k-Chordal Graphs: from Cops and Robber to Compact Routing via Treewidth[1]

Adrian Kosowski[1] Bi Li[2,3] Nicolas Nisse[2] and Karol Suchan[4,5]

[1] CEPAGE, INRIA, Univ. Bordeaux 1, France
[2] MASCOTTE, INRIA, I3S (CNRS, UNS) Sophia Antipolis, France
[3] AMSS, CAS, China
[4] Univ. Adolfo Ibanez, Facultad de Ingenieria y Ciencias, Santiago, Chile
[5] WMS, AGH - Univ. of Science and Technology, Krakow, Poland

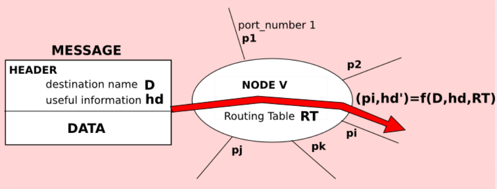
AlgoTel, la Grande Motte, 31^{st} May, 2012

[1] to be presented at ICALP'12 by Bi

(distributed) Routing in the Internet

Routing Scheme protocol that directs the traffic in a network

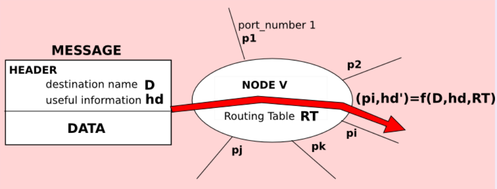
pre-requisite: computation of Routing Tables (RT)



(distributed) Routing in the Internet

Routing Scheme — protocol that directs the traffic in a network

pre-requisite: computation of Routing Tables (RT)



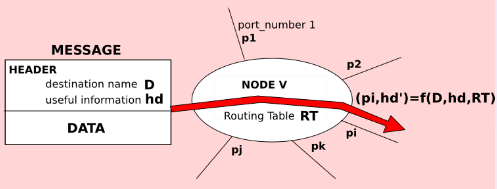
Border Gateway Protocol (BGP): (AS network)

RT's of size $O(n \log n)$ bits — "almost" the full topology

problem to compute/update $\Rightarrow$ **How to reduce their size?**

(distributed) Routing in the Internet

Routing Scheme — protocol that directs the traffic in a network
pre-requisite: — computation of Routing Tables (RT)



Border Gateway Protocol (BGP): (AS network)

RT's of size $O(n \log n)$ bits — "almost" the full topology
problem to compute/update $\Rightarrow$ **How to reduce their size?**

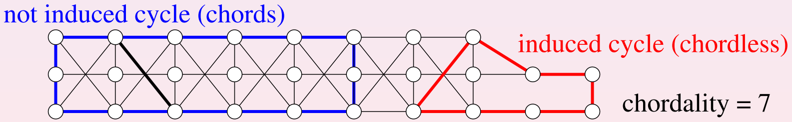
Compact routing along shortest paths

General graphs — $\Omega(n \log n)$ bits required [FG'97]
$\Rightarrow$ **need of structural properties**

Well known properties	graph parameters
small diameter (logarithmic)	($\Rightarrow$ small hyperbolicity)
power law degree distribution	
high clustering coefficient	$\Rightarrow$ **few long induced cycles**

Well known properties	graph parameters
small diameter (logarithmic)	($\Rightarrow$ small hyperbolicity)
power law degree distribution	
high clustering coefficient	$\Rightarrow$ **few long induced cycles**

Chordality of a graph G: length of **greatest induced cycle** in G



Brief related work on chordality

Complexity — chordality $\leq k$?

NP-complete	easy reduction from hamiltonian cycle
not FPT [CF'07]	no algorithm $f(k).poly(n)$ (unless $P = NP$)
FPT in planar graphs [KK'09]	Graph Minor Theory

chordality $\leq k \Rightarrow$ treewidth $\leq O(\Delta^k)$ [Bodlaender, Thilikos'97]

Compact routing schemes in graphs with chordality $\leq k$

stretch	RT's size	computation time	
$k+1$	$O(k \log^2 n)$	$poly(n)$	[Dourisboure'05]
header never changes			
$k-1$	$O(\Delta \log n)$	$O(D)$	[NRS'09]
distributed protocol to compute RT's / no header			
$O(k \log \Delta)$	$O(k \log n)$	$O(m^2)$	**[this paper]**

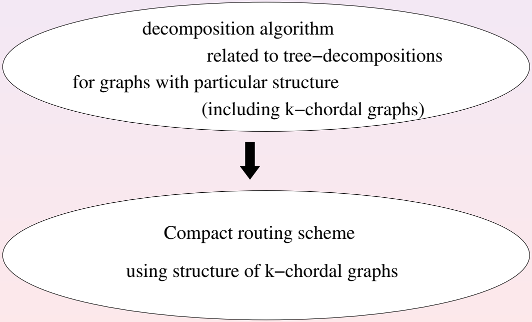
Names and Headers (if any) are of polylogarithmic size

