Distributed Graph Searching

Nicolas Nisse

project Anillo en Redes / Departamento de Ingenieria Matematica
Universidad de Chile, Santiago, Chile.

GRASTA, Ceara, February 2008

Based on joint works with: Lélia Blin, Pierre Fraigniaud, David Ilcinkas, David Soguet and Sandrine Vial

Plan

Graph Searching in a distributed way

- an unknown network and an entry (starting point) ;
- a team of robots that aims at clearing it ;
- Goal : design of a program.

Graph Searching in a distributed way

The Problem

To design a *distributed protocol* that enables the *minimum number* of searchers to clear any unknown network.
The searchers must compute themselves a strategy in the network whitout having any global knowledge of it.

Plan

Which model of graph searching ?

Drawbacks of the standard model :

1. the network is *known* (its size, its topology, etc.) ;
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 network ;
2. networks may be asynchronous ;
3. searchers cannot be teleported,
   communications must be safe,
   $\Rightarrow$ the clear part must induce a connected subgraph.

Way of clearing

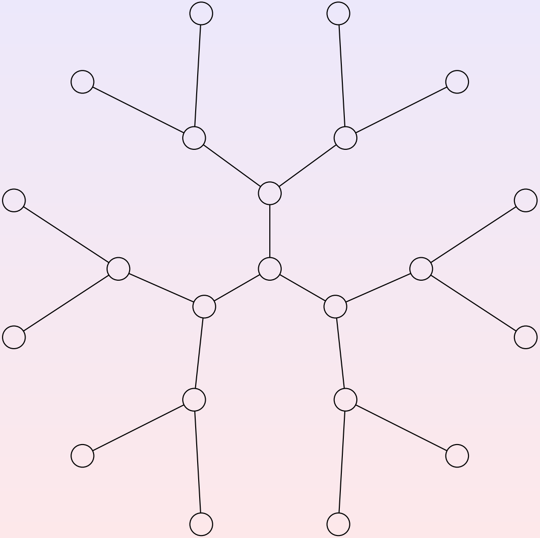
Monotone connected search strategy

- **edge-search :** An edge is cleared when it is traversed by a searcher.
- **connectedness :** At any step of the strategy, the clear part must induce a connected subgraph.
- **monotonicity :** No recontamination ever occurs. Once an edge has been cleared, it remains clear until the end.

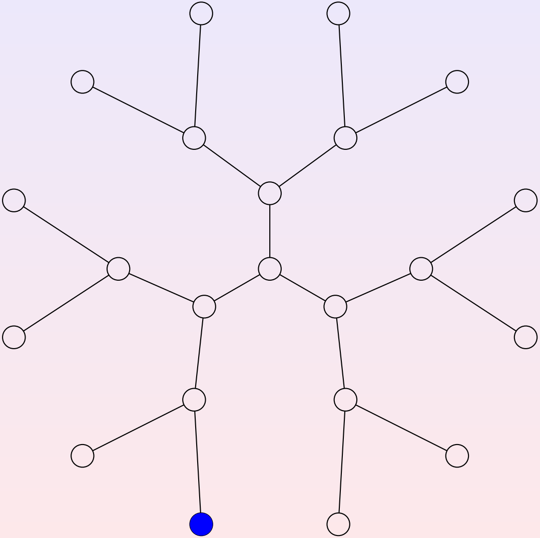
Monotone connected search number

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

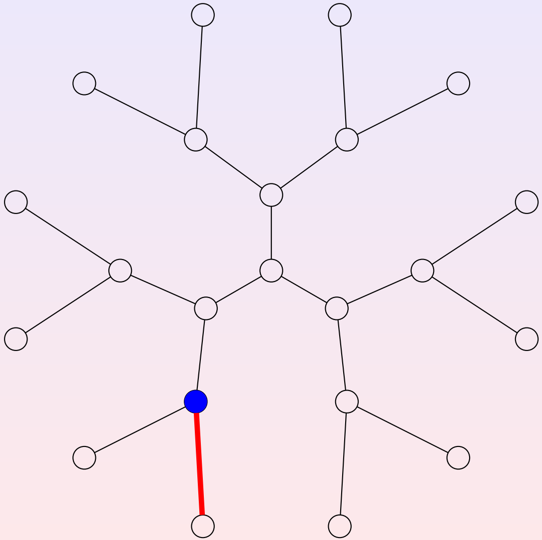
Cost of Connectedness : Example in a tree



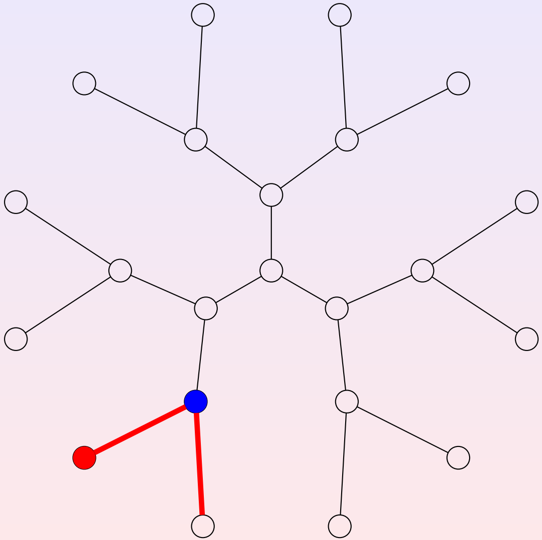
Cost of Connectedness : Example in a tree



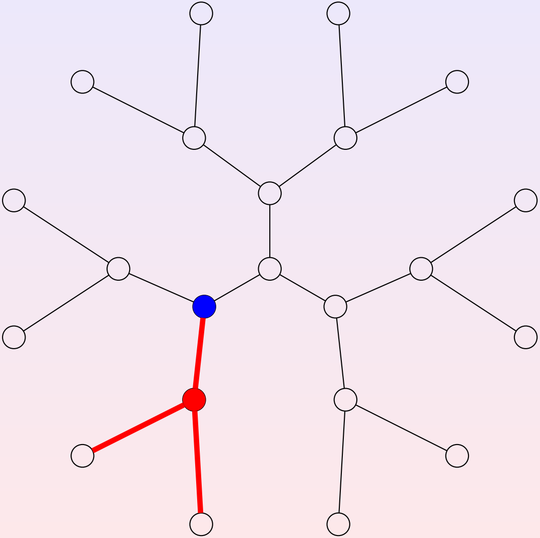
Cost of Connectedness : Example in a tree



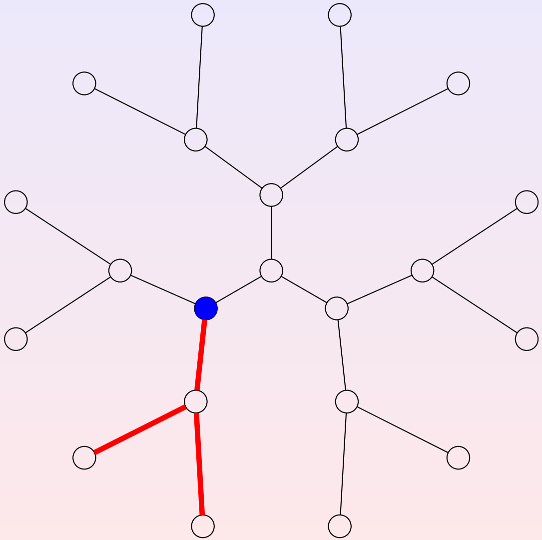
Cost of Connectedness : Example in a tree



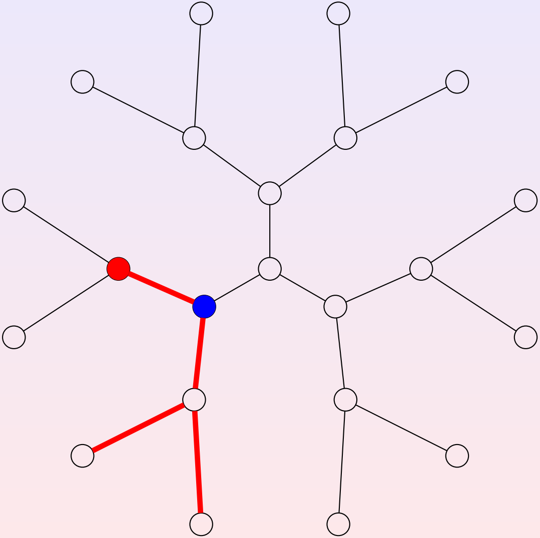
Cost of Connectedness : Example in a tree



