Pathwidth and Graph Searching Games

Nicolas Nisse

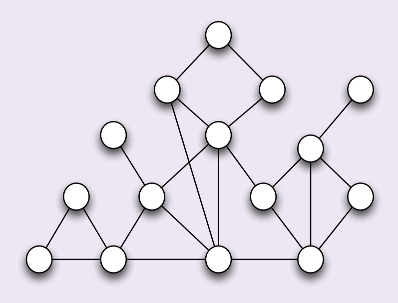
Inria, France

Univ. Nice Sophia Antipolis, CNRS, I3S, UMR 7271, Sophia Antipolis, France

COATI seminar

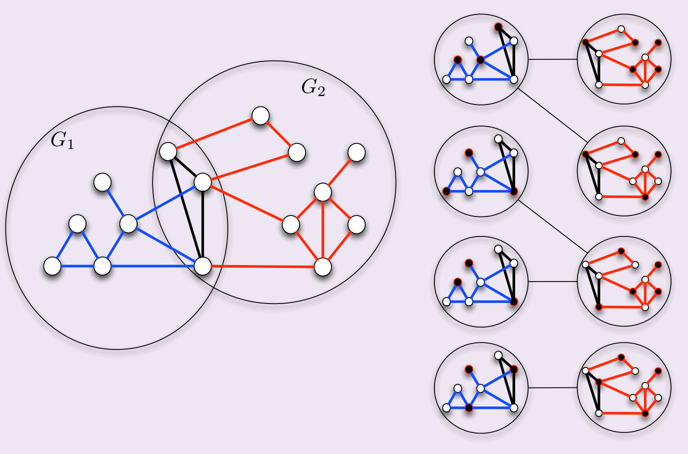
October 8th 2014

Dynamic Programming for Max. Independent Set



Let's compute a maximum independent set of this graph
Brute-force: check all subsets 2^{15}

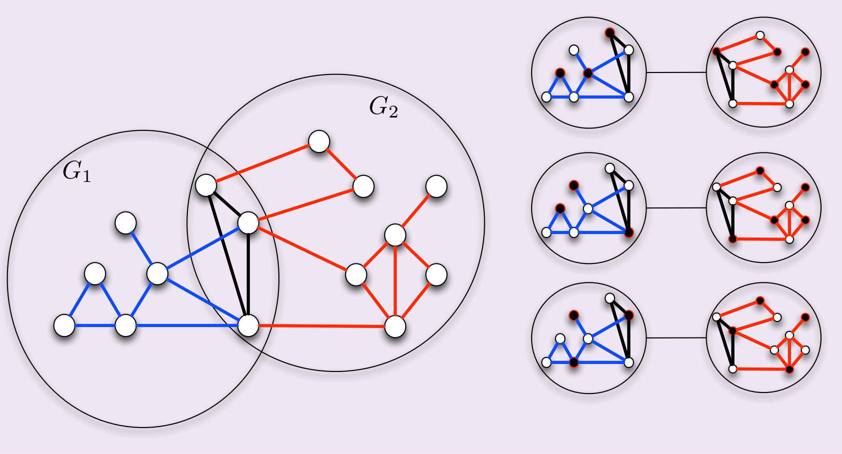
Dynamic Programming for Max. Independent Set



Brute-force: check all subsets 2^{15}

better idea (?): combine IS of G_1 and G_2 $2^8 + 2^{10} + 2^8 * 2^{10}$

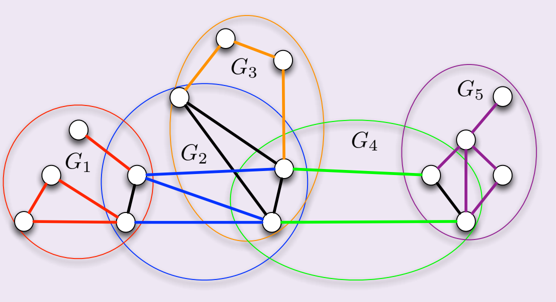
Dynamic Programming for Max. Independent Set



For any indep. set I of the Separator ($G_1 \cap G_2$), find:

- one MIS compatible with I in G_1 2^5
- one MIS compatible with I in G_2 2^7
- combine them 2^3

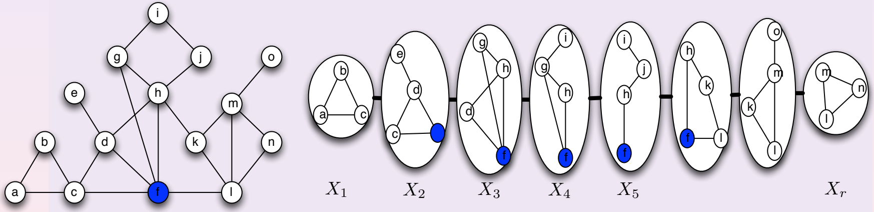
Dynamic Programming for Max. Independent Set



Going further: decompose G into more parts $\Rightarrow \#\ of\ part\ * 2^{O(size\ of\ largest\ part)}$

Path-Decomposition and Pathwidth

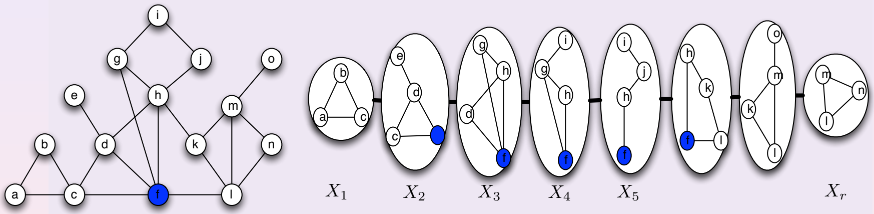
Representation of a graph $G = (V, E)$ as a Path preserving connectivity properties



Sequence $\mathcal{X} = (X_1, \cdots, X_r)$ of "bags" (set of vertices of G)
Important: intersection of two adjacent bags = separator of G

Path-Decomposition and Pathwidth

Representation of a graph $G = (V, E)$ as a Path preserving connectivity properties

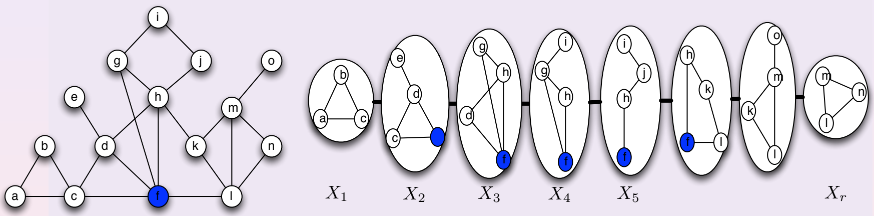


Sequence $\mathcal{X} = (X_1, \cdots, X_r)$ of "bags" (set of vertices of G)
Important: intersection of two adjacent bags = separator of G

- $\bigcup_{i \leq r} X_i = V$
- for any $e = uv \in E$, there is $i \leq r$ such that $u, v \in X_i$
- for any $i \leq j \leq k \leq r$, $X_i \cap X_k \subseteq X_j$.

Path-Decomposition and Pathwidth

Representation of a graph $G = (V, E)$ as a Path preserving connectivity properties

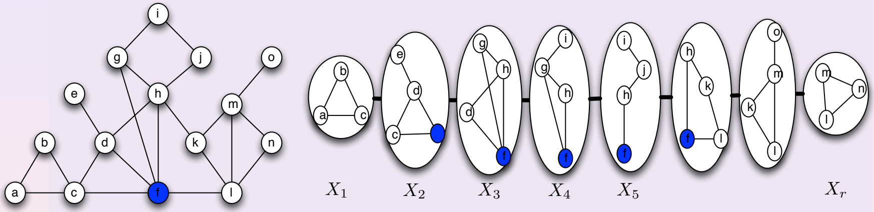


Sequence $\mathcal{X} = (X_1, \cdots, X_r)$ of "bags" (set of vertices of G)
Important: intersection of two adjacent bags = separator of G

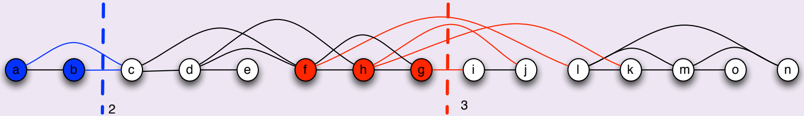
Width of $(T, \mathcal{X})$: $\max_{i \leq r} |X_i| - 1$ $\approx$ size of largest bag
Pathwidth of a graph G, $pw(G)$: min width over all path-decompositions.

Path-Decomposition and Pathwidth

Representation of a graph $G = (V, E)$ as a Path preserving connectivity properties



Equivalent definition:
Ordering of nodes $(v_1, v_2, \cdots, v_n)$ minimizing $\max_{1<i\leq n} |\{j < i \mid v_i v_j \in E\}|$.



