Jeux des gendarmes et du voleur dans les graphes.

Mineurs de graphes, stratégies connexes, et approche distribuée

Nicolas Nisse
LRI, Université Paris-Sud, France.

Soutenance de thèse
2 juillet 2007
Outline

1. Introduction
   - Motivations
   - Variants of the game
   - Definitions and Models
   - Related Works

2. Non-deterministic Graph Searching

3. Connected Graph Searching

4. Distributed Graph Searching

5. Conclusion and Further Works
Motivation: Layout problems

Numerical analysis
Reorder rows and columns of sparse matrix, Choleski factorisation, Gaussian elimination.

VLSI design
Circuits must be laid out in order to minimize physical and cost constraints.
Relationship with several graph’s parameters: bandwidth, cutwidth, profile, minimum fill-in, etc.
Motivation: Computational complexity

Models of computation: Pebble games

A DAG represents a computation circuit

Tradeoff space/time complexity

- Number of pebbles = space complexity
- Number of *moves* = time complexity

Kirousis and Papadimitriou 86

The smallest number of searchers to clear $G$ is equal to the smallest number of pebbles among acyclic orientations of $G$
Motivation: Graph minors theory

**Minor** of a graph $G$: graph obtained by a sequence of vertex or edge deletions and edge contraction.

**Wagner's conjecture**
Graphs are WQO for the minors' relation.

---

**Graph Minors, [JCTB, 1983-]**
- Robertson and Seymour prove the Wagner’s conjecture.
- Any minor closed class of graphs admits a finite obstruction set
- Tree-like decompositions of graphs excluding a minor.
General problem

Context

A fugitive is running in a graph.
A team of searchers is aiming at capturing the fugitive.

Goal

To design a strategy that capture any fugitive using the fewest searchers as possible.
Variants of graph searching games

Invisible searchers

Random walk, Aleliunas et al. [FOCS 79]. Graph’s exploration
Minimize the capture time using only one searcher.

Visible searchers

- **playing rules**: turn by turn, or simultaneous moves;
- **way to capture the fugitive**: same location, domination;
- **fugitive/searchers’ moves**: move along edges or/and jump from a vertex to another one;
- **fugitive/searchers’ velocity**;
- **fugitive’s visibility**.
Variants of graph searching games

Invisible searchers

Random walk, Aleliunas et al. [FOCS 79]. Graph’s exploration
Minimize the capture time using only one searcher.

Visible searchers

- playing rules: turn by turn, or simultaneous moves;
- way to capture the fugitive: same location, domination;
- fugitive/searchers’ moves: move along edges or/and jump from a vertex to another one;
- fugitive/searchers’ velocity;
- fugitive’s visibility.
Variants of graph searching games

Invisible searchers

Random walk, Aleliunas et al. [FOCS 79]. Graph’s exploration. Minimize the capture time using only one searcher.

Visible searchers

- **playing rules**: turn by turn, or simultaneous moves;
- way to capture the fugitive: same location, domination;
- fugitive/searchers’ moves: move along edges or/and jump from a vertex to another one;
- fugitive/searchers’ velocity;
- fugitive’s visibility.
Variants of graph searching games

Invisible searchers
Random walk, Aleliunás et al. [FOCS 79]. Graph’s exploration Minimize the capture time using only one searcher.

Visible searchers
- **playing rules**: turn by turn, or simultaneous moves;
- **way to capture the fugitive**: same location, domination;
- fugitive/searchers’ moves: move along edges or/and jump from a vertex to another one;
- fugitive/searchers’ velocity;
- fugitive’s visibility.
Variants of graph searching games

**Invisible searchers**

Random walk, Aleliunas *et al.* [FOCS 79]. Graph’s exploration. Minimize the capture time using only one searcher.