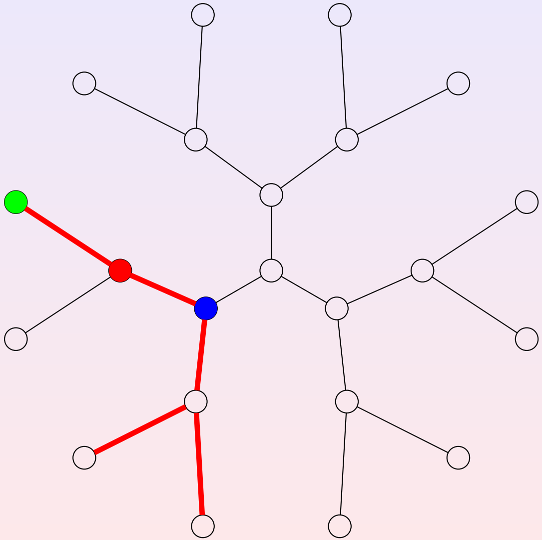
Cost of Connectedness : Example in a tree



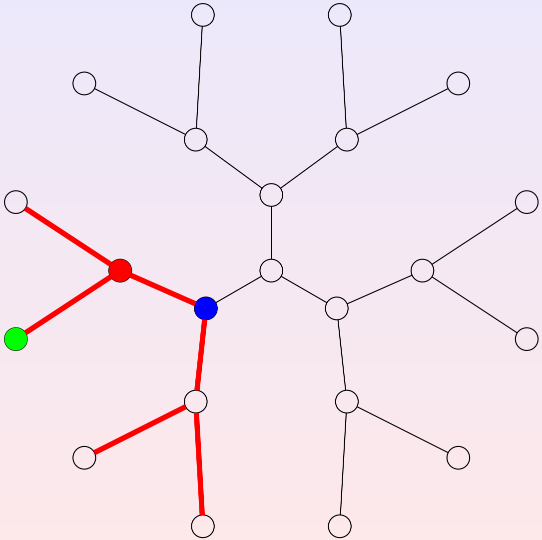
Cost of Connectedness : Example in a tree



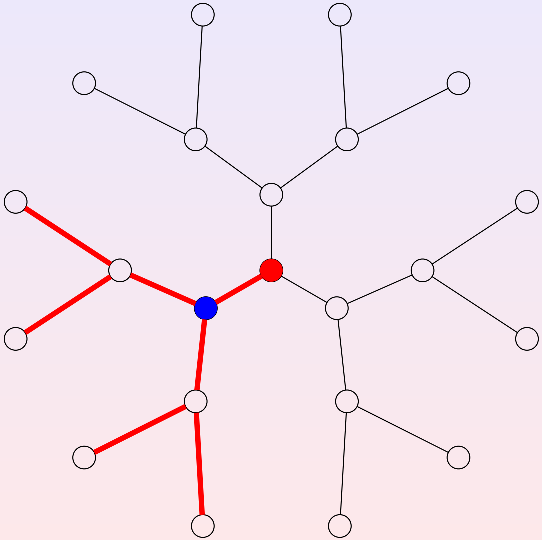
Cost of Connectedness : Example in a tree



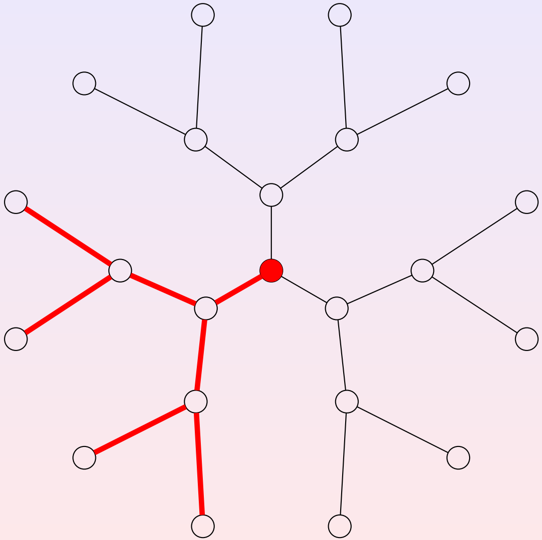
Cost of Connectedness : Example in a tree



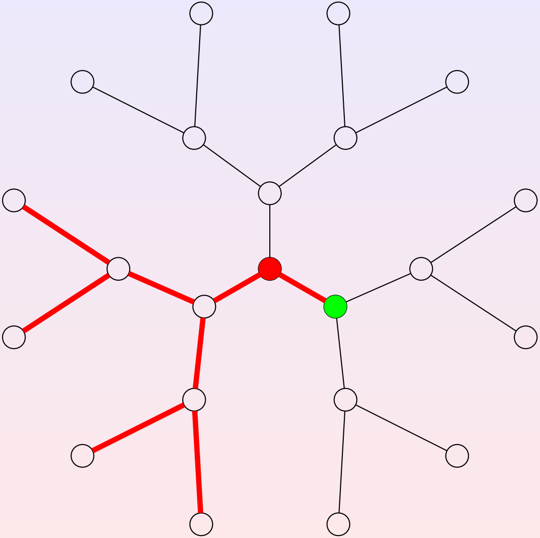
Cost of Connectedness : Example in a tree



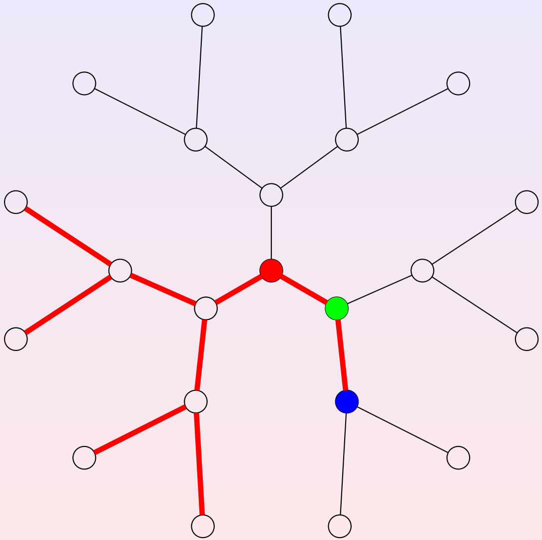
Cost of Connectedness : Example in a tree



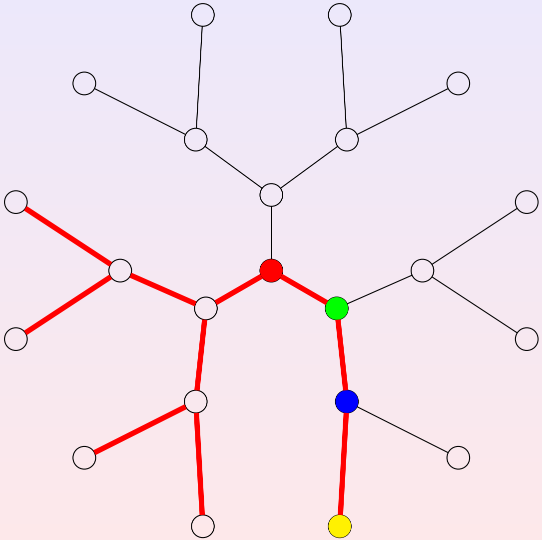
Cost of Connectedness : Example in a tree



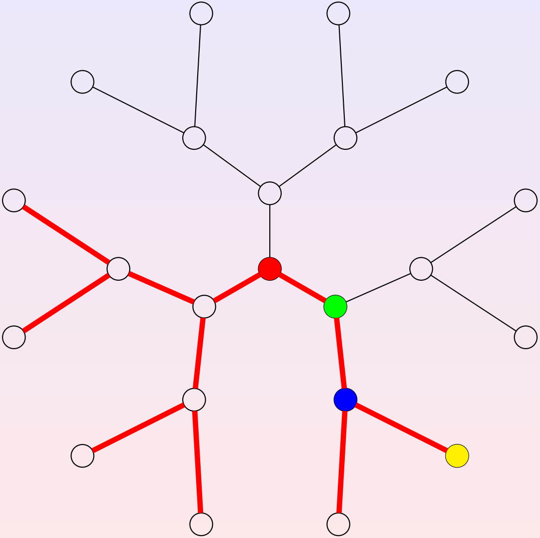
Cost of Connectedness : Example in a tree



