The Cost of Monotonicity in Distributed Graph Searching

David Ilcinkas[1] Nicolas Nisse[2] David Soguet[2]

[1] Université du Québec en Outaouais, Canada.

[2] LRI, Université Paris-Sud, France.

Opodis, December 2007

Graph searching problem

Goal:

In an undirected connected simple graph,

- in which edges are contaminated,
- a team of searchers is aiming at clearing the graph.

We want to find a strategy that clears the graph **using the minimum number of searchers**.

Applications:

- network security,
- decontaminating a set of polluted pipes,
- …

Graph searching in distributed settings

Distributed graph searching:

- The searchers compute themselves a strategy;
- The strategy must be **computed and performed** in polynomial time.

Distributed search problem:

To design a *distributed protocol* that enables the *minimum number of searchers* to clear the network in *polynomial time*.

Search strategy

The searchers move along the edges.

Search strategy

The searchers move along the edges.

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

Search strategy

The searchers move along the edges.

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

A clear edge *e* is recontaminated if a path exists between *e* and a contaminated edge, and no searchers stand on this path.

Search strategy

The searchers move along the edges.

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

A clear edge *e* is recontaminated if a path exists between *e* and a contaminated edge, and no searchers stand on this path.

A strategy consists of:

- Initially, all searchers are placed at the homebase v_0;
- sequence of moves of searcher; a searcher can move if it **does not imply recontamination**;
- until the graph is clear.

Search strategy

The searchers move along the edges.

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

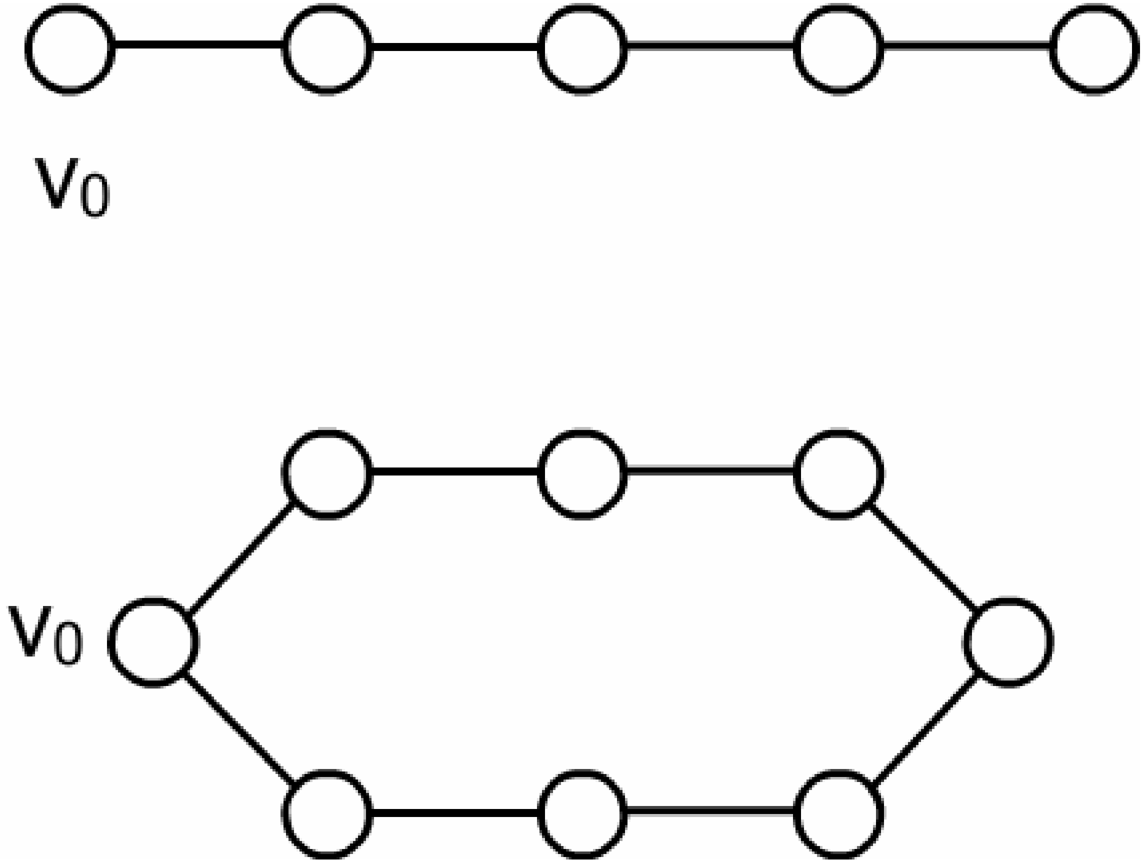
A clear edge *e* is recontaminated if a path exists between *e* and a contaminated edge, and no searchers stand on this path.

A strategy consists of:

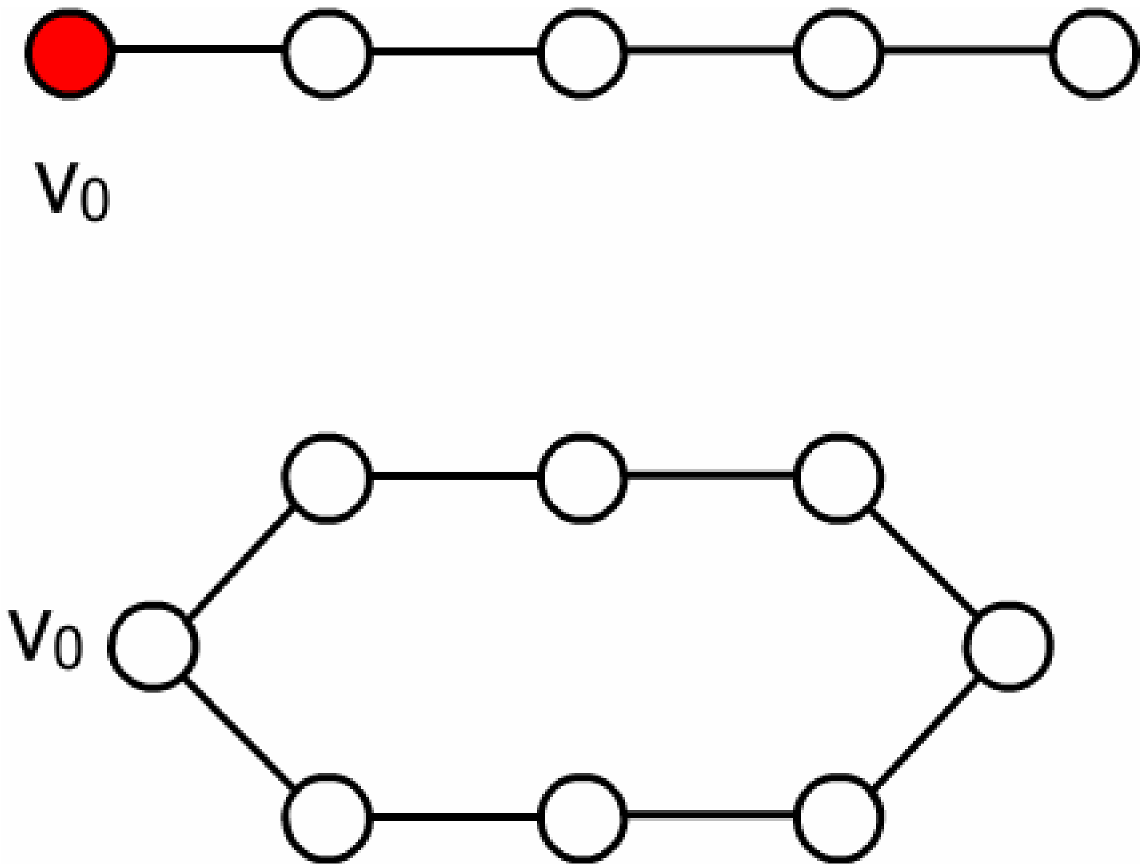
- Initially, all searchers are placed at the homebase v_0;
- sequence of moves of searcher; a searcher can move if it **does not imply recontamination**;
- until the graph is clear.

mcs(G,v_0): minimum number of searchers required to clear the graph G in this way, starting from v_0, in centralized setting.