**Visible searchers**

- **playing rules**: turn by turn, or simultaneous moves;
- **way to capture the fugitive**: same location, domination;
- **fugitive/searchers’ moves**: move along edges or/and jump from a vertex to another one;
- **fugitive/searchers’ velocity**;
- **fugitive’s visibility**.
Variants of graph searching games

Invisible searchers
Random walk, Aleliunas et al. [FOCS 79]. Graph’s exploration Minimize the capture time using only one searcher.

Visible searchers
- **playing rules**: turn by turn, or simultaneous moves;
- **way to capture the fugitive**: same location, domination;
- **fugitive/searchers’ moves**: move along edges or/and jump from a vertex to another one;
- **fugitive/searchers’ velocity**;
- **fugitive’s visibility**.
Variants of graph searching games

Invisible searchers

Random walk, Aleliunas et al. [FOCS 79]. Graph’s exploration. Minimize the capture time using only one searcher.

Visible searchers

- **playing rules**: turn by turn, or simultaneous moves;
- **way to capture the fugitive**: same location, domination;
- **fugitive/searchers’ moves**: move along edges or/and jump from a vertex to another one;
- **fugitive/searchers’ velocity**;
- **fugitive’s visibility**.
## Taxonomy of graph searching games

<table>
<thead>
<tr>
<th></th>
<th>bounded speed</th>
<th>arbitrary fast</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>visible</td>
<td>invisible</td>
</tr>
<tr>
<td><strong>turn by turn</strong></td>
<td><strong>Cops and</strong></td>
<td>X</td>
</tr>
<tr>
<td><strong>game</strong></td>
<td><strong>Robber</strong></td>
<td></td>
</tr>
<tr>
<td><strong>simultaneous</strong></td>
<td>?</td>
<td>X</td>
</tr>
<tr>
<td><strong>moves</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table:** Classification of the graph searching games
### Taxonomy of graph searching games

<table>
<thead>
<tr>
<th>fugitive's characteristics</th>
<th>turn by turn game</th>
<th>simultaneous moves</th>
</tr>
</thead>
<tbody>
<tr>
<td>bounded speed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>visible</td>
<td>Cops and Robber</td>
<td>?</td>
</tr>
<tr>
<td>invisible</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>arbitrary fast</td>
<td></td>
<td></td>
</tr>
<tr>
<td>visible</td>
<td></td>
<td>Visible graph</td>
</tr>
<tr>
<td>invisible</td>
<td></td>
<td>searching</td>
</tr>
</tbody>
</table>

**Table:** Classification of the graph searching games
Sequence of two basic operations,…

1. Place a searcher at a vertex of the graph;
2. Remove a searcher from a vertex of the graph.

…and must result in catching the fugitive

The fugitive is caught when it occupies the same vertex as a searcher and it cannot move away.

The node-search number

Let $s(G)$ be the smallest number of searchers needed to catch an invisible fugitive in a graph $G$. 
Simple example: A ternary tree
Simple example: A ternary tree
Simple example: A ternary tree

Nicolas Nisse

Jeux des gendarmes et du voleur
Simple example: A ternary tree
Simple example: A ternary tree
Simple example: A ternary tree
Simple example: A ternary tree
Simple example: A ternary tree
Simple example: A ternary tree
Simple example: A ternary tree

\[ s(T) = 3 \]
Visible fugitive

The fugitive is visible if, at every step, searchers know its position. Let $\text{vs}(G)$ be the visible search number of the graph $G$.

Obviously, for any graph $G$, $\text{vs}(G) \leq s(G)$.

In trees

For any $n$-nodes tree $T$, $s(T) \leq 1 + \log_3(n - 1)$ (tight)

Megiddo et al. [JACM 88]

For any tree $T$ (with at least 2 vertices), $\text{vs}(T) = 2$. 
Visibility of the fugitive

The fugitive is visible if, at every step, searchers know its position.

Let $\text{vs}(G)$ be the visible search number of the graph $G$.

Obviously, for any graph $G$, $\text{vs}(G) \leq s(G)$.