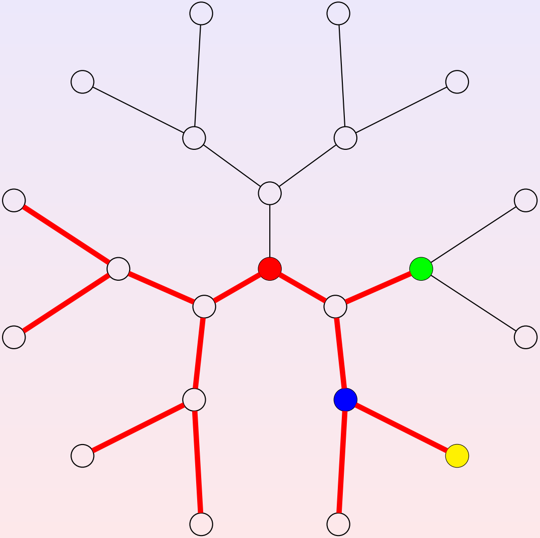
Cost of Connectedness : Example in a tree



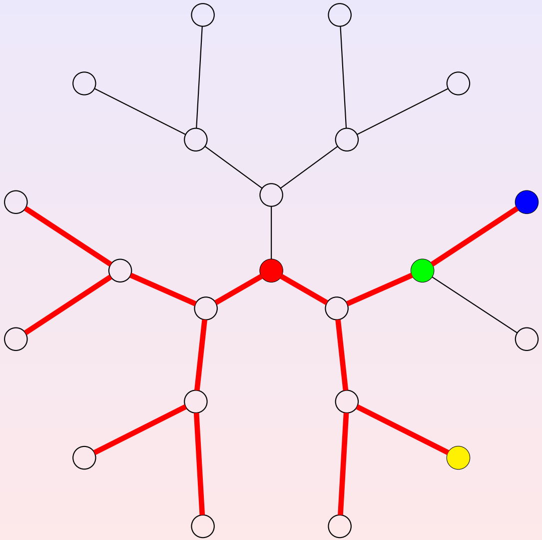
Cost of Connectedness : Example in a tree



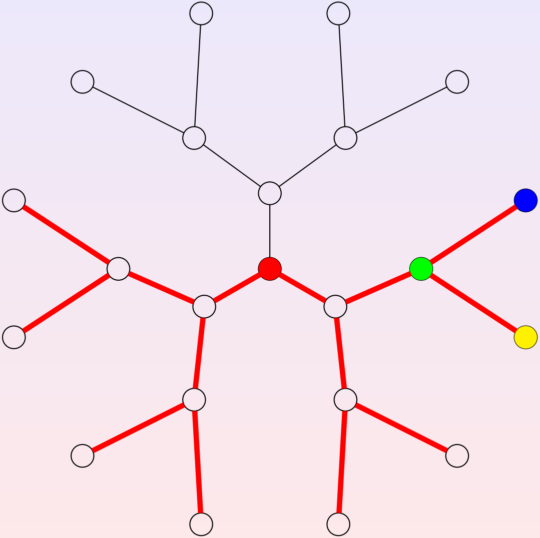
Cost of Connectedness : Example in a tree



Cost of Connectedness : Example in a tree

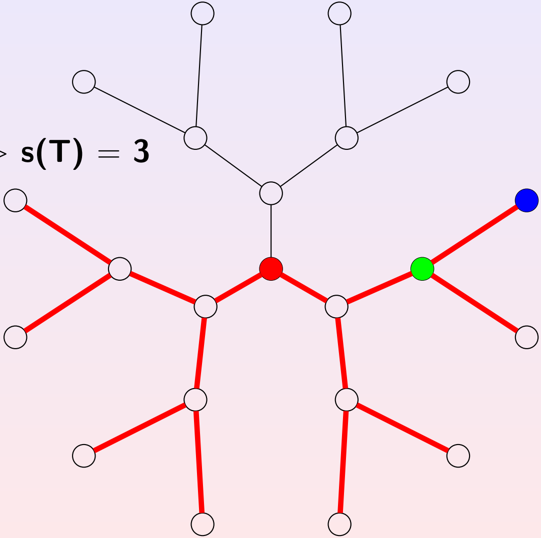


Cost of Connectedness : Example in a tree



Cost of Connectedness : Example in a tree

$\mathbf{mcs(T) = 4 > s(T) = 3}$



Cost in terms of number of searchers

Cost of connectedness

- 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]

Does Recontamination help ?

- It does not help to clear a tree in a connected way. [*Barrière, Flocchini, Fraigniaud and Santoro*. SPAA, 2002]
- There are graphs for which imposing monotonicity to connected search strategies requires strictly more searchers. [*Yang, Dyer and Alspach*. ISAAC, 2004]

Cost in terms of number of searchers

Cost of connectedness

- 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]

Does Recontamination help ?

- It does not help to clear a tree in a connected way. [*Barrière, Flocchini, Fraigniaud and Santoro*. SPAA, 2002]
- There are graphs for which imposing monotonicity to connected search strategies requires strictly more searchers. [*Yang, Dyer and Alspach*. ISAAC, 2004]

Plan

Monotone connected graph searching

Given a graph G and a vertex $v_0 \in V(G)$.

Alternative definition

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

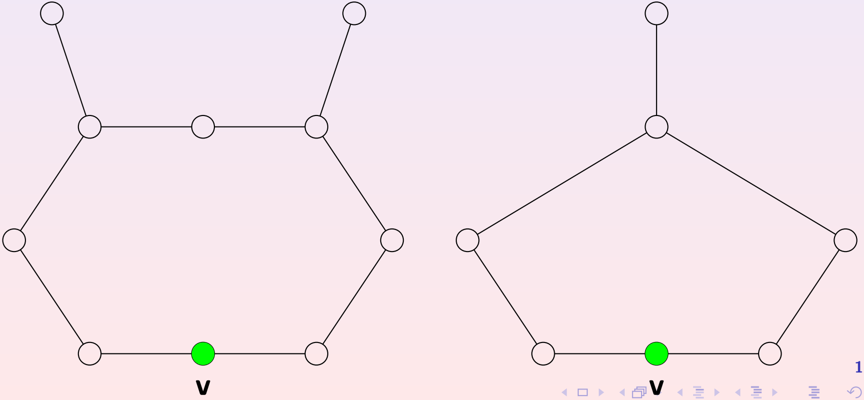
- Initially, all searchers are placed at v_0.
- One single operation is allowed : move a searcher along an edge if it does not imply recontamination.

Remarks

- The homebase remains clear during the whole strategy.
- **mcs**(G, v_0)

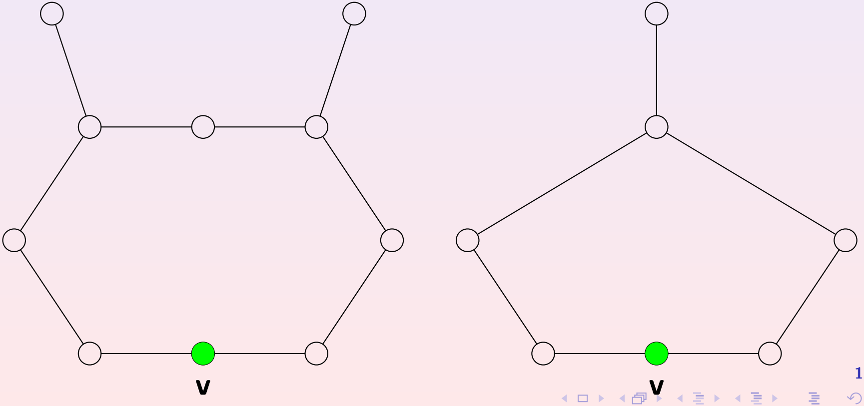
Cost of asynchronicity

Networks are asynchronous : a move takes a finite time, but no bound is known about it.
Is it more difficult to clear an asynchronous graph than a synchronous graph ? Does it cost searchers ?



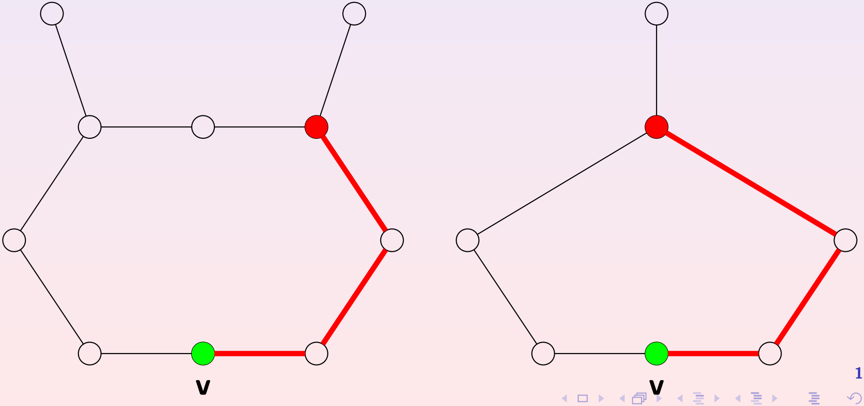
Cost of asynchronicity

Let $\mathcal{P}$ be a distributed protocol that allows to clear any unknown asynchronous graph in a monotone connected way. Is it possible for it to use the optimal number (**mcs**) of searchers ?



