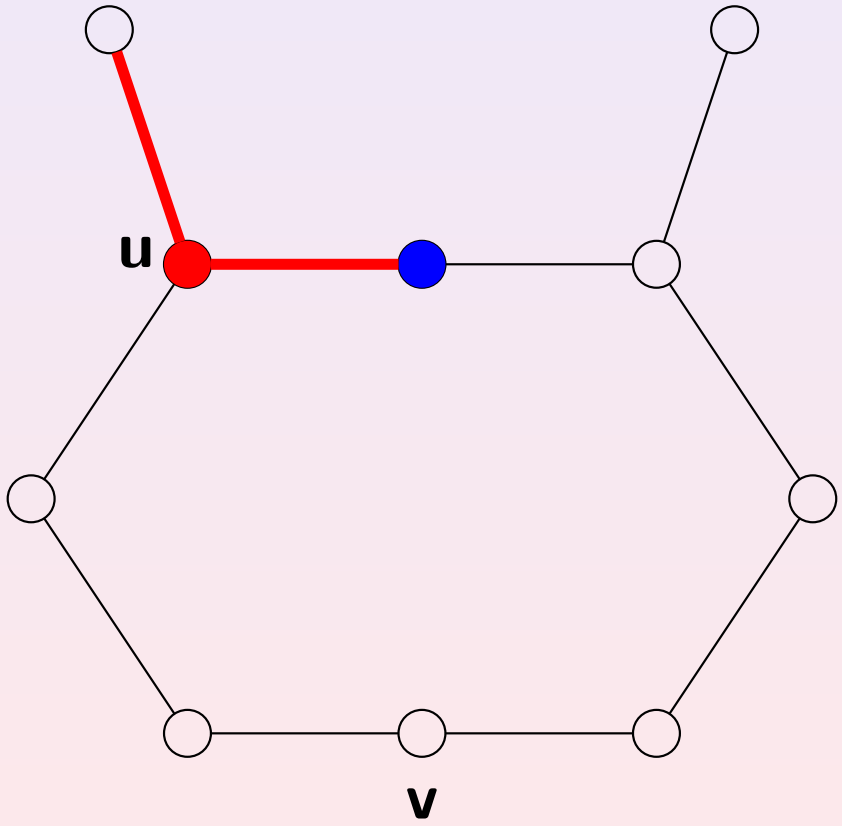
A simple example



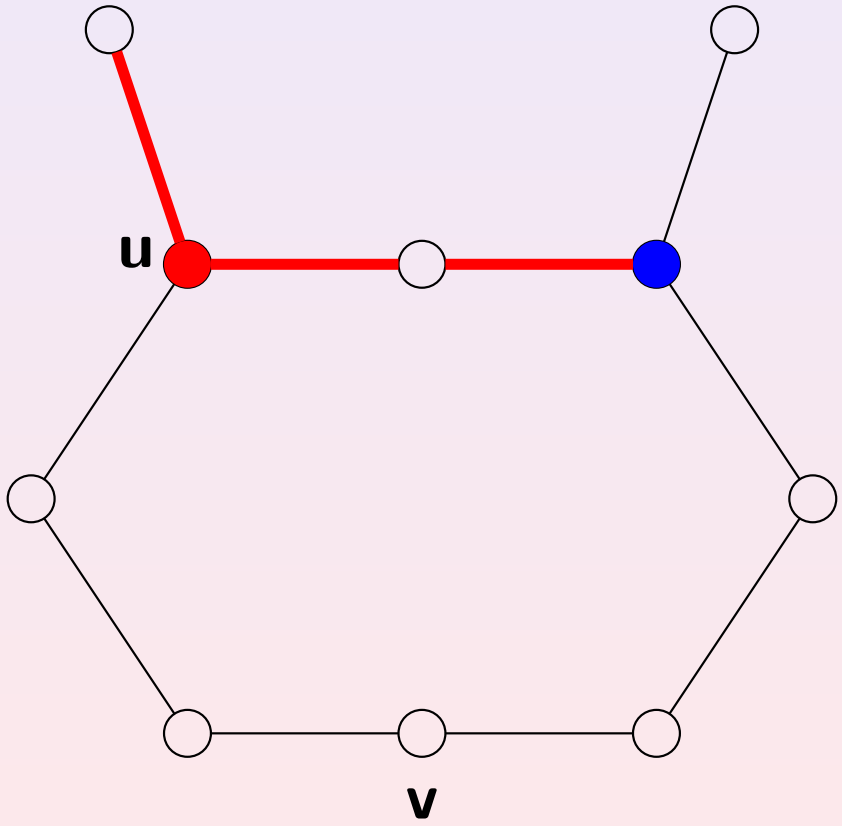
A simple example



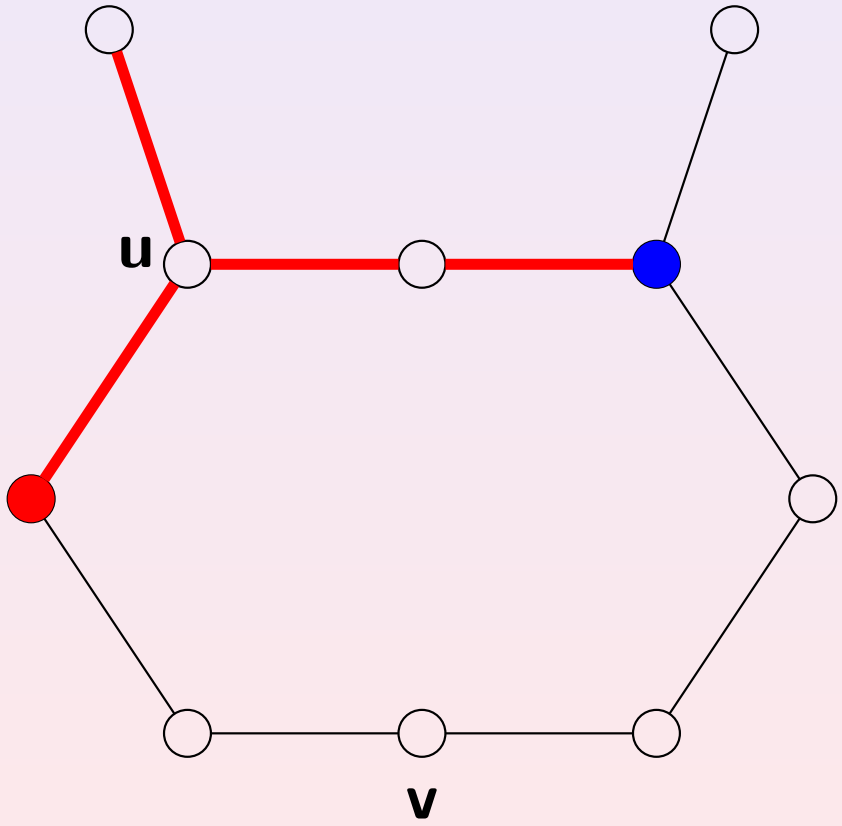
A simple example



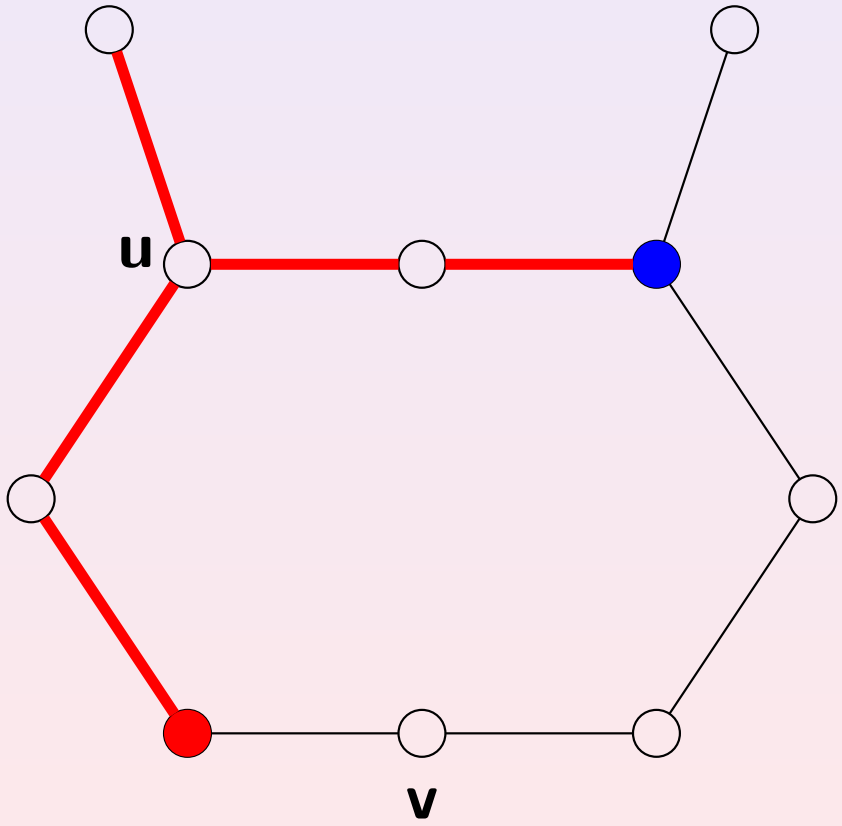
A simple example



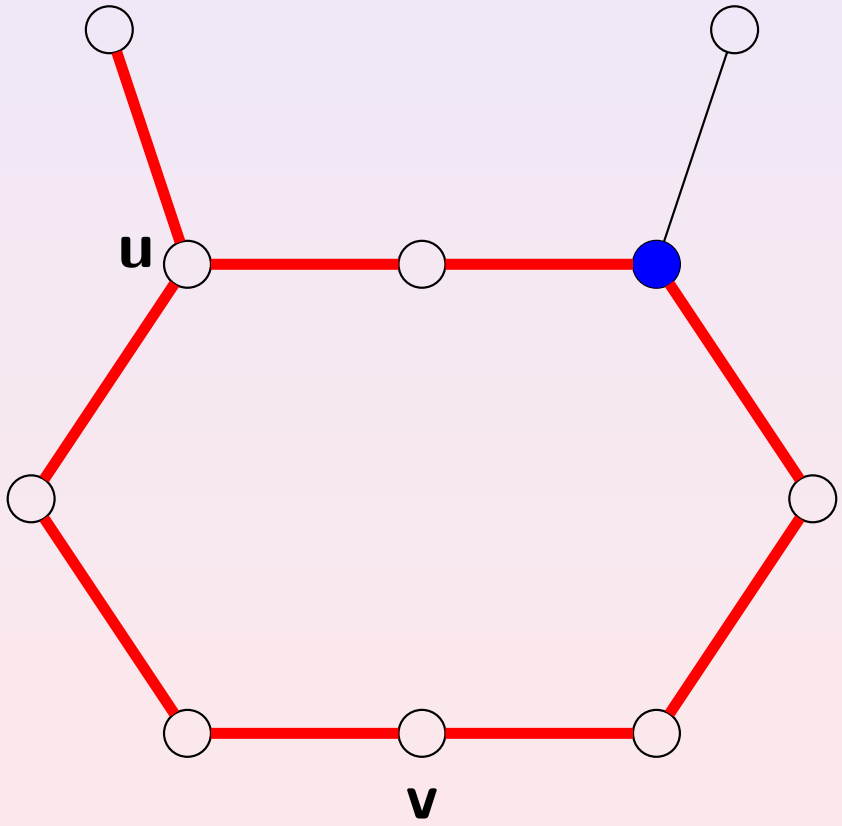
A simple example



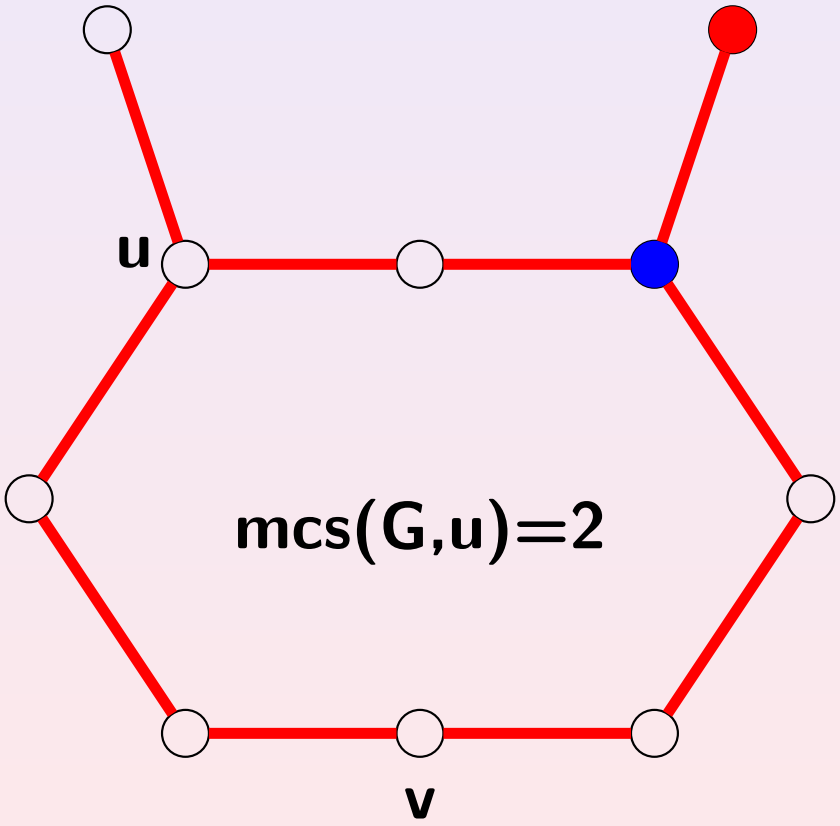
A simple example



A simple example

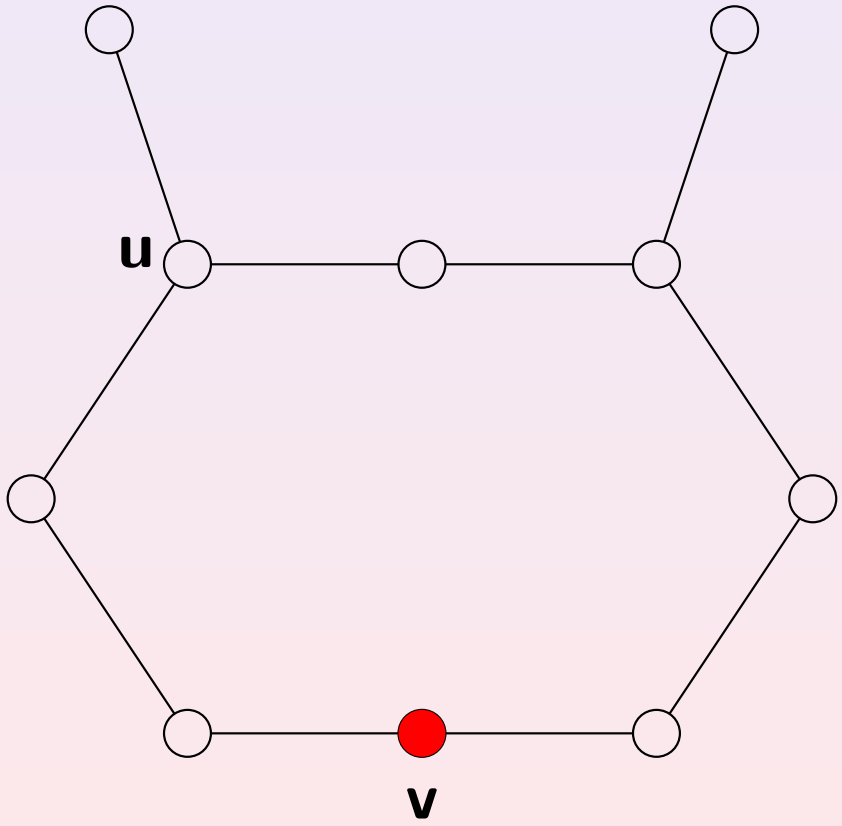


A simple example



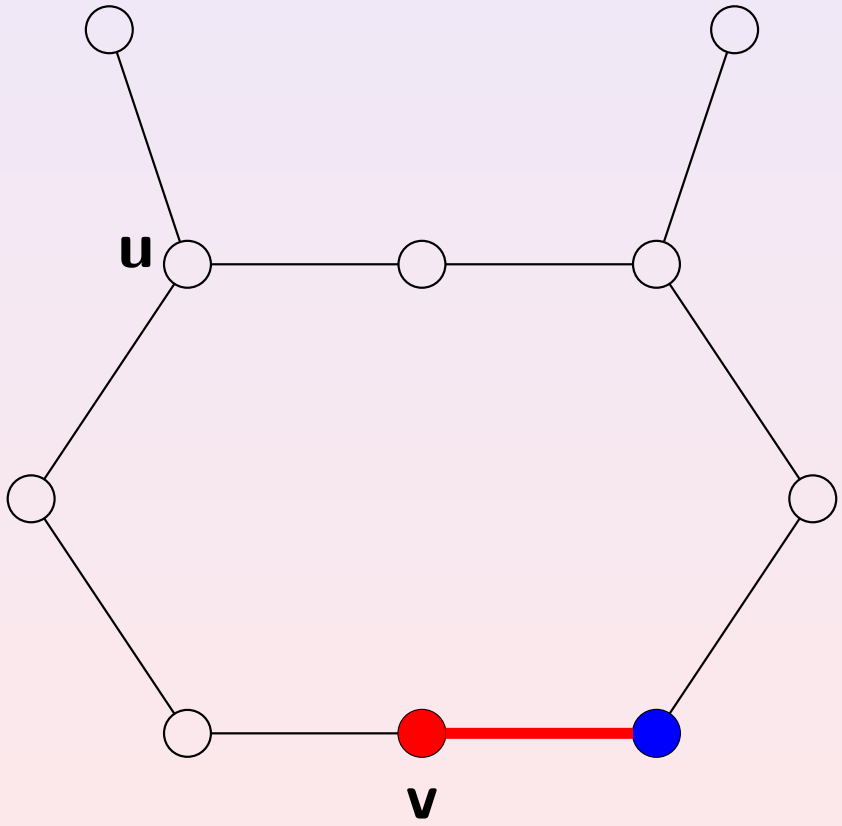
A simple example

Importance of the homebase



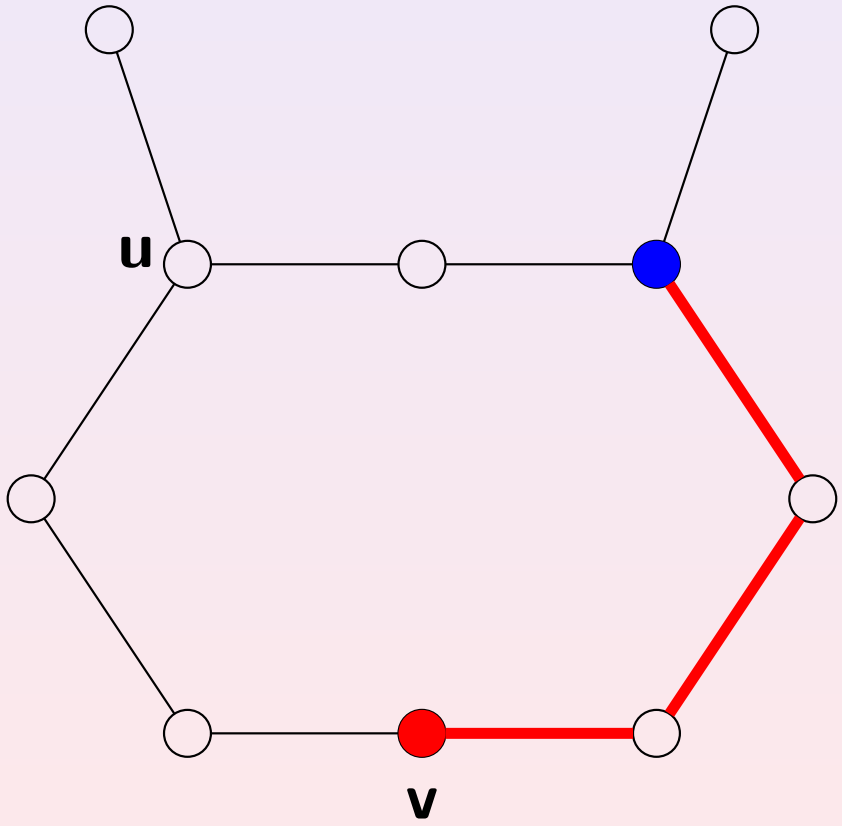