From Cops and robber to Routing via Treewidth

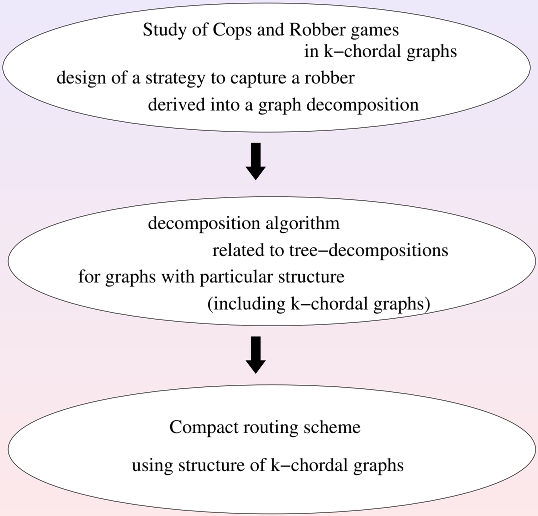
Compact routing scheme

using structure of k−chordal graphs

From Cops and robber to Routing via Treewidth



From Cops and robber to Routing via Treewidth



Our results

Theorem 1: Cops and Robber games

$k-1$ cops are sufficient to capture a robber in k-chordal graphs

Theorem 2: main result

There is a $O(m^2)$-algorithm that, in any m-edge graph G,

- either returns an induced cycle larger than k,
- or compute a tree-decomposition with each bag being the closed neighborhood of an induced path of length $\leq k-1$.

($\Rightarrow$ treewidth $\leq O(\Delta.k)$ and treelength $\leq k$)

Theorem 3: for any graph admitting such a tree-decomposition

there is a compact routing scheme using RT's of size $O(k \log n)$ bits, and achieving additive stretch $O(k \log \Delta)$.

Cops & robber games [Nowakowski and Winkler; Quilliot, 83]

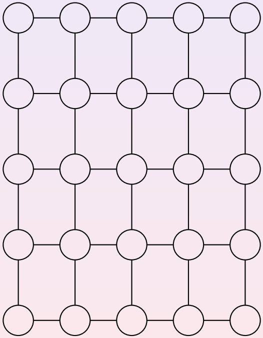
Initialization:

1. $\mathcal{C}$ places the cops;
2. $\mathcal{R}$ places the robber.

Step-by-step:

- each cop traverses at most 1 edge;
- the robber traverses at most 1 edge.

Robber captured:
A cop occupies the same vertex as the robber.



Cops & robber games [Nowakowski and Winkler; Quilliot, 83]

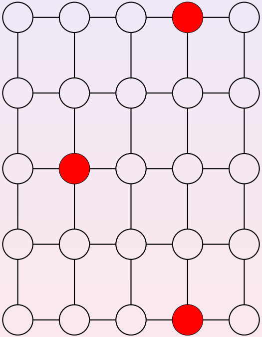
Initialization:

1. $\mathcal{C}$ places the cops;
2. $\mathcal{R}$ places the robber.

Step-by-step:

- each cop traverses at most 1 edge;
- the robber traverses at most 1 edge.

Robber captured:
A cop occupies the same vertex as the robber.



Cops & robber games [Nowakowski and Winkler; Quilliot, 83]

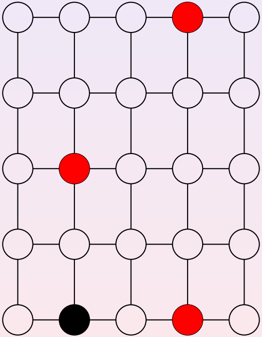
Initialization:

1. $\mathcal{C}$ places the cops;
2. $\mathcal{R}$ places the robber.

Step-by-step:

- each cop traverses at most 1 edge;
- the robber traverses at most 1 edge.

Robber captured:
A cop occupies the same vertex as the robber.



Cops & robber games [Nowakowski and Winkler; Quilliot, 83]

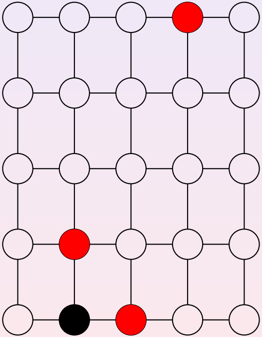
Initialization:

1. $\mathcal{C}$ places the cops;
2. $\mathcal{R}$ places the robber.

Step-by-step:

- each cop traverses at most 1 edge;
- the robber traverses at most 1 edge.

Robber captured:
A cop occupies the same vertex as the robber.