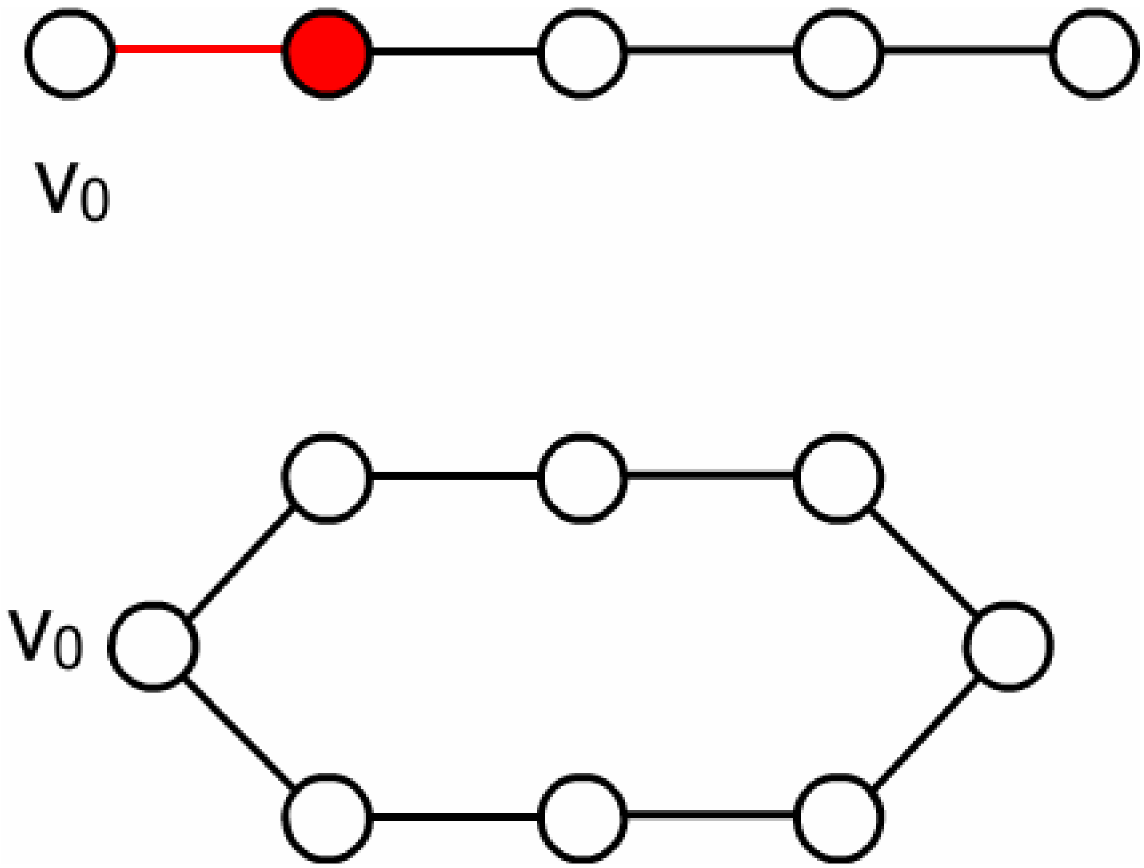
Two simple examples : the path and the ring



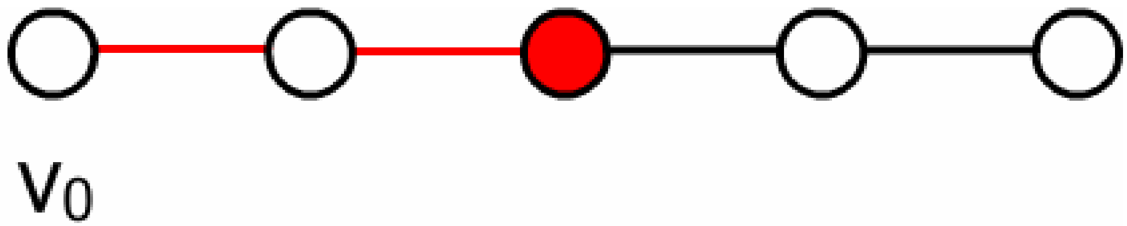
Two simple examples : the path and the ring

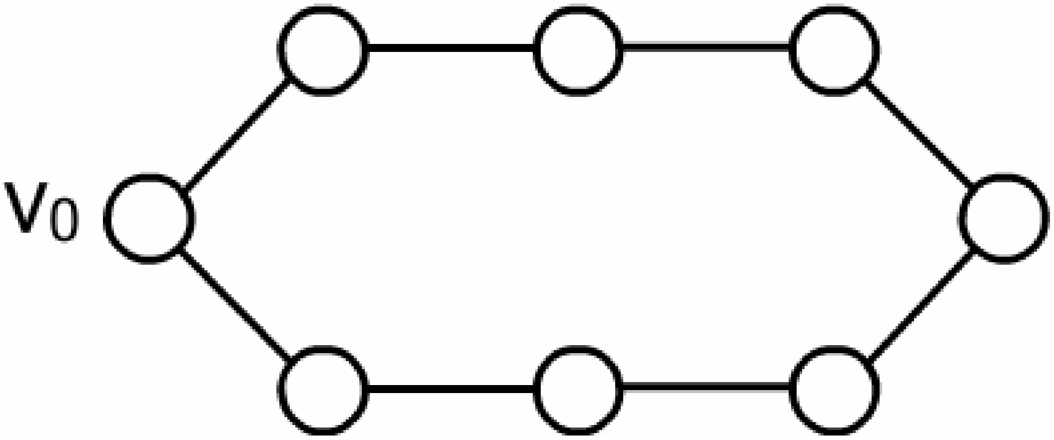


Two simple examples : the path and the ring

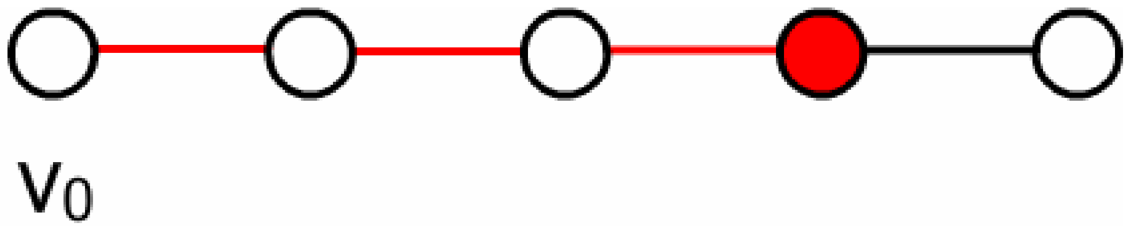


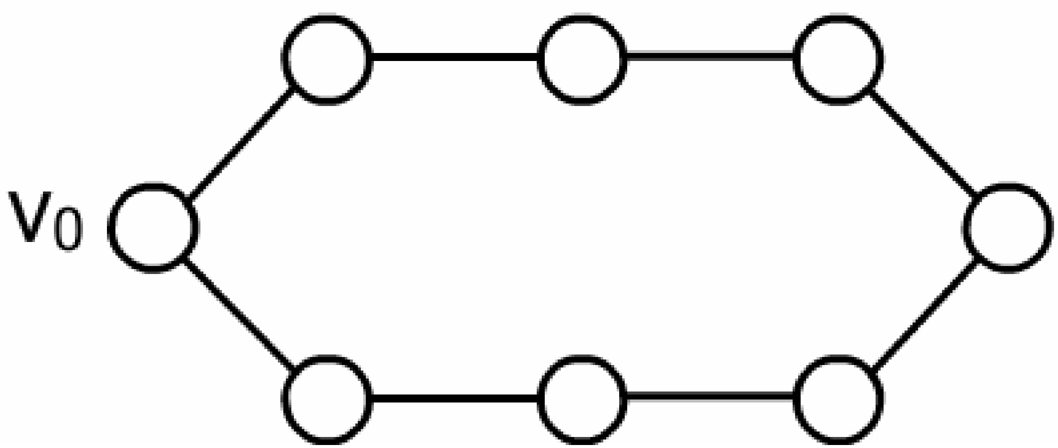
Two simple examples : the path and the ring



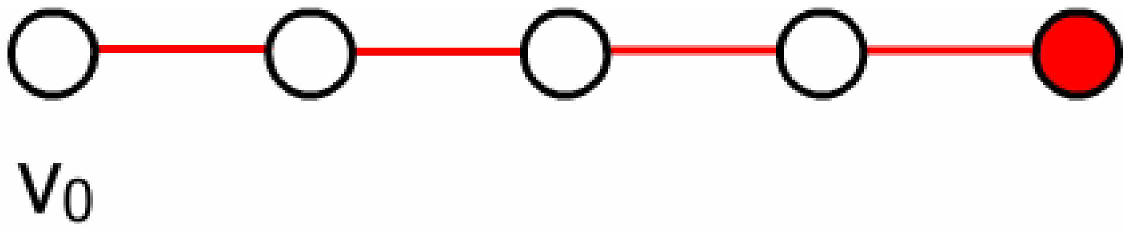


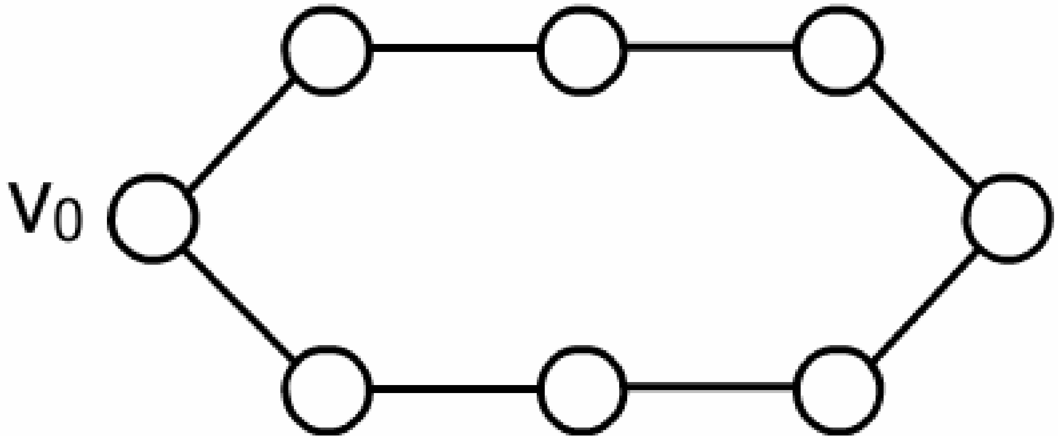
Two simple examples : the path and the ring