A simple example

Importance of the homebase



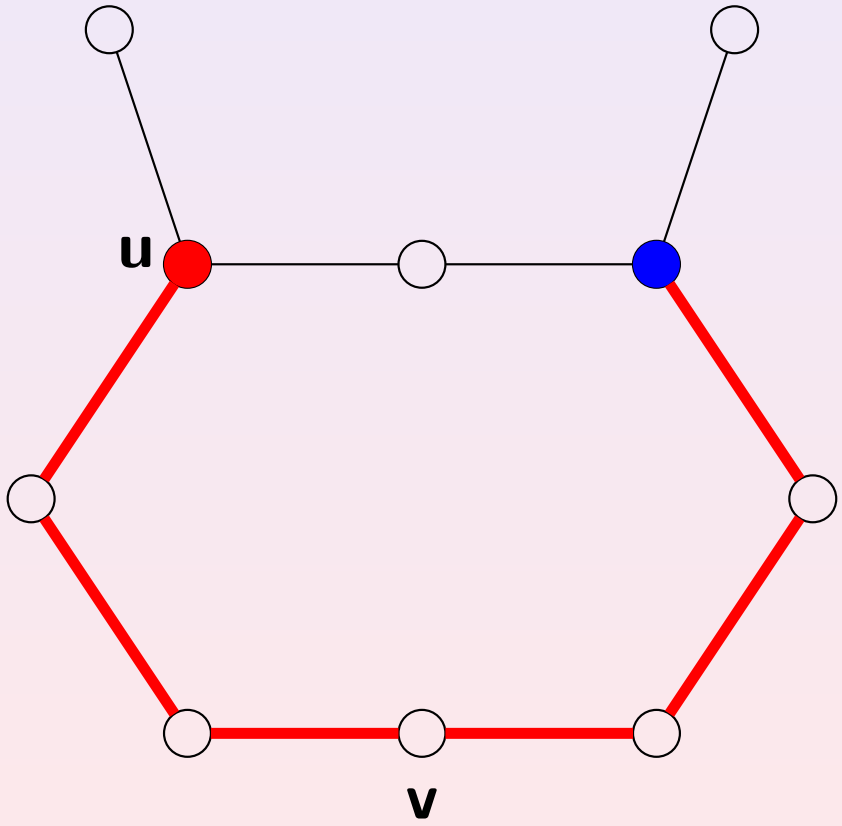
A simple example

Importance of the homebase



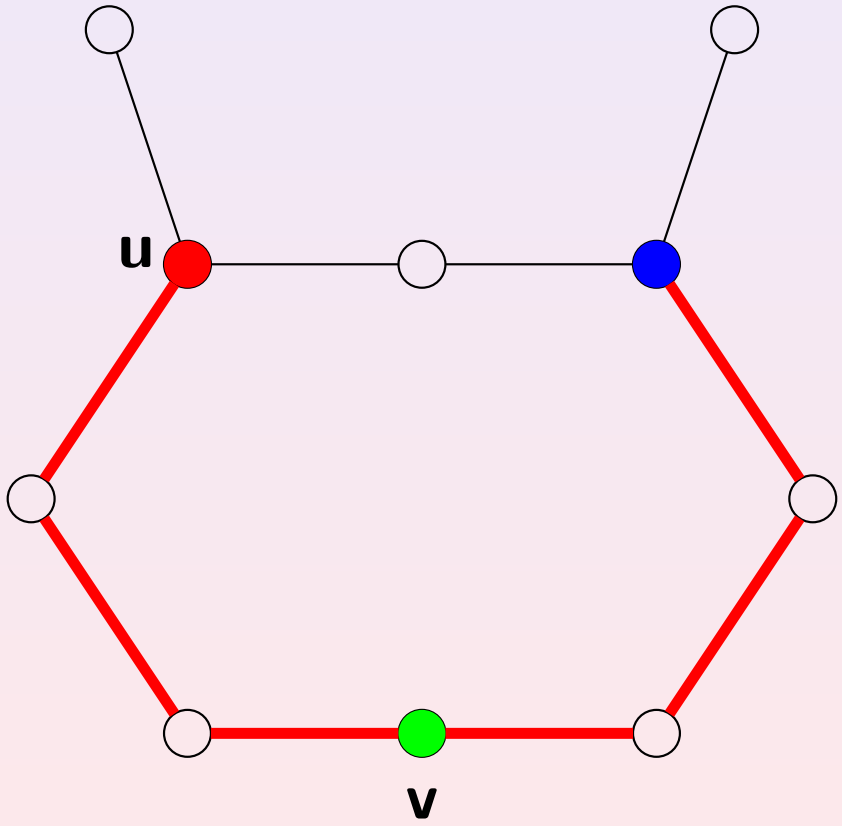
A simple example

Importance of the homebase



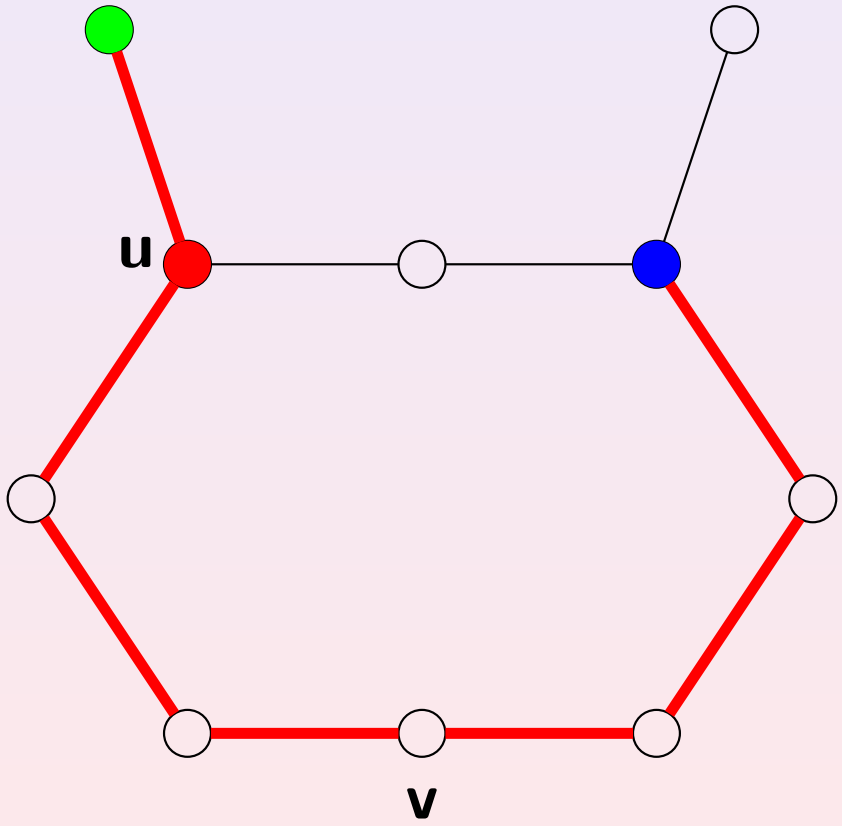
A simple example

Importance of the homebase



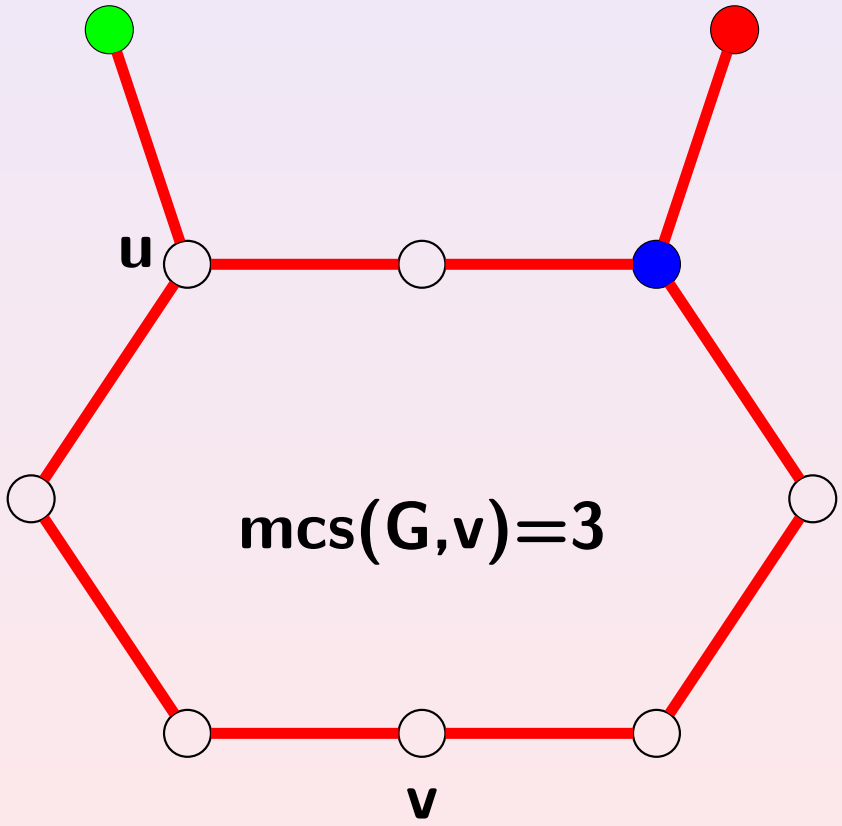
A simple example

Importance of the homebase

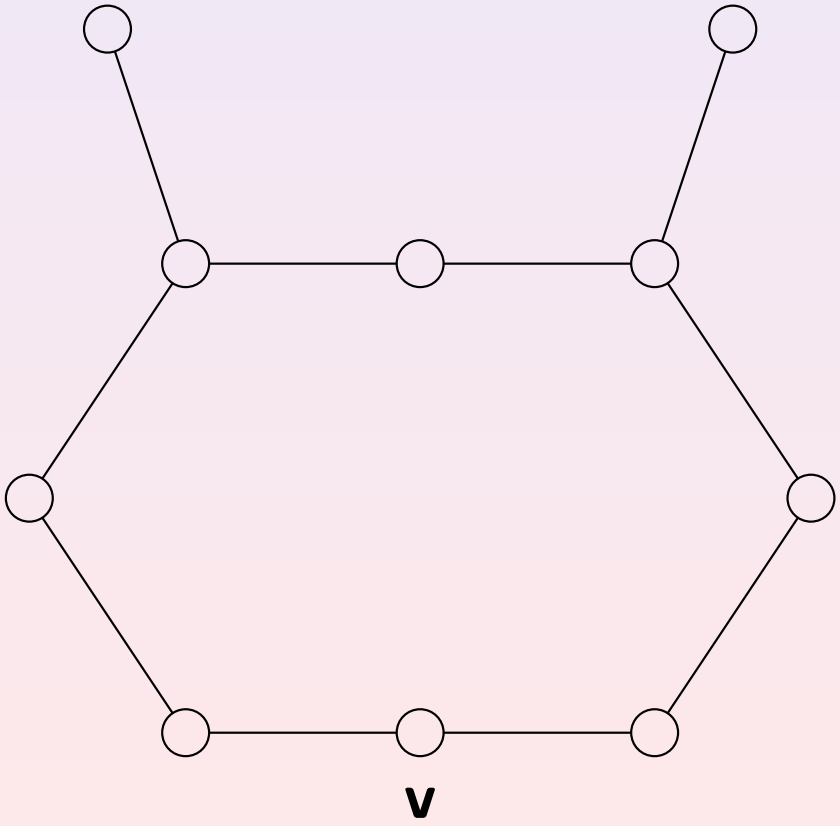


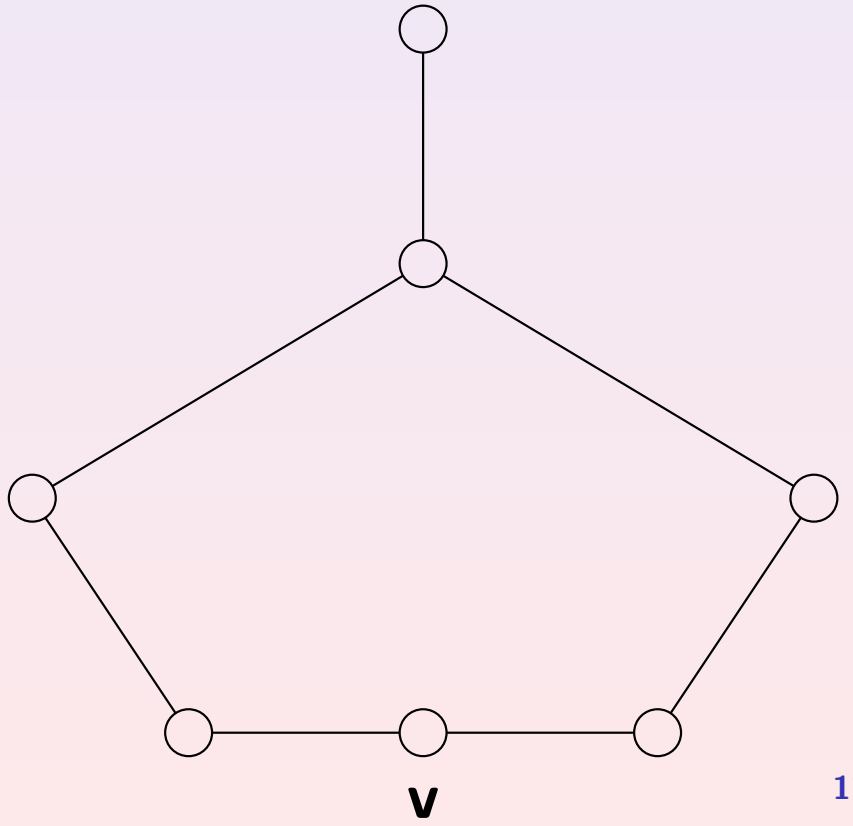
A simple example

Importance of the homebase

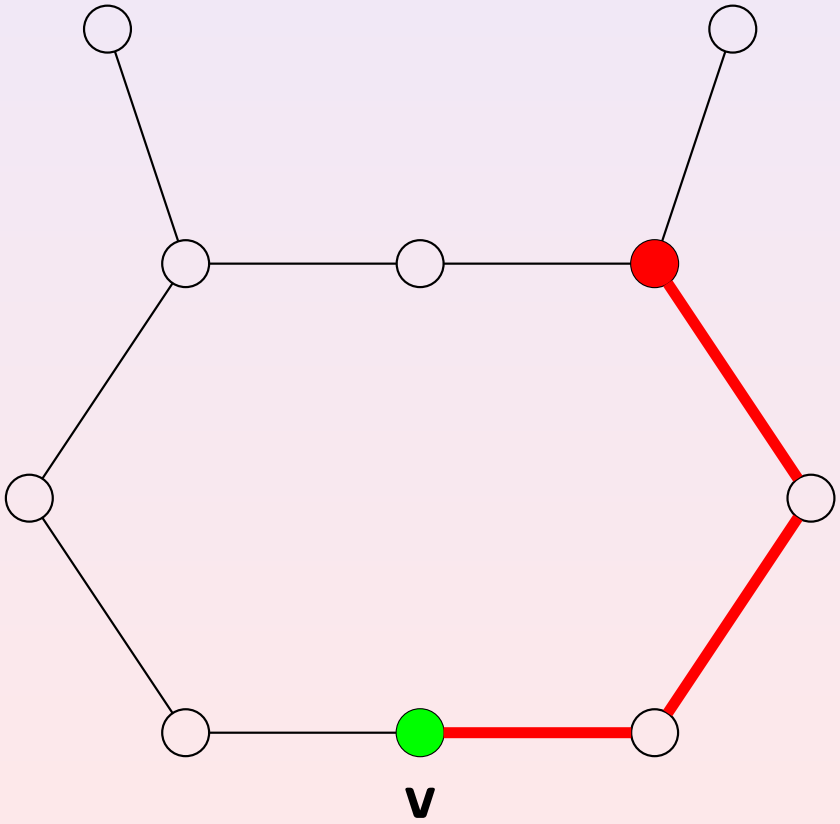