In trees

For any $n$-nodes tree $T$, $s(T) \leq 1 + \log_3(n - 1)$ (tight)

Megiddo et al. [JACM 88]

For any tree $T$ (with at least 2 vertices), $\text{vs}(T) = 2$. 
Visible graph searching in a tree
Visible graph searching in a tree
Visible graph searching in a tree
Visible graph searching in a tree
Visible graph searching in a tree
Visible graph searching in a tree

Nicolas Nisse

Jeux des gendarmes et du voleur
Visible graph searching in a tree

2 searchers are sufficient
NP-hardness

The following problems are NP-hard

**Input:** a graph $G$, an integer $k > 0$,

**Output:** $s(G) \leq k$?

Megiddo et al., [JACM 88]

**Input:** a graph $G$, an integer $k > 0$,

**Output:** $vs(G) \leq k$?

Seymour and Thomas [JCTB 93]

**Remark:** linear in the class of trees, Skodinis [JAlg 03]
The following problems are NP-hard

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>a graph $G$, an integer $k &gt; 0$,</td>
<td>$s(G) \leq k$?</td>
<td>Megiddo et al., [JACM 88]</td>
</tr>
<tr>
<td>a graph $G$, an integer $k &gt; 0$,</td>
<td>$\text{vs}(G) \leq k$?</td>
<td>Seymour and Thomas [JCTB 93]</td>
</tr>
</tbody>
</table>

**Remark:** linear in the class of trees, Skodinis [JAlg 03]

NP-membership? Certificate?
Monotonicity and NP-completeness

A vertex $v$ is **recontaminated** if the fugitive can move to $v$ after $v$ has been occupied by a searcher.

**Monotonicity**

A search strategy is **monotone** if no recontamination ever occurs. That is, a vertex is occupied by a searcher only once.

Recontamination does not help. There always exists an optimal monotone search strategy.

invisible fugitive: LaPaugh, Bienstock and Seymour [JACM 93], [JAlg 91]

visible fugitive: Seymour and Thomas [JCTB 93]

Corollary: The above problems belong to NP.

Nicolas Nisse

Jeux des gendarmes et du voleur
Monotonicity and NP-completeness

A vertex $v$ is **recontaminated** if the fugitive can move to $v$ after $v$ has been occupied by a searcher.

**Monotonicity**

A search strategy is **monotone** if no recontamination ever occurs. That is, a vertex is occupied by a searcher only once.

**Recontamination does not help**

There always exists an **optimal monotone** search strategy.

- invisible fugitive: LaPaugh, Bienstock and Seymour [JACM 93], [JAlg 91]
- visible fugitive: Seymour and Thomas [JCTB 93]

**Corollary:** The above problems belong to NP.
Monotonicity and NP-completeness

A vertex \( v \) is **recontaminated** if the fugitive can move to \( v \) after \( v \) has been occupied by a searcher.

**Monotonicity**

A search strategy is **monotone** if no recontamination ever occurs. That is, a vertex is occupied by a searcher only once.

Recontamination does not help

There always exists an optimal monotone search strategy.

- invisible fugitive: LaPaugh, Bienstock and Seymour [JACM 93] [JAlg 91]
- visible fugitive: Seymour and Thomas [JCTB 93]

**Corollary:** The above problems belong to NP.
Thanks to the monotonicity, we get:

**Search number and Pathwidth ($pw$)**

For any graph $G$, $s(G) = pw(G) + 1$,

Kinnersley [IPL 92],
Ellis, Sudborough, and Turner [Inf.Comp.94]

**Visible search number and Treewidth ($tw$)**

For any graph $G$, $vs(G) = tw(G) + 1$,

Seymour and Thomas [JCTB 93]
Outline

1. Introduction
2. Non-deterministic Graph Searching
   - Characterization
   - Monotonicity
3. Connected Graph Searching
4. Distributed Graph Searching
5. Conclusion and Further Works
Non-deterministic Graph Searching

Invisible fugitive
An **Oracle** permanently knows the position of the fugitive

One extra operation is allowed
Searchers can perform a query to the oracle: “What is the current position of the fugitive?”