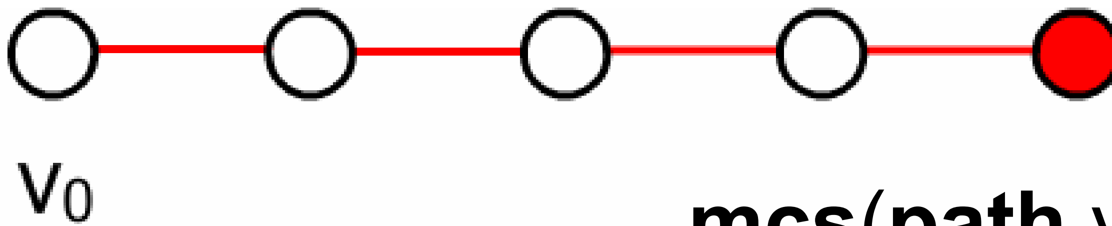


Two simple examples : the path and the ring

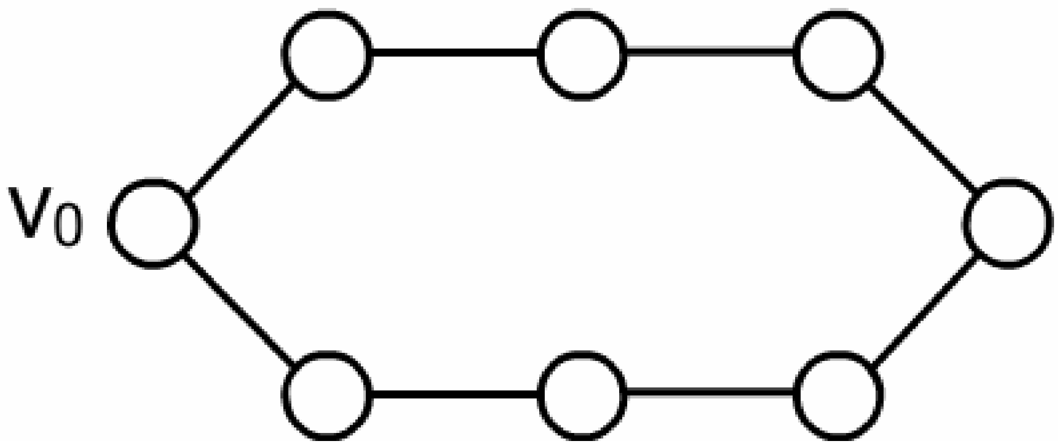




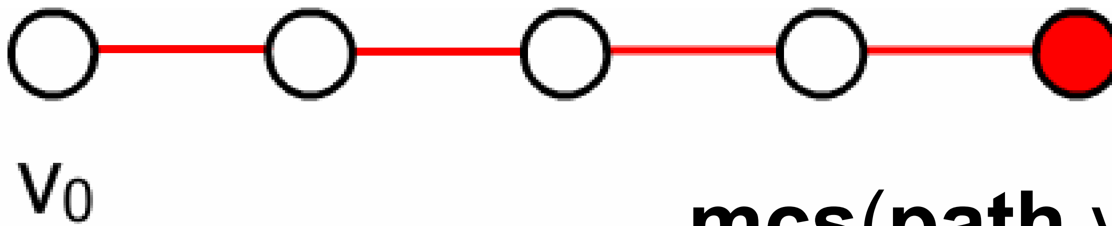
Two simple examples : the path and the ring



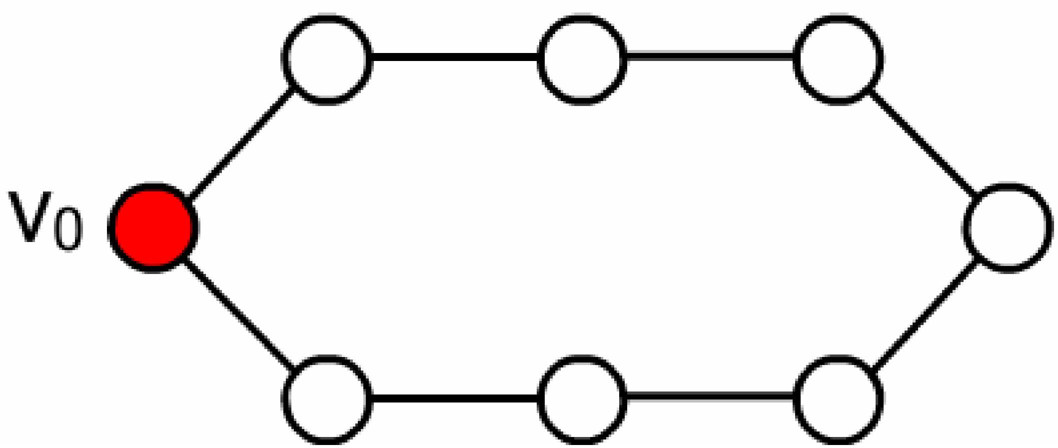
mcs(path,v_0)=1



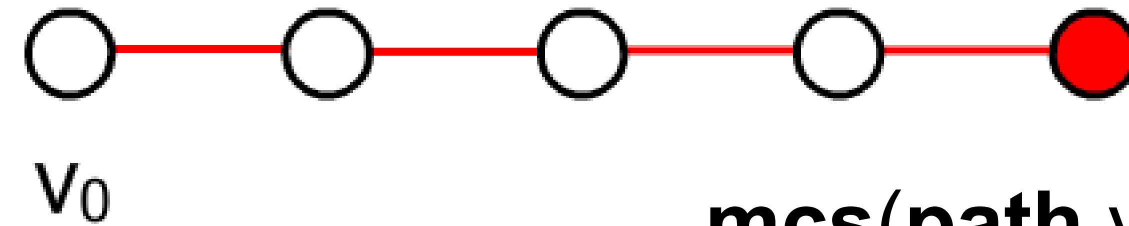
Two simple examples : the path and the ring



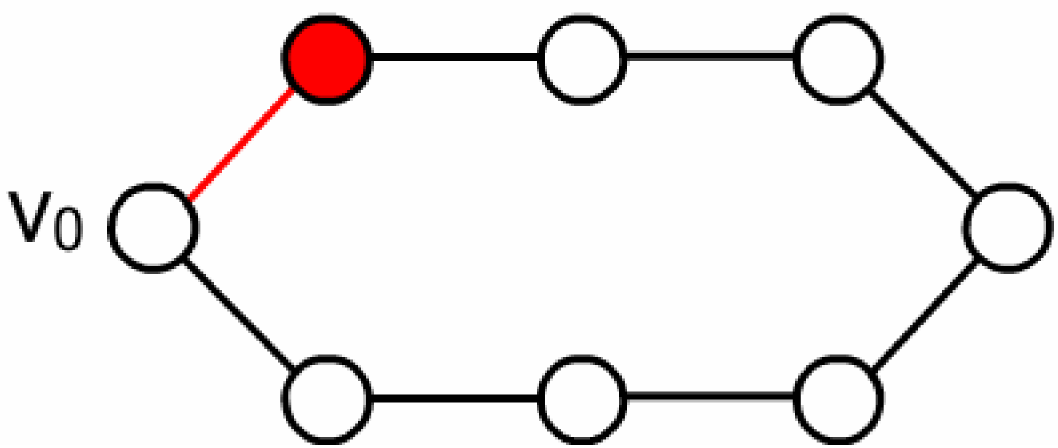
$mcs(path, v_0)=1$



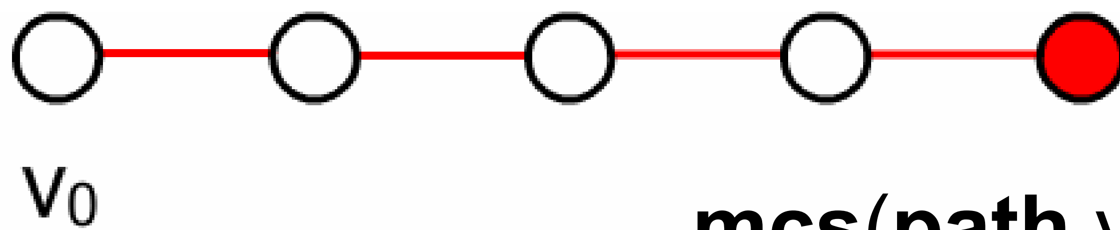
Two simple examples : the path and the ring



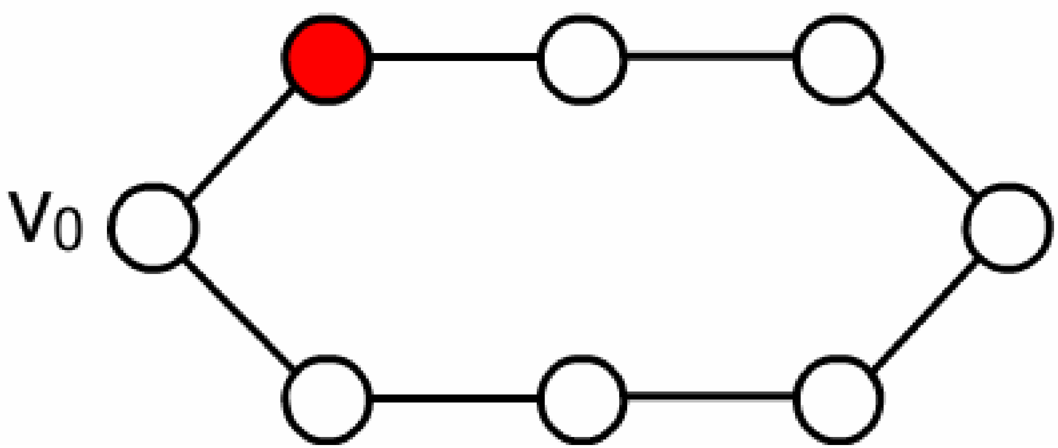
mcs(path,v_0)=1



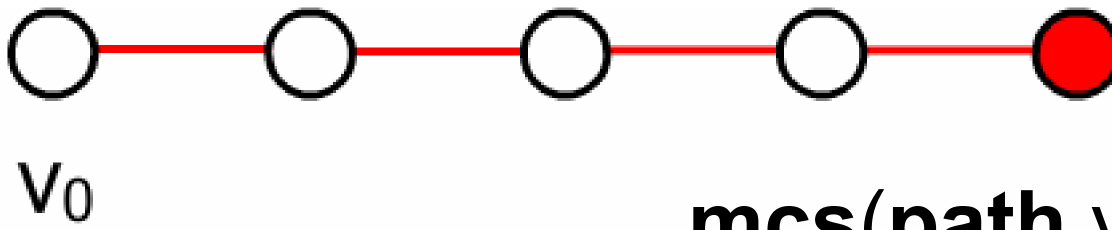
Two simple examples : the path and the ring