Algorithmic Applications and Complexity

Dynamic programming on path decomposition

MSOL Problems: "local" problems are FPT in *pw* [Courcelle'90]
e.g., coloring, independent set: $O(2^{pw} n^{O(1)})$; dominating set $O(4^{pw} n^{O(1)})$...

huge constants may be hidden (at least exponential in *pw*)
"good" decompositions must be computed

Algorithmic Applications and Complexity

Complexity to compute path-decompositions

- NP-complete to compute *pw*
  - in *planar cubic* graphs [Monien, Sudborough'88]
  - in *chordal* graphs [Gustedt'93]
- Not approximable up to additive constant (unless P=NP) [Deo, Krishnamoorthy, Langston'87]
- FPT-algorithm [Bodlaender, Kloks'96]
- Polyomial or Linear in
  - trees [Skodinis'00],
  - cographs [Bodlaender, Möhring'93],
  - split graphs [Gustedt'93], etc.
- Exponential exact algorithm [Coudert,Mazauric,N.'14]

Studying Pathwidth via Graph Searching

Team of Searchers

to Capture an invisible fugitive / Clear a contaminated graph

Rules of Graph Searching [Parsons'76]

Allowed moves

- Place a searcher at a node
- Remove a searcher from a node
- Slide a searcher along an edge

Clearing edges

- when a searcher slides along it

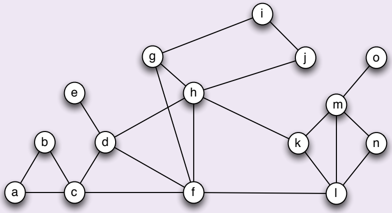
Recontamination

- if no searcher on a path from a clear edge to a contaminated one

Goal: Minimize the number of searchers needed

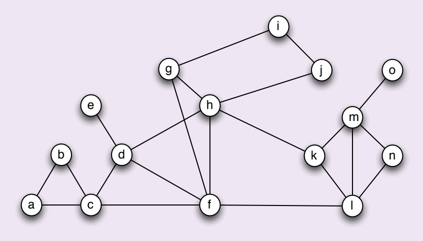
Studying Pathwidth via Graph Searching

Allowed moves: Place $P(v)$, Remove $R(v)$, Slide $S(e)$
Clearing edges: when a searcher slides along it
Recontamination: if no searcher on a path from a clear edge to a contaminated one



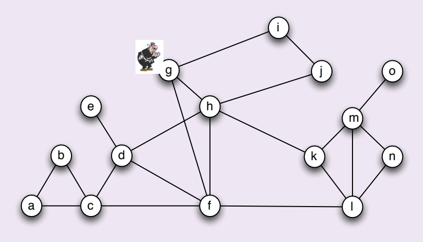
Studying Pathwidth via Graph Searching

Allowed moves: Place $P(v)$, Remove $R(v)$, Slide $S(e)$
Clearing edges: when a searcher slides along it
Recontamination: if no searcher on a path from a clear edge to a contaminated one



Studying Pathwidth via Graph Searching

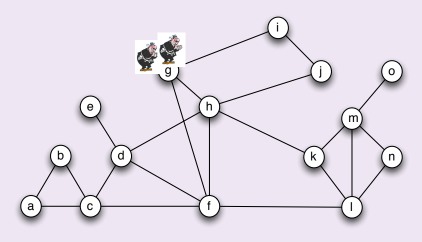
Allowed moves: Place $P(v)$, Remove $R(v)$, Slide $S(e)$
Clearing edges: when a searcher slides along it
Recontamination: if no searcher on a path from a clear edge to a contaminated one



P(g),

Studying Pathwidth via Graph Searching

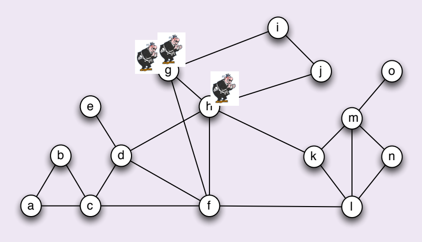
Allowed moves: Place $P(v)$, Remove $R(v)$, Slide $S(e)$
Clearing edges: when a searcher slides along it
Recontamination: if no searcher on a path from a clear edge to a contaminated one



P(g), P(g),

Studying Pathwidth via Graph Searching

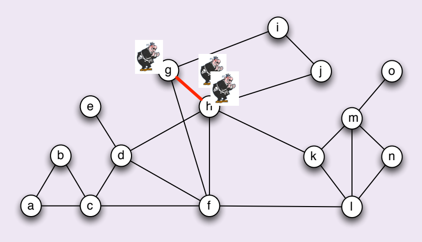
Allowed moves: Place $P(v)$, Remove $R(v)$, Slide $S(e)$
Clearing edges: when a searcher slides along it
Recontamination: if no searcher on a path from a clear edge to a contaminated one



P(g), P(g), P(h),

Studying Pathwidth via Graph Searching

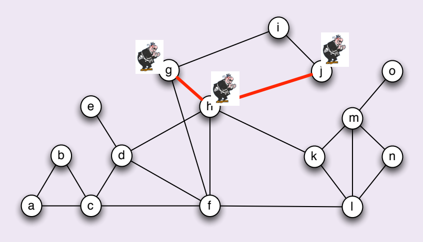
Allowed moves: Place $P(v)$, Remove $R(v)$, Slide $S(e)$
Clearing edges: when a searcher slides along it
Recontamination: if no searcher on a path from a clear edge to a contaminated one



P(g), P(g), P(h), S(gh),

Studying Pathwidth via Graph Searching

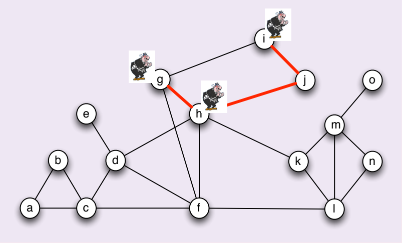
Allowed moves: Place $P(v)$, Remove $R(v)$, Slide $S(e)$
Clearing edges: when a searcher slides along it
Recontamination: if no searcher on a path from a clear edge to a contaminated one



P(g), P(g), P(h), S(gh), S(hj),

Studying Pathwidth via Graph Searching

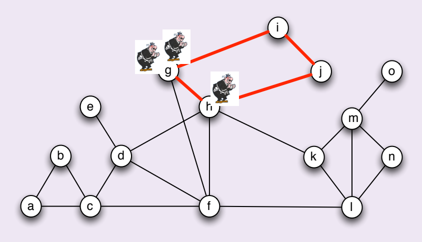
Allowed moves: Place $P(v)$, Remove $R(v)$, Slide $S(e)$
Clearing edges: when a searcher slides along it
Recontamination: if no searcher on a path from a clear edge to a contaminated one



P(g), P(g), P(h), S(gh), S(hj), S(ji),

Studying Pathwidth via Graph Searching

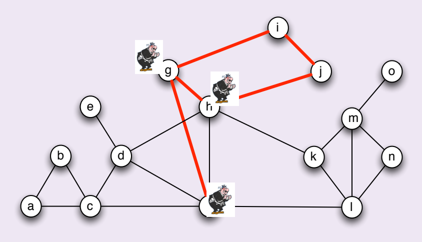
Allowed moves: Place $P(v)$, Remove $R(v)$, Slide $S(e)$
Clearing edges: when a searcher slides along it
Recontamination: if no searcher on a path from a clear edge to a contaminated one



P(g), P(g), P(h), S(gh), S(hj), S(ji), S(ih),

Studying Pathwidth via Graph Searching

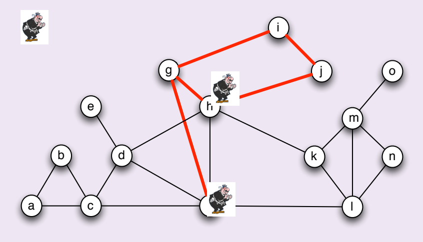
Allowed moves: Place $P(v)$, Remove $R(v)$, Slide $S(e)$
Clearing edges: when a searcher slides along it
Recontamination: if no searcher on a path from a clear edge to a contaminated one



P(g), P(g), P(h), S(gh), S(hj), S(ji), S(ih), S(gf),

Studying Pathwidth via Graph Searching

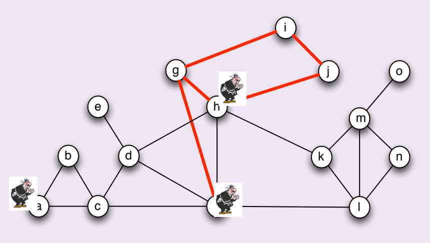
Allowed moves: Place $P(v)$, Remove $R(v)$, Slide $S(e)$
Clearing edges: when a searcher slides along it
Recontamination: if no searcher on a path from a clear edge to a contaminated one