Sequence of three basic operations
1. Place a searcher at a vertex of the graph;
2. Remove a searcher from a vertex of the graph;
3. Perform a query to the Oracle.

Tradeoff number of searchers / number of queries

$q$-limited (non-deterministic) search number, $s_q(G)$
Non-deterministic Graph Searching

Invisible fugitive
An **Oracle** permanently knows the position of the fugitive

One extra operation is allowed
Searchers can perform a query to the oracle: “What is the current position of the fugitive?”

Sequence of **three** basic operations
1. Place a searcher at a vertex of the graph;
2. Remove a searcher from a vertex of the graph;
3. Perform a query to the Oracle.

Tradeoff number of searchers / number of queries
q-limited (non-deterministic) search number, \( s_q(G) \)
Controlled Amount of Nondeterminism

\[
\begin{align*}
\text{number of searchers} & \quad \text{number of queries} \\
\text{pw}(G) + 1 & \quad \tau(G) \\
\text{tw}(G) + 1 & \quad \pi(G)
\end{align*}
\]
Still the same ternary tree

$s_0(T) = 3$
Still the same ternary tree

$s_0(T) = 3$
Still the same ternary tree

1 query
Still the same ternary tree

1 query
Still the same ternary tree

1 query
Still the same ternary tree
Still the same ternary tree

2 queries
Still the same ternary tree

2 queries
Still the same ternary tree

2 queries
Still the same ternary tree

\[ s_0(T) = s_1(T) = 3 \]
\[ s_2(T) = s_\infty(T) = 2 \]
In collaboration with F. Mazoit

For any $q \geq 0$, recontamination does not help to catch a fugitive in $G$ performing at most $q$ queries.

- Constructive proof;
- Generalize the existing proofs ($q = 0$ and $q = \infty$).

In collaboration with F.V. Fomin and P. Fraigniaud

- Equivalence between non-deterministic graph searching and branched tree-decomposition;
- Exponential exact algorithm computing $s_q(G)$ in time $O^*(2^n)$;
- $s_q(G) \leq 2 s_{q+1}(G)$ (almost tight).
Monotonicity: Search-tree

Auxiliary structure inspired by the tree-labelling [Robertson and Seymour, Graph Minor X]:

**Search-tree** = A tree $T$ labelled with subsets of $E(G)$

For any vertex $v \in V(T)$ incident to $e_1, \ldots, e_p$:

- label of $v$: $\ell(v) \subseteq E(G)$
- label of $e_i$: $\ell_v(e_i) \subseteq E(G)$

Any edge has two labels: one for each extremity.
Monotonicity: Search-tree

- Non-deterministic search strategy $\Rightarrow$ Search-tree
  - placement of searchers $\Rightarrow$ vertex of $T$
  - query $\Rightarrow$ fork (vertex of $T$ with more than one child)
  - removal of searchers $\Rightarrow$ edge of $T$
Monotonicity: Search-tree

Two Properties

1. \(\{\ell(v), \ell_v(e_1), \ell_v(e_2), \ldots, \ell_v(e_p)\}\) partition of \(E(G)\);
2. \(\forall e = \{u, v\} \in E(T), \ell_v(e)\) and \(\ell_u(e)\) are disjoint.
Monotonicity

Auxiliary structure inspired by the tree-labelling [Robertson and Seymour, Graph Minor X]:

**Search-tree** = a tree $T$ labelled with subsets of $E(G)$.

**Sketch of the proof**

- (possibly non monotone) strategy $\Rightarrow$ Search-tree
- weight function over the search-trees
- minimal search-tree $\Rightarrow$ monotone strategy
- local optimization without increasing neither the number of searcher, nor the number of queries.
Outline

1. Introduction
2. Non-deterministic Graph Searching
   - Cost of connectivity
   - Non-Monotonicity
3. Connected Graph Searching
4. Distributed Graph Searching
5. Conclusion and Further Works
Connected Graph Searching

**Limits of the Parson’s model**

- Searchers cannot move at will in a real network;
- Secured communications.

