Faster Algorithms for Graph Monopolarity

G. Philip[1] and Shrinidhi T. Sridhara[2]

[1] *Chennai Mathematical Institute, India*
[2] *Université de Bordeaux, France*

WG 2025

The Problem

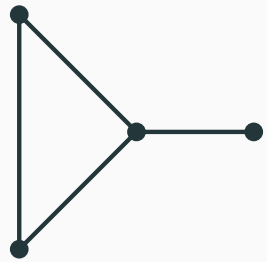
Monopolar Recognition

- Input: A graph $G = (V, E)$ on n vertices
- Question: Is there a partition $V = C \uplus I$ such that $G[C]$ is a *cluster graph* and I is an *independent set*?

The Problem

Monopolar Recognition

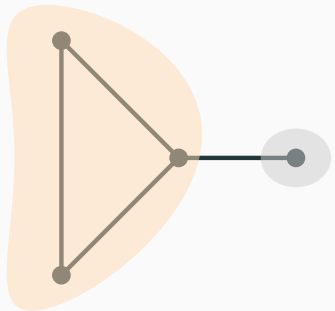
- Input: A graph $G = (V, E)$ on n vertices
- Question: Is there a partition $V = C \uplus I$ such that $G[C]$ is a *cluster graph* and I is an *independent set*?



The Problem

Monopolar Recognition

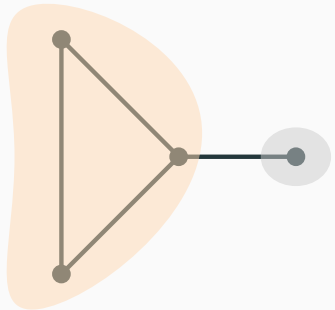
- Input: A graph $G = (V, E)$ on n vertices
- Question: Is there a partition $V = C \uplus I$ such that $G[C]$ is a *cluster graph* and I is an *independent set*?



The Problem

Monopolar Recognition

- Input: A graph $G = (V, E)$ on n vertices
- Question: Is there a partition $V = C \uplus I$ such that $G[C]$ is a *cluster graph* and I is an *independent set*?



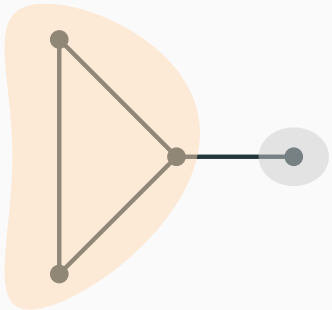
✓

Monopolar

The Problem

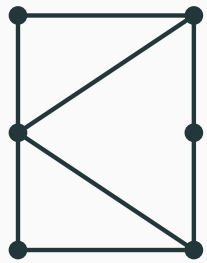
Monopolar Recognition

- Input: A graph $G = (V, E)$ on n vertices
- Question: Is there a partition $V = C \uplus I$ such that $G[C]$ is a *cluster graph* and I is an *independent set*?



✓

Monopolar



✗

Not Monopolar

Monopolarity: Some History

- Well-studied - `graphclasses.org`
- "Lighter version" of *polar* graphs*
  - Partitioning into a cluster graph and a *co-cluster* graph
  - Polar Recognition is NP-complete in very restricted classes like claw-free graphs†

*Tyshkevich and Chernyak, *Decomposition of graphs*, 1985

†Churchley and Huang, *On the polarity and monopolarity of graphs*, 2013

‡Farrugia, *Vertex-partitioning into fixed additive induced-hereditary properties is NP-hard*, 2004

§Le and Nevries, *Complexity and algorithms for recognizing polar and monopolar graphs*, 2014

Monopolarity: Some History

- Well-studied - `graphclasses.org`
- "Lighter version" of *polar* graphs*
  - Partitioning into a cluster graph and a *co-cluster* graph
  - Polar Recognition is NP-complete in very restricted classes like claw-free graphs†
- Monopolar Recognition is NP-complete‡, also in some restricted classes
  - E.g., triangle-free planar graphs§ of maximum degree 3
- Our main result: a fast exponential-time algorithm for Monopolar Recognition

---

*Tyshkevich and Chernyak, *Decomposition of graphs*, 1985

†Churchley and Huang, *On the polarity and monopolarity of graphs*, 2013

‡Farrugia, *Vertex-partitioning into fixed additive induced-hereditary properties is NP-hard*, 2004

§Le and Nevries, *Complexity and algorithms for recognizing polar and monopolar graphs*, 2014

Some Easy Exact Algorithms for Monopolar Recognition

- The straightforward algorithm:
  - Go over all partitions of V into two parts $V = C \uplus I$
  - Takes $\mathcal{O}^\star(2^n)$ time

Some Easy Exact Algorithms for Monopolar Recognition

- The straightforward algorithm:
  - Go over all partitions of V into two parts $V = C \uplus I$
  - Takes $\mathcal{O}^\star(2^n)$ time

- Easy improvement to $\mathcal{O}^\star(1.4423^n)$
  - Observation: If G has a monopolar partition, then it has one where the independent set is *inclusion-maximal*
  - Go over all *maximal* independent sets $I \subseteq V$
  - Can be done in $\mathcal{O}^\star(3^{n/3}) = \mathcal{O}^\star(1.4423^n)$ time[a]