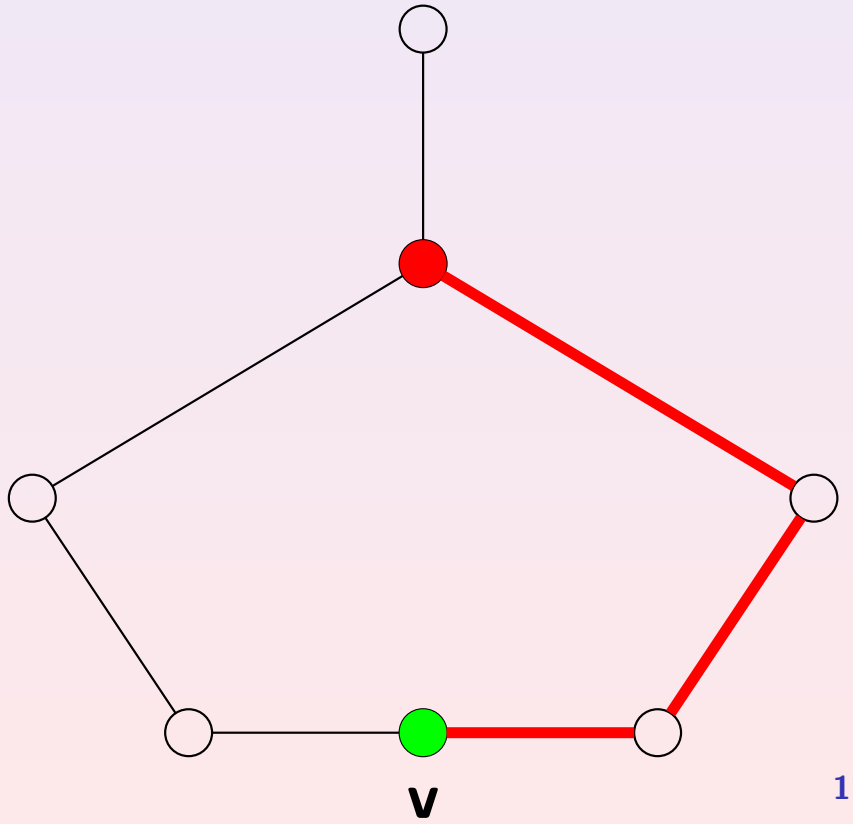
Cost of asynchronicity



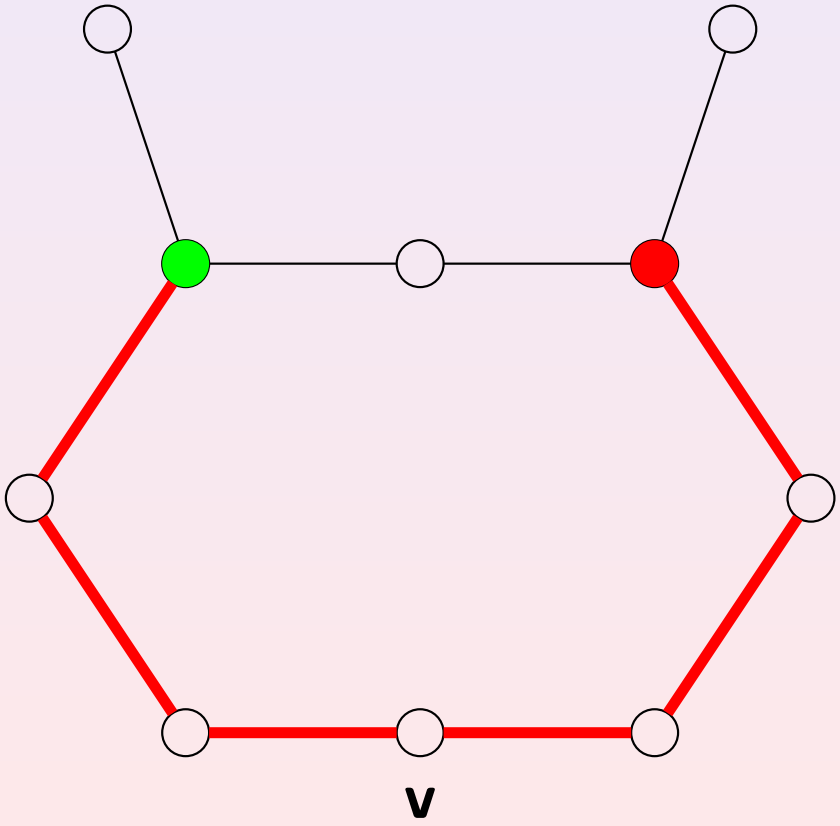


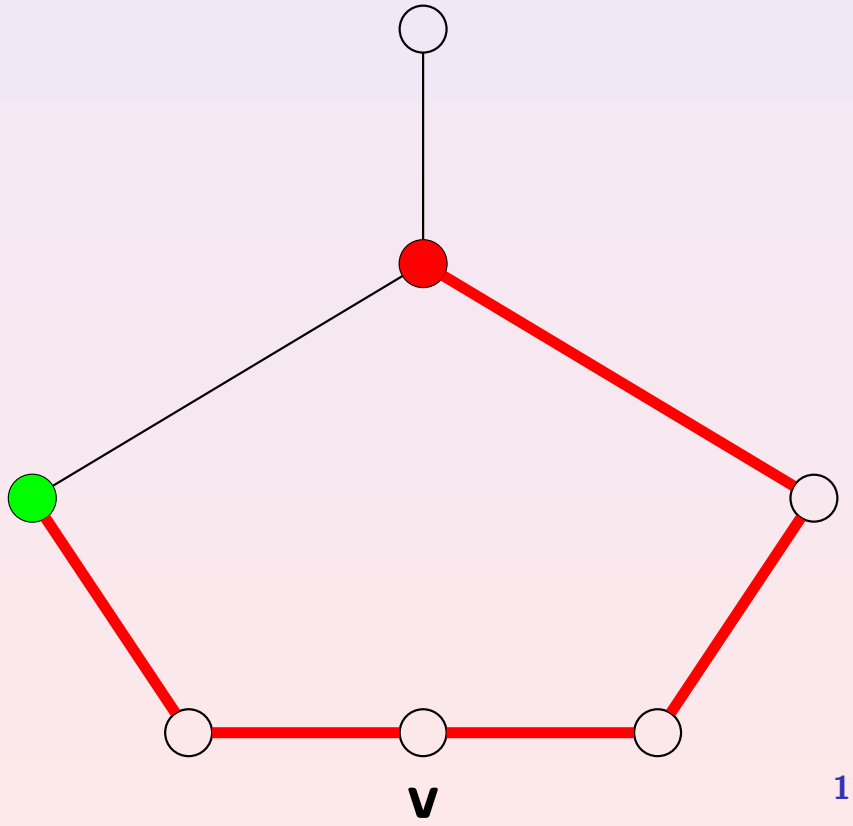
Cost of asynchronicity





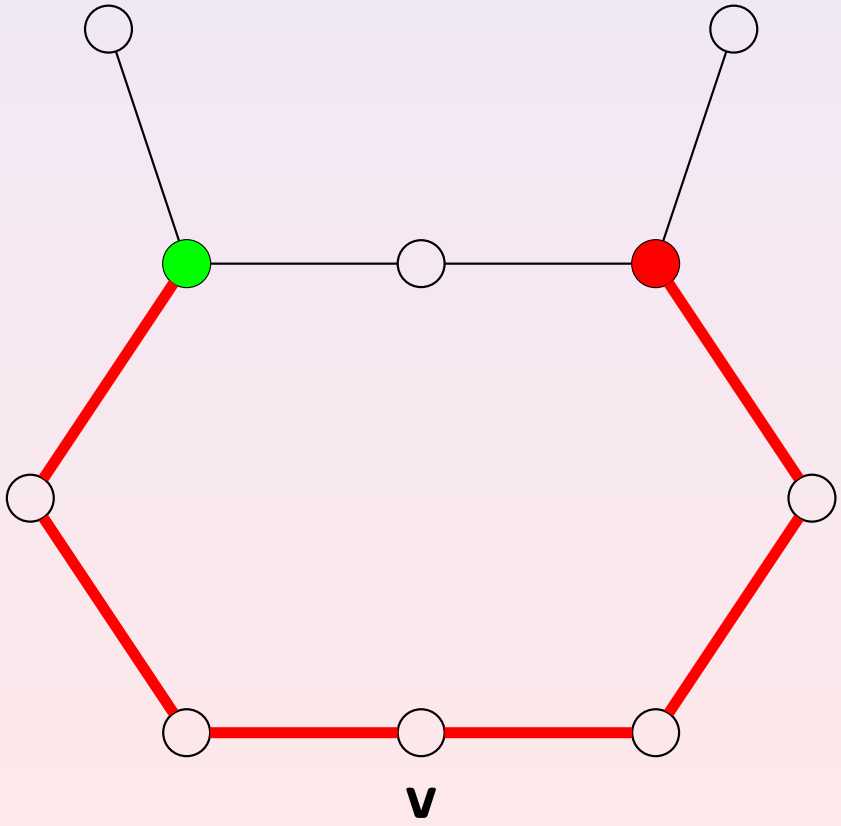
Cost of asynchronicity

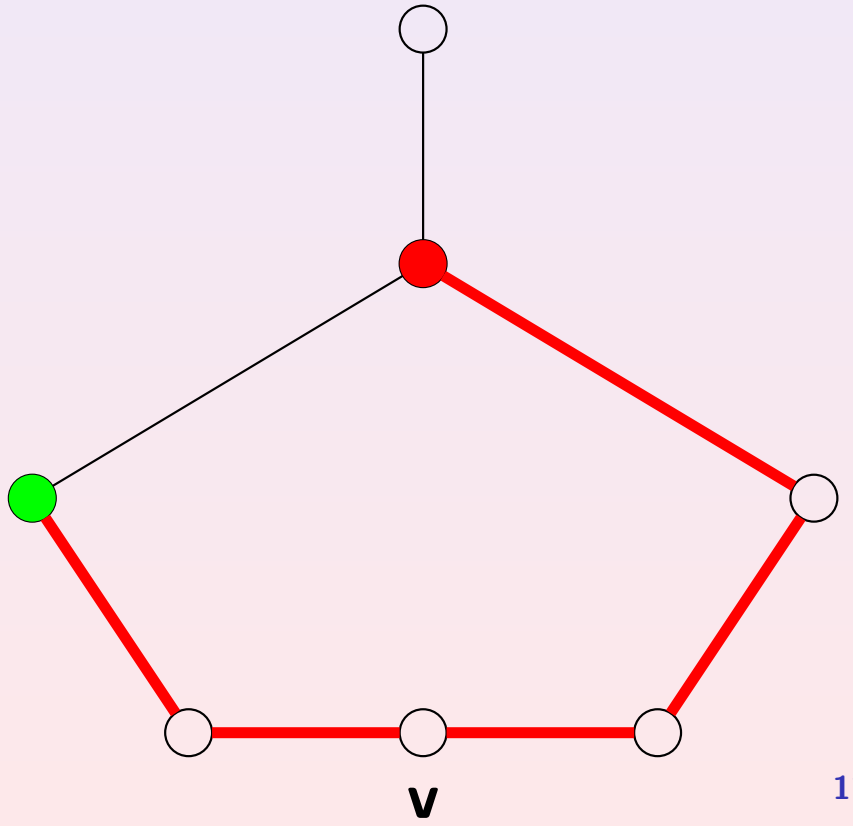




Cost of asynchronicity

The searchers cannot distinguish one graph from the other.
The two red searchers have the same local behaviour.
An extra searcher will be called in both cases.





Cost of asynchronicity

More generally

