Faster Algorithms for Graph Monopolarity

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

[1]*Chennai Mathematical Institute, India*
[2]*University of Bordeaux, France*

CMI CS Seminar

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*?

- Cluster graph: each component is a complete graph
- Independent set: there are no edges
- There is no restriction on the edges across the two parts C, I
- A graph is *monopolar* if it admits a monopolar partition*

*Ekim et al., *Polarity of chordal graphs*, 2008

The Problem

Monopolar Recognition

- Input: A graph $G = (V, E)$ on n vertices
- Question: Is G monopolar?
  - $V = C \uplus I$
    - $G[C]$: a cluster graph
    - I : an independent set

- NP-complete even in some restricted graph classes
- Our main result: a fast exponential-time algorithm for Monopolar Recognition

Some Easy Exact Algorithms for MONOPOLAR RECOGNITION

- The straightforward algorithm:
  - Go over all partitions of V into two parts $V = C \uplus I$
    - Check if $V = C \uplus I$ is a monopolar partition
  - Takes $\mathcal{O}^\star(2^n)$ time
    - n is the number of vertices in G
    - $\mathcal{O}^\star()$ hides polynomial factors in n

Some Easy Exact Algorithms for Monopolar Recognition

- The straightforward algorithm:
  - Go over all partitions of V into two parts $V = C \uplus I$
    - Check if $V = C \uplus I$ is a monopolar partition
  - Takes $\mathcal{O}^\star(2^n)$ time
    - n is the number of vertices in G
    - $\mathcal{O}^\star()$ hides polynomial factors in n
- 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 $I \subseteq V$
    - Set $C = V \setminus I$
    - Check if $V = C \uplus I$ is a monopolar partition
  - 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
- This talk: a simpler version that solves Monopolar Recognition in $\mathcal{O}^\star(2^{n/2}) = \mathcal{O}^\star(1.4142^n)$ time

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

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[a]: $\mathcal{O}^\star(2^{k_c})$ algorithm for MONOPOLAR RECOGNITION
  - k_c is the *number of cliques* on the cluster side
  - Not comparable to either of k_v, k_e

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

Monopolarity: Some History

- Well-studied
  - E.g., one of the handful of problems listed at `graphclasses.org`
- The story starts with *polar* graphs[†]
  - Generalization of both split graphs and bipartite graphs
  - Partitioning into a cluster graph and a *co-cluster* graph
  - Co-cluster $\equiv$ complete multipartite with at least one part
    - $\equiv$ edge-complement of a cluster graph
- Polar Recognition is NP-complete in very restricted classes
  - E.g.: claw-free graphs[‡]

[†] Tyshkevich and Chernyak, *Decomposition of graphs*, 1985
[‡] Churchley and Huang, *On the polarity and monopolarity of graphs*, 2013

Monopolarity: Some History

- Monopolar graphs: a "lighter" version of polar graphs
  - A simplest variant of polar graphs
  - That generalizes both split graphs and bipartite graphs
- Used to solve POLAR RECOGNITION in various graph classes
  - In polynomial time
  - E.g.: Cographs[§], chordal graphs[¶], line graphs[‖], permutation graphs[**]

§ Ekim et al., *Polar cographs*, 2008

¶ Ekim et al., *Polarity of chordal graphs*, 2008

‖ Churchley and Huang, *Line-polar graphs: characterization and recognition*, 2011

** Ekim et al., *Polar permutation graphs are polynomial-time recognisable*, 2013

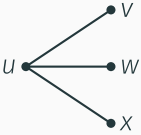
Monopolarity: Some History

- Polynomial-time algorithms for *polarity* work in two steps
- Step 1: Solve "Is the graph monopolar?"
  - Easier than the polar question in these classes
  - If Yes: we are done
- Step 2: If No:
  - Reveals lots more structure
  - Makes the polarity question easier
- MONOPOLAR RECOGNITION is NP-complete*, also in some restricted classes
  - E.g., triangle-free planar graphs† of maximum degree 3

*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

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

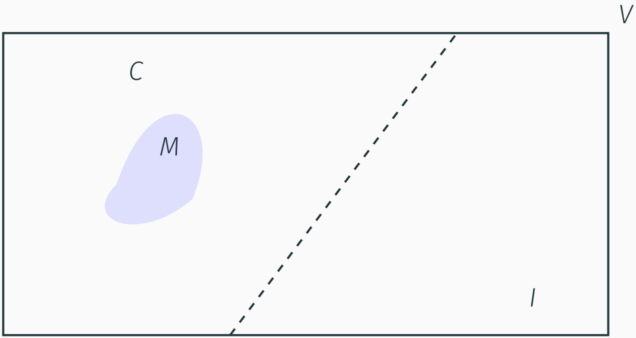
Our Exact Algorithm for Monopolarity: An Overview

- MONOPOLAR RECOGNITION is easy on claw-free graphs
- 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 …

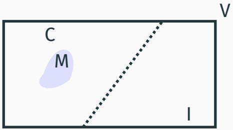
Our Exact Algorithm for Monopolarity: An Overview

- Observation: *Any* monopolar graph $G = (V, E)$ has:
  1. A monopolar partition $V = C \uplus I$, and
  2. A *claw-free modulator* M
     - This is a vertex subset whose deletion leaves a claw-free graph
     - The subgraph $G[M]$ is not necessarily claw-free!