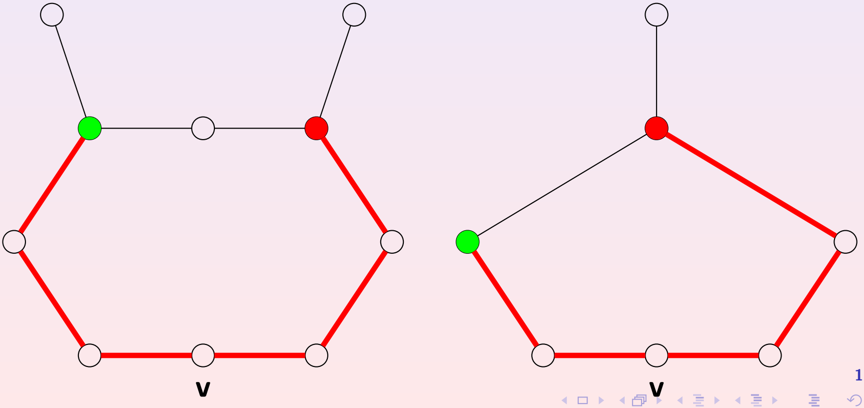
Cost of asynchronicity

Let $\mathcal{P}$ be a distributed protocol that allows to clear any unknown asynchronous graph in a monotone connected way. Is it possible for it to use the optimal number (**mcs**) of searchers ?



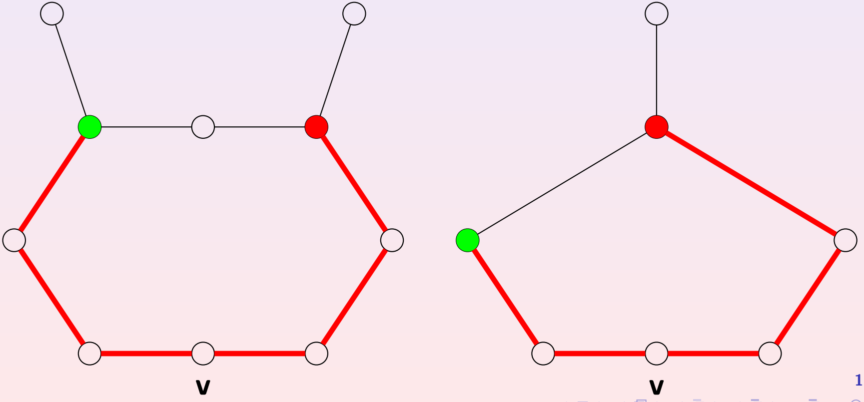
Cost of asynchronicity

Let $\mathcal{P}$ be a distributed protocol that allows to clear any unknown asynchronous graph in a monotone connected way.
Is it possible for it to use the optimal number (**mcs**) of searchers ?



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 classes of graphs such that, any distributed asynchronous graph searching protocol requires $\mathbf{mcs}(G)+1$ searchers to clear G in a connected monotone way. [*Flocchini, Huang and Luccio*. IPDPS 05]

Coordinator

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

Cost of asynchronicity

More generally

There exist classes of graphs such that, any distributed asynchronous graph searching protocol requires $\mathbf{mcs}(G) + 1$ searchers to clear G in a connected monotone way. [*Flocchini, Huang and Luccio*. IPDPS 05]

Coordinator

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

Anonymous Network

Unknown

- unknown topology
- unknown size (no upper bound)

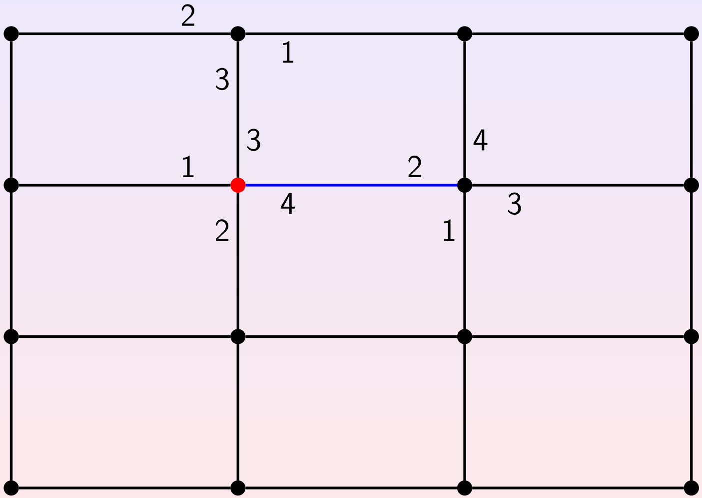
Local Memory on nodes

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

Anonymous

- No vertex labeling
- Local edge labeling

Example of an anonymous graph



Mobile Agents

Searchers

- autonomous mobile computing entities with distincs IDs,
- running the same algorithm (but the coordinator),
- Mealy automata.

Mobile Agents

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.

Distributed graph searching : related work

Protocols have been designed to clear some specific topologies.
The searchers have a prior knowledge of the topology.

Protocols to clear specific topologies

- **Mesh**. Flocchini, Luccio, and Song. [CIC 05]
- **Hypercube**. Flocchini, Huang, and Luccio. [IPDPS 05]
- **Tori**. Flocchini, Luccio, and Song. [IPDPS 06]
- **Sierpinski's graph**. Luccio. [FUN 07]

- A monotone connected strategy is performed using **mcs** $+ 1$ searchers
- Each searcher possesses $O(\log n)$ bits of memory ;
- The size of the node's whiteboard is $O(\log n)$ bits.

Distributed graph searching : related work

Protocols have been designed to clear some specific topologies.
The searchers have a prior knowledge of the topology.

Protocols to clear specific topologies

- **Mesh**. Flocchini, Luccio, and Song. [CIC 05]
- **Hypercube**. Flocchini, Huang, and Luccio. [IPDPS 05]
- **Tori**. Flocchini, Luccio, and Song. [IPDPS 06]
- **Sierpinski's graph**. Luccio. [FUN 07]

- A monotone connected strategy is performed using **mcs** $+ 1$ searchers
- Each searcher possesses $O(\log n)$ bits of memory ;
- The size of the node's whiteboard is $O(\log n)$ bits.

Distributed graph searching : related work

Protocols have been designed to clear some specific topologies. The searchers have a prior knowledge of the topology.

Protocols to clear specific topologies

- **Hypercube**. Flocchini, Huang, and Luccio. [IPDPS 05]
- **Mesh**. Flocchini, Luccio, and Song. [CIC 05]
- **Tori**. Flocchini, Luccio, and Song. [IPDPS 06]
- **Sierpinski's graph**. Luccio. [FUN 07]

Question :

Is it possible to design a distributed protocol that allows to clear any unknown graph ?

Plan

Theorem [*Blin, Fraigniaud, Nisse and Vial*. TCS 08]

We propose an distributed algorithm that enables to clear any connected, asynchronous, anonymous and unknown network G, in a connected way and starting from any homebase v_0.

1. It uses at most $k = \mathbf{mcs}(G, v_0) + 1$ searchers if $\mathbf{mcs}(G, v_0) > 1$, and $k = 1$ searcher otherwise ;
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 [*Blin, Fraigniaud, Nisse and Vial.* TCS 08]

We propose an distributed algorithm that enables to clear any connected, asynchronous, anonymous and unknown network G, in a connected way and starting from any homebase v_0.

1. It uses at most $k = \mathbf{mcs}(G, v_0) + 1$ searchers if $\mathbf{mcs}(G, v_0) > 1$, and $k = 1$ searcher otherwise ;
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.

Principles of the distributed algorithm

The Algorithm

`Initially, one searcher stands at` v_0`,` $k = 1$

`While the graph is not clear :`

- `Try all monotone connected search strategies (starting from` v_0`) using` k `searchers ;`
- `If the graph is not clear, call a new searcher (k++) ;`

Predicate

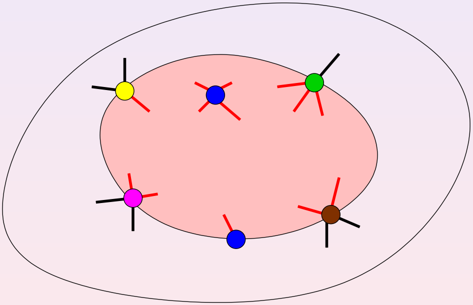
At the end of each loop, the k searchers are standing at v_0.

Basic Idea

- **to order** the possible strategies using k searchers ;
- to try all the strategies in the increasing order ;
  Somehow, we perform a guided-DFS of the graph of configurations.
- either a strategy clears the graph $\longrightarrow$ OK ;
  or after trying all the strategies, the graph remains contaminated $\longrightarrow$ one searcher more is required.