Connected Search Strategy, Barrière et al., [SPAA 02]

At any step, the cleared part of the graph must induce a connected subgraph.

Let $cs(G)$ be the connected search number of the graph $G$.

Two main questions

- What is the cost of connectivity? ratio $cs/s$?
- Monotonicity property of connected graph searching?
Connected Graph Searching

Limits of the Parson’s model
- Searchers cannot move at will in a real network;
- Secured communications.

Connected Search Strategy, Barrière et al., [SPAA 02]
At any step, the cleared part of the graph must induce a connected subgraph.
Let \( cs(G) \) be the connected search number of the graph \( G \).

Two main questions
- What is the cost of connectivity? ratio \( cs/s \)?
- Monotonicity property of connected graph searching?
Connected Graph Searching

Limits of the Parson’s model
- Searchers cannot move at will in a real network;
- Secured communications.

Connected Search Strategy, Barrière et al., [SPAA 02]
At any step, the cleared part of the graph must induce a connected subgraph.
Let $cs(G)$ be the connected search number of the graph $G$.

Two main questions
What is the cost of connectivity? ratio $cs/s$?
Monotonicity property of connected graph searching?
The cost of connectedness

In terms of number of searchers

For any tree $T$, $s(T) \leq cs(T) \leq 2s(T) - 2$. (tight)
Barrière, Flocchini, Fraigniaud, and Thilikos [WG 03]

For any connected graph $G$, $cs(G) \leq s(G) \leq 2 + \log |E(G)|$.
Fomin, Fraigniaud, and Thilikos [Tech. Rep. 04]

About monotonicity

Recontamination does not help in trees.
Barrière, Flocchini, Fraigniaud, and Santoro [SPAA 02]

Recontamination helps in general.
Alspach, Dyer, and Yang [ISAAC 04]
Results: Case of a invisible fugitive

Using the concept of *connected* tree-decomposition.

In collaboration with P. Fraigniaud

For any $n$-node connected graph $G$, $\text{cs}(G)/s(G) \leq \log n$.

Graphs with bounded chordality $k$

$(T, X)$ an optimal tree-decomposition of $G$

$\text{cs}(G) \leq (\text{tw}(G)\lceil k/2 \rceil + 1)\text{cs}(T)$.

$\Rightarrow \text{cs}(G)/s(G) \leq 2 (\text{tw}(G) + 1)$ if $G$ chordal
Sketch of proof: $\text{cs}(G) \leq s(G) \log n$

Proof by induction on $n$: $\text{cs}(G) \leq (\text{tw}(G) + 1) \log n$

For any $1 \leq i \leq r$, $G[T_i]$ is a connected subgraph with at most $n/2$ vertices.
Sketch of proof: \( \text{cs}(G) \leq s(G) \log n \)

Proof by induction on \( n \): \( \text{cs}(G) \leq (\text{tw}(G) + 1) \log n \)

There is a connected search strategy for \( G[T_1] \), using at most \( (\text{tw}(G) + 1) \log(n/2) \) searchers.
Sketch of proof: \( \text{cs}(G) \leq s(G) \log n \)

Proof by induction on \( n \): \( \text{cs}(G) \leq (\text{tw}(G) + 1) \log n \)

At most \( \text{tw}(G) + 1 \) searchers are required to protect \( G[T_1] \) from recontamination from the remaining part of \( G \).
Sketch of proof: $\text{cs}(G) \leq s(G) \log n$

Proof by induction on $n$: $\text{cs}(G) \leq (\text{tw}(G) + 1) \log n$

Then, we can terminate the clearing of $G[T_1]$. 