[a] Johnson et al., *On generating all maximal independent sets*, 1988

Our Results

- We solve Monopolar Recognition in $\mathcal{O}^\star(1.3734^n)$ time
  - Improves on the $\mathcal{O}^\star(1.4423^n)$
  - Also finds a monopolar partition, if the graph is monopolar

*Kanj et al., *Parameterized algorithms for recognizing monopolar and 2-subcolorable graphs*, 2018

Our Results

- We solve Monopolar Recognition in $\mathcal{O}^\star(1.3734^n)$ time
  - Improves on the $\mathcal{O}^\star(1.4423^n)$
  - Also finds a monopolar partition, if the graph is monopolar
- This talk: a simpler version that solves Monopolar Recognition in $\mathcal{O}^\star(2^{n/2}) = \mathcal{O}^\star(1.4142^n)$ time

*Kanj et al., *Parameterized algorithms for recognizing monopolar and 2-subcolorable graphs*, 2018

Our Results

- We solve Monopolar Recognition in $\mathcal{O}^\star(1.3734^n)$ time
  - Improves on the $\mathcal{O}^\star(1.4423^n)$
  - Also finds a monopolar partition, if the graph is monopolar
- This talk: a simpler version that solves Monopolar Recognition in $\mathcal{O}^\star(2^{n/2}) = \mathcal{O}^\star(1.4142^n)$ time
- Two FPT algorithms for Monopolar Recognition
  1. $\mathcal{O}^\star(3.076^{k_v})$; k_v is the size of a smallest vertex modulator to claw-free graphs
  2. $\mathcal{O}^\star(2.253^{k_e})$; k_e is the size of a smallest edge modulator to claw-free graphs

*Kanj et al., *Parameterized algorithms for recognizing monopolar and 2-subcolorable graphs*, 2018

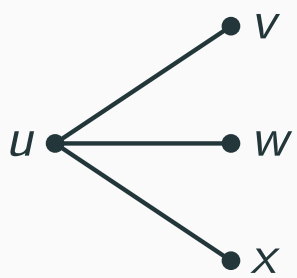
Our Results

- We solve Monopolar Recognition in $\mathcal{O}^\star(1.3734^n)$ time
  - Improves on the $\mathcal{O}^\star(1.4423^n)$
  - Also finds a monopolar partition, if the graph is monopolar
- This talk: a simpler version that solves Monopolar Recognition in $\mathcal{O}^\star(2^{n/2}) = \mathcal{O}^\star(1.4142^n)$ time
- Two FPT algorithms for Monopolar Recognition
  1. $\mathcal{O}^\star(3.076^{k_v})$; k_v is the size of a smallest vertex modulator to claw-free graphs
  2. $\mathcal{O}^\star(2.253^{k_e})$; k_e is the size of a smallest edge modulator to claw-free graphs
- Previous work*: $\mathcal{O}^\star(2^{k_c})$ algorithm for Monopolar Recognition
  - k_c is the *number of cliques* on the cluster side
  - Not comparable to either of k_v, k_e

*Kanj et al., *Parameterized algorithms for recognizing monopolar and 2-subcolorable graphs*, 2018

Our Exact Algorithm for Monopolarity: An Overview

- Known result: Monopolar Recognition is polynomial-time solvable on *claw-free* graphs[†]
  - Graphs that exclude a claw as an *induced* subgraph



[†] Churchley and Huang, *List monopolar partitions of claw-free graphs*, 2012

Our Exact Algorithm for Monopolarity: An Overview

- Monopolar Recognition is easy on claw-free graphs

Our Exact Algorithm for Monopolarity: An Overview

- Monopolar Recognition is easy on claw-free graphs
- Perhaps we can
  1. "Drill down" to a claw-free subgraph H of G,
  2. Solve Monopolar Recognition on H, and
  3. Lift the solution back to G?

Our Exact Algorithm for Monopolarity: An Overview

- Monopolar Recognition is easy on claw-free graphs
- Perhaps we can
  1. "Drill down" to a claw-free subgraph H of G,
  2. Solve Monopolar Recognition on H, and
  3. Lift the solution back to G?
- (We may need to do this for many such subgraphs H . . .)

Our Exact Algorithm for Monopolarity: An Overview

- Monopolar Recognition is easy on claw-free graphs
- Perhaps we can
  1. "Drill down" to a claw-free subgraph H of G,
  2. Solve Monopolar Recognition on H, and
  3. Lift the solution back to G?
- (We may need to do this for many such subgraphs H . . .)
- This is *roughly* what our algorithm does
  - We stop the drill-down *before* reaching a claw-free subgraph . . .
  - . . . in a certain sense . . .

Our Exact Algorithm for Monopolarity: An Overview

Lemma

Any monopolar graph with monopolar partition $V = C \uplus I$ has a claw-free modulator M such that $M \subseteq C$.

Our Exact Algorithm for Monopolarity: An Overview

Lemma

Any monopolar graph with monopolar partition $V = C \uplus I$ has a claw-free modulator M such that $M \subseteq C$.