There exist graphs G such that, any distributed asynchronous graph searching protocol requires $\mathbf{mcs}(G) + 1$ searchers to clear G in a connected monotone way. [FHL05]

Coordinator

The extra searcher, the coordinator is used to synchronize the other searchers.

Cost of asynchronicity

More generally

There exist graphs G such that, any distributed asynchronous graph searching protocol requires $\mathbf{mcs}(G) + 1$ searchers to clear G in a connected monotone way. [FHL05]

Coordinator

The extra searcher, the coordinator is used to synchronize the other searchers.

Our Model (1)

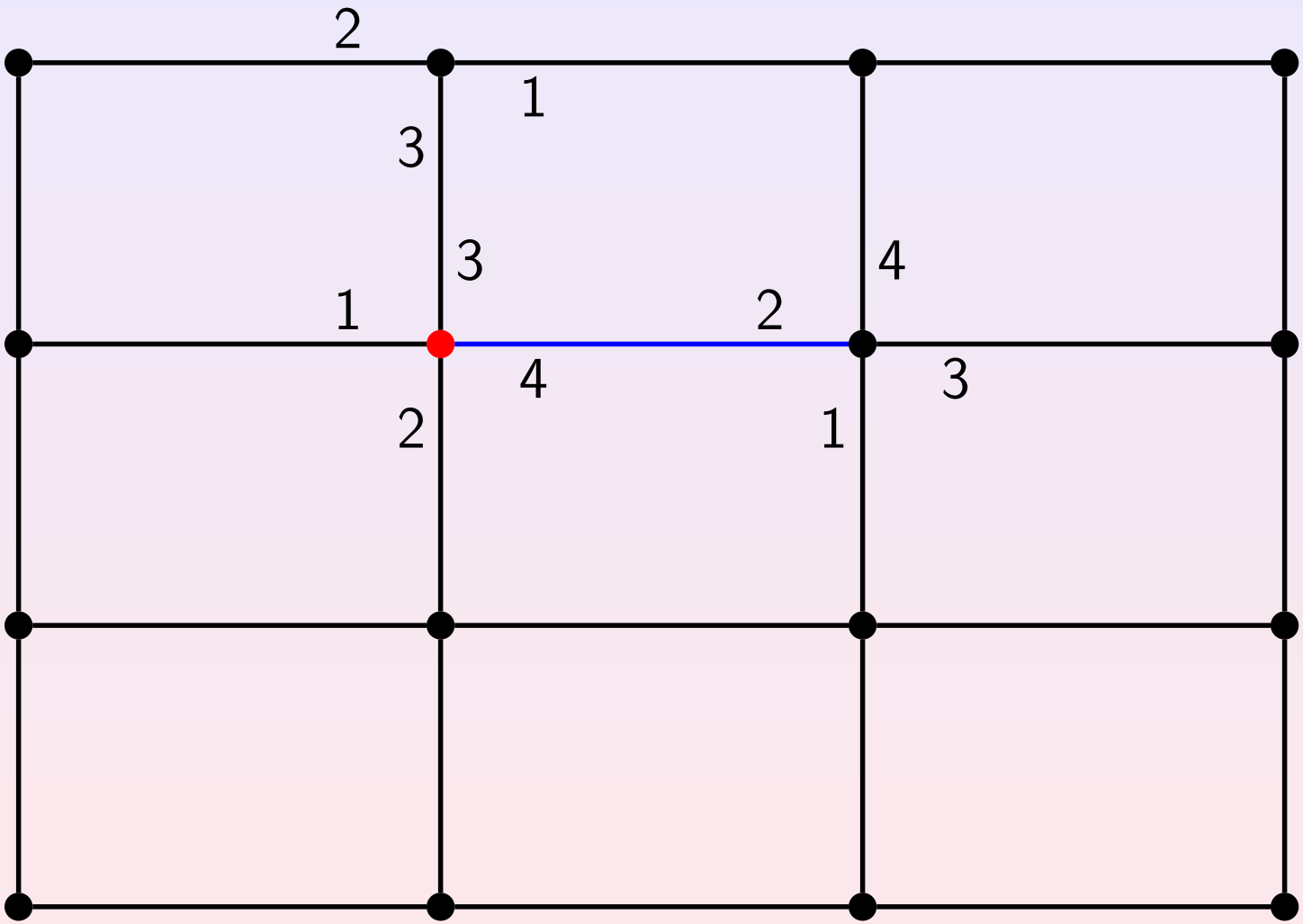
Unknown

- unknown topology
- unknown size (no upper bound)

Anonymous

- No vertex labeling
- Local edge labeling

Example of an anonymous graph



Our Model (2)

Distributed memory

- whiteboards are specific zone of local node memory,
- accessible in fair mutual exclusion.

Searchers

- autonomous mobile computing entities with distincs IDs,
- Mealy automata.

Our Model

The decision of a searcher...

- leaving a node via some specific port,
- switching state,
- writing on the whiteboard,

... is local and depends on :

- current state,
- content of the node's whiteboard,
- incoming port number.