Valid moves

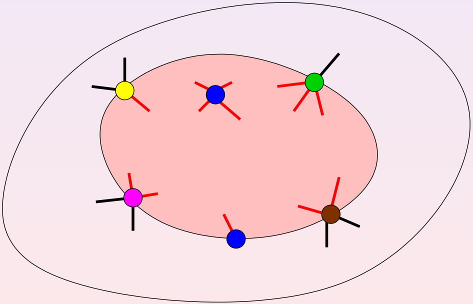
Two kinds of moves are compatible with a monotone connected strategy



Valid moves

Two kinds of moves are compatible with a monotone connected strategy

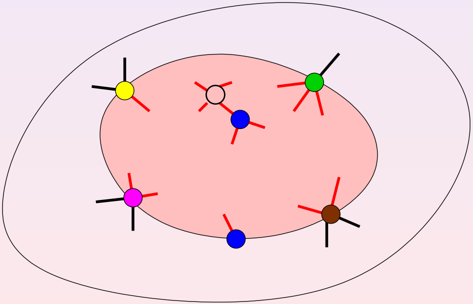
(1) A searcher at a clear (or guarded) vertex can move through any clear port.



Valid moves

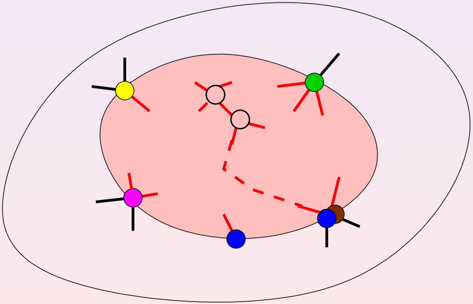
Two kinds of moves are compatible with a monotone connected strategy

(1) A searcher at a clear (or guarded) vertex can move through any clear port.



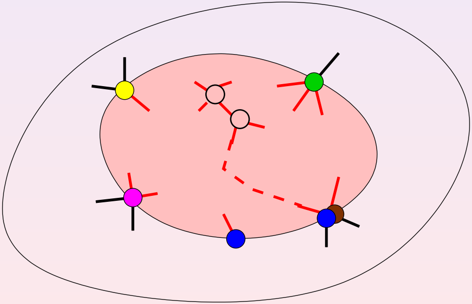
Valid moves

Two kinds of moves are compatible with a monotone connected strategy
(1) A searcher at a clear (or guarded) vertex can move through the clear part **to help another searcher**,



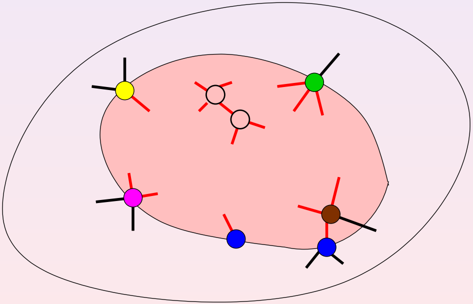
Valid moves

Two kinds of moves are compatible with a monotone connected strategy
(1) A searcher at a clear (or guarded) vertex can move **to help another searcher**, and then **to clear an edge**.



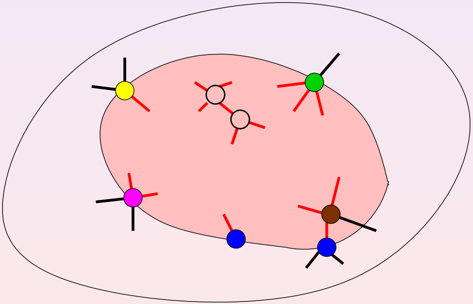
Valid moves

Two kinds of moves are compatible with a monotone connected strategy
(1) A searcher at a clear (or guarded) vertex can move **to help another searcher**, and then **to clear an edge**.



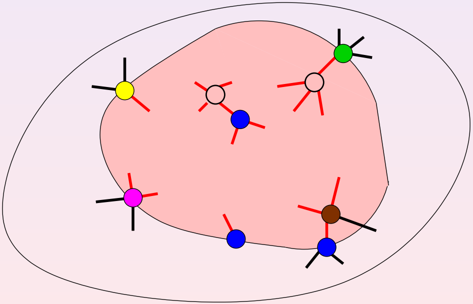
Valid moves

Two kinds of moves are compatible with a monotone connected strategy
(2) A searcher at a vertex incident to only one contaminated port can move **to clear the corresponding edge**.



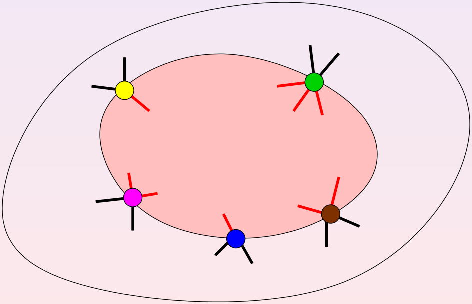
Valid moves

Two kinds of moves are compatible with a monotone connected strategy
(2) A searcher at a vertex incident to only one contaminated port can move **to clear the corresponding edge**.



Valid moves

Two kinds of moves are compatible with a monotone connected strategy
Such a configuration is the result of **a failing strategy** and will lead to **backtracking**.



Ordering on moves and strategies

Representation of valid moves

(i, j, p) denotes : “searcher i joins searcher j and the smallest searcher follows the port p to clear the corresponding edge”.
(i, i, p) denotes : “searcher i follows the port p to clear the corresponding edge”.
The moves are ordered in the lexicographical order.

Ordering on the sequences of valid moves

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

Proof of Correctness

1. only valid moves are performed $\longrightarrow$ only valid strategies are performed.
2. valid strategies are performed in the lexicographical order $\longrightarrow$ all valid strategies are performed.
3. if the graph is clear, the algorithm stops and if all strategies with k searchers have been tried, the algorithm ask for an extra searcher $\longrightarrow$ our algorithm terminates

Searchers know that the graph is clear when all of them are occupying some vertex with no incident contaminated edge.

Some difficulties we have to adress

Only one move at time :

the coordinator acts like a token.

- It looks for the searcher that can perform the smallest valid move ;
- If none valid move is possible, it looks for the last searcher that has moved.

Some difficulties we have to adress

Only one move at time :

the coordinator acts like a token.

It must be possible to backtrack :

all actions performed by the searchers are written in stacks distributed on the whiteboards.

- The last searcher that has moved backtracks its last action ;
- Then, the coordinator looks for the next valid move to be performed.

Some difficulties we have to adress

Only one move at time :

the coordinator acts like a token.

It must be possible to backtrack :

all actions performed by the searchers are written in stacks distributed on the whiteboards.

A searcher must be able to find another searcher :

A trace of any searcher is written in arrays on the whiteboards.

- When leaving a node, a searcher writes the port it used.

Drawbacks of our Protocol

Whiteboards of size $O(m \log n)$

It would be interesting to reduce it.

Finite automata

Our Protocol can be implemented using finite automata in the class of graph with bounded **mcs**. Is it possible to do better ?

Monotonicity

The strategy performed by our Protocol is not monotone (we need to backtrack).
Not surprising but bothersome because the clearing may be performed in exponential time.

Plan

How to force Monotonicity ?

Recall : The Problem

To design a *distributed protocol* for the minimum number of searchers (i.e., **mcs**) to clear any network in a monotone connected way.
The searchers must compute themselves a strategy in the network whitout having any global knowledge of it.

How to force Monotonicity ?

Recall : The Problem

To design a *distributed protocol* for the minimum number of searchers (i.e., **mcs**) to clear any network in a monotone connected way.
The searchers must compute themselves a strategy in the network whitout having any global knowledge of it.

If the *monotonicity property* is relaxed $\longrightarrow$ OK ;

If the searchers know the topology of the network in which they are launched $\longrightarrow$ OK for some topologies (mesh, hypercube, etc.)

How to force Monotonicity ?

Recall : The Problem

To design a *distributed protocol* for the minimum number of searchers (i.e., **mcs**) to clear any network in a monotone connected way.
The searchers must compute themselves a strategy in the network whitout having any global knowledge of it.

Two opposite Approaches

- If we impose the number of searchers to be optimal, What is the **minimum quantity of information** that must be provided about the graph ?
- If we impose the graph to be unknown, What is the **minimum number of searchers** that must be used ?