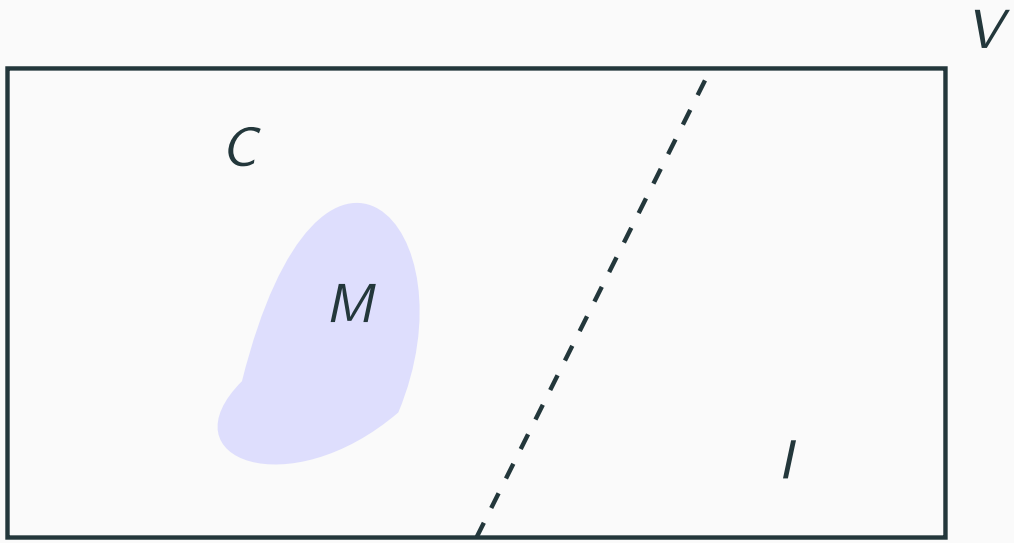
- This is a vertex subset whose deletion leaves a claw-free graph

Our Exact Algorithm for Monopolarity: An Overview

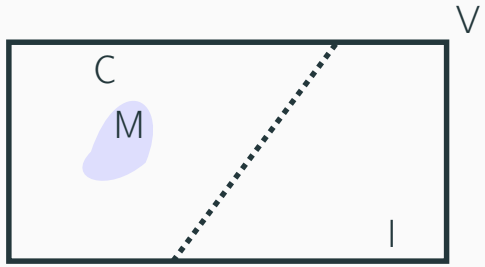
Lemma

Any monopolar graph with monopolar partition $V = C \uplus I$ has a claw-free modulator M such that $M \subseteq C$.

- This is a vertex subset whose deletion leaves a claw-free graph
- The subgraph $G[M]$ is not necessarily claw-free!

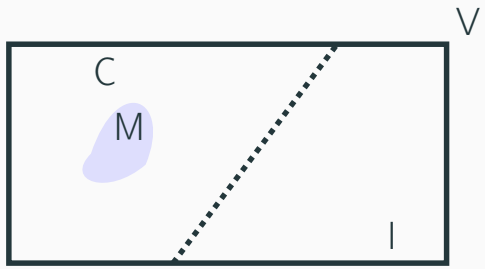


Our Exact Algorithm for Monopolarity: An Overview



- If G is monopolar, then G has a claw-free modulator M which lives entirely inside the cluster part of *some* monopolar partition of G.
  - We say that such an M is *good*.

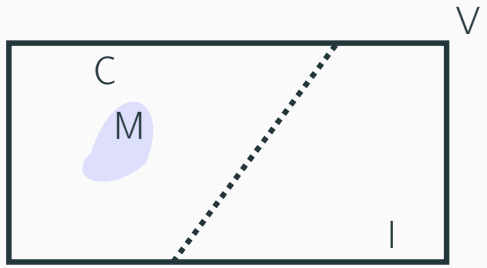
Our Exact Algorithm for Monopolarity: An Overview



- If G is monopolar, then G has a claw-free modulator M which lives entirely inside the cluster part of *some* monopolar partition of G.
  - We say that such an M is *good*.

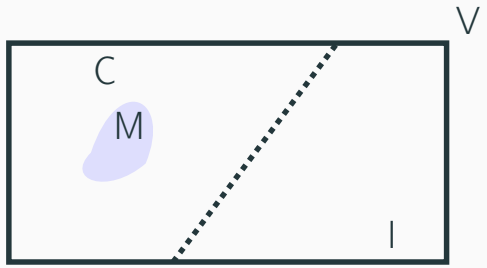
- It is not true that an *arbitrary* claw-free modulator of G can be made to live inside the cluster part of G.

Our Exact Algorithm for Monopolarity: An Overview



- Every monopolar graph G has a good claw-free modulator M.

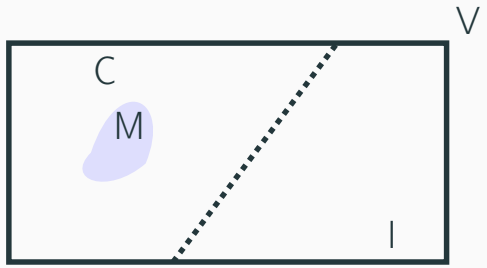
Our Exact Algorithm for Monopolarity: An Overview



- Every monopolar graph G has a good claw-free modulator M.

- Our algorithm (simplified version):
  1. Branches on induced claws: *guess* a good claw-free modulator M

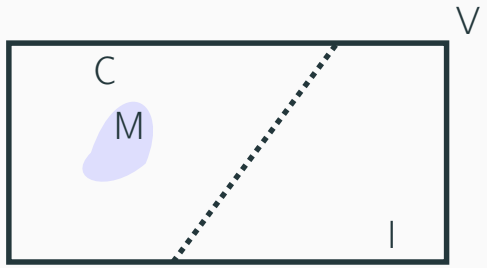
Our Exact Algorithm for Monopolarity: An Overview



- Every monopolar graph G has a good claw-free modulator M.