Theorem

For any connected, asynchronous, and anonymous network G, and any $u_0 \in V(G)$, we propose an distributed algorithm that enables clearing G using searchers starting from the homebase u_0, and initially unaware of G.

1. It uses at most $k = \mathbf{mcs}(G, u_0) + 1$ searchers if $\mathbf{mcs}(G, u_0) > 1$, and $k = 1$ searcher if $\mathbf{mcs}(G, u_0) = 1$;
2. Every searcher involved in the search strategy computed uses $O(\log k)$ bits of memory;
3. During the execution, at most $O(m \log n)$ bits of information are stored at every whiteboard.

Theorem

For any connected, asynchronous, and anonymous network G, and any $u_0 \in V(G)$, we propose an distributed algorithm that enables clearing G using searchers starting from the homebase u_0, and initially unaware of G.

1. It uses at most $k = \mathbf{mcs}(G, u_0) + 1$ searchers if $\mathbf{mcs}(G, u_0) > 1$, and $k = 1$ searcher if $\mathbf{mcs}(G, u_0) = 1$;
2. Every searcher involved in the search strategy computed uses $O(\log k)$ bits of memory;
3. During the execution, at most $O(m \log n)$ bits of information are stored at every whiteboard.

Properties

- after the execution of our algorithm, an optimal monotone connected search strategy for G is written on the vertices'whiteboards in a distributed way.
- in the class of graphs with bounded **mcs**, searchers can be implemented by finite automata.

Principles of the distributed algorithm (1)

The Algorithm

Initially, one searcher stands at v_0, $k = 1$
While the graph is not clear :

- Try all monotone connected search strategies using k searchers ;
- If the graph is not clear, call a new searcher ;

Predicate

At the end of each loop, the k searchers are standing at v_0.

Principles of the distributed algorithm (2)

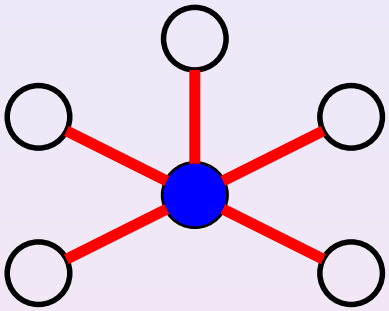
To prove the distributed algorithm :

- We propose a centralized algorithm that clear any graph G in a connected way, starting from $v_0 \in V(G)$ and using $\mathbf{mcs}(G, v_0) + 1$ searchers.
- We prove the equivalence between the algorithms.

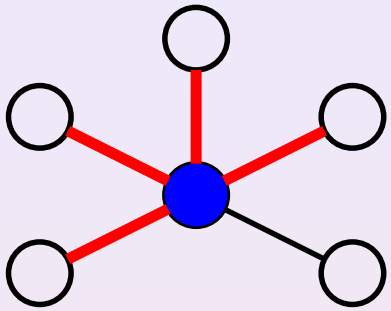
Proof

1. proof of the centralized algorithm,
2. we prove that both algorithms perform the same strategies in the same order

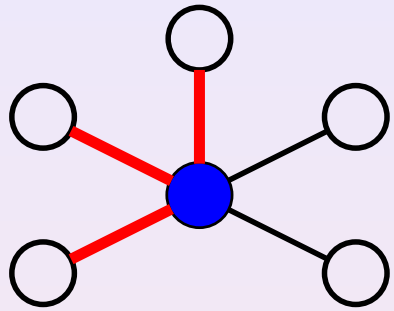
Valid moves



all edges
are cleared

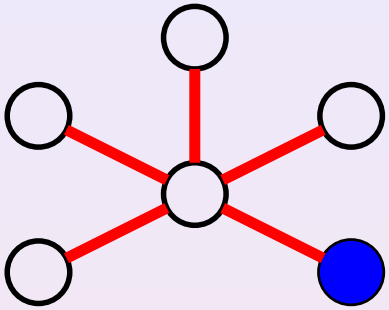


one edge
is contaminated

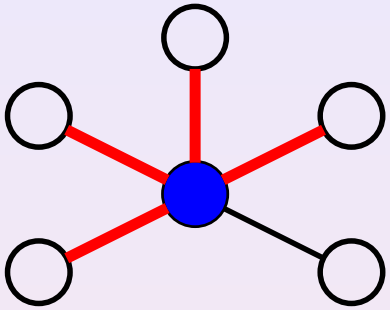


more than one edge
are contaminated

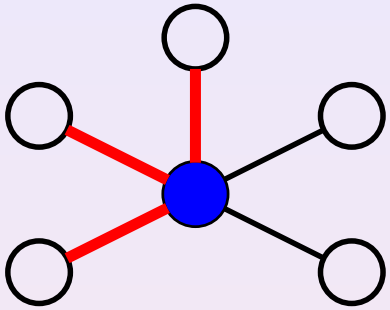
Valid moves



all edges
are cleared



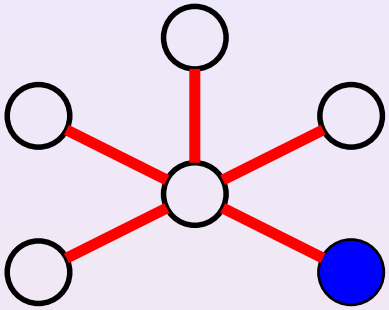
one edge
is contaminated



more than one edge
are contaminated

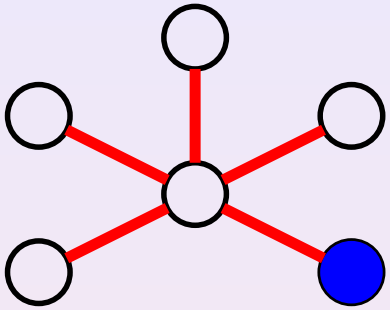
Any move is Valid

Valid moves



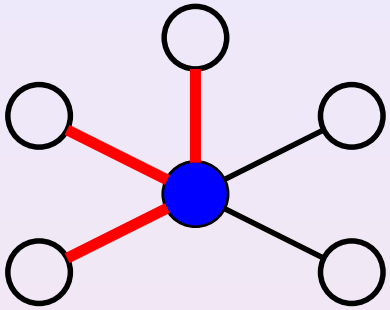
all edges
are cleared

Any move is Valid



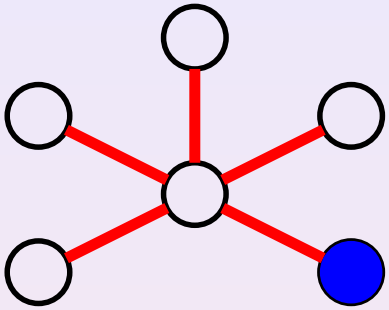
one edge
is contaminated

One move is Valid



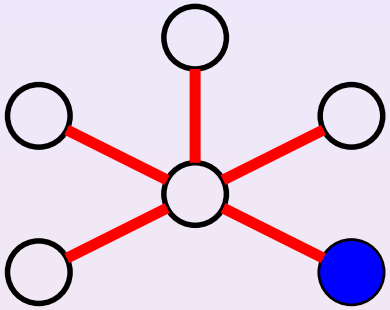
more than one edge
are contaminated

Valid moves



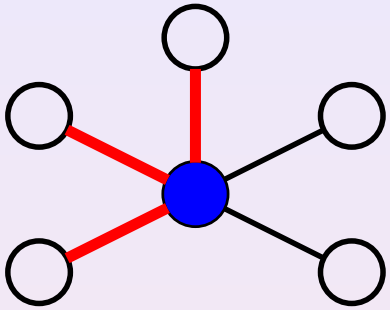
all edges
are cleared

Any move is Valid



one edge
is contaminated

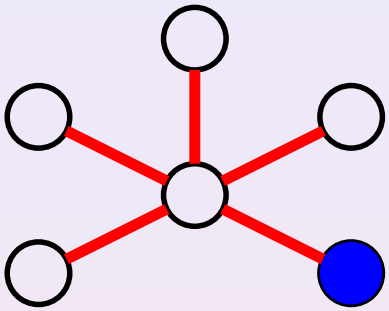
One move is Valid



more than one edge
are contaminated

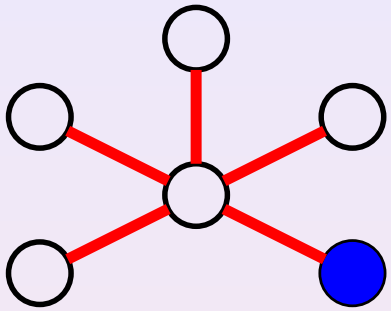
No moves are Valid

Valid moves



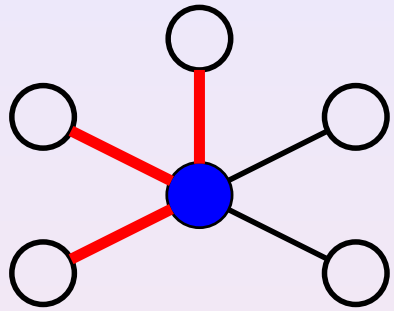
all edges
are cleared

Any move is Valid



one edge
is contaminated

One move is Valid



more than one edge
are contaminated

No moves are Valid

We order the valid moves and the strategies

A sequence of valid moves corresponds to a partial monotone connected search strategy.
The sequence are ordered in the lexicographic order.

The centralized Algorithm

$k \geq 1$ being given, the centralized algorithm tries every strategy using k searchers, starting from v_0.