Cops & robber games [Nowakowski and Winkler; Quilliot, 83]

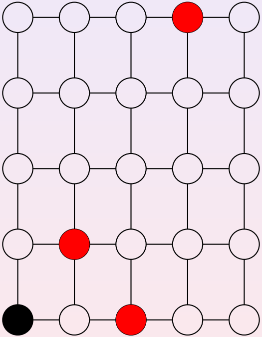
Initialization:

1. $\mathcal{C}$ places the cops;
2. $\mathcal{R}$ places the robber.

Step-by-step:

- each cop traverses at most 1 edge;
- the robber traverses at most 1 edge.

Robber captured:
A cop occupies the same vertex as the robber.



Cops & robber games [Nowakowski and Winkler; Quilliot, 83]

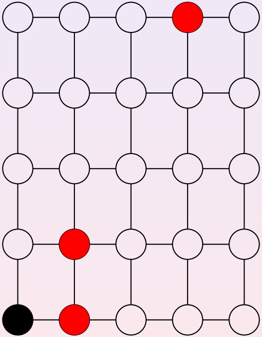
Initialization:

1. $\mathcal{C}$ places the cops;
2. $\mathcal{R}$ places the robber.

Step-by-step:

- each cop traverses at most 1 edge;
- the robber traverses at most 1 edge.

Robber captured:
A cop occupies the same vertex as the robber.



Cops & robber games [Nowakowski and Winkler; Quilliot, 83]

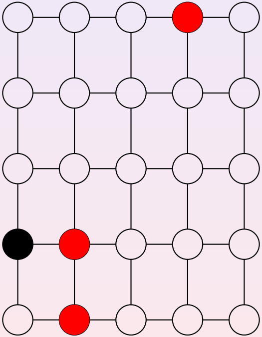
Initialization:

1. $\mathcal{C}$ places the cops;
2. $\mathcal{R}$ places the robber.

Step-by-step:

- each cop traverses at most 1 edge;
- the robber traverses at most 1 edge.

Robber captured:
A cop occupies the same vertex as the robber.



Cops & robber games [Nowakowski and Winkler; Quilliot, 83]

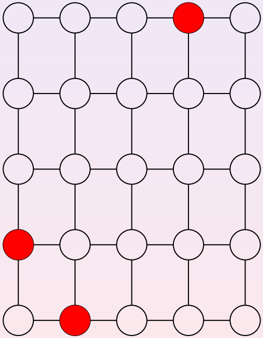
Initialization:

1. $\mathcal{C}$ places the cops;
2. $\mathcal{R}$ places the robber.

Step-by-step:

- each cop traverses at most 1 edge;
- the robber traverses at most 1 edge.

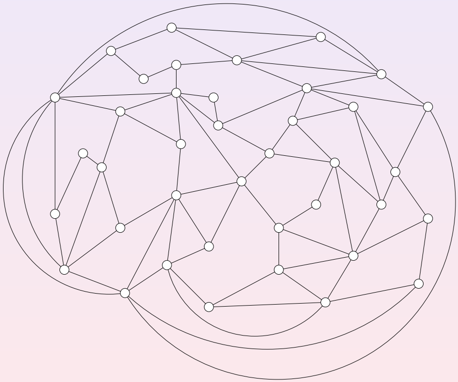
Robber captured:
A cop occupies the same vertex as the robber.



Cop number

$cn(G)$ minimum number of cops to capture any robber

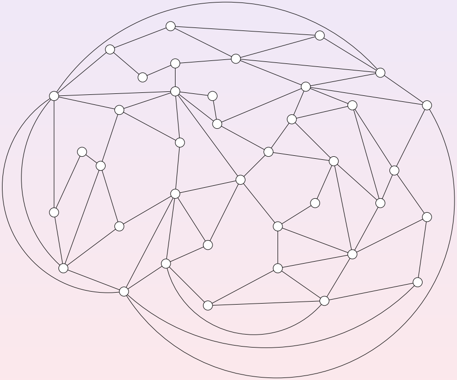
Determine $cn(G)$ for the following graph G?



Cop number

$cn(G)$ minimum number of cops to capture any robber

Determine $cn(G)$ for the following graph G? ≤ 3



$cn(G) \leq 3$ for any planar graph G [Aigner, Fromme, 84]

Cops & robber games: the graph structure helps!!