Nicolas Nisse
Jeux des gendarmes et du voleur
Sketch of proof: $\text{cs}(G) \leq s(G) \log n$

Proof by induction on $n$: $\text{cs}(G) \leq (\text{tw}(G) + 1) \log n$

The $(\text{tw}(G) + 1) \log(n/2)$ searchers can be used to clear another subgraph $G[T_i]$, and so on...
Sketch of proof: $\text{cs}(G) \leq s(G) \log n$

Proof by induction on $n$: $\text{cs}(G) \leq (\text{tw}(G) + 1) \log n$

Connected search strategy using at most $(\text{tw}(G) + 1) \log n$ searchers. Thus, $\text{cs}(G) \leq s(G) \log n$
Results: Case of a visible fugitive

In collaboration with P. Fraigniaud

For any $n$-node graph $G$, $\text{cvs}(G)/\text{vs}(G) \leq \log n$ tight for monotone strategies: $\text{mcvs}(G)/\text{vs}(G) \geq \Omega(\log n)$.

In collaboration with P. Fraigniaud

In visible connected graph searching, recontamination helps

For any $k \geq 4$, there exists a graph $G$ such that $\text{cvs}(G) = 4k + 1$ and any monotone connected visible search strategy uses at least $4k + 2$ searchers.
Recontamination helps in visible connected graph searching

Let $G$ be the graph below: $mcvs(G) > cvs(G) = 4$. 

mcvs, cvs, symmetry axis
Recontamination helps in visible connected graph searching

Let $G$ be the graph below: $\text{mcvs}(G) > \text{cvs}(G) = 4$. 
Recontamination helps in visible connected graph searching

Let $G$ be the graph below: $\text{mcvs}(G) > \text{cvs}(G) = 4$. 

symmetry axis
Recontamination helps in visible connected graph searching

Let $G$ be the graph below: $mcvs(G) > cvs(G) = 4$. 

Let $G$ be the graph below: $mcvs(G) > cvs(G) = 4$. 

symmetry axis
Recontamination helps in visible connected graph searching

Let $G$ be the graph below: $\text{mcvs}(G) > \text{cvs}(G) = 4$. 
Recontamination helps in visible connected graph searching

Let $G$ be the graph below: $mcvs(G) > cvs(G) = 4$. 
Recontamination helps in visible connected graph searching

Let $G$ be the graph below: $\text{mcvs}(G) > \text{cvs}(G) = 4$. 

\[
\text{symmetry axis}
\]
Non-monotonicity

Recontamination helps in visible connected graph searching

Let $G$ be the graph below: $\text{mcvs}(G) > \text{cvs}(G) = 4$. 

symmetry axis
Recontamination helps in visible connected graph searching

Let $G$ be the graph below: $\text{mcvs}(G) > \text{cvs}(G) = 4$. 

symmetry axis
Recontamination helps in visible connected graph searching

Let $G$ be the graph below: $\text{mcvs}(G) > \text{cvs}(G) = 4$. 

symmetry axis
Non-monotonicity

Recontamination helps in visible connected graph searching

Let $G$ be the graph below: $\text{mcvs}(G) > \text{cvs}(G) = 4$. 

symmetry axis
Recontamination helps in visible connected graph searching

Let $G$ be the graph below: $\text{mcvs}(G) > \text{cvs}(G) = 4.$
Recontamination helps in visible connected graph searching

Let $G$ be the graph below: $mcvs(G) > cvs(G) = 4$. 

symmetry axis
Outline

1. Introduction
2. Non-deterministic Graph Searching
3. Connected Graph Searching
   - Model
   - Distributed Protocols
4. Conclusion and Further Works
Graph searching in a distributed way

Distributed search problem

To design a *distributed protocol* that enables the *minimum number* of searchers to clear the network. The searchers must compute *themselves* a strategy.

We consider connected search strategies.

$mcs$ refers to the smallest number of searchers required to catch an invisible fugitive in a monotone connected way.
Graph searching in a distributed way

Distributed search problem

To design a distributed protocol that enables the minimum number of searchers to clear the network. The searchers must compute themselves a strategy.

We consider connected search strategies.

mcs refers to the smallest number of searchers required to catch an invisible fugitive in a monotone connected way.
Distributed graph searching: model

The searchers
- autonomous mobile computing entities with distinct IDs;
- automata with $O(\log n)$ bits of memory;
- their decision is computed locally.