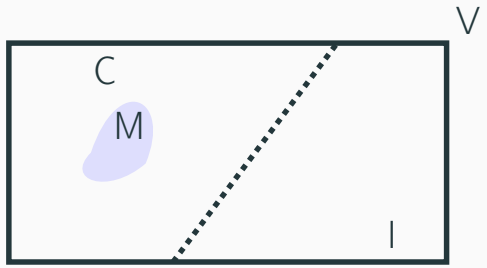
- Our algorithm (simplified version):
  1. Branches on induced claws: *guess* a good claw-free modulator M
  2. Checks for a monopolar partition $V = C \uplus I$ with $M \subseteq C$
     - In polynomial time

Our Exact Algorithm for Monopolarity: An Overview



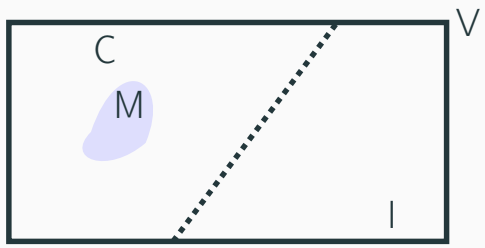
- Every monopolar graph G has a good claw-free modulator M.
- Our algorithm (simplified version):
  1. Branches on induced claws: *guess* a good claw-free modulator M
  2. Checks for a monopolar partition $V = C \uplus I$ with $M \subseteq C$
     - In polynomial time
- The first step needs some care, to make the branching faster

Our Exact Algorithm for Monopolarity: An Overview



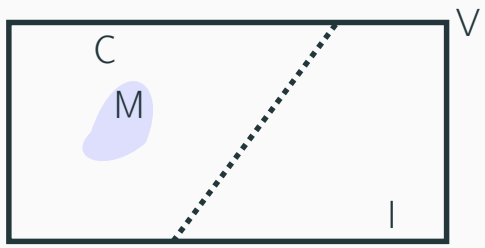
- Every monopolar graph G has a good claw-free modulator M.
- Our algorithm (simplified version):
  1. Branches on induced claws: *guess* a good claw-free modulator M
  2. Checks for a monopolar partition $V = C \uplus I$ with $M \subseteq C$
     - In polynomial time
- The first step needs some care, to make the branching faster
- The second step is the non-trivial part

Our Exact Algorithm for Monopolarity: The non-trivial part



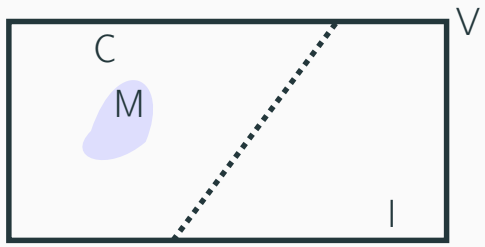
- Goal: Check if a claw-free modulator $M \subseteq V$ is good or not
  1. If $G[M]$ is not a cluster graph: discard this guess

Our Exact Algorithm for Monopolarity: The non-trivial part



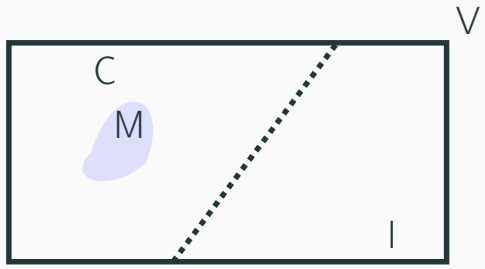
- Goal: Check if a claw-free modulator $M \subseteq V$ is good or not
  1. If $G[M]$ is not a cluster graph: discard this guess
  2. If $G - M$ is not monopolar: G is not monopolar either

Our Exact Algorithm for Monopolarity: The non-trivial part



- Goal: Check if a claw-free modulator $M \subseteq V$ is good or not
  1. If $G[M]$ is not a cluster graph: discard this guess
  2. If $G - M$ is not monopolar: G is not monopolar either
  3. If $G - M$ is monopolar: we are stuck
     - Not clear how to check if M is good
     - $G - M$ can have exponentially-many monopolar partitions
     - (E.g.: if $G - M$ is a matching.)
     - How to "extend" M to the "correct" cluster part of G?

Our Exact Algorithm for Monopolarity: The non-trivial part



- We show that we can do the non-trivial part in polynomial time

Lemma

Given an arbitrary graph G and a claw-free vertex modulator M of G, we can check if G has a monopolar partition where all of M belongs to the cluster part in polynomial time.

Checking if a given claw-free modulator is good

Our (simpler) algorithm $\mathcal{A}$ solves

- Input: Graph $G = (V, E)$, claw-free vertex modulator $M \subseteq V$
  - Where $G[M]$ is a cluster graph
- Question: Is there a monopolar partition $V = C \uplus I$ with $M \subseteq C$?

A special case, solved by Le and Nevries[‡]

- Input: *Claw-free* graph $G = (V, E)$, vertex subset $M \subseteq V$
  - Where $G[M]$ is a cluster graph
- Question: Is there a monopolar partition $V = C \uplus I$ with $M \subseteq C$?