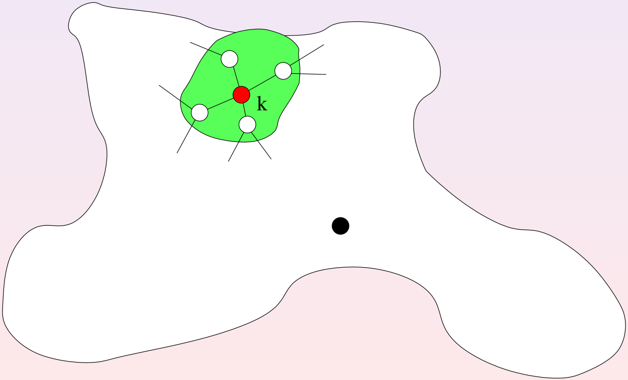
- G with **girth** g (min induced cycle) and **min degree** d: $cn(G) \geq d^g$ [Frankl 87]
- $\exists$ n-node graphs G (projective plane): $cn(G) = \Theta(\sqrt{n})$ [Frankl 87]
- G with **dominating set** k: $cn(G) \leq k$ [folklore]
- **Planar graph** G: $cn(G) \leq 3$ [Aigner, Fromme, 84]
- **Minor free graph** G excluding a minor H: $cn(G) \leq |E(H)|$ [Andreae, 86]
- G with **genus** g: $cn(G) \leq 3/2g + 3$ [Schröder, 01]
- G with **treewidth** t: $cn(G) \leq t/2 + 1$ [Joret, Kaminsk,Theis 09]
- G **random graph** (Erdös Reyni): $cn(G) = O(\sqrt{n})$ [Bollobas *et al.* 08]
- **any** n-node graph G: $cn(G) = O(\frac{n}{2^{(1+o(1))\sqrt{\log n}}})$ [Lu,Peng 09, Scott,Sudakov 10]

Theorem 1

G with chordality k: $cn(G) \leq k - 1$.

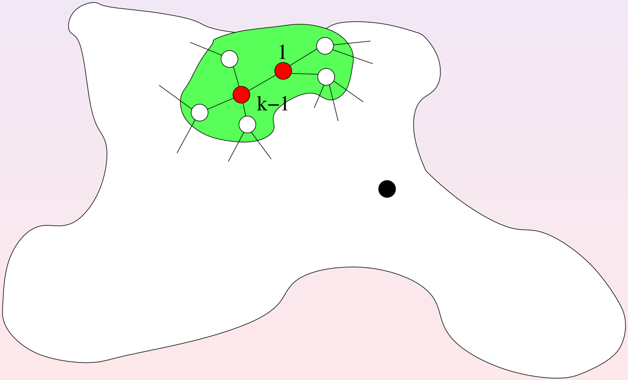
initialization: all k cops in one arbitrary node $P = \{v_1\}$

invariant: Cops always occupy an induced path $P = \{v_1, \cdots, v_i\}$



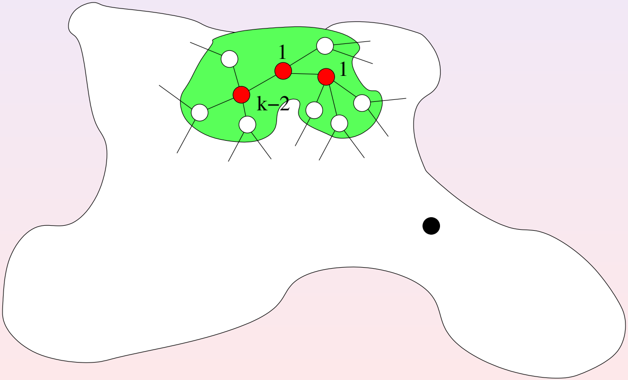
initialization: all k cops in one arbitrary node $P = \{v_1\}$
invariant: Cops always occupy an induced path $P = \{v_1, \cdots, v_i\}$
algorithm:

extension: if $w \in N(v_1) \cup N(v_i)$, Pw induced and $N(w) \cap C_{robber} \neq \emptyset$



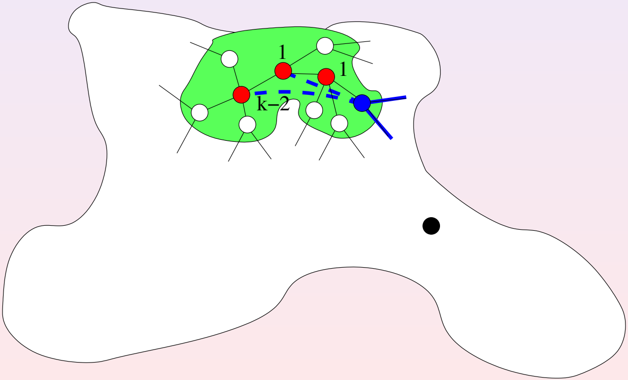
initialization: all k cops in one arbitrary node $P = \{v_1\}$
invariant: Cops always occupy an induced path $P = \{v_1, \cdots, v_i\}$
algorithm:

extension: if $w \in N(v_1) \cup N(v_i)$, Pw induced and $N(w) \cap C_{robber} \neq \emptyset$



initialization: all k cops in one arbitrary node $P = \{v_1\}$
invariant: Cops always occupy an induced path $P = \{v_1, \cdots, v_i\}$
algorithm:

extension: if $w \in N(v_1) \cup N(v_i)$, Pw induced and $N(w) \cap C_{robber} \neq \emptyset$