P(g), P(g), P(h), S(gh), S(hj), S(ji), S(ih), S(gf), R(g),

Studying Pathwidth via Graph Searching

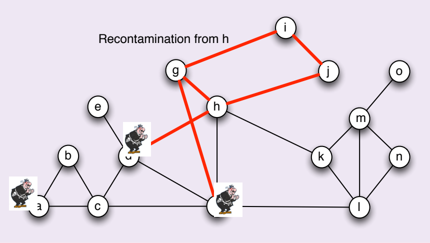
Allowed moves: Place $P(v)$, Remove $R(v)$, Slide $S(e)$
Clearing edges: when a searcher slides along it
Recontamination: if no searcher on a path from a clear edge to a contaminated one



P(g), P(g), P(h), S(gh), S(hj), S(ji), S(ih), S(gf), R(g), P(a),

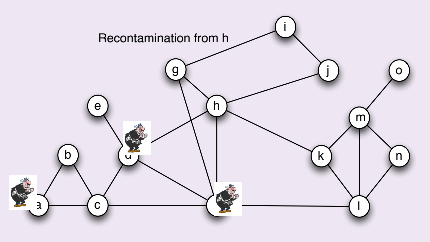
Studying Pathwidth via Graph Searching

Allowed moves: Place $P(v)$, Remove $R(v)$, Slide $S(e)$
Clearing edges: when a searcher slides along it
Recontamination: if no searcher on a path from a clear edge to a contaminated one



P(g), P(g), P(h), S(gh), S(hj), S(ji), S(ih), S(gf), R(g), P(a), S(hd),

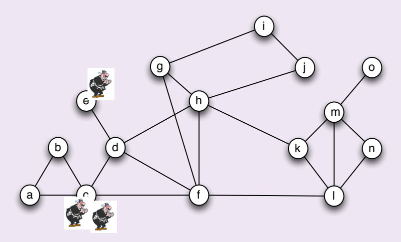
Allowed moves: Place $P(v)$, Remove $R(v)$, Slide $S(e)$
Clearing edges: when a searcher slides along it
Recontamination: if no searcher on a path from a clear edge to a contaminated one



P(g), P(g), P(h), S(gh), S(hj), S(ji), S(ih), S(gf), R(g), P(a), S(hd), Recontamination, let's start again

Studying Pathwidth via Graph Searching

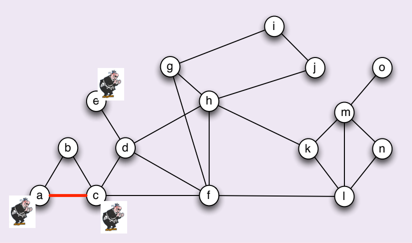
Allowed moves: Place $P(v)$, Remove $R(v)$, Slide $S(e)$
Clearing edges: when a searcher slides along it
Recontamination: if no searcher on a path from a clear edge to a contaminated one



P(c), P(c), P(e),

Studying Pathwidth via Graph Searching

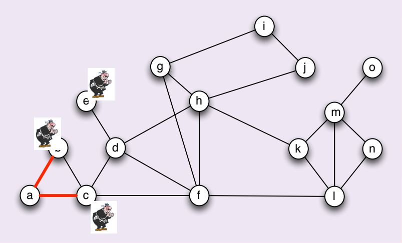
Allowed moves: Place $P(v)$, Remove $R(v)$, Slide $S(e)$
Clearing edges: when a searcher slides along it
Recontamination: if no searcher on a path from a clear edge to a contaminated one



P(c), P(c), P(e), S(ca),

Studying Pathwidth via Graph Searching

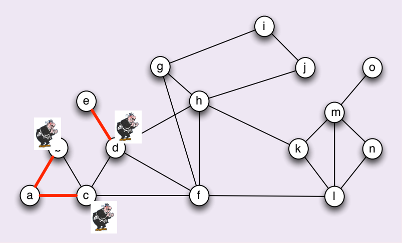
Allowed moves: Place $P(v)$, Remove $R(v)$, Slide $S(e)$
Clearing edges: when a searcher slides along it
Recontamination: if no searcher on a path from a clear edge to a contaminated one



P(c), P(c), P(e), S(ca), S(ab),

Studying Pathwidth via Graph Searching

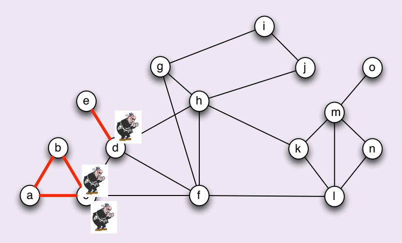
Allowed moves: Place $P(v)$, Remove $R(v)$, Slide $S(e)$
Clearing edges: when a searcher slides along it
Recontamination: if no searcher on a path from a clear edge to a contaminated one



P(c), P(c), P(e), S(ca), S(ab), S(ed),

Studying Pathwidth via Graph Searching

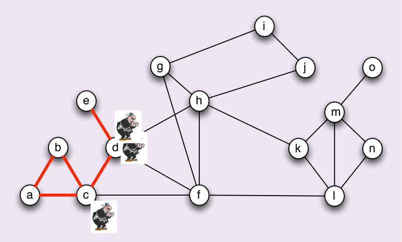
Allowed moves: Place $P(v)$, Remove $R(v)$, Slide $S(e)$
Clearing edges: when a searcher slides along it
Recontamination: if no searcher on a path from a clear edge to a contaminated one



P(c), P(c), P(e), S(ca), S(ab), S(ed), S(bc),

Studying Pathwidth via Graph Searching

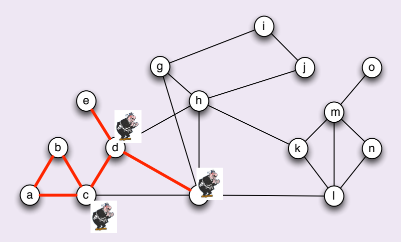
Allowed moves: Place $P(v)$, Remove $R(v)$, Slide $S(e)$
Clearing edges: when a searcher slides along it
Recontamination: if no searcher on a path from a clear edge to a contaminated one



P(c), P(c), P(e), S(ca), S(ab), S(ed), S(bc), S(cd),

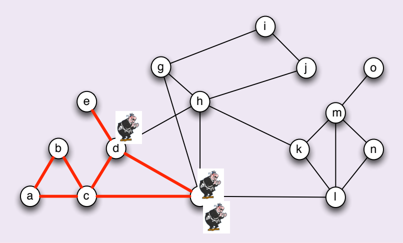
Studying Pathwidth via Graph Searching

Allowed moves: Place $P(v)$, Remove $R(v)$, Slide $S(e)$
Clearing edges: when a searcher slides along it
Recontamination: if no searcher on a path from a clear edge to a contaminated one



P(c), P(c), P(e), S(ca), S(ab), S(ed), S(bc), S(cd), S(df),

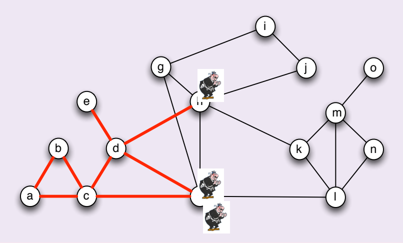
Allowed moves: Place $P(v)$, Remove $R(v)$, Slide $S(e)$
Clearing edges: when a searcher slides along it
Recontamination: if no searcher on a path from a clear edge to a contaminated one



P(c), P(c), P(e), S(ca), S(ab), S(ed), S(bc), S(cd), S(df), S(cf),

Studying Pathwidth via Graph Searching

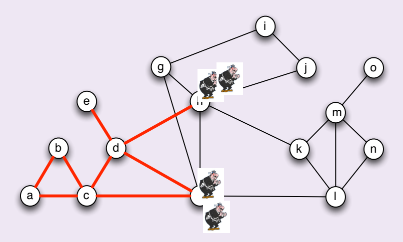
Allowed moves: Place $P(v)$, Remove $R(v)$, Slide $S(e)$
Clearing edges: when a searcher slides along it
Recontamination: if no searcher on a path from a clear edge to a contaminated one



P(c), P(c), P(e), S(ca), S(ab), S(ed), S(bc), S(cd), S(df), S(cf), S(dh),

Studying Pathwidth via Graph Searching

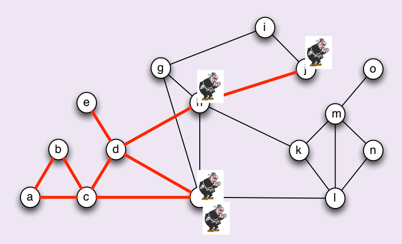
Allowed moves: Place $P(v)$, Remove $R(v)$, Slide $S(e)$
Clearing edges: when a searcher slides along it
Recontamination: if no searcher on a path from a clear edge to a contaminated one



P(c), P(c), P(e), S(ca), S(ab), S(ed), S(bc), S(cd), S(df), S(cf), S(dh), P(h),

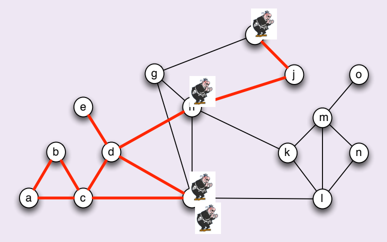
Studying Pathwidth via Graph Searching

Allowed moves: Place $P(v)$, Remove $R(v)$, Slide $S(e)$
Clearing edges: when a searcher slides along it
Recontamination: if no searcher on a path from a clear edge to a contaminated one