- We generalize this to *arbitrary* graphs G and their claw-free modulators M

[‡]Le and Nevries, *Complexity and algorithms for recognizing polar and monopolar graphs*, 2014

Our Algorithms for Monopolarity: High-level overview

- Algorithm $\mathcal{A}$:
  1. Runs in polynomial time
  2. Checks if a given claw-free modulator M is good

Our Algorithms for Monopolarity: High-level overview

- Algorithm $\mathcal{A}$:
  1. Runs in polynomial time
  2. Checks if a given claw-free modulator M is good

- The $\mathcal{O}^\star(2^{n/2})$ algorithm for Monopolar Recognition:
  1. Branch on induced claws:
     - Go over all potentially good claw-free modulators M
     - In $\mathcal{O}^\star(2^{n/2})$ time
  2. Apply algorithm $\mathcal{A}$ on each such M

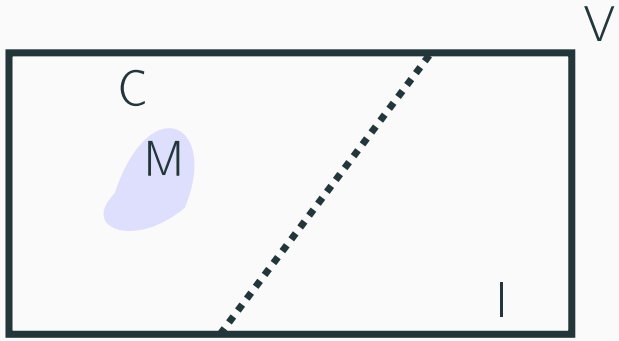
Our Algorithms for Monopolarity: High-level overview

- Algorithm $\mathcal{A}$:
  1. Runs in polynomial time
  2. Checks if a given claw-free modulator M is good

- The $\mathcal{O}^\star(2^{n/2})$ algorithm for Monopolar Recognition:
  1. Branch on induced claws:
     - Go over all potentially good claw-free modulators M
     - In $\mathcal{O}^\star(2^{n/2})$ time
  2. Apply algorithm $\mathcal{A}$ on each such M

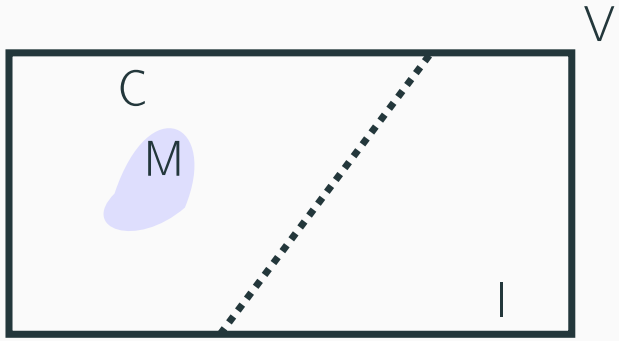
- The FPT algorithms:
  1. Find *one* claw-free vertex modulator
     - Using standard techniques
  2. Guess the intersection of M with this modulator
  3. Compute the rest of M
  4. Apply algorithm $\mathcal{A}$ on this M

Checking if a claw-free modulator M is good



"Is my M like this?"

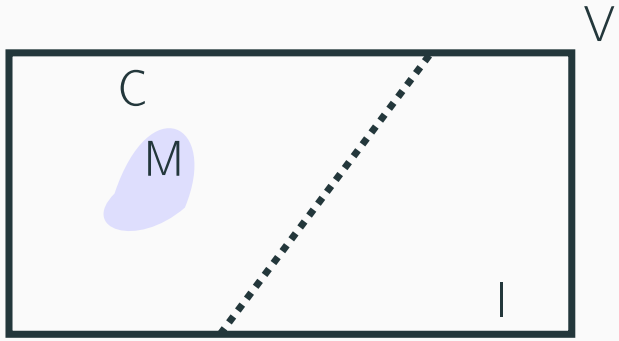
Checking if a claw-free modulator M is good



"Is my M like this?"

- If no induced P_3: trivial

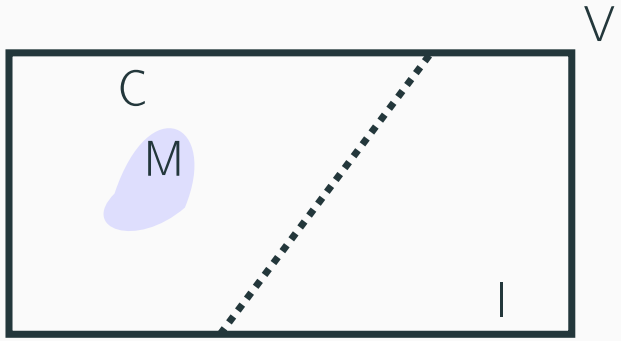
Checking if a claw-free modulator M is good



"Is my M like this?"

- If no induced P_3: trivial
- If G has induced P_3s: Look at small induced subgraphs that *contain* them
  - How they interact with M
  - Where they could potentially lie in a monopolar partition

Checking if a claw-free modulator M is good



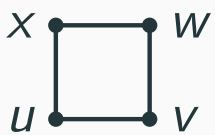
"Is my M like this?"