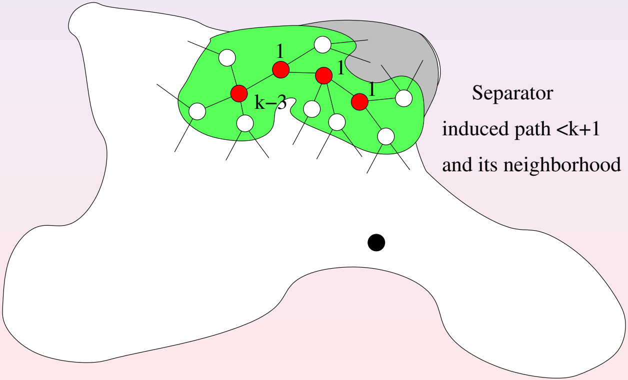


initialization: all k cops in one arbitrary node $P = \{v_1\}$

invariant: Cops always occupy an induced path $P = \{v_1, \cdots, v_i\}$

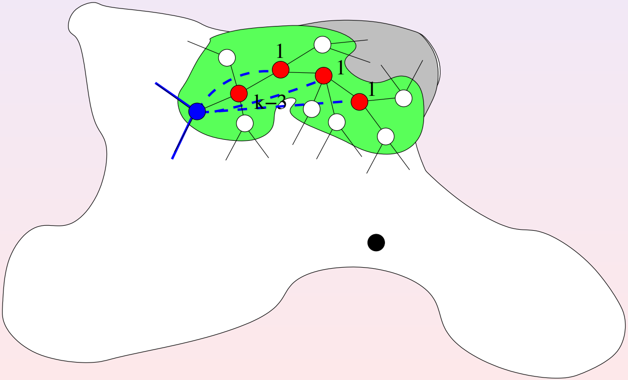
algorithm:

extension: if $w \in N(v_1) \cup N(v_i)$, Pw induced and $N(w) \cap C_{robber} \neq \emptyset$



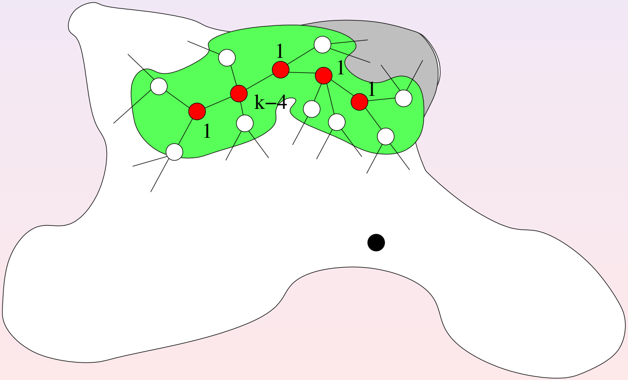
initialization: all k cops in one arbitrary node $P = \{v_1\}$
invariant: Cops always occupy an induced path $P = \{v_1, \cdots, v_i\}$
algorithm:

extension: if $w \in N(v_1) \cup N(v_i)$, Pw induced and $N(w) \cap C_{robber} \neq \emptyset$



initialization: all k cops in one arbitrary node $P = \{v_1\}$
invariant: Cops always occupy an induced path $P = \{v_1, \cdots, v_i\}$
algorithm:

extension: if $w \in N(v_1) \cup N(v_i)$, Pw induced and $N(w) \cap C_{robber} \neq \emptyset$

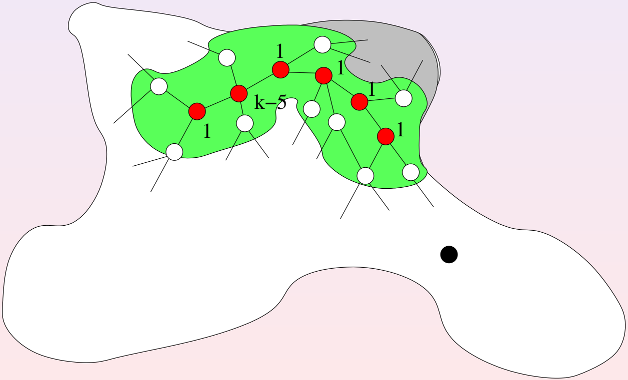


initialization: all k cops in one arbitrary node $P = \{v_1\}$

invariant: Cops always occupy an induced path $P = \{v_1, \cdots, v_i\}$

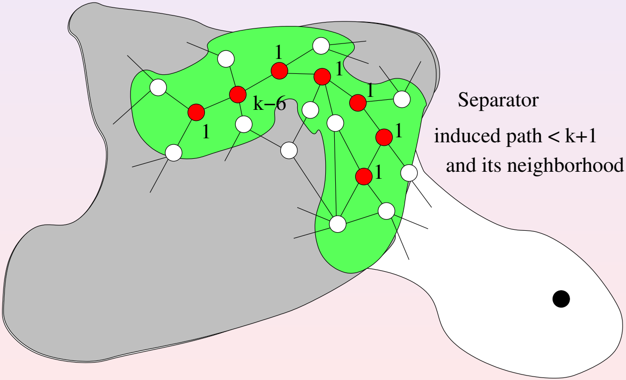
algorithm:

extension: if $w \in N(v_1) \cup N(v_i)$, Pw induced and $N(w) \cap C_{robber} \neq \emptyset$