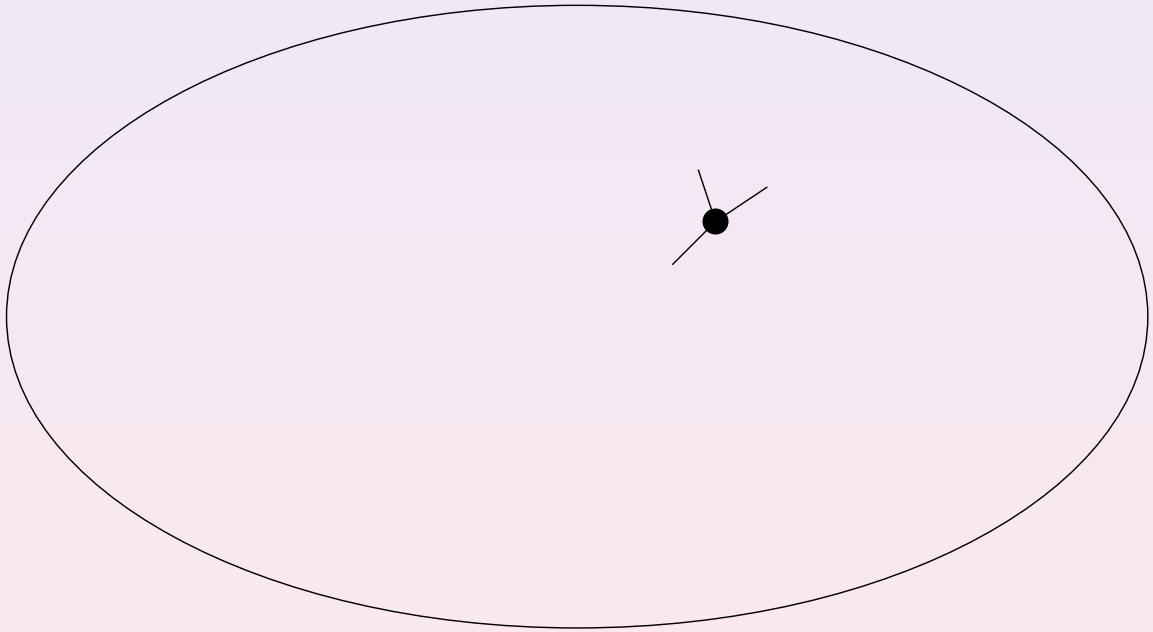
Initially, all searchers are at v_0 ;
Valid moves are performing one by one ;

- the smallest possible move is performed,
- if no valid move is possible, the last move performed is backtracked.

Remark : the latter case occurs when any searcher stands alone at a vertex with at least two incident contaminated edges.

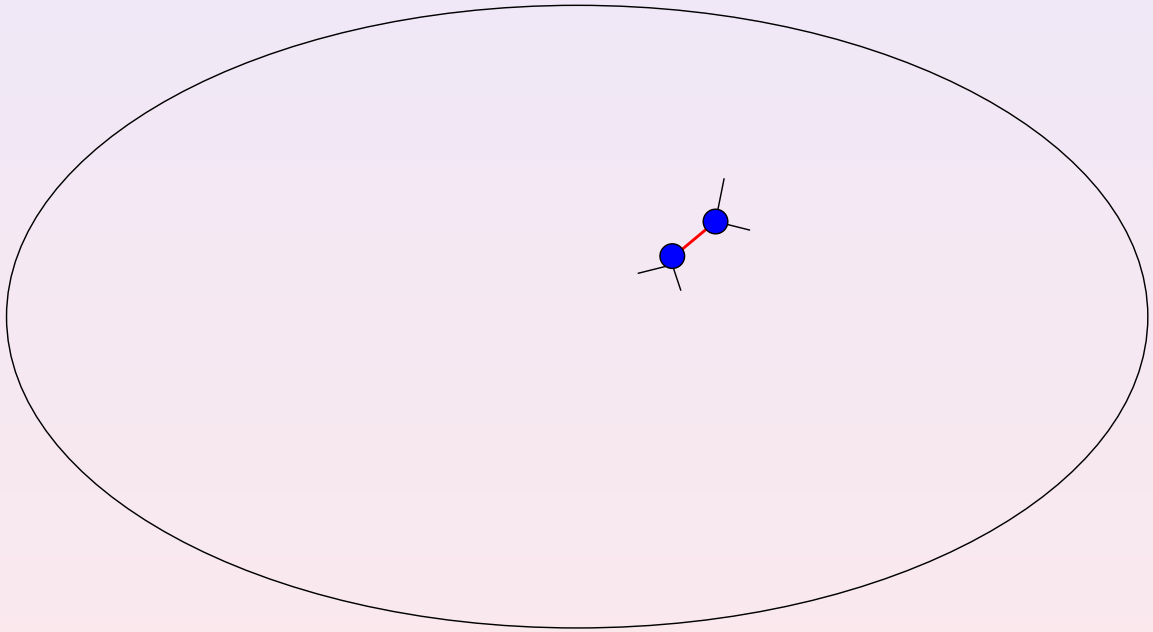
The centralized Algorithm

The algorithm tries all strategies for k=1,2,...



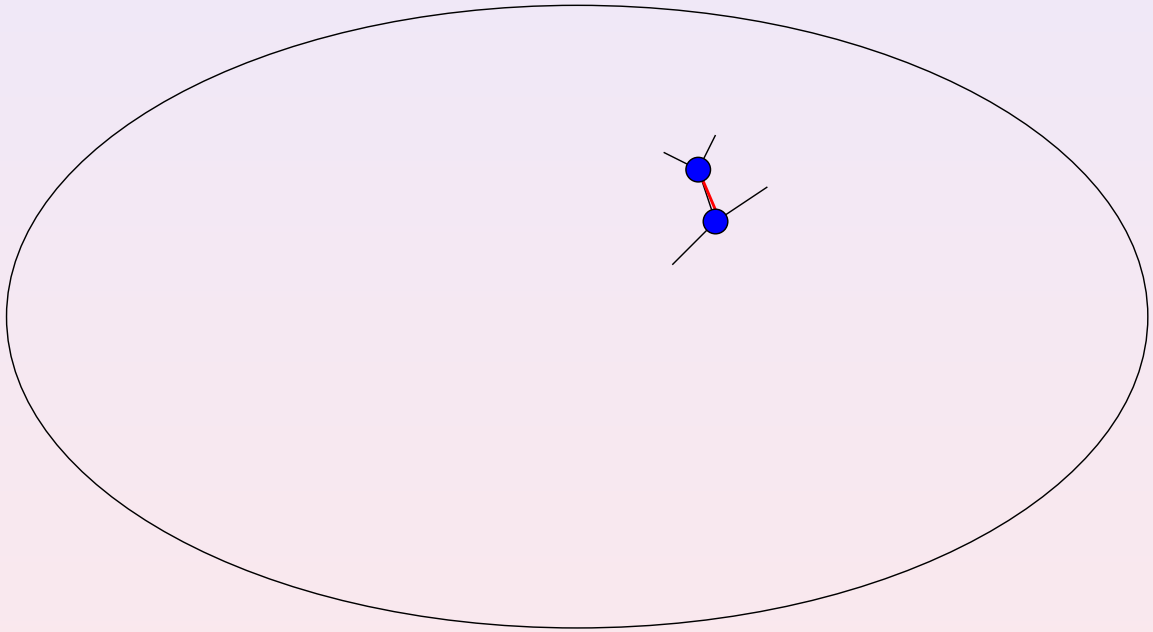
The centralized Algorithm

The algorithm tries all strategies for k=1,2,...



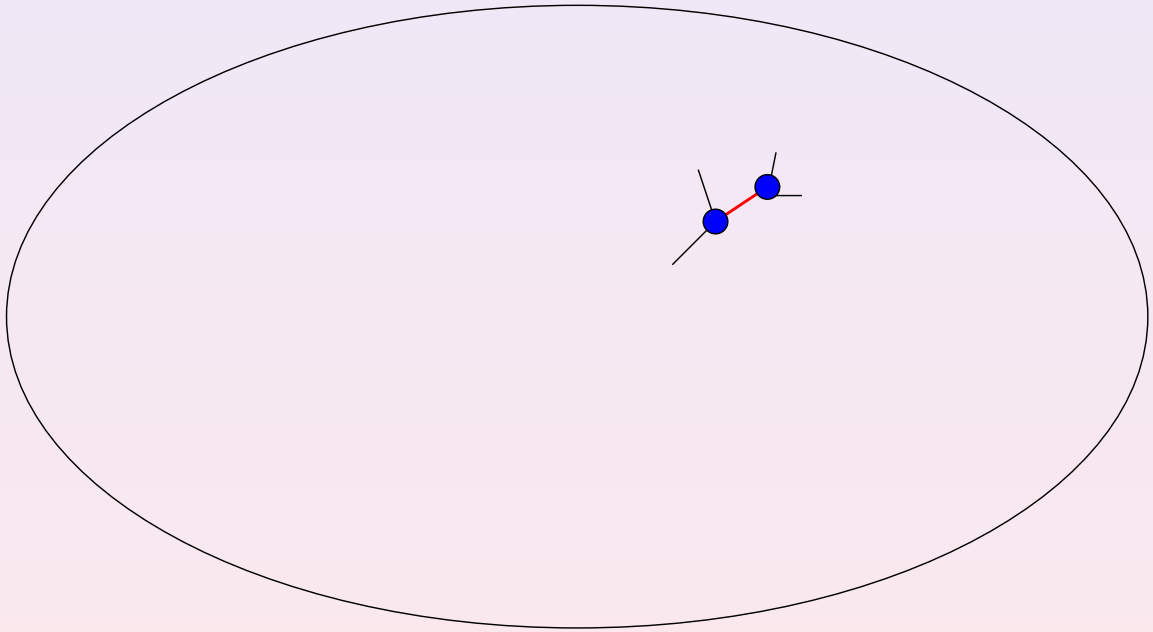
The centralized Algorithm

The algorithm tries all strategies for k=1,2,...



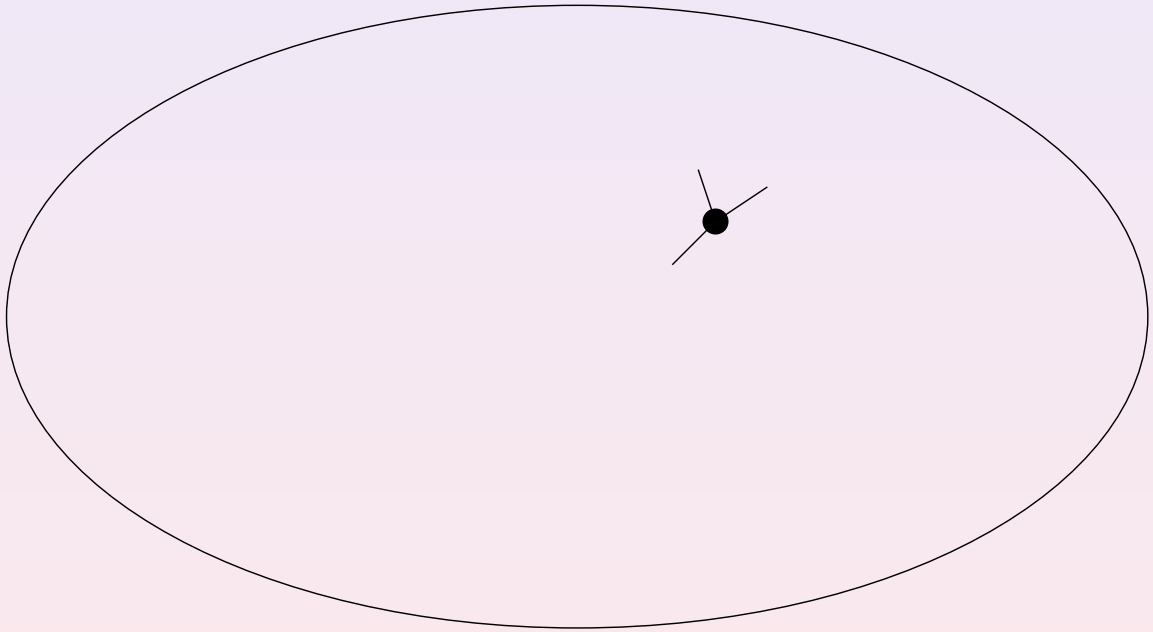
The centralized Algorithm

The algorithm tries all strategies for k=1,2,...