- If no induced P_3: trivial
- If G has induced P_3s: Look at small induced subgraphs that *contain* them
  - How they interact with M
  - Where they could potentially lie in a monopolar partition
- Use this to construct a 2SAT formula $\phi \equiv M$ is good
  - Building on Le and Nevries' arguments
  - (These worked when the graph G had no claws)

Small induced subgraphs

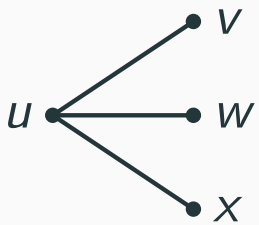
- The six connected graphs on four vertices:

(a) P_4

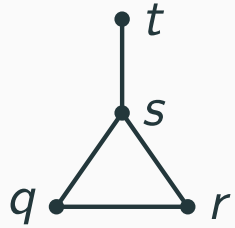


(b) C_4

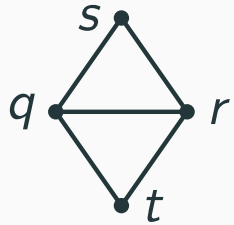
(c) K_4



(d) Claw



(e) Paw
$P(s, t)$

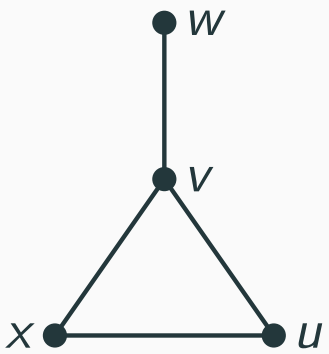


(f) Diamond
$D(s, t)$

- Each induced P_3 is a part of at least one such subgraph.

Handling the P_3s

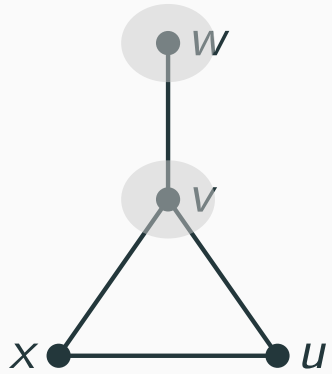
- Example[§]: Induced P_3 is part of a Paw $P(v, w)$.



[§]Le and Nevries, *Complexity and algorithms for recognizing polar and monopolar graphs*, 2014

Handling the P_3s

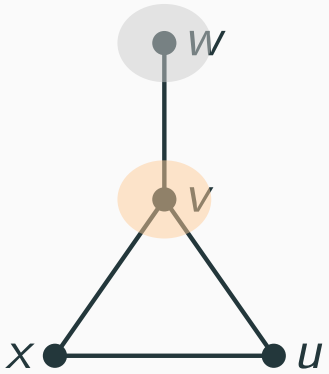
- Example[§]: Induced P_3 is part of a Paw $P(v, w)$.



[§]Le and Nevries, *Complexity and algorithms for recognizing polar and monopolar graphs*, 2014

Handling the P_3s

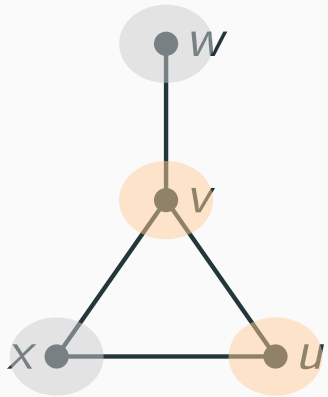
- Example[§]: Induced P_3 is part of a Paw $P(v, w)$.



[§]Le and Nevries, *Complexity and algorithms for recognizing polar and monopolar graphs*, 2014

Handling the P_3s

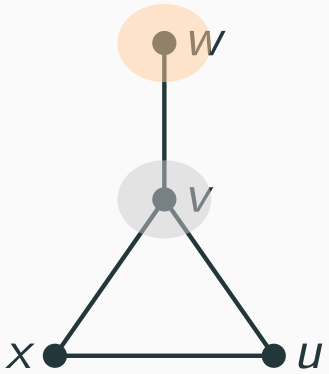
- Example[§]: Induced P_3 is part of a Paw $P(v, w)$.



[§]Le and Nevries, *Complexity and algorithms for recognizing polar and monopolar graphs*, 2014

Handling the P_3s

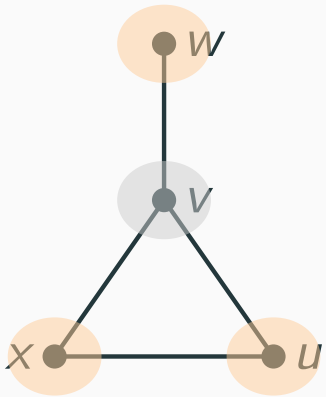
- Example[§]: Induced P_3 is part of a Paw $P(v, w)$.



[§]Le and Nevries, *Complexity and algorithms for recognizing polar and monopolar graphs*, 2014

Handling the P_3s

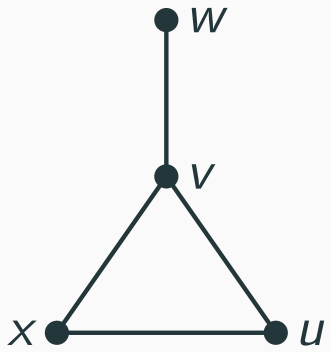
- Example[§]: Induced P_3 is part of a Paw $P(v, w)$.



[§]Le and Nevries, *Complexity and algorithms for recognizing polar and monopolar graphs*, 2014

Handling the P_3s

- Example[§]: Induced P_3 is part of a Paw $P(v, w)$.



- So, in any monopolar partition, either v or w is in the cluster part.
- Likewise, we decide constraints for other small graphs.
- Along with the main constraint of all vertices in M should be assigned to cluster part, we get a 2SAT instance.