  , where $M \subseteq C$ holds.

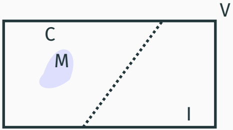


Our Exact Algorithm for Monopolarity: An Overview



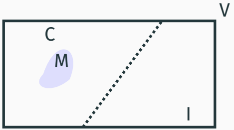
- If G is monopolar, then G has a claw-free modulator M which lives entirely inside the cluster part of *some* monopolar partition of G.
    - We say that such an M is *good*.
- It is not true that an *arbitrary* claw-free modulator of G can be made to live inside the cluster part of G.

Our Exact Algorithm for Monopolarity: An Overview



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

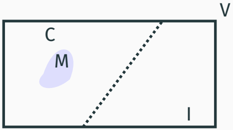
Our Exact Algorithm for Monopolarity: The non-trivial part



- 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
  2. $G - M$ is claw-free; we check if $G - M$ is monopolar
     - Recall: This can be done in polynomial time[§]
  3. If $G - M$ is not monopolar: G is not monopolar either
     - Discard this guess
  4. 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?

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

Our Exact Algorithm for Monopolarity: The non-trivial part



- We show that we can do the non-trivial part in polynomial time
- We design an algorithm $\mathcal{A}$ which takes as input:
  1. an arbitrary graph G and
  2. a claw-free vertex modulator M of G

  runs in polynomial time, and checks if G has a monopolar partition where all of M belongs to the cluster part.

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 given claw-free modulator is good

Our 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 $M \subseteq C$?

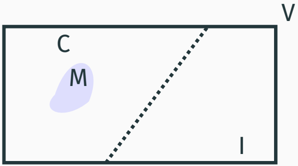
A special case, solved by Le and Nevries[¶]

- 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

Checking if a claw-free modulator M is good



"Is my M like this?"

- Given: $G[M]$ is a cluster graph, and $G - M$ is claw-free
- If G has no induced P_3s:
  - G is already a cluster graph
  - The answer is, trivially, yes
- So we assume: G has induced P_3s

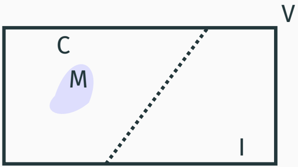
Checking if a claw-free modulator M is good

- G has induced P_3s
- We look at these P_3s and small induced subgraphs that *contain* these P_3s
  - How they interact with M
  - Where they could potentially lie in a monopolar partition
- And use this information to construct a 2SAT formula ϕ which is satisfiable if and only if M is good
  - Building on Le and Nevries' arguments
  - (These worked when the graph G had no claws)

Constructing the 2SAT Formula ϕ

- One variable x_v for each vertex v
- Goal: x_v will be True in a satisfying assignment of ϕ if and only if there is a monopolar partition:
  1. that "extends" M, and,
  2. in which vertex v is in the *independent set*

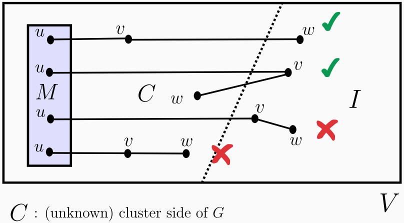
Constructing the 2SAT Formula ϕ



"Is my M like this?"

- Some easy clauses:
  - $\neg x_v$ for each vertex $v \in M$
    - Such a vertex v must be in the cluster part
  - $(\neg x_u \vee \neg x_v)$ for each edge $uv \in E$
    - Both end-points of an edge cannot *together* be in the independent set

The 2SAT Formula ϕ: clauses for induced P_3s which intersect M



C : (unknown) cluster side of G

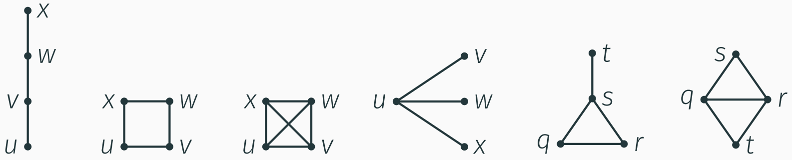
I : (unknown) independent side of G

M : (known) claw-free modulator of G

- For a vertex $u \in M$ and induced P_3s uvw or vuw
  - Add the clause $(x_v \vee x_w)$
    - All of the P_3 cannot be in the cluster graph $G[C]$

More clauses in ϕ: small induced subgraphs

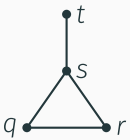
- The six connected graphs on four vertices:



(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.