The network
- undirected connected graph;
- local orientation of the edges;
- whiteboards on vertices (zone of local memory);
- asynchronous environment.
Distributed graph searching: related work

The searchers **have a prior knowledge** of the topology.

<table>
<thead>
<tr>
<th>Protocols to clear specific topologies</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tree.</strong> Barrière <em>et al.</em>, [SPAA 02]</td>
</tr>
<tr>
<td><strong>Mesh.</strong> Flocchini, Luccio, and Song. [CIC 05]</td>
</tr>
<tr>
<td><strong>Hypercube.</strong> Flocchini, Huang, and Luccio. [IPDPS 05]</td>
</tr>
<tr>
<td><strong>Tori.</strong> Flocchini, Luccio, and Song. [IPDPS 06]</td>
</tr>
<tr>
<td><strong>Sierpinski’s graph.</strong> Luccio. [FUN 07]</td>
</tr>
</tbody>
</table>

A monotone connected and strategy is performed using **mcs** + 1 searchers.

**Remark:**
The extra searcher is due to the asynchronicity of the network and it is necessary [CIC 05].
Distributed graph searching: related work

The searchers have a prior knowledge of the topology.

**Protocols to clear specific topologies**

- **Tree.** Barrière et al., [SPAA 02]
- **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 and strategy is performed using mcs + 1 searchers.

**Remark:**

The extra searcher is due to the asynchronicity of the network and it is necessary [CIC 05].
In collaboration with L. Blin, P. Fraigniaud and S. Vial

Distributed protocol that enable $\text{mcs}(G) + 1$ searchers to clear an unknown graph $G$ in a connected way.

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

In collaboration with D. Soguet

$\Theta(n \log n)$ bits of information must be provided to the searchers to clear a unknown graph in a monotone connected way.
In collaboration with L. Blin, P. Fraigniaud and S. Vial

Distributed protocol that enable \( \text{mcs}(G) + 1 \) searchers to clear an unknown graph \( G \) in a connected way.

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

In collaboration with D. Soguet

\( \Theta(n \log n) \) bits of information must be provided to the searchers to clear a unknown graph in a monotone connected way.
In collaboration with L. Blin, P. Fraigniaud and S. Vial

Distributed protocol that enable $mcs(G) + 1$ searchers to clear an unknown graph $G$ in a connected way.

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

In collaboration with D. Soguet

$\Theta(n \log n)$ bits of information must be provided to the searchers to clear an unknown graph in a monotone connected way.
1. Introduction
2. Non-deterministic Graph Searching
3. Connected Graph Searching
4. Distributed Graph Searching
5. Conclusion and Further Works
Summary of the results

Non-deterministic graph searching
A unified approach of visible and invisible graph searching
Unified proof of monotonicity.

Connected graph searching
Upper bounds for the ratio \( cs/s \)
Case of a visible fugitive

Distributed graph searching
Distributed protocol to clear an unknown graph
Amount of information required for monotonicity
Open Problems

Non-deterministic graph searching
Explicit FPT Algorithm?
Polynomial-time algorithm in trees?

Connected graph searching
\( cs/s \) ?
FPT Algorithm?
NP-membership?

Distributed graph searching
Tradeoff between amount of information and number of searchers?
Further Works

Directed graph decompositions...
- Directed treewidth. [Johnson et al., 95]
- DAG-width. [Obdrzalek, and Berwanger et al. 06]
- Kelly-width. [Hunter and Kreutzer, 07]

... and related directed graph searching games
- Monotonicity ? [Barat 06, Adler 07]
  **Open problem:** Is the graph searching game corresponding to DAG-width (resp., Kelly-width) monotone?

Matroid decompositions
- Matroid’s treewidth, [Hlineny and Whittle, 06]
- Matroid’s branchwidth [Mazoit and Thomassé, 06]
Thank's