P(c), P(c), P(e), S(ca), S(ab), S(ed), S(bc), S(cd), S(df), S(cf), S(dh), P(h), S(hj),

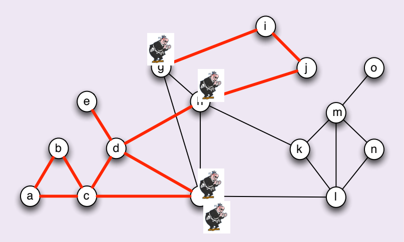
Allowed moves: Place $P(v)$, Remove $R(v)$, Slide $S(e)$
Clearing edges: when a searcher slides along it
Recontamination: if no searcher on a path from a clear edge to a contaminated one



P(c), P(c), P(e), S(ca), S(ab), S(ed), S(bc), S(cd), S(df), S(cf), S(dh), P(h), S(hj), S(ji),

Studying Pathwidth via Graph Searching

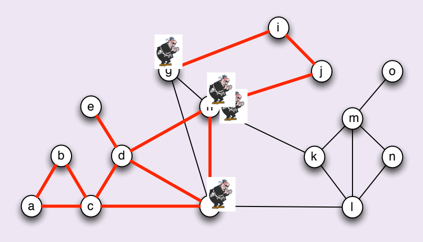
Allowed moves: Place $P(v)$, Remove $R(v)$, Slide $S(e)$
Clearing edges: when a searcher slides along it
Recontamination: if no searcher on a path from a clear edge to a contaminated one



P(c), P(c), P(e), S(ca), S(ab), S(ed), S(bc), S(cd), S(df), S(cf), S(dh), P(h), S(hj), S(ji), S(ig),

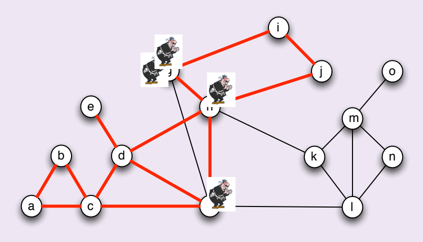
Studying Pathwidth via Graph Searching

Allowed moves: Place $P(v)$, Remove $R(v)$, Slide $S(e)$
Clearing edges: when a searcher slides along it
Recontamination: if no searcher on a path from a clear edge to a contaminated one



P(c), P(c), P(e), S(ca), S(ab), S(ed), S(bc), S(cd), S(df), S(cf), S(dh), P(h), S(hj), S(ji), S(ig), S(fh),

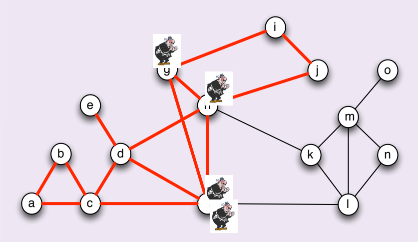
Allowed moves: Place $P(v)$, Remove $R(v)$, Slide $S(e)$
Clearing edges: when a searcher slides along it
Recontamination: if no searcher on a path from a clear edge to a contaminated one



P(c), P(c), P(e), S(ca), S(ab), S(ed), S(bc), S(cd), S(df), S(cf), S(dh), P(h), S(hj), S(ji), S(ig), S(fh), S(hg),

Studying Pathwidth via Graph Searching

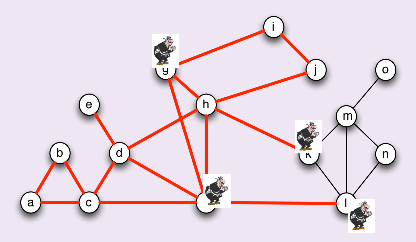
Allowed moves: Place $P(v)$, Remove $R(v)$, Slide $S(e)$
Clearing edges: when a searcher slides along it
Recontamination: if no searcher on a path from a clear edge to a contaminated one



P(c), P(c), P(e), S(ca), S(ab), S(ed), S(bc), S(cd), S(df), S(cf), S(dh), P(h), S(hj), S(ji), S(ig), S(fh), S(hg), S(gf),

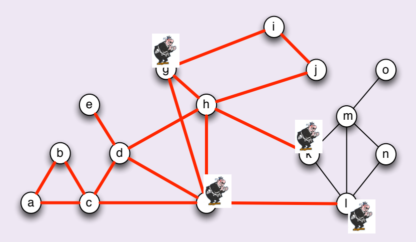
Studying Pathwidth via Graph Searching

Allowed moves: Place $P(v)$, Remove $R(v)$, Slide $S(e)$
Clearing edges: when a searcher slides along it
Recontamination: if no searcher on a path from a clear edge to a contaminated one



P(c), P(c), P(e), S(ca), S(ab), S(ed), S(bc), S(cd), S(df), S(cf), S(dh), P(h), S(hj), S(ji), S(ig), S(fh), S(hg), S(gf), S(hk), S(fl), etc. $\Rightarrow$ 4 searchers are sufficient

Studying Pathwidth via Graph Searching



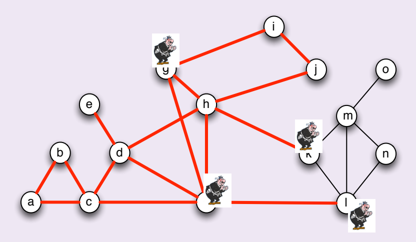
P(c), P(c), P(e), S(ca), S(ab), S(ed), S(bc), S(cd), S(df), S(cf), S(dh), P(h), S(hj), S(ji), S(ig), S(fh), S(hg), S(gf), S(hk), S(fl), etc. $\Rightarrow$ 4 searchers are sufficient

Relationship with path-decomposition

Induces an sequence on vertices: each time a **contaminated** node becomes occupied

$(c, e, a, b, d, f, h, j, i, g, k, l \cdots)$

Studying Pathwidth via Graph Searching



P(c), P(c), P(e), S(ca), S(ab), S(ed), S(bc), S(cd), S(df), S(cf), S(dh), P(h), S(hj), S(ji), S(ig), S(fh), S(hg), S(gf), S(hk), S(fl), etc. $\Rightarrow$ 4 searchers are sufficient

Relationship with path-decomposition

Induces an sequence on vertices: each time a **contaminated** node becomes occupied

$(c, e, a, b, d, f, h, j, i, g, k, l \cdots)$

If there is no recontamination: It is an ordering, i.e., a path-decomposition

Many variants of Graph Searching

		Edge-Search [Parsons'76]	Node-Search [Kirousis-Papdimitriou'86]	Mixed Search [Bienstock, Seymour'91]
Allowed moves	Place	yes	yes	yes
	Remove	yes	yes	yes
	Slide	yes	no	yes
Clearing moves	Slide	yes	no	yes
	2 ends occupied	no	yes	yes
Min. # of searchers		$es(G)$	$ns(G)$	$s(G)$

Many variants of Graph Searching

		Edge-Search [Parsons'76]	Node-Search [Kirousis-Papdimitriou'86]	Mixed Search [Bienstock, Seymour'91]
Allowed moves	Place	yes	yes	yes
	Remove	yes	yes	yes
	Slide	yes	no	yes
Clearing moves	Slide	yes	no	yes
	2 ends occupied	no	yes	yes
Min. # of searchers		$es(G)$	$ns(G)$	$s(G)$

Theorem [Bienstock, Seymour'91]

Three previous variants are monotone

i.e., there always exists an optimal strategy without recontamination

Consequence: for any graph G, $ns(G) = pw(G) + 1$.

Many variants of Graph Searching

		Edge-Search [Parsons'76]	Node-Search [Kirousis-Papdimitriou'86]	Mixed Search [Bienstock, Seymour'91]
Allowed moves	Place	yes	yes	yes
	Remove	yes	yes	yes
	Slide	yes	no	yes
Clearing moves	Slide	yes	no	yes
	2 ends occupied	no	yes	yes
Min. # of searchers		$es(G)$	$ns(G)$	$s(G)$

Link between variants (easy exercise)

For any graph G

- $s(G) \leq ns(G) \leq s(G) + 1$
- $s(G) \leq es(G) \leq s(G) + 1$
- $es(G) - 1 \leq ns(G) \leq es(G) + 1$

Many variants of Graph Searching

		Edge-Search [Parsons'76]	Node-Search [Kirousis-Papdimitriou'86]	Mixed Search [Bienstock, Seymour'91]
Allowed moves	Place	yes	yes	yes
	Remove	yes	yes	yes
	Slide	yes	no	yes
Clearing moves	Slide	yes	no	yes
	2 ends occupied	no	yes	yes
Min. # of searchers		$es(G)$	$ns(G)$	$s(G)$

Link between variants (easy exercise)

For any graph G

- $s(G) \leq ns(G) \leq s(G) + 1$
- $s(G) \leq es(G) \leq s(G) + 1$
- $es(G) - 1 \leq ns(G) \leq es(G) + 1$

Star: $s = ns = es = 2$, **Path:** $es = s = 1$, $ns = 2$; **In** $K_{r,r}$**:** $ns = s = r + 1$; $es = r + 2$

Complexity issues

	pathwidth pw (node-search ns)	edge-search es	mixed-search s
planar graphs with bounded maximum degree	NP-complete [Monien, Sudborough'88]		
split graphs	P [Gustedt'93]	P [Peng et al'00]	linear [FominHM10]
star-like graphs with ≥ 2 peripheral nodes per peripheral clique	NP-complete [Gustedt'93]	?	?
cographs	P [Bodlaender, M'93]	linear [GolovachHM12]	P [Heggernes, Mihai'08]

Open Problems

- Graph class where complexity differs ?
- Complexity of deciding if $pw(G)/es(G)/s(G)$ differ ?