Results

Extra Information

$\Theta(n \log n)$ bits of information must be provided to the **mcs**(G) searchers to clear any n-node graph G in a distributed monotone connected way.
[*Nisse and Soguet*. SIROCCO 07]

Extra Searchers

$\Theta(\frac{n}{\log n}) mcs(G)$ searchers are necessary and sufficient to clear any unknown n-node graph G in a distributed connected *monotone* way. [*Ilcinkas, Nisse and Soguet*. OPODIS 07]

To clear a graph with advice

What is the information that must be given to the searchers such that it exists a distributed protocol that enables them to clear all graphs in a monotone connected and optimal way ?

What kind of knowledge ?

Qualitative information

- Topology, size, diameter of the network ...

Quantitative information : advice [Fraigniaud *et al.* PODC06]

- Measure the **minimum number of bits of information** to **efficiently** perform a distributed task.

To clear a graph with advice

What is the information that must be given to the searchers such that it exists a distributed protocol that enables them to clear all graphs in a monotone connected and optimal way ?

What kind of knowledge ?

Qualitative information

- Topology, size, diameter of the network ...

Quantitative information : advice [Fraigniaud *et al.* PODC06]

- Measure the **minimum number of bits of information** to **efficiently** perform a distributed task.

To clear a graph with advice

What is the information that must be given to the searchers such that it exists a distributed protocol that enables them to clear all graphs in a monotone connected and optimal way ?

What kind of knowledge ?

Qualitative information

- Topology, size, diameter of the network ...

Quantitative information : advice [Fraigniaud *et al.* PODC06]

- Measure the **minimum number of bits of information** to **efficiently** perform a distributed task.

Advice, size of advice [Fraigniaud *et al.* 06]

A distributed problem $\mathcal{P}$

Instance of $\mathcal{P}$ (for example a graph G)

Advice : information that can be used to solve $\mathcal{P}$ *efficiently*

Information is modeled by

- An oracle $\mathcal{O}$ that assigns at any instance G a string of bits $\mathcal{O}(G)$ that is distributed on the vertices of G.
- size of advice $|\mathcal{O}(G)|$

Examples

- wake-up (linear number of messages) : $\Theta(n \log n)$ bits ;
- broadcast (linear number of messages) : $O(n)$ bits ;
- tree exploration, MST, graph coloring ...

Advice, size of advice [Fraigniaud *et al.* 06]

A distributed problem $\mathcal{P}$

Instance of $\mathcal{P}$ (for example a graph G)

Advice : information that can be used to solve $\mathcal{P}$ *efficiently*

Information is modeled by

- An oracle $\mathcal{O}$ that assigns at any instance G a string of bits $\mathcal{O}(G)$ that is distributed on the vertices of G.
- size of advice $|\mathcal{O}(G)|$

Examples

- wake-up (linear number of messages) : $\Theta(n \log n)$ bits ;
- broadcast (linear number of messages) : $O(n)$ bits ;
- tree exploration, MST, graph coloring ...

Advice, size of advice [Fraigniaud *et al.* 06]

Problem : distributed search problem
Instance : an unknown graph G and a homebase $v_0 \in V(G)$.
Advice : to monotoneously clear G using $\mathbf{mcs}(G)$ searchers.

Information is modeled by

- An oracle $\mathcal{O}$ that distributes a string of bits $\mathcal{O}(G, v_0)$ on the vertices of G.
- size of advice $|\mathcal{O}(G, v_0)|$

Question :
What is the minimum size of advice such that it exists a distributed protocol that efficiently solves the distributed search problem ?

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

Upper bound

$O(n \log n)$ bits of advice are sufficient to solve the distributed search problem.

We design an oracle $\mathcal{O}$ of size $O(n \log n)$ bits and a distributed protocol $\mathcal{P}$ using $\mathcal{O}$ that clears all graphs in a monotone connected and optimal way.

- Automata with $O(\log n)$ bits of memory.
- Node's whiteboard of size $O(\log n)$ bits.

Let S be a monotone connected and optimal strategy for G.
$S \longrightarrow$ order on the vertices and a spanning tree T of G.
Roughly, our oracle "encodes" T on the vertices of G.

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

Upper bound

$O(n \log n)$ bits of advice are sufficient to solve the distributed search problem.

We design an oracle $\mathcal{O}$ of size $O(n \log n)$ bits and a distributed protocol $\mathcal{P}$ using $\mathcal{O}$ that clears all graphs in a monotone connected and optimal way.

- Automata with $O(\log n)$ bits of memory.
- Node's whiteboard of size $O(\log n)$ bits.

Let S be a monotone connected and optimal strategy for G.
$S \rightarrow$ order on the vertices and a spanning tree T of G.
Roughly, our oracle "encodes" T on the vertices of G.

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

Upper bound

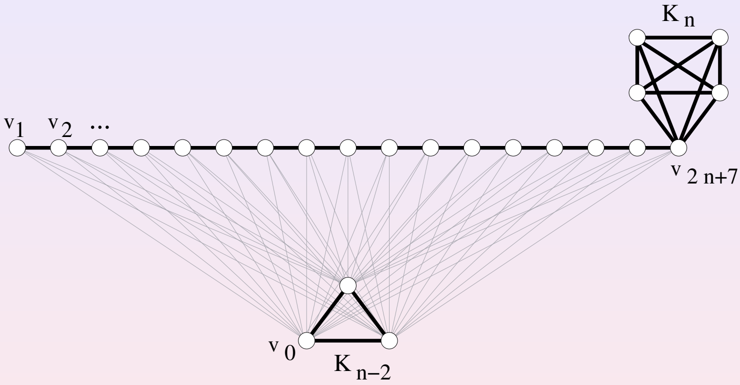
$O(n \log n)$ bits of advice are sufficient to solve the distributed search problem.

We design an oracle $\mathcal{O}$ of size $O(n \log n)$ bits and a distributed protocol $\mathcal{P}$ using $\mathcal{O}$ that clears all graphs in a monotone connected and optimal way.

- Automata with $O(\log n)$ bits of memory.
- Node's whiteboard of size $O(\log n)$ bits.

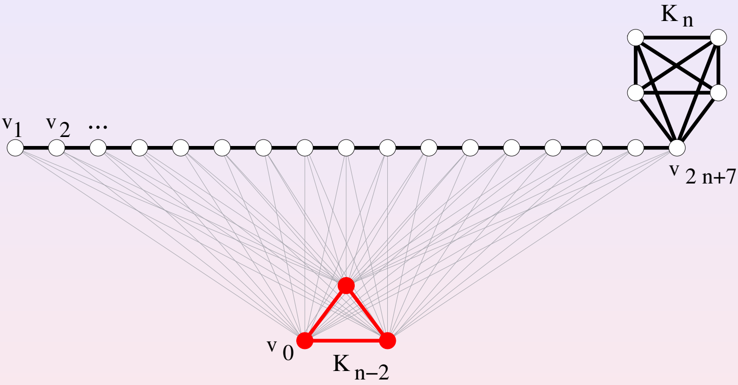
Let S be a monotone connected and optimal strategy for G.
$S \longrightarrow$ order on the vertices and a spanning tree T of G.
Roughly, our oracle "encodes" T on the vertices of G.

The lower bound : $\Omega(n \log n)$



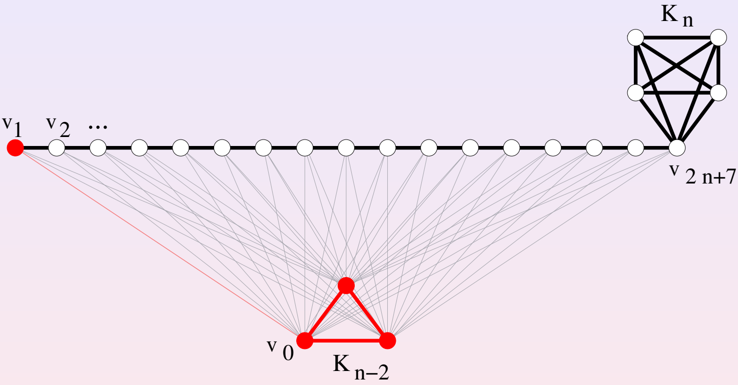
Class of graphs $(G_n)_{n \in N}$ (The figure corresponds to G_5).
All the monotone connected and optimal strategies in this class are **strongly constrained**.

The lower bound : $\Omega(n \log n)$



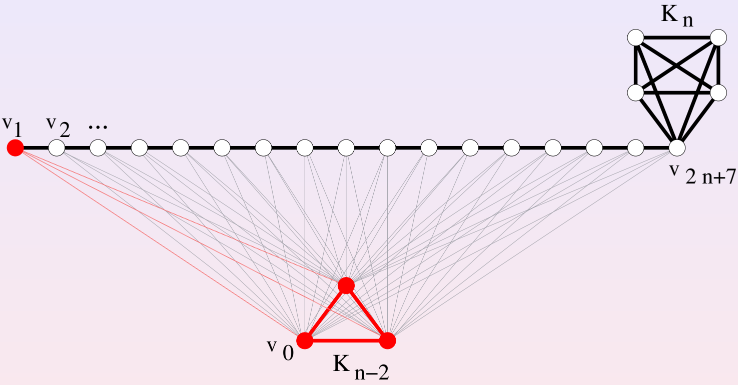
Class of graphs $(G_n)_{n \in N}$ (The figure corresponds to G_5).
All the monotone connected and optimal strategies in this class are **strongly constrained**.