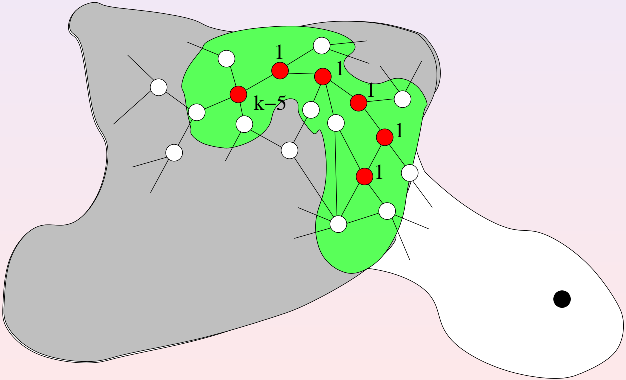


initialization: all k cops in one arbitrary node $P = \{v_1\}$
invariant: Cops always occupy an induced path $P = \{v_1, \cdots, v_i\}$
algorithm:

extension: if $w \in N(v_1) \cup N(v_i)$, Pw induced and $N(w) \cap C_{robber} \neq \emptyset$



initialization: all k cops in one arbitrary node $P = \{v_1\}$
invariant: Cops always occupy an induced path $P = \{v_1, \cdots, v_i\}$
algorithm: *retraction:* if v_1 or v_i cannot be extended, else
extension: if $w \in N(v_1) \cup N(v_i)$, Pw induced and $N(w) \cap C_{robber} \neq \emptyset$

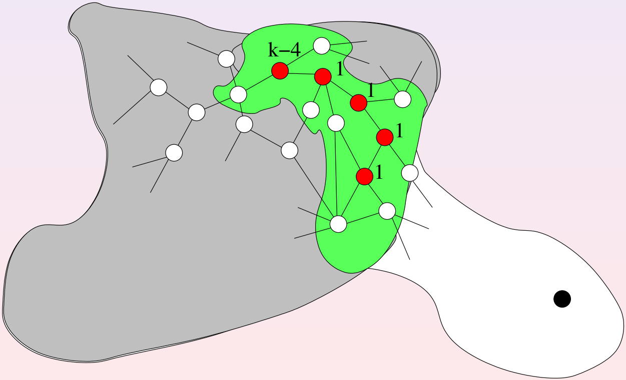


initialization: all k cops in one arbitrary node $P = \{v_1\}$

invariant: Cops always occupy an induced path $P = \{v_1, \cdots, v_i\}$

algorithm: *retraction:* if v_1 or v_i cannot be extended, else

extension: if $w \in N(v_1) \cup N(v_i)$, Pw induced and $N(w) \cap C_{robber} \neq \emptyset$

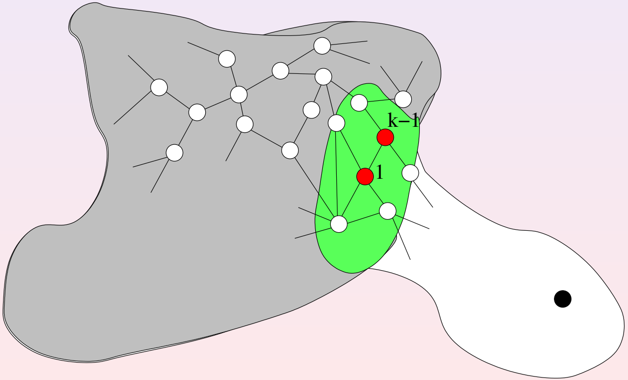


initialization: all k cops in one arbitrary node $P = \{v_1\}$

invariant: Cops always occupy an induced path $P = \{v_1, \cdots, v_i\}$

algorithm: *retraction:* if v_1 or v_i cannot be extended, else

extension: if $w \in N(v_1) \cup N(v_i)$, Pw induced and $N(w) \cap C_{robber} \neq \emptyset$