[§]Le and Nevries, *Complexity and algorithms for recognizing polar and monopolar graphs*, 2014

Handling the remaining P_3s

- But, our graph may have induced P_3s far away from set M - no neighbours in M, not part of any small subgraphs.

¶Plus some more conditions

Handling the remaining P_3s

- But, our graph may have induced P_3s far away from set M - no neighbours in M, not part of any small subgraphs.
- We show that we can get rid of such P_3s and then apply Le and Nevries's 2SAT lemma
  - This is our main technical insight

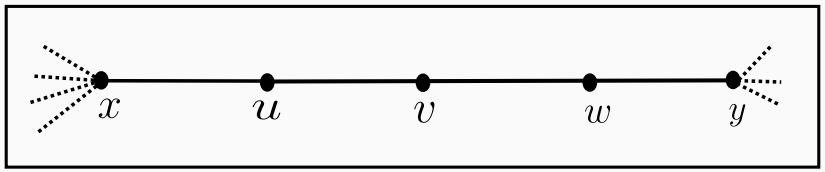
¶Plus some more conditions

Handling the remaining P_3s

- But, our graph may have induced P_3s far away from set M - no neighbours in M, not part of any small subgraphs.
- We show that we can get rid of such P_3s and then apply Le and Nevries's 2SAT lemma
  - This is our main technical insight

Lemma

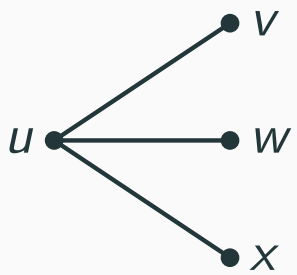
Let M be a claw-free vertex modulator of graph G. If uvw is an induced P_3 in G which does not have a neighbour[¶] in M, then *all* the vertices u, v, w are of degree exactly 2 in G.



[¶]Plus some more conditions

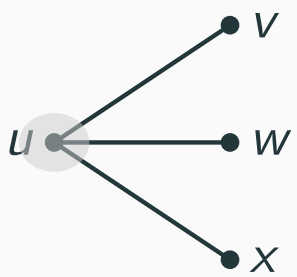
Branching on an induced claw

How to branch to a claw-free modulator M?



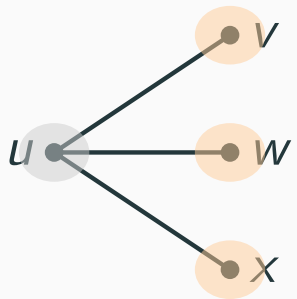
Branching on an induced claw

How to branch to a claw-free modulator M?



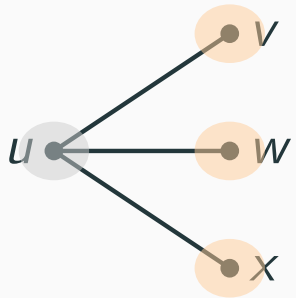
Branching on an induced claw

How to branch to a claw-free modulator M?



Branching on an induced claw

How to branch to a claw-free modulator M?

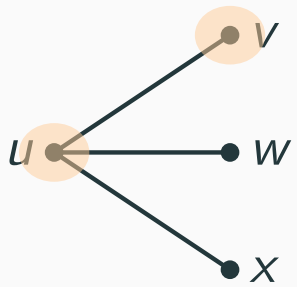


Put v, w, x to M

Measure drops by 4

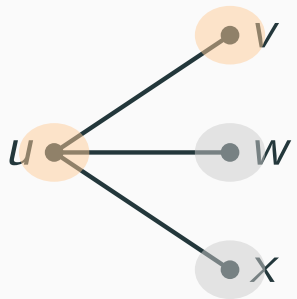
Branching on an induced claw

How to branch to a claw-free modulator M?



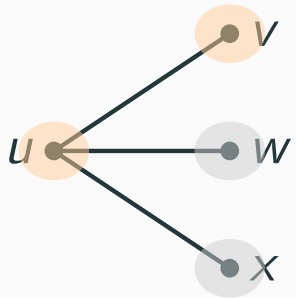
Branching on an induced claw

How to branch to a claw-free modulator M?



Branching on an induced claw

How to branch to a claw-free modulator M?

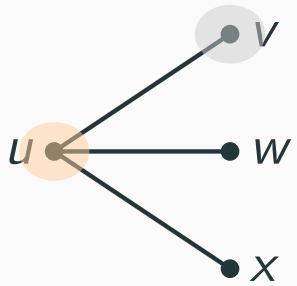


Put u, v to M

Measure drops by 4

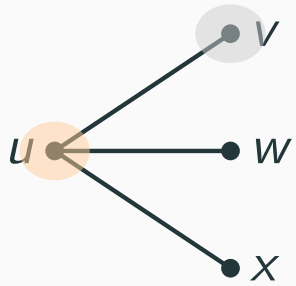
Branching on an induced claw

How to branch to a claw-free modulator M?



Branching on an induced claw

How to branch to a claw-free modulator M?

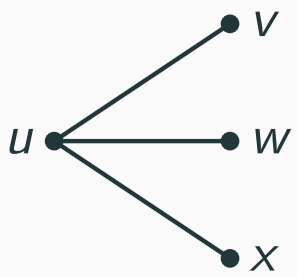


Put u to M, no other implication

Measure drops by just 2

Branching on an induced claw

How to branch to a claw-free modulator M?