The lower bound : $\Omega(n \log n)$



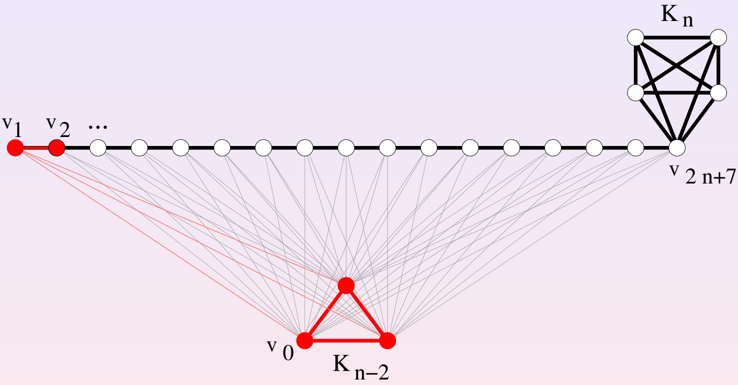
Class of graphs $(G_n)_{n \in N}$ (The figure corresponds to G_5).
All the monotone connected and optimal strategies in this class are **strongly constrained**.

The lower bound : $\Omega(n \log n)$



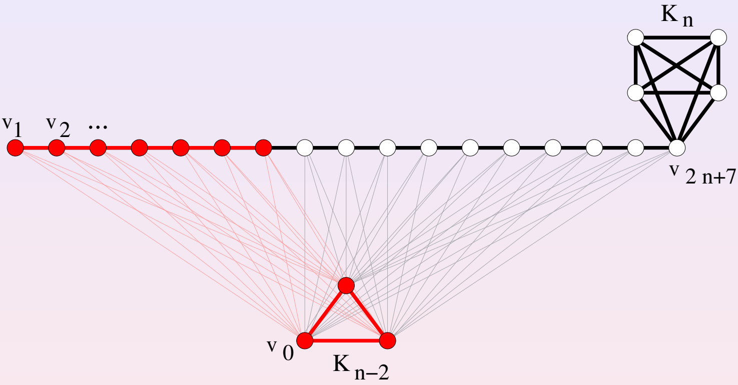
Class of graphs $(G_n)_{n \in N}$ (The figure corresponds to G_5).
All the monotone connected and optimal strategies in this class are **strongly constrained**.

The lower bound : $\Omega(n \log n)$



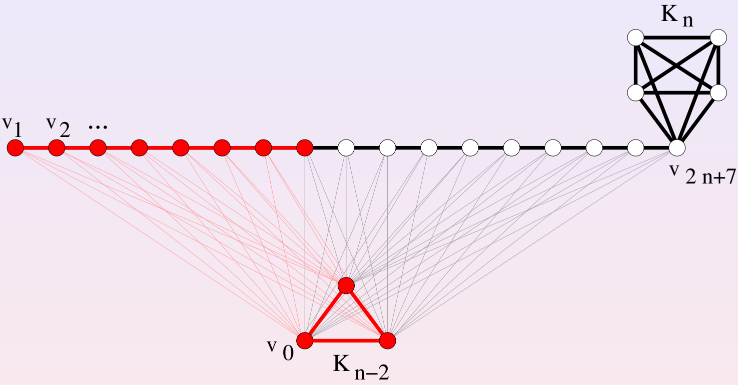
Class of graphs $(G_n)_{n \in N}$ (The figure corresponds to G_5).
All the monotone connected and optimal strategies in this class are **strongly constrained**.

The lower bound : $\Omega(n \log n)$



Class of graphs $(G_n)_{n \in N}$ (The figure corresponds to G_5).
All the monotone connected and optimal strategies in this class are **strongly constrained**.

The lower bound : $\Omega(n \log n)$



Class of graphs $(G_n)_{n \in N}$ (The figure corresponds to G_5).
All the monotone connected and optimal strategies in this class are **strongly constrained**.

The lower bound

Let G_n be the previous graph.
Let L be the set of the local orientations of G_n

Lemma 1

Let f be a particular string of bits of advice. Let $\mathcal{P}$ be a protocol that solves the distributed search problem. $\mathcal{P}$ can clear at most $A = |L|(\frac{1}{n-2})^n$ instances of L.

Lemma 2 [Fraigniaud *et al.* 06]

An oracle $\mathcal{O}$ of size q gives at most $t = (q+1)2^q C^n_{n+q}$ different strings of bits of advice.

The lower bound

Let G_n be the previous graph.
Let L be the set of the local orientations of G_n

Lemma 1

Let f be a particular string of bits of advice. Let $\mathcal{P}$ be a protocol that solves the distributed search problem. $\mathcal{P}$ can clear at most $A = |L|(\frac{1}{n-2})^n$ instances of L.

Lemma 2 [Fraigniaud *et al.* 06]

An oracle $\mathcal{O}$ of size q gives at most $t = (q+1)2^q C^n_{n+q}$ different strings of bits of advice.

The lower bound

- Lemma 1 : $\mathcal{P}$ cannot clear more than A instances of L.
- Lemma 2 : An oracle $\mathcal{O}$ of size q gives at most t strings.

According to lemma 2 :

It exists at least $B = |L|/t$ instances with the same advice.

Asymptotically, if $q = \alpha n \log n$ and $\alpha < 1/4$, $B > A$.

Hence, any protocol using an oracle of size less than $\Omega(n \log n)$ cannot clear all instances of L.

To clear a graph with extra searchers

[*Ilcinkas, Nisse and Soguet*. OPODIS 07]

$\Theta(\frac{n}{\log n}) mcs(G)$ searchers are necessary and sufficient to clear any unknown n-node graph G in a distributed connected *monotone* way.

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

Distributed Protocol allowing $O(\frac{n}{\log n} mcs(G))$ searchers to clear any unknown graph G in a monotone connected way.

- Automata with $O(\log n)$ bits of memory.
- Node's whiteboard of size $O(\log n)$ bits.

Main issue of our protocol

- maintains a dynamic rooted tree S
  - S is a tree of degree at most 3;
  - $V(S)$ is the set of vertices occupied by the searchers;
  - S is a minor of the clear part of G.
- at each step, the protocol tries to clear an edge of G that insures that S becomes as close as possible to a complete tree of degree 3.

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

Distributed Protocol allowing $O(\frac{n}{\log n} mcs(G))$ searchers to clear any unknown graph G in a monotone connected way.

- Automata with $O(\log n)$ bits of memory.
- Node's whiteboard of size $O(\log n)$ bits.

Main issue of our protocol

- maintains a dynamic rooted tree S
  - S is a tree of degree at most 3 ;
  - $V(S)$ is the set of vertices occupied by the searchers ;
  - S is a minor of the clear part of G.
- at each step, the protocol tries to clear an edge of G that insures that S becomes as close as possible to a complete tree of degree 3.

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

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

At each step, our protocol insures 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 isomorphic to T_{k-1} $\Rightarrow k = O(\log|V(T_k)|) = O(\mathbf{s}(T_k)) = O(\mathbf{mcs}(G, v_0))$

Then, if N is the number of searchers used by the protocol,

$$N \leq \frac{|V(T_k)|}{\log|V(T_k)|}\log|V(T_k)| = O(\frac{n}{\log n}\mathbf{mcs}(G, v_0))$$

Idea of the lower bound $\Omega(n/\log n) mcs(G)$

Let $\mathcal{P}$ be any distributed protocol to clear any unknown graph in a monotone connected way.

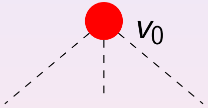
A turn by turn game between $\mathcal{P}$ and a adversary $\mathcal{A}$

- $\mathcal{P}$ and $\mathcal{A}$ play alternatively, starting with $\mathcal{P}$;
- $\mathcal{P}$ aims at clearing a graph that $\mathcal{A}$ gradually builds ;
- $\mathcal{A}$ builds the graph in order to force $\mathcal{P}$ to use the maximum number of searchers.

Idea of the lower bound $\Omega(n/\log n)mcs(G)$

The adversary $\mathcal{A}$ will gradually build a n-node tree T of maximum degree 3, with root v_0.

Example for $n = 10$.