Complexity issues

	pathwidth pw (node-search ns)	edge-search es	mixed-search s
planar graphs with bounded maximum degree	NP-complete [Monien, Sudborough'88]		
split graphs	P [Gustedt'93]	P [Peng et al'00]	linear [FominHM10]
star-like graphs with ≥ 2 peripheral nodes per peripheral clique	NP-complete [Gustedt'93]	?	?
cographs	P [Bodlaender, M'93]	linear [GolovachHM12]	P [Heggernes, Mihai'08]

Open Problems

- Graph class where complexity differs ?
- Complexity of deciding if $pw(G)/es(G)/s(G)$ differ ?

Study new variants of Graph Searching

to understand/approximate Pathwidth ?

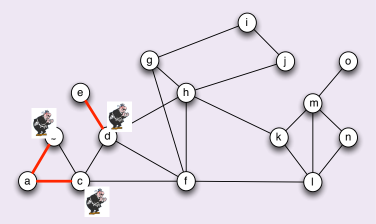
- Connected Graph Searching
- Exclusive Graph Searching

Connected Graph Searching

Connected Graph Searching [Barriere et al.'02]

"cleared" area must be always connected

Connected search number $cs(G)$: # min of Cops



example of non-connected step

Connected Graph Searching

Connected Graph Searching [Barriere et al.'02]

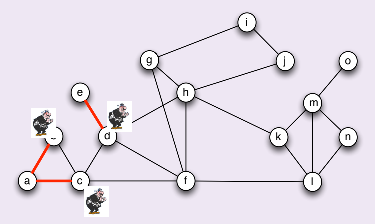
"cleared" area must be always connected

Connected search number $cs(G)$: # min of Cops

$\forall$ graph G, $cs(G) \leq 2pw(G) + O(1)$ [Dereniowski'12]

not monotone [Yang,Dyer,Alspach DM'09]

- open question: in NP?
- open question: FPT?



example of non-connected step

Connected Graph Searching

Connected Graph Searching [Barriere et al.'02]

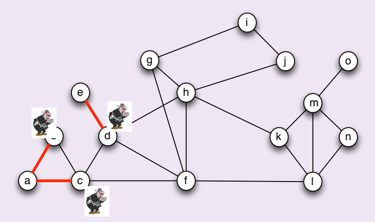
"cleared" area must be always connected

Connected search number $cs(G)$: # min of Cops

$\forall$ graph G, $cs(G) \leq 2pw(G) + O(1)$ [Dereniowski'12]

not monotone [Yang,Dyer,Alspach DM'09]

- open question: in NP?
- open question: FPT?



example of non-connected step

Approximate Pathwidth via connected Search?

pw is NP-hard in weighted trees [Mihai,Todinca FAW'09]

3-approximation for *cs* in weighted trees [Dereniowski TCS'12]

Other graph classes (chordal,...) ?

Exclusive Graph Searching

Exclusive Graph Searching [Burman,Blin,N.'12]

New Constraint

- Exclusivity: at most one searcher per node

Allowed moves

- Only initially: place some searchers on distinct nodes
- then, only slide is allowed (in particular: no searchers may be added)

Clearing edges

- when a searcher slides along it **OR** if both ends occupied

Recontamination: if no searcher on a path from a clear edge to a contaminated one

Exclusive Graph Searching

Exclusive Graph Searching [Burman,Blin,N.'12]

New Constraint

- Exclusivity: at most one searcher per node

Allowed moves

- Only initially: place some searchers on distinct nodes
- then, only slide is allowed (in particular: no searchers may be added)

Clearing edges

- when a searcher slides along it **OR** if both ends occupied

Recontamination: if no searcher on a path from a clear edge to a contaminated one

Exclusive Graph Searching

Exclusive Graph Searching [Burman,Blin,N.'12]

New Constraint

- Exclusivity: at most one searcher per node

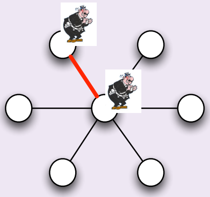
Allowed moves

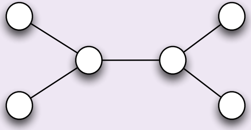
- Only initially: place some searchers on distinct nodes
- then, only slide is allowed (in particular: no searchers may be added)

Clearing edges

- when a searcher slides along it **OR** if both ends occupied

Recontamination: if no searcher on a path from a clear edge to a contaminated one





Exclusive Graph Searching

Exclusive Graph Searching [Burman,Blin,N.'12]

New Constraint

- Exclusivity: at most one searcher per node

Allowed moves

- Only initially: place some searchers on distinct nodes
- then, only slide is allowed (in particular: no searchers may be added)

Clearing edges

- when a searcher slides along it **OR** if both ends occupied

Recontamination: if no searcher on a path from a clear edge to a contaminated one

Exclusive Graph Searching

Exclusive Graph Searching [Burman,Blin,N.'12]

New Constraint

- Exclusivity: at most one searcher per node

Allowed moves

- Only initially: place some searchers on distinct nodes
- then, only slide is allowed (in particular: no searchers may be added)

Clearing edges

- when a searcher slides along it **OR** if both ends occupied

Recontamination: if no searcher on a path from a clear edge to a contaminated one

Exclusive Graph Searching

Exclusive Graph Searching [Burman,Blin,N.'12]

New Constraint

- Exclusivity: at most one searcher per node

Allowed moves

- Only initially: place some searchers on distinct nodes
- then, only slide is allowed (in particular: no searchers may be added)

Clearing edges

- when a searcher slides along it **OR** if both ends occupied

Recontamination: if no searcher on a path from a clear edge to a contaminated one

Exclusive Graph Searching

Exclusive Graph Searching [Burman,Blin,N.'12]

New Constraint

- Exclusivity: at most one searcher per node

Allowed moves

- Only initially: place some searchers on distinct nodes
- then, only slide is allowed (in particular: no searchers may be added)

Clearing edges

- when a searcher slides along it **OR** if both ends occupied

Recontamination: if no searcher on a path from a clear edge to a contaminated one

Exclusive Graph Searching

Exclusive Graph Searching [Burman,Blin,N.'12]

New Constraint

- Exclusivity: at most one searcher per node

Allowed moves

- Only initially: place some searchers on distinct nodes
- then, only slide is allowed (in particular: no searchers may be added)

Clearing edges

- when a searcher slides along it **OR** if both ends occupied

Recontamination: if no searcher on a path from a clear edge to a contaminated one

Exclusive Graph Searching

Exclusive Graph Searching [Burman,Blin,N.'12]

New Constraint

- Exclusivity: at most one searcher per node

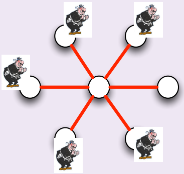
Allowed moves

- Only initially: place some searchers on distinct nodes
- then, only slide is allowed (in particular: no searchers may be added)

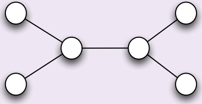
Clearing edges

- when a searcher slides along it **OR** if both ends occupied

Recontamination: if no searcher on a path from a clear edge to a contaminated one



xs(star)$=\Delta - 1$



Exclusive Graph Searching

Exclusive Graph Searching [Burman,Blin,N.'12]

New Constraint

- Exclusivity: at most one searcher per node

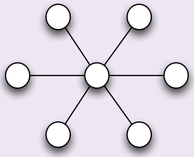
Allowed moves

- Only initially: place some searchers on distinct nodes
- then, only slide is allowed (in particular: no searchers may be added)

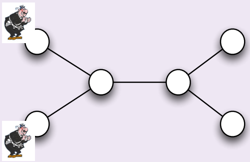
Clearing edges

- when a searcher slides along it **OR** if both ends occupied

Recontamination: if no searcher on a path from a clear edge to a contaminated one



$xs(star) = \Delta - 1$



Exclusive Graph Searching

Exclusive Graph Searching [Burman,Blin,N.'12]