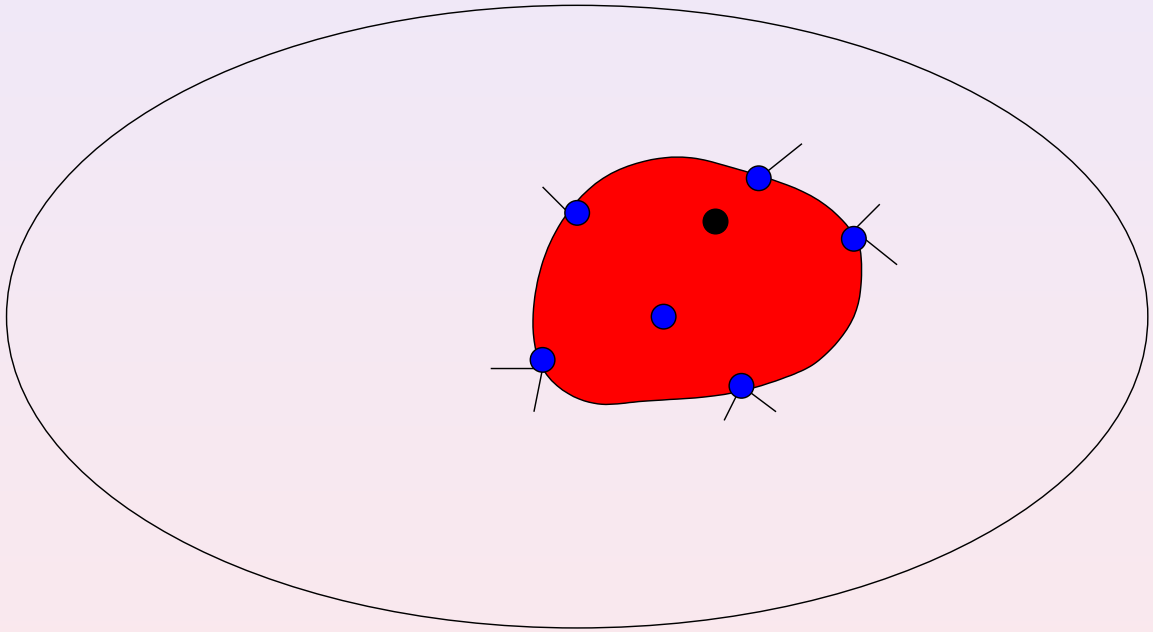
The centralized Algorithm

The algorithm tries all strategies for k=1,2,...



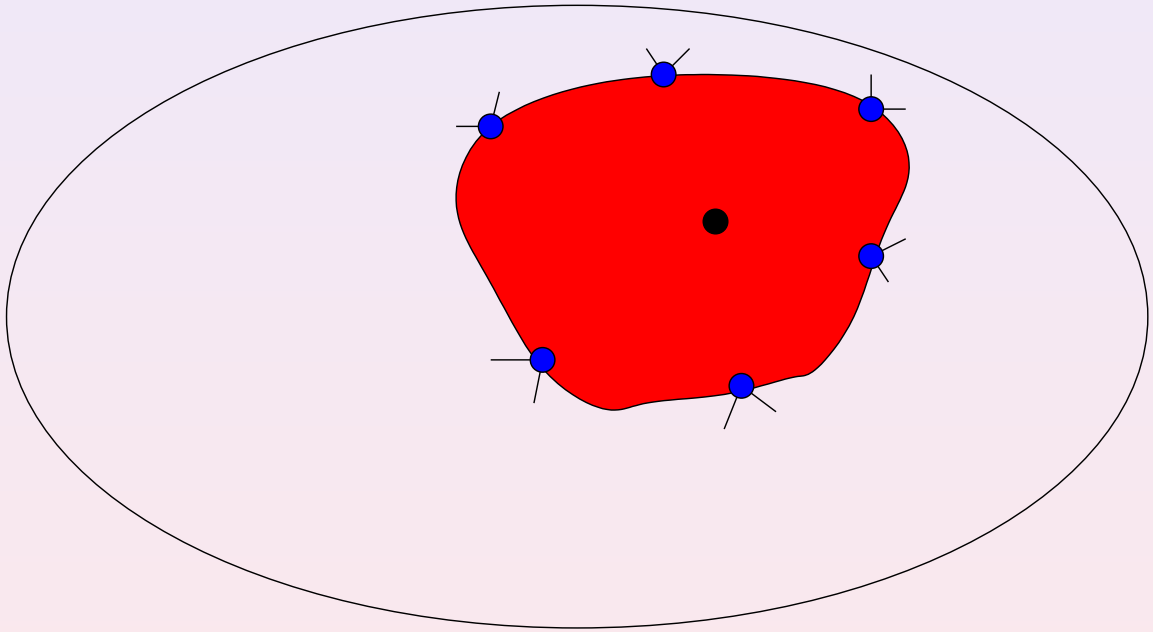
The centralized Algorithm

The algorithm tries all strategies for k=1,2,...



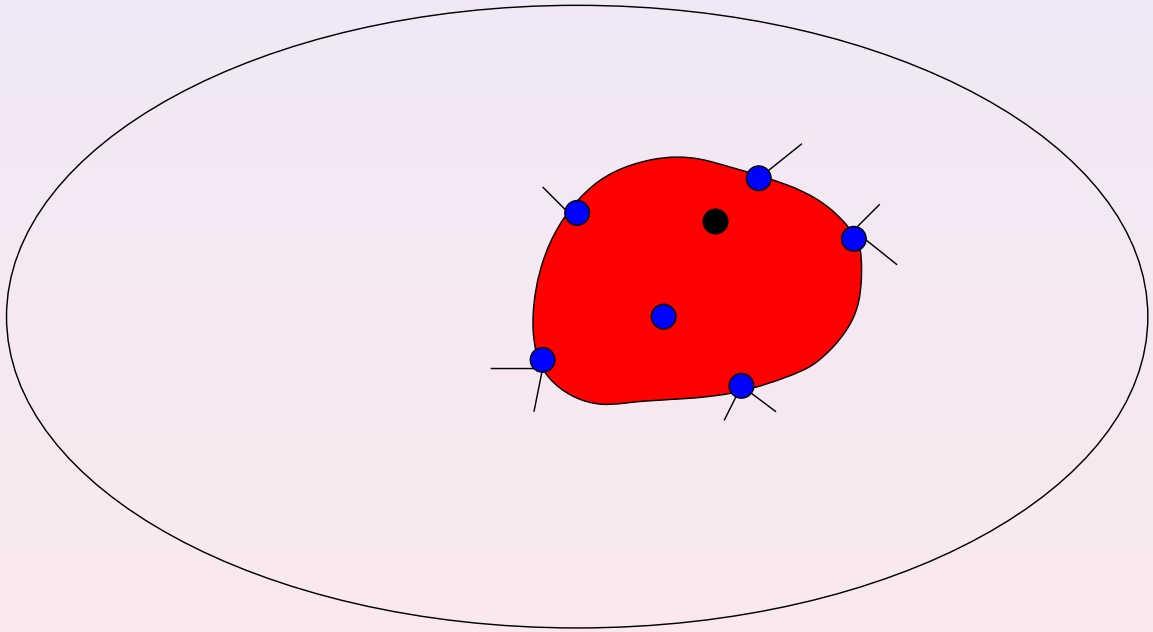
The centralized Algorithm

The algorithm tries all strategies for k=1,2,...



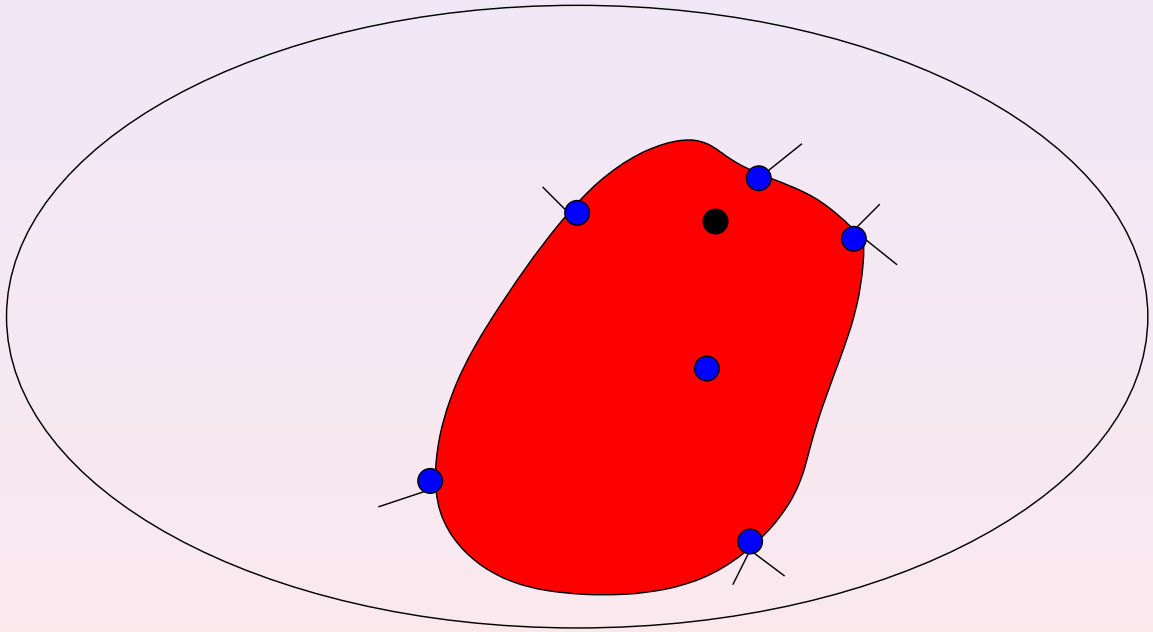
The centralized Algorithm

The algorithm tries all strategies for k=1,2,...



The centralized Algorithm

The algorithm tries all strategies for k=1,2,...



Further Works

Open problems

- Is there exist an algorithm using finite automata ?
- How to reduce the size of whiteboards ?
- What is the amount of knowledge that is required to clear a graph in a polynomial time ?