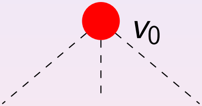
Idea of the lower bound $\Omega(n/\log n)mcs(G)$

The adversary $\mathcal{A}$ will gradually build a n-node tree T of maximum degree 3, with root v_0.

Turn i of $\mathcal{P}$

- $\mathcal{P}$ chooses a searcher ;
- and moves it toward a vertex v along an edge e of T.

Example for $n = 10$.



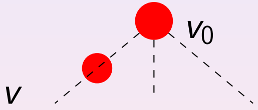
Idea of the lower bound $\Omega(n/\log n)mcs(G)$

The adversary $\mathcal{A}$ will gradually build a n-node tree T of maximum degree 3, with root v_0.

Turn i of $\mathcal{P}$

- $\mathcal{P}$ chooses a searcher ;
- and moves it toward a vertex v along an edge e of T.

Example for $n = 10$.



Idea of the lower bound $\Omega(n/\log n)mcs(G)$

The adversary $\mathcal{A}$ will gradually build a n-node tree T of maximum degree 3, with root v_0.

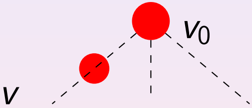
Turn i of $\mathcal{P}$

- $\mathcal{P}$ chooses a searcher ;
- and moves it toward a vertex v along an edge e of T.

Turn i of $\mathcal{A}$

- if v was not discovered yet, $\mathcal{A}$ decides v is incident to 2 non-explored vertices.
- otherwise, $\mathcal{A}$ skips its turn.

Example for $n = 10$.



Idea of the lower bound $\Omega(n/\log n) mcs(G)$

The adversary $\mathcal{A}$ will gradually build a n-node tree T of maximum degree 3, with root v_0.

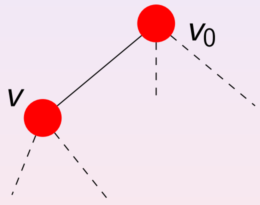
Turn i of $\mathcal{P}$

- $\mathcal{P}$ chooses a searcher ;
- and moves it toward a vertex v along an edge e of T.

Turn i of $\mathcal{A}$

- if v was not discovered yet, $\mathcal{A}$ decides v is incident to 2 non-explored vertices.
- otherwise, $\mathcal{A}$ skips its turn.

Example for $n = 10$.



Idea of the lower bound $\Omega(n/\log n) mcs(G)$

The adversary $\mathcal{A}$ will gradually build a n-node tree T of maximum degree 3, with root v_0.

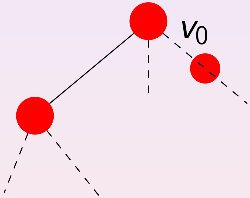
Turn i of $\mathcal{P}$

- $\mathcal{P}$ chooses a searcher ;
- and moves it toward a vertex v along an edge e of T.

Turn i of $\mathcal{A}$

- if v was not discovered yet, $\mathcal{A}$ decides v is incident to 2 non-explored vertices.
- otherwise, $\mathcal{A}$ skips its turn.

Example for $n = 10$.



Idea of the lower bound $\Omega(n/\log n)mcs(G)$

The adversary $\mathcal{A}$ will gradually build a n-node tree T of maximum degree 3, with root v_0.

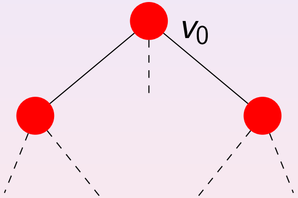
Turn i of $\mathcal{P}$

- $\mathcal{P}$ chooses a searcher ;
- and moves it toward a vertex v along an edge e of T.

Turn i of $\mathcal{A}$

- if v was not discovered yet, $\mathcal{A}$ decides v is incident to 2 non-explored vertices.
- otherwise, $\mathcal{A}$ skips its turn.

Example for $n = 10$.



Idea of the lower bound $\Omega(n/\log n)mcs(G)$

The adversary $\mathcal{A}$ will gradually build a n-node tree T of maximum degree 3, with root v_0.

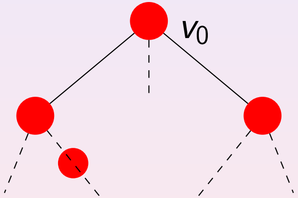
Turn i of $\mathcal{P}$

- $\mathcal{P}$ chooses a searcher ;
- and moves it toward a vertex v along an edge e of T.

Turn i of $\mathcal{A}$

- if v was not discovered yet, $\mathcal{A}$ decides v is incident to 2 non-explored vertices.
- otherwise, $\mathcal{A}$ skips its turn.

Example for $n = 10$.



Idea of the lower bound $\Omega(n/\log n)mcs(G)$

The adversary $\mathcal{A}$ will gradually build a n-node tree T of maximum degree 3, with root v_0.

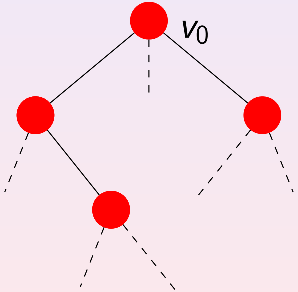
Turn i of $\mathcal{P}$

- $\mathcal{P}$ chooses a searcher ;
- and moves it toward a vertex v along an edge e of T.

Turn i of $\mathcal{A}$

- if v was not discovered yet, $\mathcal{A}$ decides v is incident to 2 non-explored vertices.
- otherwise, $\mathcal{A}$ skips its turn.

Example for $n = 10$.



Idea of the lower bound $\Omega(n/\log n)mcs(G)$

The adversary $\mathcal{A}$ will gradually build a n-node tree T of maximum degree 3, with root v_0.

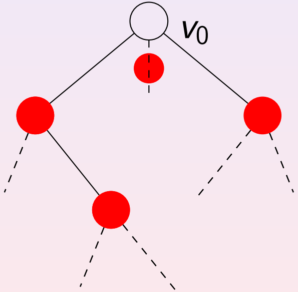
Turn i of $\mathcal{P}$

- $\mathcal{P}$ chooses a searcher ;
- and moves it toward a vertex v along an edge e of T.

Turn i of $\mathcal{A}$

- if v was not discovered yet, $\mathcal{A}$ decides v is incident to 2 non-explored vertices.
- otherwise, $\mathcal{A}$ skips its turn.

Example for $n = 10$.



The game terminates when $V(T) = n$.

Idea of the lower bound $\Omega(n/\log n)mcs(G)$

The adversary $\mathcal{A}$ will gradually build a n-node tree T of maximum degree 3, with root v_0.

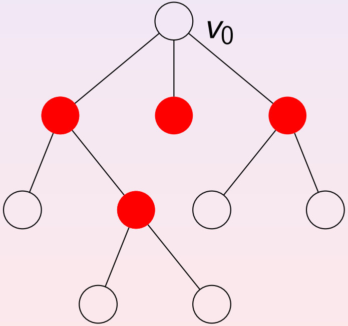
Turn i of $\mathcal{P}$

- $\mathcal{P}$ chooses a searcher ;
- and moves it toward a vertex v along an edge e of T.

Turn i of $\mathcal{A}$

- if v was not discovered yet, $\mathcal{A}$ decides v is incident to 2 non-explored vertices.
- otherwise, $\mathcal{A}$ skips its turn.

Example for $n = 10$.



The game terminates when $V(T) = n$.

Idea of the lower bound $\Omega(n/\log n)$

When the game terminates :

T is a tree with at least $(n+2)/2$ leaves.
$\Rightarrow \mathcal{P}$ used at least $k \geq n/4$ searchers

Since T is a tree, $\mathbf{mcs}(T, v_0) = O(\log n)$
[*Megiddo et al.* 88, *Barrière et al.* 03]

Then $\mathcal{P}$ used $\Omega(\frac{n}{\log n}\mathbf{mcs}(T, v_0))$ searchers.

Plan

Conclusion

Is it possible to clear any graph in a distributed manner ?

Yes

Distributed protocol that clears any graph G using $\mathbf{mcs}(G) + 1$ searchers.

What if monotonicity is required ?

Extra information

$\Theta(n \log n)$ bits of information necessary and sufficient if $\mathbf{mcs}(G) + 1$ searchers are used.

Extra searchers

$\Theta(\frac{n}{\log n}) mcs(G)$ searchers necessary and sufficient if no knowledge about the graph is provided.

Further Work

Monotonicity

Tradeoff : number of searchers / amount of information

Relaxation of monotonicity/connectedness constraints

Distributed protocol to clear a graph G using $\mathbf{s}(G)$ searchers.
($\mathbf{s}(G)$ denotes the "classical" search number of G)

- Random algorithms
- Graph's decompositions
- ...

Thank you