New Constraint

- Exclusivity: at most one searcher per node

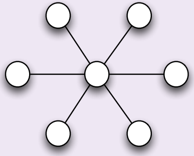
Allowed moves

- Only initially: place some searchers on distinct nodes
- then, only slide is allowed (in particular: no searchers may be added)

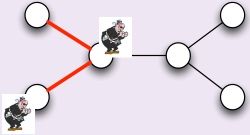
Clearing edges

- when a searcher slides along it **OR** if both ends occupied

Recontamination: if no searcher on a path from a clear edge to a contaminated one



$xs(star) = \Delta - 1$



Exclusive Graph Searching

Exclusive Graph Searching [Burman,Blin,N.'12]

New Constraint

- Exclusivity: at most one searcher per node

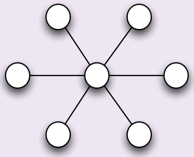
Allowed moves

- Only initially: place some searchers on distinct nodes
- then, only slide is allowed (in particular: no searchers may be added)

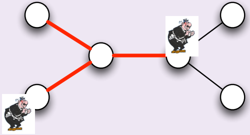
Clearing edges

- when a searcher slides along it **OR** if both ends occupied

Recontamination: if no searcher on a path from a clear edge to a contaminated one



$xs(star) = \Delta - 1$



Exclusive Graph Searching [Burman,Blin,N.'12]

New Constraint

- Exclusivity: at most one searcher per node

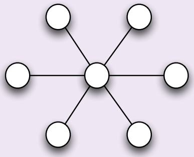
Allowed moves

- Only initially: place some searchers on distinct nodes
- then, only slide is allowed (in particular: no searchers may be added)

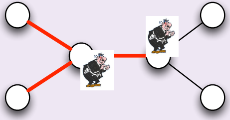
Clearing edges

- when a searcher slides along it **OR** if both ends occupied

Recontamination: if no searcher on a path from a clear edge to a contaminated one



$xs(star) = \Delta - 1$



Exclusive Graph Searching [Burman,Blin,N.'12]

New Constraint

- Exclusivity: at most one searcher per node

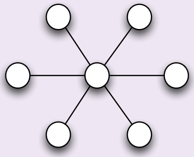
Allowed moves

- Only initially: place some searchers on distinct nodes
- then, only slide is allowed (in particular: no searchers may be added)

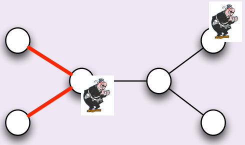
Clearing edges

- when a searcher slides along it **OR** if both ends occupied

Recontamination: if no searcher on a path from a clear edge to a contaminated one



$xs(star) = \Delta - 1$



Recontamination!

Exclusive Graph Searching

Exclusive Graph Searching [Burman,Blin,N.'12]

New Constraint

- Exclusivity: at most one searcher per node

Allowed moves

- Only initially: place some searchers on distinct nodes
- then, only slide is allowed (in particular: no searchers may be added)

Clearing edges

- when a searcher slides along it **OR** if both ends occupied

Recontamination: if no searcher on a path from a clear edge to a contaminated one

$xs(star) = \Delta - 1$

Exclusive Graph Searching

Exclusive Graph Searching [Burman,Blin,N.'12]

New Constraint

- Exclusivity: at most one searcher per node

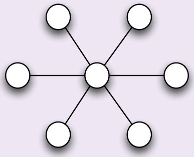
Allowed moves

- Only initially: place some searchers on distinct nodes
- then, only slide is allowed (in particular: no searchers may be added)

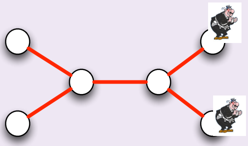
Clearing edges

- when a searcher slides along it **OR** if both ends occupied

Recontamination: if no searcher on a path from a clear edge to a contaminated one



$xs(star) = \Delta - 1$



No optimal monotone strategy :(

Results on Exclusive Graph Searching

Exclusive Graph Searching

new constraint: at most one Cop per node at every step [Blin,Burman,N.'13]
(Cops can slide along edges)

$xs(G)$: min # of Cops $mxs(G)$: min # of Cops for monotone strategies

Results on Exclusive Graph Searching

Exclusive Graph Searching

new constraint: at most one Cop per node at every step [Blin,Burman,N.'13]
(Cops can slide along edges)

$xs(G)$: min # of Cops $mxs(G)$: min # of Cops for monotone strategies

variant not monotone ($xs(G)$ may differ from $mxs(G)$) [Blin,Burman,N.'13]
For any graph G with max. degree Δ, $s(G) \leq xs(G) \leq (\Delta - 1)(s(G) + 1)$

Results on Exclusive Graph Searching

Exclusive Graph Searching

new constraint: at most one Cop per node at every step [Blin,Burman,N.'13]
(Cops can slide along edges)

$xs(G)$: min # of Cops $mxs(G)$: min # of Cops for monotone strategies

variant not monotone ($xs(G)$ may differ from $mxs(G)$) [Blin,Burman,N.'13]
For any graph G with max. degree Δ, $s(G) \leq xs(G) \leq (\Delta - 1)(s(G) + 1)$

About complexity: Computing xs is

- NP-hard in planar graphs with max degree 3 [Markou,N.,Pérennes]
- polynomial in trees [Blin,Burman,N.'13]
- linear in cographs [Markou,N.,Pérennes]

	pathwidth [Gustedt'93]	monotone exclusive-search [Markou,N.,Pérennes]
split graphs	**P**	**NP-complete**
star-like graphs with ≥ 2 peripheral nodes per clique	**NP-complete**	**P**

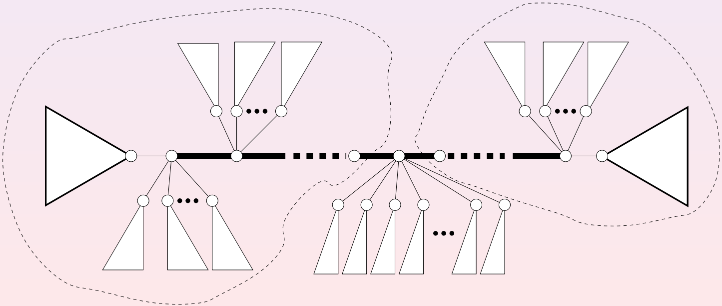
Exclusive Graph Searching in trees

Theorem: characterization of trees with $xs(T) \leq k$ [Blin,Burman,N.'13]

Let $k \geq 1$. For any tree T, $xs(T) \leq k$ iff for any node v:

1. v has degree at most $k+1$;
2. for any branch B at v, $xs(B) \leq k$;
3. for any even $i > 1$, at most i branches B at v have $xs(B) \geq k - i/2 + 1$.

Gives a polynomial-time algorithm using dynamic programming

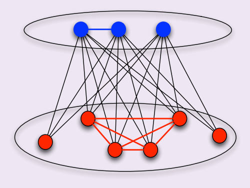


Exclusive Graph Searching in Cograph

Reminder: a graph is a cograph if

- single vertex, or
- disjoint union $G_1 \bigcup G_2$ of 2 cographs, or
- join $G_1 \otimes G_2$ of 2 cographs (add complete bipartite between G_1 and G_2)

The decomposition can be obtained in linear time [Corneil, Perl, Steward'85]



Exclusive search is monotone in cographs and linear-time algo. [Markou,N.,Pérennes]

Let G_1 and G_2 two cographs.

- $xs(G_1 \bigcup G_2) = xs(G_1) + xs(G_2)$
- $xs(G_1 \otimes G_2) \approx \min\{xs(G_1) + |V(G_2)|, xs(G_2) + |V(G_1)|\}$

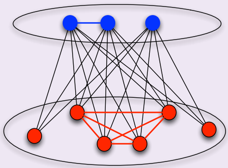
(small difference if G_1 or G_2 are not connected, but can be well characterized)

Exclusive Graph Searching in Cograph

Reminder: a graph is a cograph if

- single vertex, or
- disjoint union $G_1 \bigcup G_2$ of 2 cographs, or
- join $G_1 \otimes G_2$ of 2 cographs (add complete bipartite between G_1 and G_2)

The decomposition can be obtained in linear time [Corneil, Perl, Steward'85]



Exclusive search is monotone in cographs and linear-time algo. [Markou,N.,Pérennes]

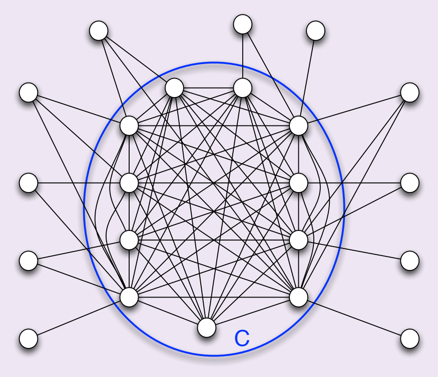
Let G_1 and G_2 two cographs.