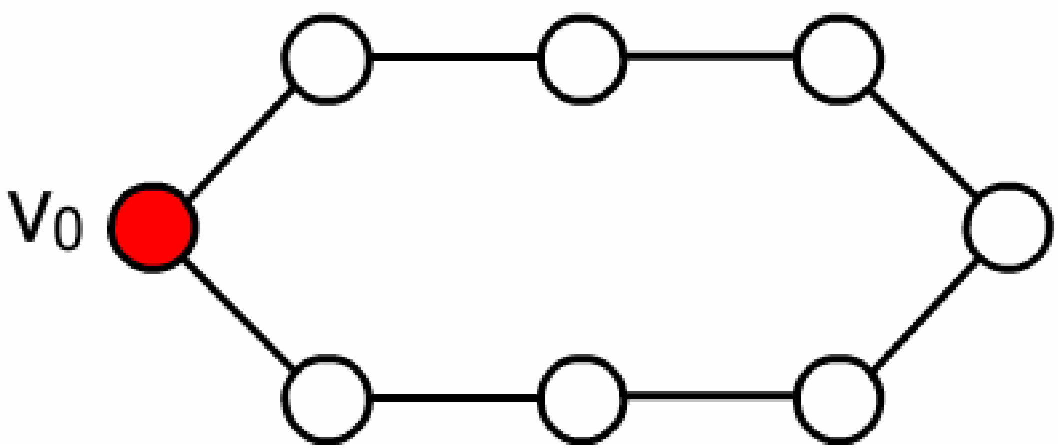
mcs(path,v_0)=1



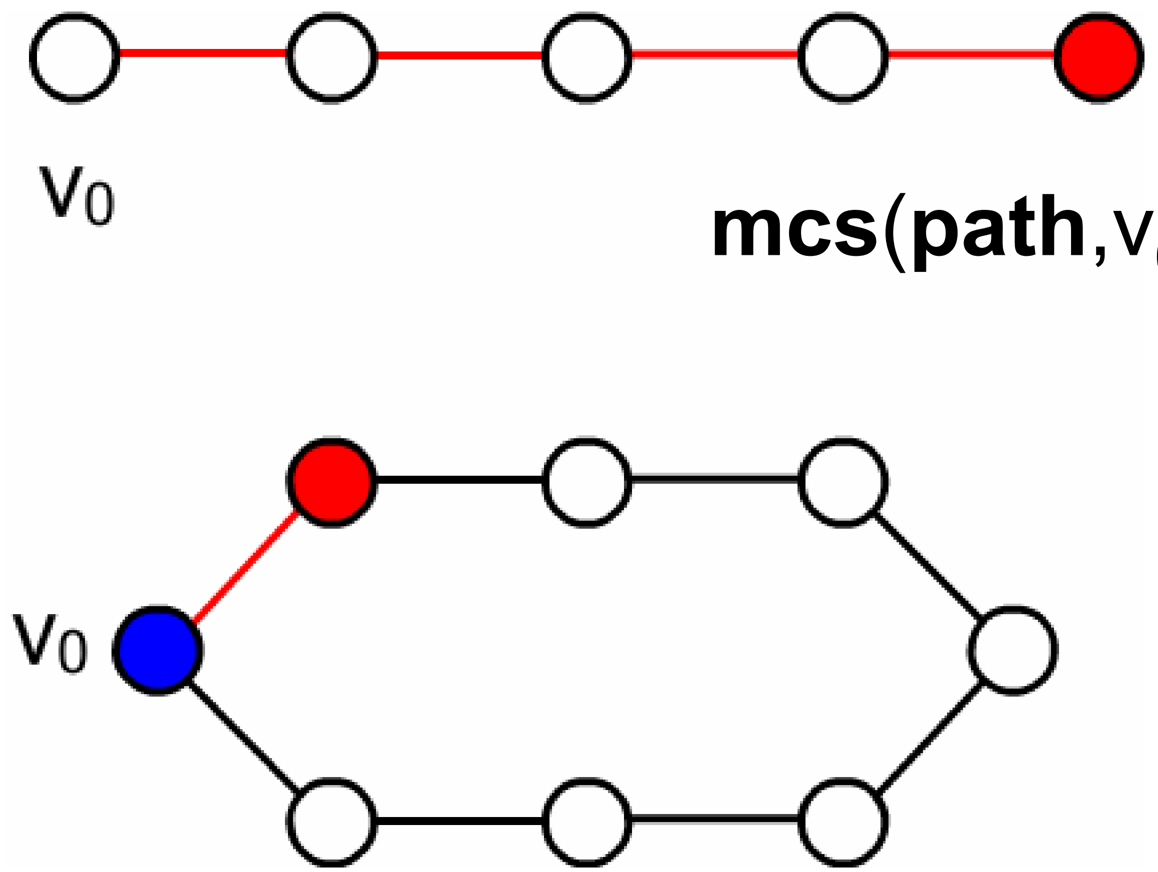
Two simple examples : the path and the ring