More clauses in ϕ: small induced subgraphs



Paw $P(s, t)$

- For an induced paw $P(s, t)$: Add $(x_s \vee x_t)$ to ϕ
    - If both s, t are in the cluster, then neither of q, r can be in the cluster
    - But both of q, r can't be in the independent set, either
- Similar clauses for diamonds and C_4s
- And for K_4s and P_4s with at least one vertex in M

Handling the remaining P_3s

- All other induced P_3s—if any—are "far away" from M
  - They have no neighbour in M

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

Handling the remaining P_3s

- All other induced P_3s—if any—are "far away" from M
  - They have no neighbour in M

- If there are no "far-away" P_3s, we are done:

Lemma[‖]

Let $S \subseteq V(G)$ be such that

1. $G[S]$ is a cluster graph, and
2. no induced P_3 is "far away" from S.

Let ϕ be the 2SAT formula constructed as described above. Then G has a monopolar partition with S in the cluster part, if and only if ϕ is satisfiable.

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

Handling the remaining P_3s

- All other induced P_3s—if any—are "far away" from M
  - They have no neighbour in M

- If there are no "far-away" P_3s, we are done:

Lemma[‖]

Let $S \subseteq V(G)$ be such that

1. $G[S]$ is a cluster graph, and
2. no induced P_3 is "far away" from S.

Let ϕ be the 2SAT formula constructed as described above. Then G has a monopolar partition with S in the cluster part, if and only if ϕ is satisfiable.

- But there *could* be induced P_3s far away from M ...

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

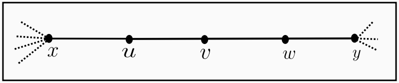
Handling the remaining P_3s

- Our graph may have induced P_3s far away from set M
- If we could somehow get rid of all such P_3s
  - Then we can apply Le and Nevries' lemma
- We show that we can indeed get rid of such P_3s
  - This is our main technical insight

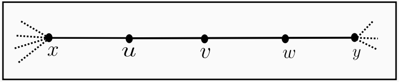
Handling the remaining P_3s

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

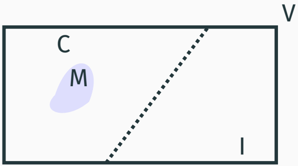


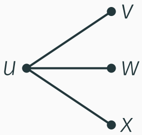
- Observation: Consider an induced P_3 uvw with $deg(u) = deg(v) = deg(w) = 2$. Let $x \neq v, y \neq v$ be the "other" neighbours of u, w, respectively. If we know the parts of a monopolar partition to which u, w, x, y belong, then we can fix the part of v as well.
    - $u \in C, w \in C$: v goes to I
    - $u \in I, w \in I$: v goes to C
    - $u \in I, w \in C, y \in I$: v goes to C
    - Similarly for the remaining cases
        - Some of them, a bit more complex
- We can *safely* delete v and recursively solve for $G - v$

Putting it all together ...

- We get rid of induced P_3s that have no neighbours in M
  - Delete their middle vertices
- We express the remaining problem on subgraph (say) G' as a 2SAT instance ϕ, and solve it
  - Invoking Le and Nevries' lemma
- A satisfying assignment for ϕ yields a monopolar partition of G' that "extends" M
  - We can put back the deleted vertices, while fixing their "side" (cluster or independent)
- If ϕ is not satisfiable, then there is no such monopolar partition for G', and so also for G.
- Thus we solve the base case of our branching, in polynomial time

Branching on an induced claw





- We branch on where u, v go in the monopolar partition $V = C \uplus I$
  1. $u \in I \implies \{v, w, x\} \subseteq C$
     - Add v, w, x to M
  2. $u \in C, v \in C \implies \{w, x\} \subseteq I$
     - Add u, v to M
  3. $u \in C, v \in I$ (no further implication)
     - Add u to M
- Three-way branching
- The number of undecided vertices drops by $(4, 4, 2)$
- Branching tree with $\mathcal{O}^\star(2^{n/2})$ nodes

Summarizing ...

- The $\mathcal{O}^\star(1.414^n)$ algorithm:
  - Branch on induced claws to guess a "good" $M \subseteq V$
  - Verify the guess using the 2SAT idea
- The $\mathcal{O}^\star(1.3734^n)$ algorithm:
  - Branch on induced chairs (a.k.a *forks*) to guess a "good" $M \subseteq V$
  - Verify the guess using the 2SAT idea
  - Both parts are more involved
- The two FPT algorithms: $\mathcal{O}^\star(3.076^{k_v})$, $\mathcal{O}^\star(2.253^{k_e})$
  - Use standard techniques to find *one* modulator
  - Guess the part of M that lives "in" the modulator
    - This gives us the rest of M as well
  - Verify the guess using the 2SAT idea

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!