- $xs(G_1 \bigcup G_2) = xs(G_1) + xs(G_2)$
- $xs(G_1 \otimes G_2) \approx \min\{xs(G_1) + |V(G_2)|, xs(G_2) + |V(G_1)|\}$

(small difference if G_1 or G_2 are not connected, but can be well characterized)

Exclusive Graph Searching in Split Graphs

Split Graph: $G = (I \cup C, E)$ if C induces a clique and I induces an independent set



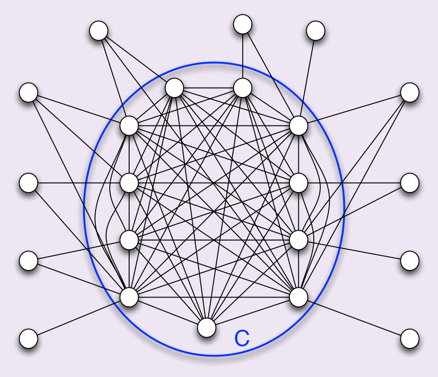
Exclusive Graph Searching in Split Graphs

Split Graph: $G = (I \cup C, E)$ if C induces a clique and I induces an independent set

Charaterization of monotone strategies

$mxs(G) \leq k \Leftrightarrow$ it exists a particular strategy

-
-
-



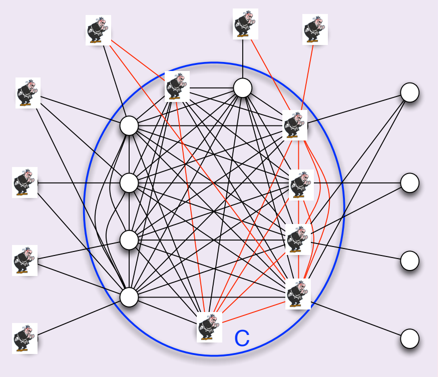
Exclusive Graph Searching in Split Graphs

Split Graph: $G = (I \cup C, E)$ if C induces a clique and I induces an independent set

Charaterization of monotone strategies

$mxs(G) \leq k \Leftrightarrow$ it exists a particular strategy

-
-
-



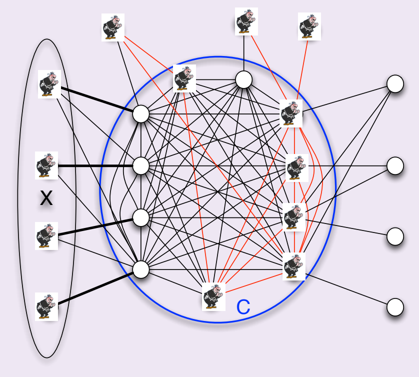
Exclusive Graph Searching in Split Graphs

Split Graph: $G = (I \cup C, E)$ if C induces a clique and I induces an independent set

Charaterization of monotone strategies

$mxs(G) \leq k \Leftrightarrow$ it exists a particular strategy

- slide along a matching from $X \subseteq I$ to C
-
-



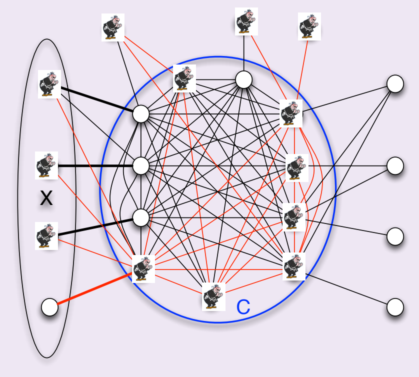
Exclusive Graph Searching in Split Graphs

Split Graph: $G = (I \cup C, E)$ if C induces a clique and I induces an independent set

Charaterization of monotone strategies

$mxs(G) \leq k \Leftrightarrow$ it exists a particular strategy

- slide along a matching from $X \subseteq I$ to C
-
-

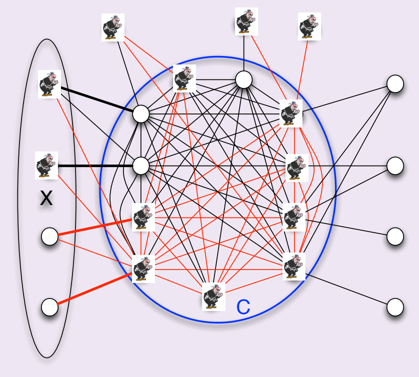


Split Graph: $G = (I \cup C, E)$ if C induces a clique and I induces an independent set

Charaterization of monotone strategies

$mxs(G) \leq k \Leftrightarrow$ it exists a particular strategy

- slide along a matching from $X \subseteq I$ to C
-
-



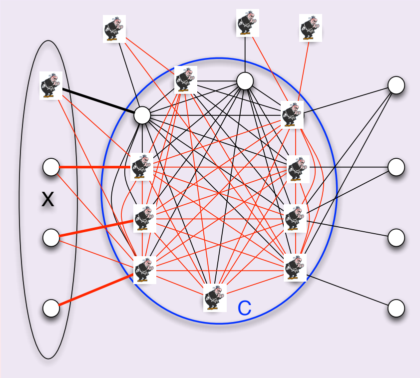
Exclusive Graph Searching in Split Graphs

Split Graph: $G = (I \cup C, E)$ if C induces a clique and I induces an independent set

Charaterization of monotone strategies

$mxs(G) \leq k \Leftrightarrow$ it exists a particular strategy

- slide along a matching from $X \subseteq I$ to C
-
-



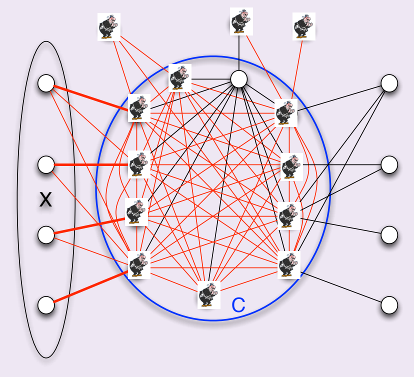
Exclusive Graph Searching in Split Graphs

Split Graph: $G = (I \cup C, E)$ if C induces a clique and I induces an independent set

Charaterization of monotone strategies

$mxs(G) \leq k \Leftrightarrow$ it exists a particular strategy

- slide along a matching from $X \subseteq I$ to C
-
-



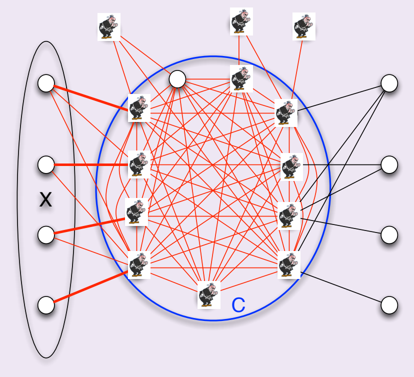
Exclusive Graph Searching in Split Graphs

Split Graph: $G = (I \cup C, E)$ if C induces a clique and I induces an independent set

Charaterization of monotone strategies

$mxs(G) \leq k \Leftrightarrow$ it exists a particular strategy

- slide along a matching from $X \subseteq I$ to C
- may slide along ONE edge in C
-



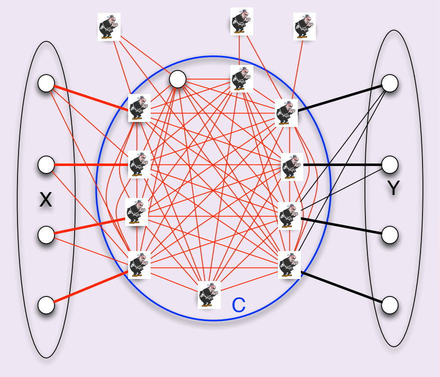
Exclusive Graph Searching in Split Graphs

Split Graph: $G = (I \cup C, E)$ if C induces a clique and I induces an independent set

Charaterization of monotone strategies

$mxs(G) \leq k \Leftrightarrow$ it exists a particular strategy

- slide along a matching from $X \subseteq I$ to C
- may slide along ONE edge in C
- slide along a matching from C to $Y \subseteq I \setminus X$



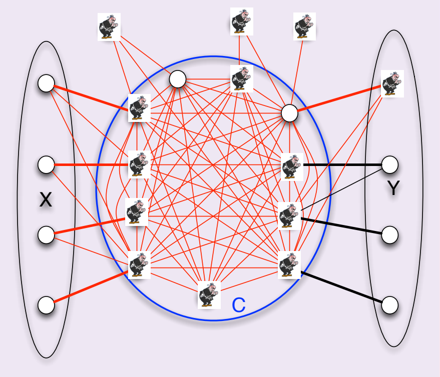
Exclusive Graph Searching in Split Graphs

Split Graph: $G = (I \cup C, E)$ if C induces a clique and I induces an independent set

Charaterization of monotone strategies

$mxs(G) \leq k \Leftrightarrow$ it exists a particular strategy

- slide along a matching from $X \subseteq I$ to C
- may slide along ONE edge in C
- slide along a matching from C to $Y \subseteq I \setminus X$