mcs(path,v_0)=1

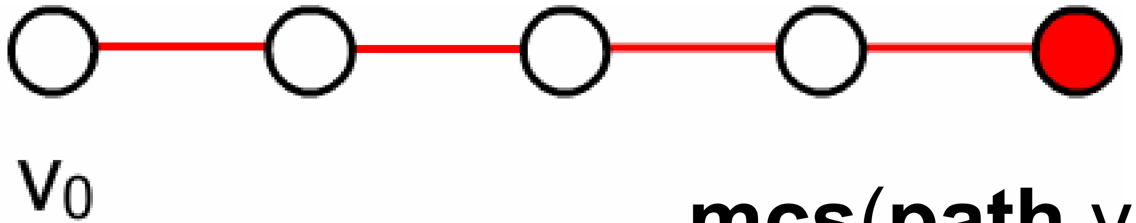


Two simple examples : the path and the ring

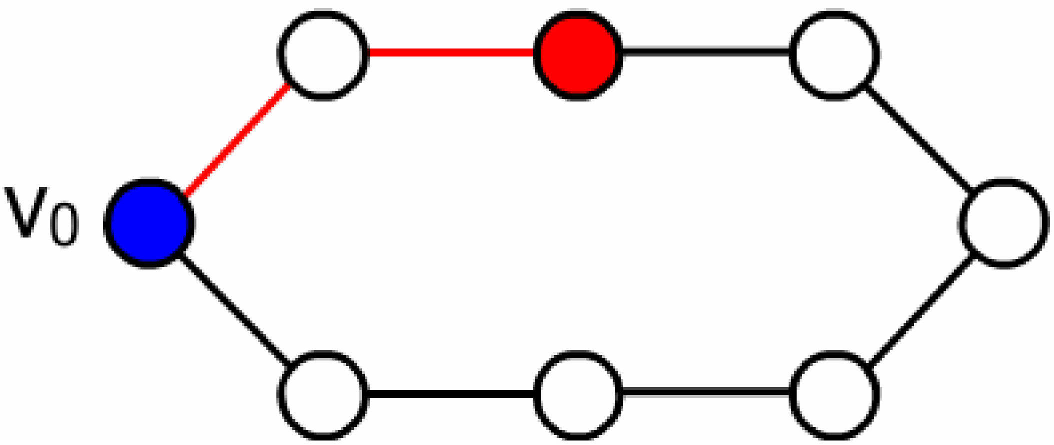


mcs(path,v_0)=1

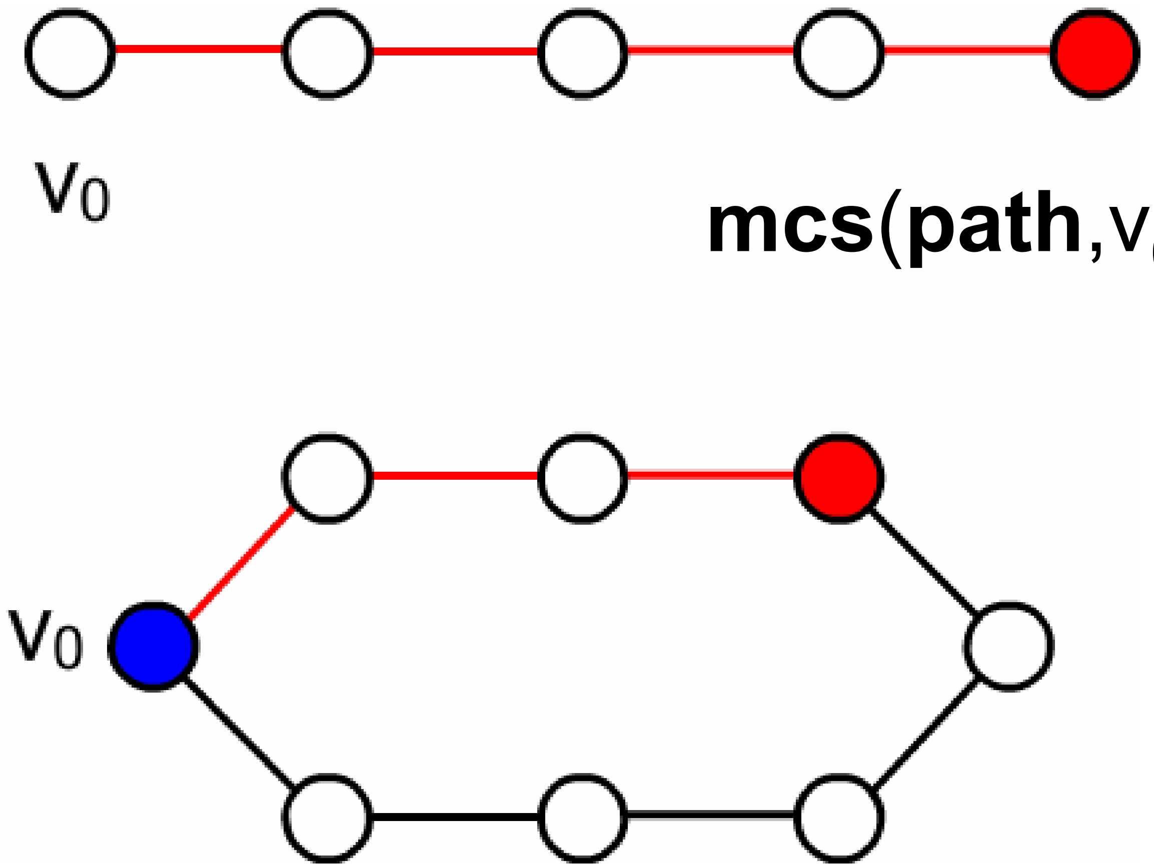
Two simple examples : the path and the ring



mcs(path,v_0)=1

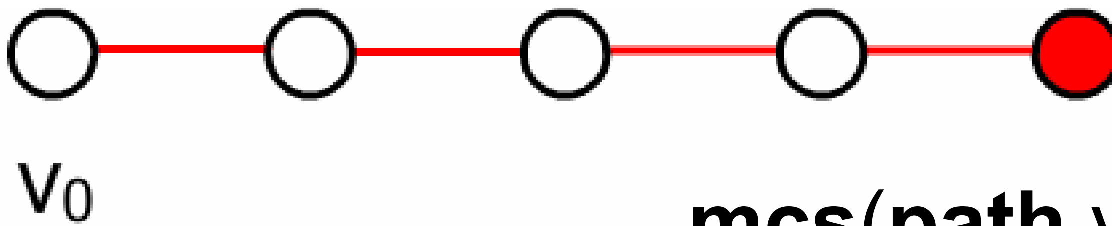


Two simple examples : the path and the ring

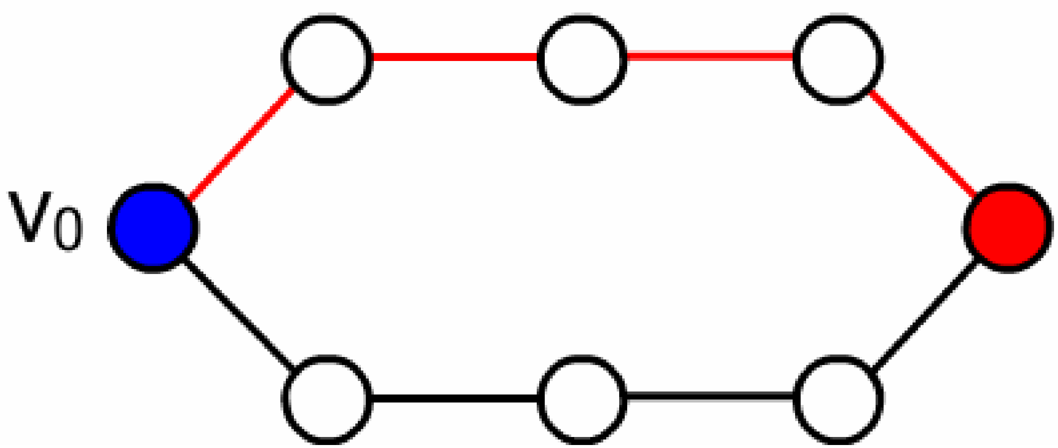


mcs(path,v_0)=1

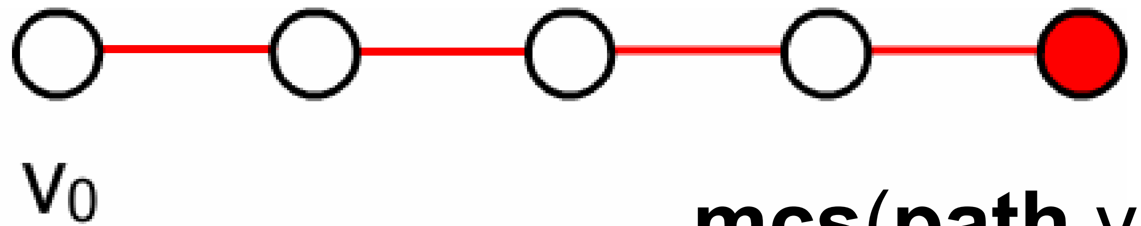
Two simple examples : the path and the ring



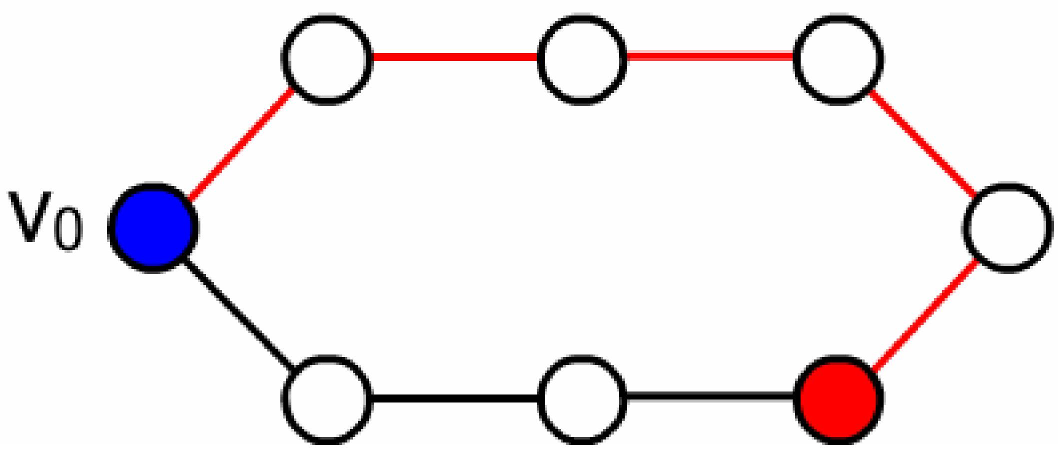
mcs(path,v_0)=1



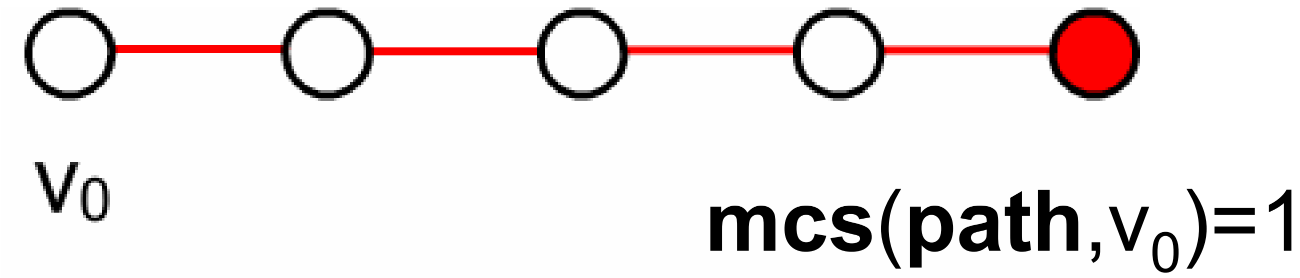
Two simple examples : the path and the ring

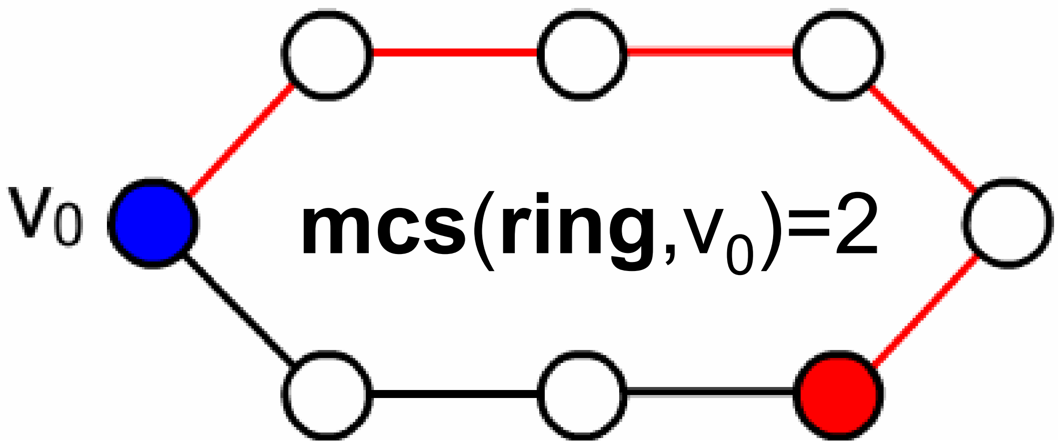


mcs(path,v_0)=1



Two simple examples : the path and the ring





Monotone connected search strategy

Monotone connected strategy:

- Monotonicity: the contaminated part of the graph never grows (i.e., no recontamination can occur)

  $\Rightarrow$ **polynomial time**
- Connectivity: the cleared part is connected

  $\Rightarrow$ **safe communications**

Monotone connected search strategy

Monotone connected strategy:

- Monotonicity: the contaminated part of the graph never grows (i.e., no recontamination can occur)

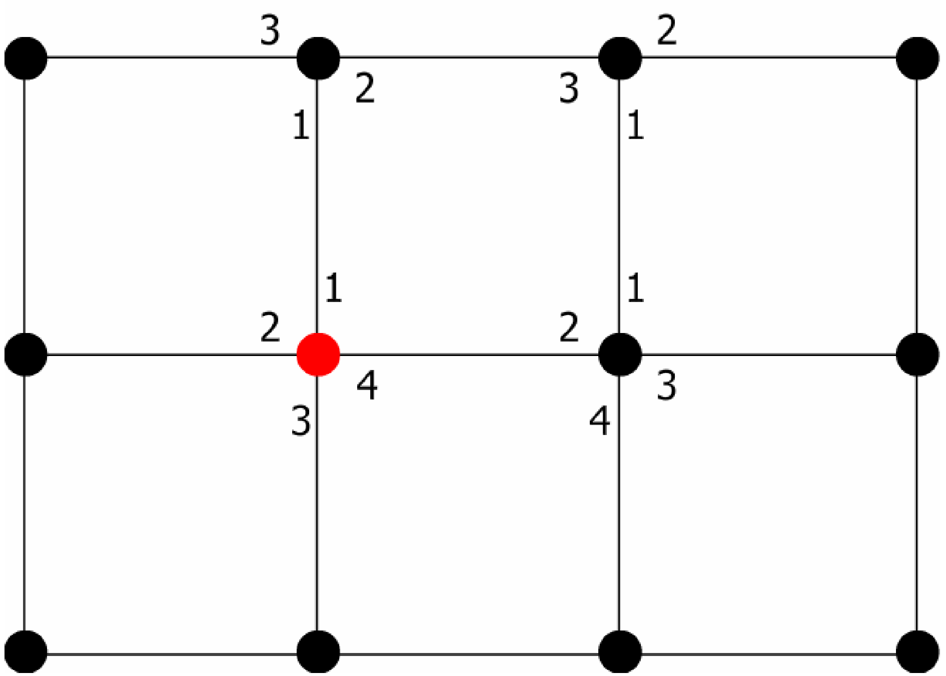
  ⇒ **polynomial time**
- Connectivity: the cleared part is connected

  ⇒ **safe communications**

Remark: The problem of **computing mcs**(G,v_0) and the corresponding monotone connected strategy is **NP-complete in a centralized setting** [Megiddo *et al*. 1988].

Model : Environment

- Undirected connected simple graph;
- Local orientation of the edges;
- Whiteboard (zone of local memory);
- Synchronous/asynchronous environment.



Model : the searchers

Autonomous mobile computing entities with memory and distinct IDs.

Decision is computed locally and depends on:

- its current state;
- the states of the other searchers present at the vertex;
- information on the whiteboard;
- if appropriate the incoming port number.

A searcher can decide to:

- leave a vertex via a specific port number;
- write, read or erase information on the whiteboard;
- switch its state.

Related work 1/2

The searchers have no prior information about the graph.

Protocol to clear an unknown graph

Distributed chasing of network intruders

[Blin, Fraignaud, Nisse and Vial. 2006]

A connected and optimal strategy is performed.

Related work 1/2

The searchers have no prior information about the graph.

Protocol to clear an unknown graph

Distributed chasing of network intruders

[Blin, Fraignaud, Nisse and Vial. 2006]

A connected and optimal strategy is performed.

Problem:

the strategy is not monotone and may be performed in exponential time.

Related work 2/2

The searchers have a prior knowledge about the graph.

Related work 2/2

The searchers have a prior knowledge about the graph.

Protocols to clear specific topologies

- **Mesh** [Flocchini, Luccio and Song. 2005]
- **Hypercube** [Flocchini, Huang and Luccio. 2005]
- **Tori** [Flocchini, Luccio and Song. 2006]
- **Siperski's graph** [Luccio. 2007]

A monotone connected and optimal strategy is performed.

Related work 2/2

The searchers have a prior knowledge about the graph.

Protocols to clear specific topologies

- **Mesh** [Flocchini, Luccio and Song. 2005]
- **Hypercube** [Flocchini, Huang and Luccio. 2005]
- **Tori** [Flocchini, Luccio and Song. 2006]
- **Siperski's graph** [Luccio. 2007]

A monotone connected and optimal strategy is performed.

Protocol to clear a graph with advice

Θ(n log n) bits of advice (information) are necessary and sufficient to clear any n nodes graph **in a monotone connected and optimal way** [Nisse and Soguet. 2007].

Graph searching in distributed settings

Distributed search problem:

To design a *distributed protocol* that enables the *searchers* to clear the network in a monotone connected and optimal way.

Graph searching in distributed settings

Distributed search problem:

To design a *distributed protocol* that enables the *searchers* to clear the network in a monotone connected and optimal way.

Relaxed distributed search problem:

To design a *distributed protocol* that enables the *searchers* to clear the network in a monotone connected but not necessary optimal way.

Problem

A natural question is :

Compared to the optimal number of searchers in a centralized setting, how many additional searchers are necessary and sufficient, to clear in a monotone and connected way any unknown graph, in a decentralized manner?

Quality and competitive ratio of a protocol

The quality of a protocol $\mathscr{P}$ to clear a graph G starting from v_0 is measured by comparing the number of searchers it used to the number **mcs**(G, v_0).

The competitive ratio of a protocol $\mathscr{P}$ is the quality of the protocol $\mathscr{P}$, maximized over all graphs and all starting nodes.

Our results

Upper bound:

The relaxed distributed search problem can be solved by a protocol of competitive ratio $O(n / \log n)$.

Our results

Upper bound:

The relaxed distributed search problem can be solved by a protocol of competitive ratio O(n / log n).

We design a protocol that use at most O((n/log n) **mcs**(G,v_0)) searchers to clear any graph G in a monotone connected way, starting from $v_0 \in V_G$. The searchers use at most O(log n) bits of memory, and whiteboards are of size O(n) bits.

Our results

Upper bound:

The relaxed distributed search problem can be solved by a protocol of competitive ratio $O(n / \log n)$.

Lower bound:

Any protocol for solving the relaxed distributed search problem has competitive ratio $\Omega(n / \log n)$.

Our results

Upper bound:

The relaxed distributed search problem can be solved by a protocol of competitive ratio $O(n / \log n)$.

Lower bound:

Any protocol for solving the relaxed distributed search problem has competitive ratio $\Omega(n / \log n)$.

For any distributed protocol $\mathscr{P}$, there exists a constant *c* such that for any sufficiently large n, there exists a n-node graph G, and $v_0 \in V(G)$, such that $\mathscr{P}$ requires at least $(cn / \log n)$ **mcs**(G, v_0) searchers to clear G starting from v_0.

Idea of the upper bound : $O(n / \log n)$

Definition: A graph H is a minor of a graph G if H is a subgraph of a graph obtained by a succession of edge contractions of G.

Idea of the upper bound : O(n / log n)

Definition: A graph H is a minor of a graph G if H is a subgraph of a graph obtained by a succession of edge contractions of G.

Main issue of the protocol clearing a graph G:

- maintain a dynamic rooted tree S;
  - S is a tree of degree at most 3;
  - and S is a minor of the clear part of G.
- at each step, the protocol tries to clear an edge of G such that S becomes as close as possible to a complete tree of degree 3.

Idea of the upper bound : O(n / log n)

Let $\mathbf{T_k}$ be a complete tree of degree 3 of depth k.

Idea of the upper bound : O(n / log n)

Let $\mathbf{T_k}$ be a complete tree of degree 3 of depth k.

At each step, the protocol is such that:

- V(S) is the set of vertices of G occupied by a searcher

Idea of the upper bound : $O(n / \log n)$

Let $\mathbf{T_k}$ be a complete tree of degree 3 of depth k.

At each step, the protocol is such that:

- V(S) is the set of vertices of G occupied by a searcher
  - ⇒ if k is the maximum depth of S, the protocol uses at most $|V(T_k)|$ searchers.

Idea of the upper bound : O(n / log n)

Let $\mathbf{T_k}$ be a complete tree of degree 3 of depth k.

At each step, the protocol is such that:

- V(S) is the set of vertices of G occupied by a searcher

  $\Rightarrow$ if k is the maximum depth of S, the protocol uses at most $|V(T_k)|$ searchers.

- S is a minor of G, and S has depth $k \geq 1$ iff there exists a previous step such that S was T_{k-1}.

Idea of the upper bound : $O(n / \log n)$

Let T_k be a complete tree of degree 3 of depth k.

At each step, the protocol is such that:

- V(S) is the set of vertices of G occupied by a searcher
  - $\Rightarrow$ if k is the maximum depth of S, the protocol uses at most $|V(T_k)|$ searchers.
- S is a minor of G, and S has depth $k \geq 1$ iff there exists a previous step such that S was T_{k-1}.
  - $\Rightarrow O(k) \leq O(mcs(G, v_0))$

Idea of the upper bound : $O(n / \log n)$

Let T_k be a complete tree of degree 3 of depth k.