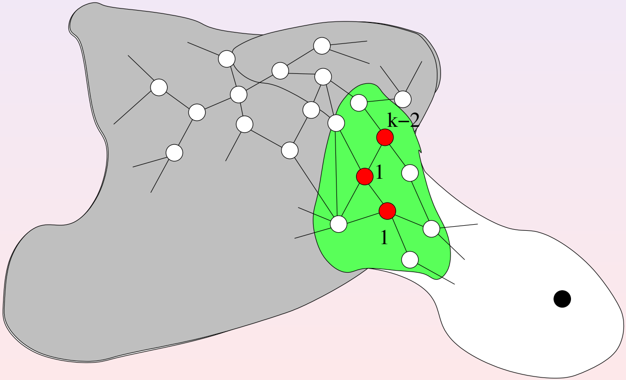


initialization: all k cops in one arbitrary node $P = \{v_1\}$

invariant: Cops always occupy an induced path $P = \{v_1, \cdots, v_i\}$

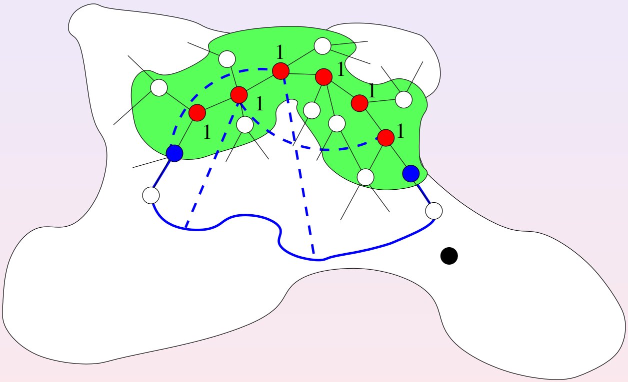
algorithm: *retraction:* if v_1 or v_i cannot be extended, else

extension: if $w \in N(v_1) \cup N(v_i)$, Pw induced and $N(w) \cap C_{robber} \neq \emptyset$



Capture in k-chordal graphs: worm's strategy

$\{v_1, \cdots, v_i\}$ occupied: if no retraction $\Rightarrow$ induced cycle $\geq i+1$

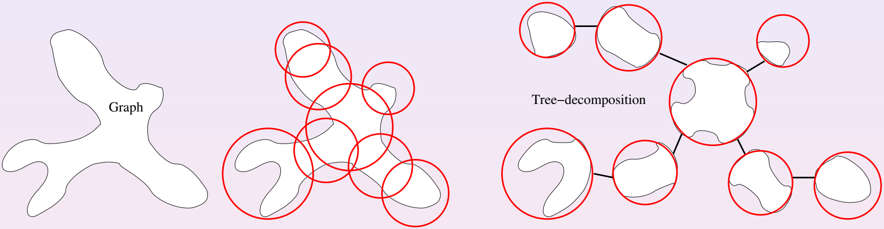


Theorem 1 greedy algorithm

worw's strategy uses $\leq k-1$ cops in k-chordal graphs

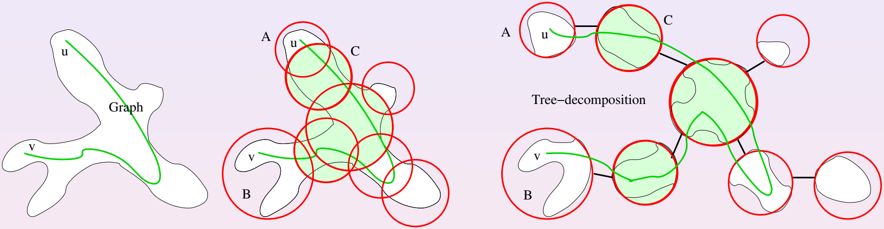
Tree-decomposition/treewidth (unformal)

Pieces (subgraphs) with tree-like structure (bag=separator)

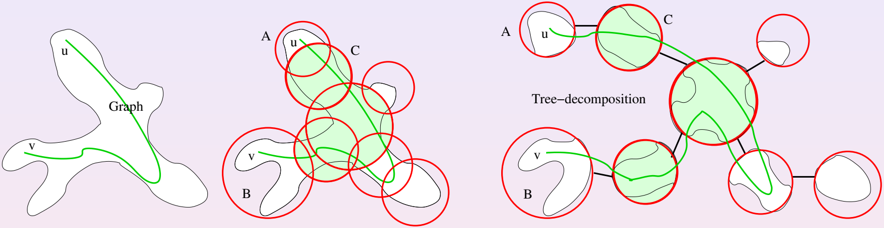


Tree-decomposition/treewidth (unformal)

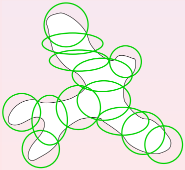
Pieces (subgraphs) with tree-like structure (bag=separator)



Pieces (subgraphs) with tree-like structure (bag=separator)