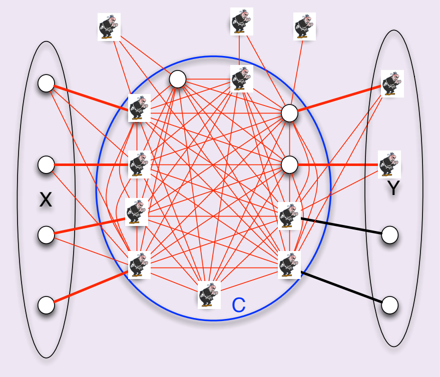
Exclusive Graph Searching in Split Graphs

Split Graph: $G = (I \cup C, E)$ if C induces a clique and I induces an independent set

Charaterization of monotone strategies

$mxs(G) \leq k \Leftrightarrow$ it exists a particular strategy

- slide along a matching from $X \subseteq I$ to C
- may slide along ONE edge in C
- slide along a matching from C to $Y \subseteq I \setminus X$



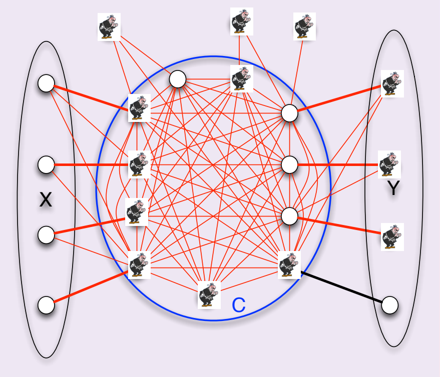
Exclusive Graph Searching in Split Graphs

Split Graph: $G = (I \cup C, E)$ if C induces a clique and I induces an independent set

Charaterization of monotone strategies

$mxs(G) \leq k \Leftrightarrow$ it exists a particular strategy

- slide along a matching from $X \subseteq I$ to C
- may slide along ONE edge in C
- slide along a matching from C to $Y \subseteq I \setminus X$



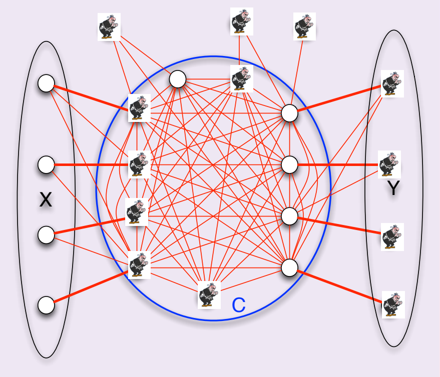
Exclusive Graph Searching in Split Graphs

Split Graph: $G = (I \cup C, E)$ if C induces a clique and I induces an independent set

Charaterization of monotone strategies

$mxs(G) \leq k \Leftrightarrow$ it exists a particular strategy

- slide along a matching from $X \subseteq I$ to C
- may slide along ONE edge in C
- slide along a matching from C to $Y \subseteq I \setminus X$



Exclusive Graph Searching in Split Graphs

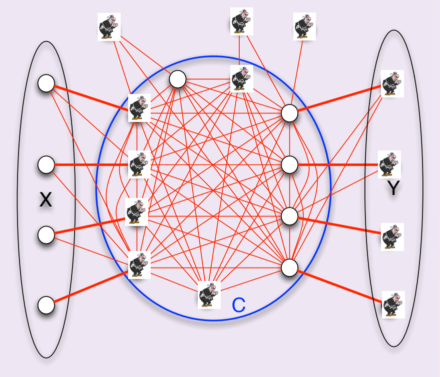
Split Graph: $G = (I \cup C, E)$ if C induces a clique and I induces an independent set

Charaterization of monotone strategies

$mxs(G) \leq k \Leftrightarrow$ it exists a particular strategy

- slide along a matching from $X \subseteq I$ to C
- may slide along ONE edge in C
- slide along a matching from C to $Y \subseteq I \setminus X$

It uses $k = |V| - |X| - |Y| \ (-1)$ searchers



Exclusive Graph Searching in Split Graphs

Split Graph: $G = (I \cup C, E)$ if C induces a clique and I induces an independent set

Charaterization of monotone strategies

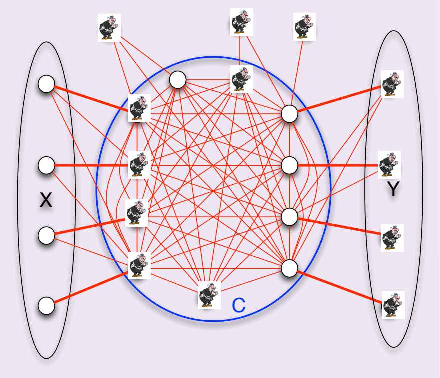
$mxs(G) \leq k \Leftrightarrow$ it exists a particular strategy

- slide along a matching from $X \subseteq I$ to C
- may slide along ONE edge in C
- slide along a matching from C to $Y \subseteq I \setminus X$

It uses $k = |V| - |X| - |Y|$ (-1) searchers

Find particular subsets as large as possible

$$X = \{x_1, \cdots, x_r\} \textit{ and } N(x_i) \setminus \bigcup_{j<i} N(x_j) \neq \emptyset.$$



Exclusive Graph Searching in Split Graphs

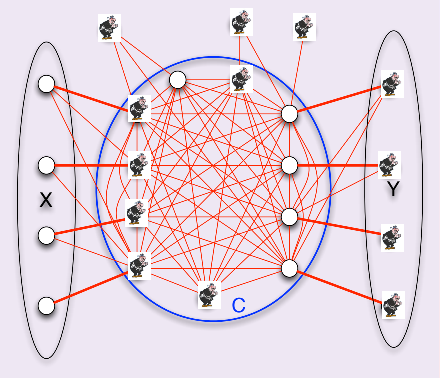
Split Graph: $G = (I \cup C, E)$ if C induces a clique and I induces an independent set

Charaterization of monotone strategies

$mxs(G) \leq k \Leftrightarrow$ it exists a particular strategy

- slide along a matching from $X \subseteq I$ to C
- may slide along ONE edge in C
- slide along a matching from C to $Y \subseteq I \setminus X$

It uses $k = |V| - |X| - |Y|$ (-1) searchers



New problem:

input: $\{S_1, \cdots, S_n\}$ subsets of ground set A
output: a sequence $(S_{i_1}, \cdots, S_{i_r})$ such that $(\bigcup_{j \leq k} S_{i_j})_k$ strictly increasing and r is maximum

NP-hard (reduction from MIN-SAT)

Exclusive Graph Searching in Split Graphs

Split Graph: $G = (I \cup C, E)$ if C induces a clique and I induces an independent set

Charaterization of monotone strategies

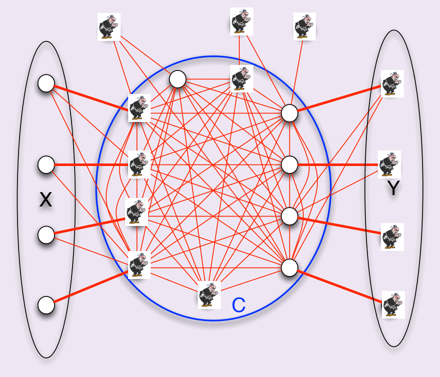
$mxs(G) \leq k \Leftrightarrow$ it exists a particular strategy

- slide along a matching from $X \subseteq I$ to C
- may slide along ONE edge in C
- slide along a matching from C to $Y \subseteq I \setminus X$

It uses $k = |V| - |X| - |Y| \ (-1)$ searchers

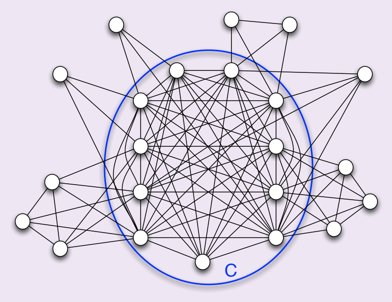
Theorem [Markou,N.,Pérennes]

Computing *mxs* is NP-complete in split graphs (contrary to pathwidth)



Exclusive Graph Searching in Split Graphs

Star-like: One Central clique C_0 and Peripheral cliques intersecting only in C_0



Theorem: Strategies are very constrained [Markou,N.,Pérennes]

G a star-like graph with each peripheral clique has at least two peripheral nodes.

1. Either there is an edge of C_0 that does not belong to any peripheral clique, and $mxs(G) = |V(G)| - r - 1$,
2. or $mxs(G) = |V(G)| - r$.

Perspectives on Exclusive Graph Searching

- Are there graph classes where *pw* is NP-complete
  and *xs* (*mxs*) in *P* and provide good approximation of *pw*?
  (or *vice-versa*)
- Can *xs* (or *mxs*) be approximated?
- *xs* in NP?
- *xs* (or *mxs*) FPT?
- $xs = mxs$ in split graphs?
- ...