At each step, the protocol is such that:

- V(S) is the set of vertices of G occupied by a searcher
  - $\Rightarrow$ if k is the maximum depth of S, the protocol uses at most $|V(T_k)|$ searchers.
- S is a minor of G, and S has depth $k \geq 1$ iff there exists a previous step such that S was T_{k-1}.
  - $\Rightarrow O(k) \leq O(mcs(G,v_0))$

Then it can be proved that, if N is the number of searchers used by S:

$$N = O\left(\frac{n}{\log n} \times mcs(G, v_0)\right)$$

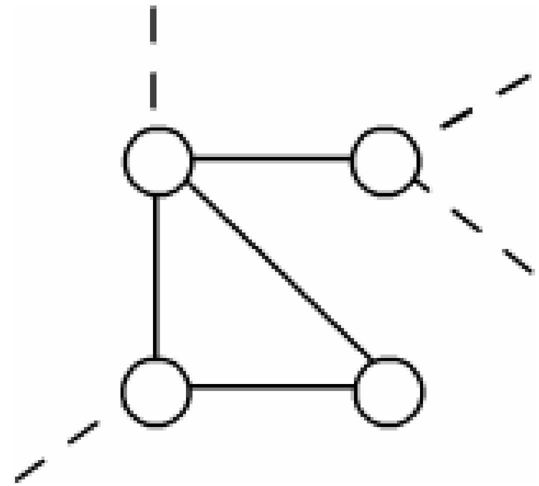
Partial graph

A partial graph is:

- a graph which can have edges with only one end;
- a half-edge is an edge with one single end;
- a full-edge is an edge with two ends.

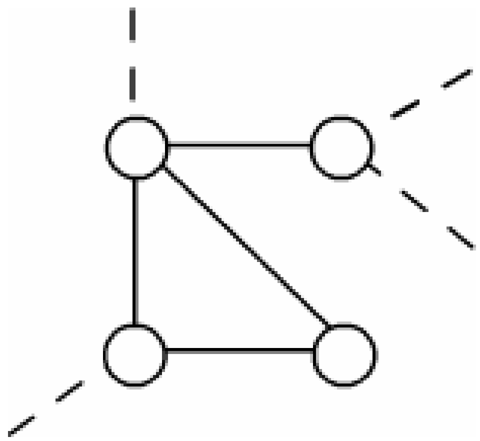
——— denotes a full-edge

– – – – denotes a half-edge

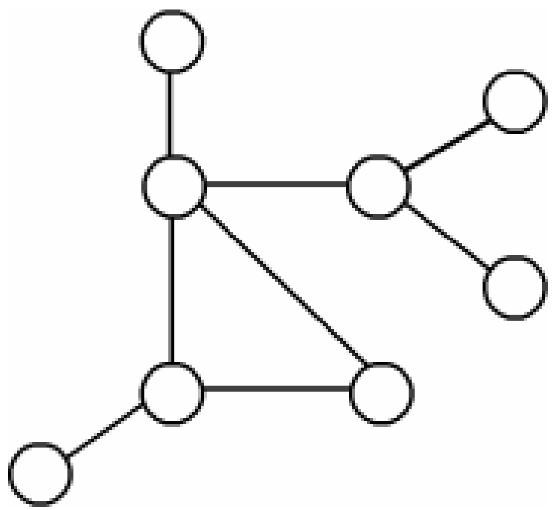


a partial graph G = (V,H,F)

Partial graph



G = (V,H,F)



the graph G^+ obtained by adding a degree-one end to any half-edge of G.

Lower bound : $\Omega(n / \log n)$

Let $\mathscr{P}$ be any protocol which solves the relaxed distributed search problem.

Game turn by turn between $\mathscr{P}$ and adversary A:

- $\mathscr{P}$ and A play alternatively, starting with $\mathscr{P}$;
- $\mathscr{P}$ clears the graph in a monotone connected way;
- The role of A is to force $\mathscr{P}$ to use the maximum number of searchers.

The Game turn by turn

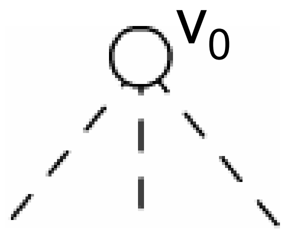
The adversary A gradually builds a $n \geq 5$ nodes tree T of degree at most 3.

The Game turn by turn

The adversary A gradually builds a n≥5 nodes tree T of degree at most 3.

Initially all searchers are placed at v_0, with T_1 is the partial graph:

The partial graph T_1



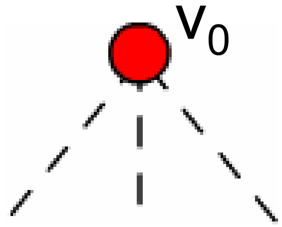
The Game turn by turn

The adversary A gradually builds a $n \geq 5$ nodes tree T of degree at most 3.
Initially all searchers are placed at v_0, with T_1 is the partial graph:

The turn i of $\mathscr{P}$:

- $\mathscr{P}$ chooses a searcher;

Example with n=10



The Game turn by turn

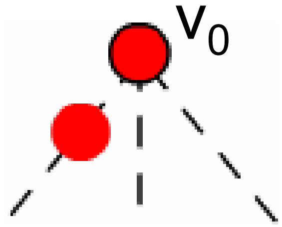
The adversary A gradually builds a $n \geq 5$ nodes tree T of degree at most 3.

Initially all searchers are placed at v_0, with T_1 is the partial graph:

The turn i of $\mathscr{P}$:

- $\mathscr{P}$ chooses a searcher;
- and moves it along an edge e of T_i.

Example with n=10



The Game turn by turn

The adversary A gradually builds a n≥5 nodes tree T of degree at most 3.
Initially all searchers are placed at v_0, with T_1 is the partial graph:

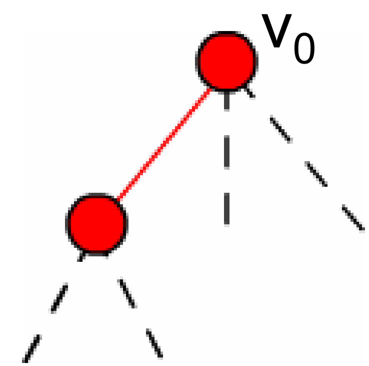
The turn i of $\mathscr{P}$:

- $\mathscr{P}$ chooses a searcher;
- and moves it along an edge e of T_i.

The turn i of A:

- If e is a half-edge, A adds a new end v to e such that v is incident to two new half-edges.

Example with n=10



The Game turn by turn

The adversary A gradually builds a $n \geq 5$ nodes tree T of degree at most 3.
Initially all searchers are placed at v_0, with T_1 is the partial graph:

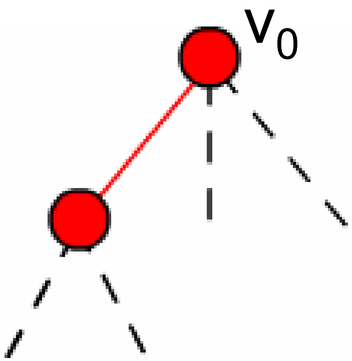
The turn i of 𝒫:

- 𝒫 chooses a searcher;
- and moves it along an edge e of T_i.

The turn i of A:

- If e is a half-edge, A adds a new end v to e such that v is incident to two new half-edges.
- If e is a full-edge, A skips is turn.

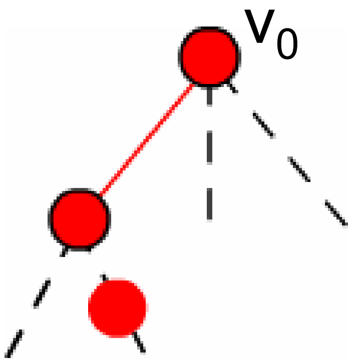
Example with n=10



The Game turn by turn

The adversary A gradually builds a $n \geq 5$ nodes tree T of degree at most 3.
Initially all searchers are placed at v_0, with T_1 is the partial graph:

Example with n=10



$|V(T_2^+)| = 6$

The turn i of 𝒫:

- 𝒫 chooses a searcher;
- and moves it along an edge e of T_i.

The turn i of A:

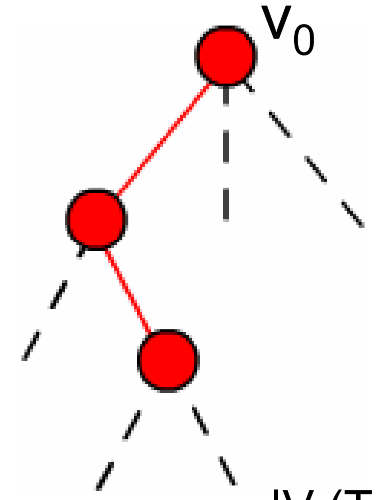
- If e is a half-edge, A adds a new end v to e such that v is incident to two new half-edges.
- If e is a full-edge, A skips is turn.

The Game turn by turn

The adversary A gradually builds a $n \geq 5$ nodes tree T of degree at most 3.

Initially all searchers are placed at v_0, with T_1 is the partial graph:

Example with n=10



The turn i of $\mathscr{P}$:

- $\mathscr{P}$ chooses a searcher;
- and moves it along an edge e of T_i.

The turn i of A:

- If e is a half-edge, A adds a new end v to e such that v is incident to two new half-edges.
- If e is a full-edge, A skips is turn.