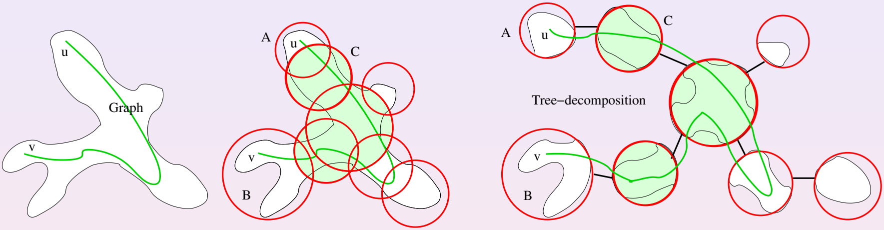


Usually, try to minimize the largest bag (treewidth)

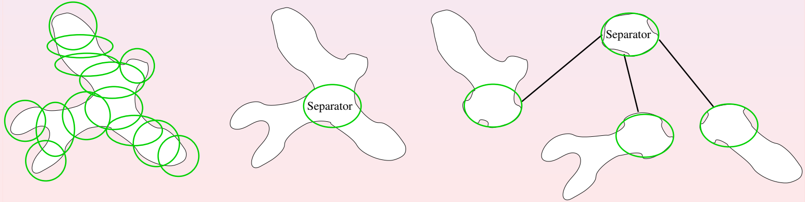


Tree-decomposition/treewidth (unformal)

Pieces (subgraphs) with tree-like structure (bag=separator)

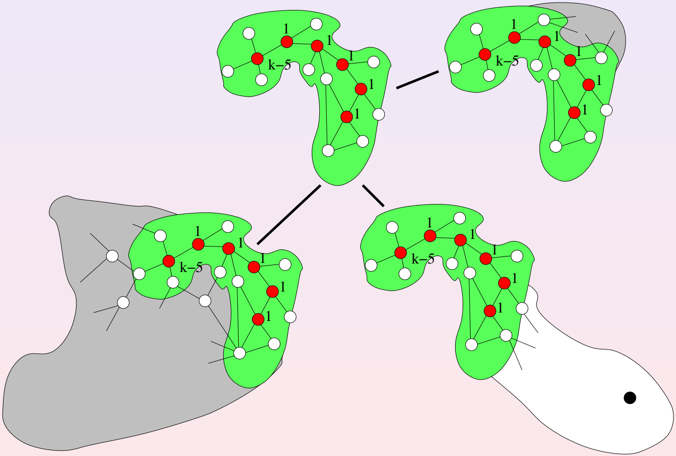


Computation: find a separator with desired properties, then induction



Tree-decomposition with k-induced paths

From k-worm's strategy



Tree-decomposition with k-induced paths

k-worm strategy $\Rightarrow$ decomposition with separator$=$ k-caterpillar

Theorem 2: main result

There is a $O(m^2)$-algorithm that, in any m-edge graph G,

- either returns an induced cycle larger than k,
- or compute a tree-decomposition with each bag being the closed neighborhood of an induced path of length $\leq k-1$.

In case of k-chordal graphs:
$\Rightarrow$ treewidth $\leq O(\Delta.k)$ (improves [Bodlaender,Thilikos'97] result)
$\Rightarrow$ treelength $\leq k$
$\Rightarrow$ hyperbolicity $\leq 3k/2$

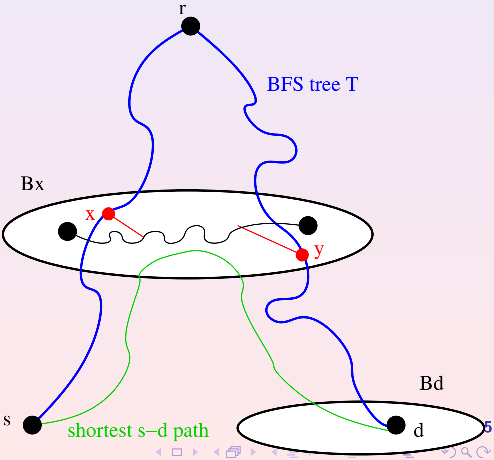
Application to compact routing

stretch $O(k \log \Delta)$ with RT's of size $O(k \log n)$ bits

BFS-tree T, tree-decomposition D with k-caterpillar separators

From s to d

1. follow the path to r in T
   until find x such that
   B_x is an ancestor of B_d in D
   stretch: $+k$
2. in B_x, find y an ancestor of d in T
   stretch: $+k \log \Delta$
3. follow the path to d in T
   stretch: $+k$



Further work

Routing

improve the stretch of our routing scheme

implementation in graphs with "few" long induced cycles

Decompositions

- complexity of computing decomposition with k-induced path, minimizing k
- algorithmic uses of such decompositions
- other structures of bags

Cops and robber

Conjecture: For any connected n-node graph G, $cn(G) = O(\sqrt{n})$. [Meyniel 87]