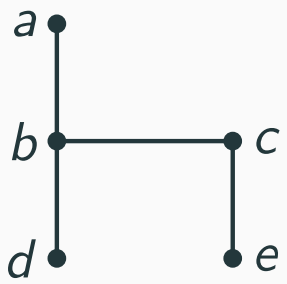
- Three-way branching
- The number of undecided vertices drops by $(4, 4, 2)$
- Branching tree with $O(2^{n/2})$ nodes

The faster $\mathcal{O}^*(1.3734^n)$ algorithm

- Main insights of the faster algorithm are:

The faster $\mathcal{O}^\star(1.3734^n)$ algorithm

- Main insights of the faster algorithm are:
  - The non-trivial part of deciding if a chair-free vertex modulator is good can still be solved in *polynomial* time.
  - There exists a strategy to branch *faster* towards these modulators.



Our algorithm $\mathcal{A}$ solves

- Input: Graph $G = (V, E)$, chair-free vertex modulator $M \subseteq V$
  - Where $G[M]$ is a cluster graph
- Question: Is there a monopolar partition $V = C \uplus I$ with $M \subseteq C$?

Some open problems

1. Can we improve on the exact exponential running time?

[‖] Kanj et al., *Parameterized algorithms for recognizing monopolar and 2-subcolorable graphs*, 2018

Some open problems

1. Can we improve on the exact exponential running time?
   - ETH lower bound[‖]: $\mathcal{O}^\star(2^{\Omega(m+n)}) = \mathcal{O}^\star(2^{\Omega(n)})$

[‖] Kanj et al., *Parameterized algorithms for recognizing monopolar and 2-subcolorable graphs*, 2018

Some open problems

1. Can we improve on the exact exponential running time?
   - ETH lower bound[‖]: $\mathcal{O}^\star(2^{\Omega(m+n)}) = \mathcal{O}^\star(2^{\Omega(n)})$
2. Can we solve *polar* recognition faster than $\mathcal{O}^\star(2^n)$?

[‖] Kanj et al., *Parameterized algorithms for recognizing monopolar and 2-subcolorable graphs*, 2018

Some open problems

1. Can we improve on the exact exponential running time?
   - ETH lower bound[‖]: $\mathcal{O}^\star(2^{\Omega(m+n)}) = \mathcal{O}^\star(2^{\Omega(n)})$
2. Can we solve *polar* recognition faster than $\mathcal{O}^\star(2^n)$?
   - Perhaps using our algorithm for monopolarity as a starting point?

[‖] Kanj et al., *Parameterized algorithms for recognizing monopolar and 2-subcolorable graphs*, 2018

Some open problems

1. Can we improve on the exact exponential running time?
   - ETH lower bound[‖]: $\mathcal{O}^\star(2^{\Omega(m+n)}) = \mathcal{O}^\star(2^{\Omega(n)})$
2. Can we solve *polar* recognition faster than $\mathcal{O}^\star(2^n)$?
   - Perhaps using our algorithm for monopolarity as a starting point?
3. FPT algorithms for Monopolar Recognition *without* finding a small modulator to claw-free graphs?

[‖] Kanj et al., *Parameterized algorithms for recognizing monopolar and 2-subcolorable graphs*, 2018

Some open problems

1. Can we improve on the exact exponential running time?
   - ETH lower bound[‖]: $\mathcal{O}^\star(2^{\Omega(m+n)}) = \mathcal{O}^\star(2^{\Omega(n)})$
2. Can we solve *polar* recognition faster than $\mathcal{O}^\star(2^n)$?
   - Perhaps using our algorithm for monopolarity as a starting point?
3. FPT algorithms for Monopolar Recognition *without* finding a small modulator to claw-free graphs?
   - The bottleneck in our algorithms is finding these modulators.

[‖] Kanj et al., *Parameterized algorithms for recognizing monopolar and 2-subcolorable graphs*, 2018

Some open problems

1. Can we improve on the exact exponential running time?
   - ETH lower bound[‖]: $\mathcal{O}^\star(2^{\Omega(m+n)}) = \mathcal{O}^\star(2^{\Omega(n)})$
2. Can we solve *polar* recognition faster than $\mathcal{O}^\star(2^n)$?
   - Perhaps using our algorithm for monopolarity as a starting point?
3. FPT algorithms for Monopolar Recognition *without* finding a small modulator to claw-free graphs?
   - The bottleneck in our algorithms is finding these modulators.
4. Find a (vertex, or edge) modulator to claw-free graphs in faster FPT time?

[‖] Kanj et al., *Parameterized algorithms for recognizing monopolar and 2-subcolorable graphs*, 2018

Some open problems

1. Can we improve on the exact exponential running time?
   - ETH lower bound[‖]: $\mathcal{O}^\star(2^{\Omega(m+n)}) = \mathcal{O}^\star(2^{\Omega(n)})$
2. Can we solve *polar* recognition faster than $\mathcal{O}^\star(2^n)$?
   - Perhaps using our algorithm for monopolarity as a starting point?
3. FPT algorithms for Monopolar Recognition *without* finding a small modulator to claw-free graphs?
   - The bottleneck in our algorithms is finding these modulators.
4. Find a (vertex, or edge) modulator to claw-free graphs in faster FPT time?
   - We used the d-Hitting Set algorithm as a black box

[‖] Kanj et al., *Parameterized algorithms for recognizing monopolar and 2-subcolorable graphs*, 2018

Thank you!