The Game turn by turn

The adversary A gradually builds a $n \geq 5$ nodes tree T of degree at most 3.
Initially all searchers are placed at v_0, with T_1 is the partial graph:

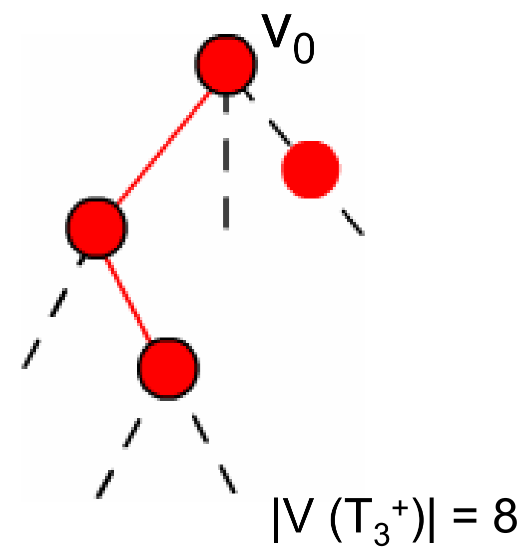
The turn i of 𝒫:

- 𝒫 chooses a searcher;
- and moves it along an edge e of T_i.

The turn i of A:

- If e is a half-edge, A adds a new end v to e such that v is incident to two new half-edges.
- If e is a full-edge, A skips is turn.

Example with n=10



The Game turn by turn

The adversary A gradually builds a n≥5 nodes tree T of degree at most 3.
Initially all searchers are placed at v_0, with T_1 is the partial graph:

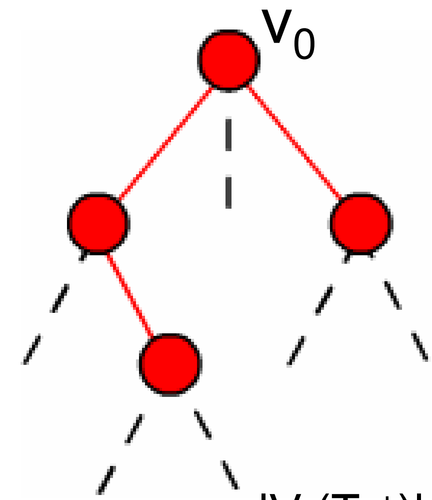
The turn i of $\mathscr{P}$:

- $\mathscr{P}$ chooses a searcher;
- and moves it along an edge e of T_i.

The turn i of A:

- If e is a half-edge, A adds a new end v to e such that v is incident to two new half-edges.
- If e is a full-edge, A skips is turn.

Example with n=10

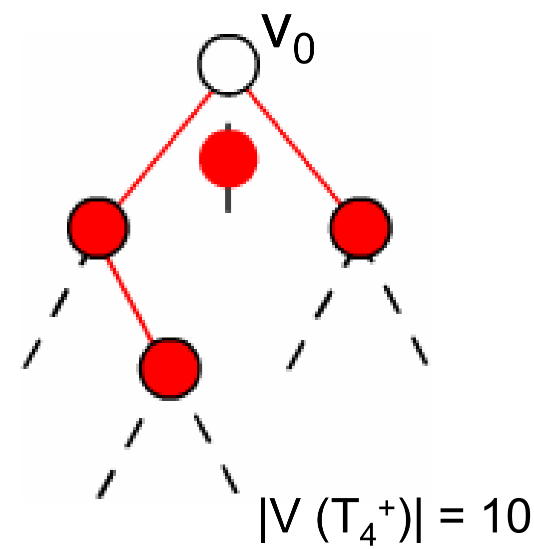


$|V(T_4^+)| = 10$

The Game turn by turn

The adversary A gradually builds a $n \geq 5$ nodes tree T of degree at most 3.
Initially all searchers are placed at v_0, with T_1 is the partial graph:

Example with n=10



The turn i of 𝒫:

- 𝒫 chooses a searcher;
- and moves it along an edge e of T_i.

The turn i of A:

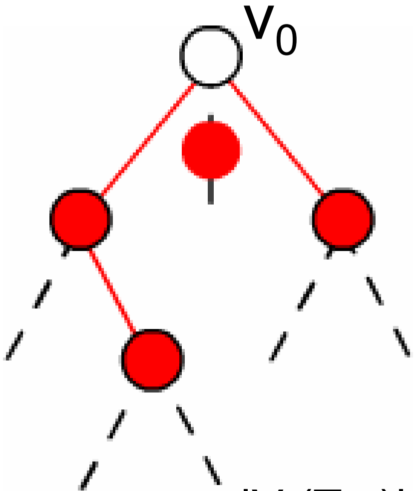
- If e is a half-edge, A adds a new end v to e such that v is incident to two new half-edges.
- If e is a full-edge, A skips is turn.

The Game turn by turn

The adversary A gradually builds a $n \geq 5$ nodes tree T of degree at most 3.

Initially all searchers are placed at v_0, with T_1 is the partial graph:

Example with n=10



$|V(T_4^+)| = 10$

The turn i of 𝒫:

- 𝒫 chooses a searcher;
- and moves it along an edge e of T_i.

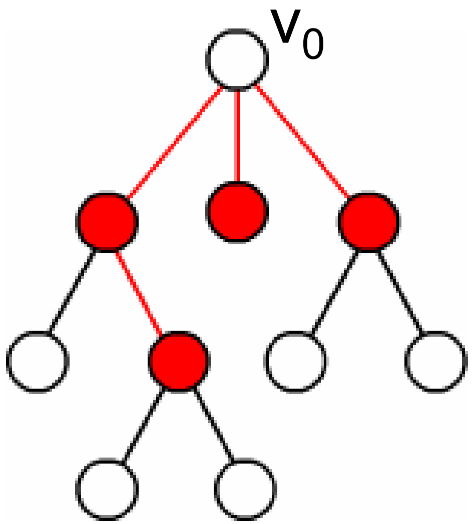
The turn i of A:

- If e is a half-edge, A adds a new end v to e such that v is incident to two new half-edges.
- If e is a full-edge, A skips is turn.
- When $|V(T_p^+)| = n$. A decides that the graph T is actually T_p^+.

The Game turn by turn

The adversary A gradually builds a $n \geq 5$ nodes tree T of degree at most 3.
Initially all searchers are placed at v_0, with T_1 is the partial graph:

Example with n=10



The turn i of 𝒫:

- 𝒫 chooses a searcher;
- and moves it along an edge e of T_i.

The turn i of A:

- If e is a half-edge, A adds a new end v to e such that v is incident to two new half-edges.
- If e is a full-edge, A skips is turn.
- When $|V(T_p^+)| = n$. A decides that the graph T is actually T_p^+.

Lower bound : $\Omega(n / \log n)$

The tree T is such that:

- T is a tree with at least $(n + 2)/2$ leaves.

  $\Rightarrow$ P uses at least $k \geq n/4$ searchers.

Lower bound : $\Omega(n / \log n)$

The tree T is such that:

- T is a tree with at least $(n + 2)/2$ leaves.

  $\Rightarrow$ P uses at least $k \geq n/4$ searchers.
- **s**(G): smallest number of searchers that are necessary to clear a graph G without the constraints of monotonicity and connectivity.

Theorem: Let G be a tree with $n \geq 2$ vertices,

s(G)$\leq 1 + \log_3(n - 1)$ [Megiddo *et al.* 1988]

For any $v_0 \in V(G)$, **mcs**$(G, v_0) \leq 2$**s**$(G) - 1$ [Barrière *et al.* 2003]

$\Rightarrow$ **mcs**$(T, v_0) \leq 2(1 + \log_3(n - 1))$

Lower bound : $\Omega(n / \log n)$

The tree T is such that:

- T is a tree with at least $(n + 2)/2$ leaves.

  $\Rightarrow$ P uses at least $k \geq n/4$ searchers.
- **s**(G): smallest number of searchers that are necessary to clear a graph G without the constraints of monotonicity and connectivity.

Theorem: Let G be a tree with $n \geq 2$ vertices,

s(G)$\leq 1 + \log_3(n - 1)$ [Megiddo *et al.* 1988]

For any $v_0 \in V(G)$, **mcs**$(G, v_0) \leq 2$**s**$(G) - 1$ [Barrière *et al.* 2003]

$\Rightarrow$ **mcs**$(T, v_0) \leq 2(1 + \log_3(n - 1))$

Then it can be proved that there is a constant $c > 0$ such that for any $n \geq 5$ we have

$$k \geq c\left(\frac{n}{\log n}\right)\mathbf{mcs}(T, v_0)$$

Conclusion and Perspectives

In decentralized settings:

- $\Theta(n \log n)$ bits of information are necessary and sufficient to clear any n-node graph in a monotone connected and optimal way.
- Any distributed protocol aiming at clearing any unknown n-node graph in a monotone connected way has competitive ratio $\Theta(n / \log n)$.

Conclusion and Perspectives

In decentralized settings:

- $\Theta(n \log n)$ bits of information are necessary and sufficient to clear any n-node graph in a monotone connected and optimal way.
- Any distributed protocol aiming at clearing any unknown n-node graph in a monotone connected way has competitive ratio $\Theta(n / \log n)$.

It would be interesting to establish a tradeoff between the number of searchers that are required to clear any graph G in a monotone connected way and the amount of information that must be provided to the searchers.

Conclusion and Perspectives

In decentralized settings:

- $\Theta(n \log n)$ bits of information are necessary and sufficient to clear any n-node graph in a monotone connected and optimal way.
- Any distributed protocol aiming at clearing any unknown n-node graph in a monotone connected way has competitive ratio $\Theta(n / \log n)$.

It would be interesting to establish a tradeoff between the number of searchers that are required to clear any graph G in a monotone connected way and the amount of information that must be provided to the searchers.

An other problem is to improve the competitive ratio of a search protocol by allowing the search strategy to be not monotone while it is performed in polynomial time.